Whitepaper Encryption in Microsoft Azure and Amazon AWS Cloud

Management Summary

Problem statement / relevance

Cloud (IaaS and PaaS) are used by 92% of the educational institutions[1]. The use of the cloud brings economy of scale, continuous new features, and enables IT departments to switch from Capital expense (CAPEX) into Operational Expense (OPEX)[2]. The cloud also changes the security exposure of an organisation caused by for example the increased integration opportunities and configuration options.

Reducing impact of data breach

The technical capability of encryption can be used to reduce the impact of certain classes of data breaches, since encrypted data can not be read without the decryption key. Nothing is for free so the use of encryption will provide a tradeoff in usability, performance, complexity and costs. This paper aims to provide insight into this tradeoff.

Improving security

We define security as the level of knowing that what occurs in reality is according to the intentional design. This defines insecure as something that happens and you did not foresee, and as something you intend to see and does not occur.

To improve security in the cloud you can adopt a process of automated configuration and validation, this is called Secure Software Development Lifecycle[3] (SSDLC). This whitepaper mostly focuses on data storage and touches on data transport in the code and deployment phases of the SSDLC.

The following five functions[4] need to be addressed to improve security: Identify, Protect, Detect, Respond and Recover, and are defined by the US National Institute of Standards and Technology (NIST). This whitepaper focuses itself on the function *Protect*.

Please note that this paper addresses a very specific set of issues. The advice and analysis in this whitepaper addresses the encryption of data at rest and in transit with regards to the options available when working with block storage attached to virtual machine instance, object storage, file storage, managed relational databases and related Key Management features. Many factors that impact the security of the solution, as a whole, are dependent on much more as for example secure deployment, secure identity management, etc.

Research questions

This whitepaper addresses the question: "*How can encryption of data at rest and data in transit be used to improve the security of cloud usage with regards to access to data and the impact of a data-breach, in the context of NL higher education institutions for the Microsoft Azure and Amazon AWS cloud.*"

[1] https://www.acm.nl/system/files/documents/Marktstudie%20cloudservices_DEF_Openbaar.pdf
[2] https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/strategy/business-outcomes/fiscal-outcomes
[3] https://dodcio.defense.gov/Portals/0/Documents/DoD%20Enterprise%20DevSecOps%20Reference%20Design%20v1.0_Public%20Release.pdf
[4] https://www.nist.gov/cyberframework/online-learning/five-functions

This whitepaper is intended for CISOs and cloud engineers working in the Dutch higher educational sector, to inform them on the nuances regarding the following research questions:

1) Where do encryption keys live and where does the encryption happen?
    a) What are the Key Management options, e.g. Cloud Service Provider managed, customer managed, external, etc?
    b) How are keys created, stored and managed in the AWS and Azure cloud?
    c) What are the available security best practices for managing keys with regards to key access policies, key rotation, logging of CRUD operations on keys and use of keys, etc.?
    d) Is the Key Management Solution tamper proof for physical attacks certified for the FIPS 140-2 standard[5]?
    e) Is encryption in transit applied between the client and service (in case of server-side encryption)?
2) Which encryption algorithms are available and which options are available?
    a) Do the available options depend on the tier of the service/ application?
3) What are the risks?
    a) What is the exposure when an encryption key is retrieved by another actor?
    b) What is the impact on the complexity of the solution and the responsibilities in regards to service management?
4) What are the costs?
    a) What are the pricing models that are relevant to the use case?
    b) What are the ballpark costs associated with certain services?

Use cases

We scope our research to a specific set of services offered by AWS and Microsoft Azure, namely

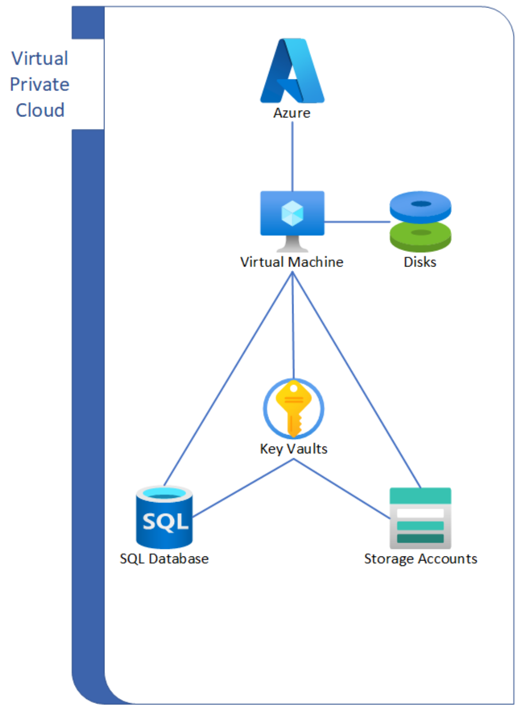
- Block storage (Amazon EBS and EC2 instance storage, Azure managed disks)
- Object storage (Amazon S3, Azure blob storage)
- File sharing (Amazon FSx for Windows File Server, Azure Files)
- Managed database (Amazon RDS, Azure SQL and Azure Database for MySQL & PostgreSQL)

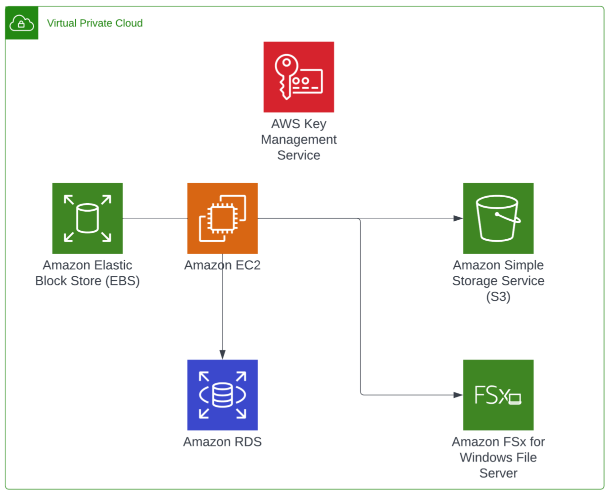
These four services are commonly used in two-tier application architectures. The following two main use cases serve as examples for educational institutions.

1. **Business Application Solution**
    a. Use Case: Timetable solution for educational institutions.
    b. Components: Application VM + block storage volumes + object storage container + Managed SQL database + File share.
2. **Analytics Solution**
    a. Use Case: Local processing of research data with SPSS/R.

[5] https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp3617.pdf

b. Components: Notebook VM + block storage volumes + object storage container + File share.




Virtual Private Cloud
AWS Key Management Service
Amazon Elastic Block Store (EBS)
Amazon EC2
Amazon Simple Storage Service (S3)
Amazon RDS
Amazon FSx for Windows File Server

Summary and conclusions

Encryption of data is one part of many steps in a secure software development lifecycle (SSDLC). It can be used to reduce the impact of certain classes of data breaches, since encrypted data can not be read without the decryption key. Additionally, (decryption) key access policies can be used to reduce data access to specific users and/or systems.

Both AWS and Microsoft Azure offer key management services and encryption options that enable customers to encrypt data using Platform-managed-, Customer-Managed- or Customer-Provided keys (keys hosted in a customer key store). These services are integrated with client libraries provided by the cloud service providers to enable Client-Side Encryption, and other cloud services to enable Server-Side Encryption.

While these key management services and encryption options enable customers to protect data against unauthorised access. Protection against unauthorised access by or via the cloud provider, using Server-Side Encryption, depends on technical measures and procedural measures taken by the provider.

When customers use Client-Side Encryption for object storage the cloud service providers have no technical means to read the customer data. We conclude with a high degree of confidence that this is also the case for data on Amazon EBS virtual disks used with Amazon EC2 virtual or bare-metal machines that run on Nitro hosts (all current generation EC2 instances). The same, finally, is true for using Microsoft SQL Server Always Encrypted, which leverages Client-Side Encryption.

For all other services investigated, protection depends in part on procedural measures. And while these measures are strong, there are technical means for the providers to get access to customer data. Customers have to rely on contractual agreements, such as SCCs for data protection. This paper does not aim to address the legal risks or quantify the risks of data access by cloud providers that have the technical means to access customer data. These risks have been investigated elsewhere. For instance, the Dutch NCSC has published reports on the risks of data access by cloud providers under the US CLOUD Act[6, 7].

When encryption is used to protect data at rest, it is important to be in control of the encryption key lifecycle and key access policies to control and monitor data access. Both AWS and Microsoft Azure allow customers to maintain such control when using Customer-Managed Keys. Customers have to explicitly choose to use Customer-Managed Keys; by default the cloud providers use Platform-Managed Keys managed by them, with little visibility or control for the customer.

We recommend that customers use Customer-Managed Keys backed by a HSM, to protect their data. This recommendation aligns with the controls and requirements of the NBA maturity model for information security adopted by the Dutch education sector[8].

When encryption is used to protect data in transit, it is important to achieve end-to-end encryption. For data transferred between customers and cloud services, TLS endpoints are available. For data transferred between cloud services, data is encrypted across cloud provider's data centres. Within data centres, data is not always encrypted on the network layer. Based on our investigation we recommend that Azure Files with NFS not be used for sensitive data, and that Azure server-side encryption for managed disks not be used for disk volumes that contain sensitive data because the data is not encrypted in flight.

Our investigation does not cover encryption in use or confidential computing. These are technologies that promise to further enhance privacy of data that is processed. We aim to investigate encryption in use and confidential computing in a future project.

The ensuing sections describe our findings in more detail.

Key management services

The AWS and Microsoft Azure clouds both offer encryption key management services, called AWS Key Management Service (AWS KMS), and Azure Key Vault & Managed HSM, respectively.

The key management services and their integration with compute, storage, and database services use *envelope encryption* for data at rest, an approach to encryption with different types of keys. Data encryption keys (DEKs) are used to protect customer data, and key encryption keys (KEKs) are used to protect the data encryption keys. Data encryption keys need to be available to the compute,

[6] https://www.ncsc.nl/documenten/publicaties/2022/augustus/16/cloud-act-memo
[7] https://www.ncsc.nl/documenten/rapporten/2022/november/23/cloud-act-requests
[8] https://www.nba.nl/intern-en-overheidsaccountants/volwassenheidsmodel-informatiebeveiliging/

database and storage services to facilitate transparent encryption and decryption of the customer data. The key encryption keys are stored in tightly secured systems, designed to ensure that the key material never leaves the secure boundary.

AWS and Microsoft implemented a number of technical- and procedural processes to protect the DEKs and the KEKs. Information about the processes is available in audit reports available to their customers under NDA. Because of the non-public nature of the audit reports, we cannot describe audit details in this document. Our general finding is that AWS has defined more explicit and detailed security controls about cryptography in their ISO 27001, SOC 2 and C5 reports than Microsoft[9], thereby providing more insight into the organisational controls to safeguard encryption keys.

Key encryption keys on AWS come in three types: *AWS owned* keys, *AWS managed* keys and *Customer managed* keys (CMKs). The key life cycles of AWS owned and AWS managed keys are managed by AWS. AWS owned keys exist outside the customer's environment; AWS managed keys are created by AWS in the customer's cloud environment. AWS managed key logging is available in the customer's cloud environment. No logging is available to customers for AWS owned keys. In the discussion below we refer to AWS owned and AWS managed keys collectively as Platform-Managed Keys (PMKs).

On Azure there are two types of key encryption keys: Platform-Managed Keys - also known as Microsoft managed keys or service managed keys - (PMKs), and Customer-Managed Keys (CMKs). The key life cycles of PMKs are managed by Azure. PMKs exist outside the customer's environment, therefore, no logging is available in the customer's cloud environment.

When a CMK is used, whether on AWS or on Azure, the customer controls the key lifecycle, and can configure key access policies that determine who can use which CMKs for which purposes. This is a crucial aspect of cloud encryption. Overbroad access permissions for keys negate the protections offered by encrypting data: anybody who can use a CMK, can potentially read data protected with that key. In addition, access logs for key usage are available for CMKs (who used which key, at what time and for which kind of operation).

CMKs can be generated by AWS KMS and Azure Key Vault and Azure Managed HSM. Alternatively, key material can be generated by the customer outside the cloud environment and imported into AWS KMS or Azure Key Vault to be used as a CMK. There are minor differences in the options for rotating and deleting cloud-generated vs. imported keys.

On the technology side, we found that both providers offer options to have KEKs stored in certified Hardware Security Modules. Azure offers Azure Key Vault Premium, certified to FIPS 140-2 Level 2, and Azure Managed HSM, certified to FIPS 140-2 Level 3. AWS KMS CMKs are stored in HSMs, certified to FIPS 140-2 Level 3. AWS CloudHSM, available as an external key store for KMS, is also certified to FIPS 140-2 Level 3. AWS designs their own HSMs, and the FIPS 140-2 certification allows their customers to view detailed information about the system security

[9] Audit reports are available under NDA to AWS customers in the AWS Console, and to Microsoft customers in the Microsoft Service Trust portal.

policies[10]. Azure uses commercially available equipment. At present, Azure Key Vault uses nCipher nShield HSMs and Azure Managed HSM uses Marvell LiquidSecurity HSM adapters, but this may change over time.

Key storage systems certified to higher levels of the FIPS 140-2 standard offer stronger security guarantees. The appendix NIST FIPS 140 standard contains an overview of the differences between levels 1, 2, and 3. When working with sensitive data, we recommend that customers use HSM-backed keys. When working with highly sensitive data, we recommend that FIPS 140-2 Level 3 certified HSMs be used. On AWS this is the default for customer managed KMS keys; on Azure this means using Azure Managed HSM.

AWS CloudHSM and Azure Managed HSM, additionally, provide single-tenant security domains where the customer is in complete control of all keys, including root keys.

AWS CloudHSM keys can be used to encrypt data at rest for the services in scope via the *custom key stores* feature of AWS KMS. This feature also enables use of on-premises key stores. In a technical sense, on-premises key stores do not improve the security of key storage very much, while introducing availability risks. This is further discussed in the External Key store section. Azure does not offer options to use keys stored in on-premises key stores for Server-Side encryption.

Azure Managed HSM implements the Key Vault API to encrypt data at rest for the services in scope, using a single-tenant security domain in an HSM environment. Managed HSM is not supported for all services within the scope of our research. Most notably, Azure Managed HSM cannot be used to protect keys used with Azure Disk Encryption[11].

Encryption of virtual disks

When customers use virtual machines in the cloud, data is stored on virtual disk volumes. AWS and Azure support encrypting virtual disk volumes. Every volume is encrypted with one or more unique DEKs, depending on the volume size. A set of DEKs is protected with a KEK.

There are two types of virtual disks: virtual disks backed by storage in the hypervisor host, and virtual disks backed by a remote storage system attached via a network. When using network-attached disks, the lifecycle of the disk is separate from the lifecycle of the VM. Host-based disks have a lifecycle tied to that of the VM. It is recommended that host-based disks be used only for temporary data. Our discussion therefore focuses on network-attached virtual disks.

AWS's virtual disk service, Elastic Block Store (EBS), allows all volume types to be encrypted in the same way, whether the volume is HDD-backed or SSD-backed. On modern VM instance types, the encryption and decryption is handled by the AWS Nitro storage controller, a custom ASIC attached to the hypervisor host. Because crypto operations are done in hardware, IOPS performance is not affected by encryption. Encryption does add some latency to disk IO operations. Data is

[10] https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4523
[11] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption-overview

encrypted using AES256-XTS before it is transmitted to the EBS system over the network. A volume DEK is available to the Nitro controller while the volume is mounted. When using Nitro-based VM instances, AWS staff cannot access the encryption keys.

Microsoft's Azure Managed Disk service offers multiple encryption models. Not all models can be used for all volume types. The number of combinations of encryption models and key storage options is large, and the public documentation makes it hard to understand which combinations are supported.

Azure Service-side encryption encrypts and decrypts data at the storage service. Data transferred between a hypervisor host and the storage nodes is not encrypted. Therefore service-side encryption should be avoided for sensitive data. Host-based encryption encrypts and decrypts data on the hypervisor host. When host-based encryption is used, data is encrypted on the network between the hypervisor host and the storage nodes. This is also the case for Azure Disk Encryption (ADE). The difference is that with ADE the data is encrypted inside the VM instance, before it traverses the hypervisor stack. This further reduces the risk of data being accessible to the cloud provider. Microsoft uses AES256 algorithms for the encryption, but declined to share which AES mode is used. Crypto operations are performed in software, and affect IOPS performance. Host-based encryption and ADE are not supported for the fastest volume types (Ultra Disks and Premium SSD v2). This means there is a risk that data can be intercepted on the network when using the fastest volume types. There does not seem to be any ways to mitigate this risk.

Encryption of file shares

AWS and Azure offer file sharing services that support the NFS and SMB protocols. SMB file shares can be protected using CMKs. Data is not vulnerable to interception on the network between client and server, because encryption in transit is available. Access management at the user level is supported, with integration to Active Directory for user management.

Amazon Elastic File System is AWS's native NFS file sharing service. It supports encryption at rest with CMKs, encryption in transit, and access management at the user level. Azure Files, built on Azure Storage, supports NFS and SMB file sharing. For NFS, Azure Files does not support encryption in transit, nor user level access management. Customers who need NFS support for sensitive data in Azure are recommended to consider the Azure NetApp Files service or Azure Marketplace solutions that may offer better encryption and access management features. Our investigation did not cover the encryption or service details of Azure NetApp Files or Amazon FSx for NetApp ONTAP; we focused our investigation on the cloud native file sharing services.

Amazon and Microsoft do not share details about the protection of DEKs in the file sharing services. We have to assume there is a residual risk that they are technically able to decrypt data stored when required to do so as part of a law enforcement order.

Encryption of object storage

Amazon S3 and Azure Blob Storage are services that allow users to store large quantities of data in a flat namespace using a REST interface. Both services support server-side encryption as well as client-side encryption.

Server-side encryption supports the use of CMKs and HSM-backed keys. Every data object/blob is, by default, encrypted with one or more unique DEKs, depending on the object size. DEKs need to be available to S3/Blob Storage for transparent crypto operations. AWS and Azure do not share specific details about the protection of DEKs when in use. We have to assume there is a residual risk that they are technically able to decrypt data stored when required to do so as part of a law enforcement order.

Data objects can be Client-side encrypted before sending them to S3 or Azure Blob Storage, without ever transmitting the encryption key to the provider, offering a higher degree of protection. This approach, while more secure, is less practical than using server-side encryption and does not support all object/blob storage use cases, without the customer implementing user access and encryption key protection themselves.

Middle ground is found by using the client-side encryption libraries offered by AWS and Microsoft. These simplify the client-side encryption/decryption operations, and allow KMS and Key Vault to be used, respectively for storing KEKs. This approach builds on the strong security foundation of AWS and Microsoft Azure for managing KEKs, while not exposing DEKs to their object/blob storage services. Note that the DEKs are handled by the AWS KMS and Azure Key Vault services.

Encryption in managed databases

When using managed databases, the risk landscape is different because the cloud provider manages not only the virtual compute and storage, but also the database engine that runs atop. The cloud provider must, as part of the service, be able to access the virtual compute instances that run the database engine in order to manage the database engine and the databases. Management activities include installing DB engine updates, updating the underlying operating system, copying database log files, creating databases, creating backups, etc. This means that even if the database files and various logs are persisted on encrypted volumes, there are risks that the provider accesses the contents of customer databases, for instance when compelled by a law enforcement order.

Fortunately, additional encryption options are available to mitigate this risk. The first option is to use the Transparent Data Encryption (TDE) feature of the SQL Server and Oracle database engines. When TDE is enabled, pages are encrypted before being persisted to disk. On AWS this is supported for both engines; Azure does not offer a managed Oracle database service. Because of the nature of the services, there is still a risk of data access by the provider.

Microsoft SQL Server offers yet another encryption option, dubbed Always Encrypted. This feature allows the database user to specify columns that are to be encrypted within the database, rendering the data in those columns inaccessible to the cloud provider by encrypting the data client-side. Microsoft offers an integration between SQL Server Always Encrypted and Azure Key Vault. No such integration between SQL Server and AWS KMS is available; AWS users who want to use the Always Encrypted feature need to manage their own encryption keys. The Always Encrypted feature offers the best protection against data access by the cloud provider. The downside is that it is not transparent to the database user, because there are limitations to querying encrypted data in encrypted columns. Azure partially addresses these limitations using secure enclaves to execute sensitive operations. These secure enclaves are backed by software and hardware-based trusted execution environments such as Virtualization-based security (VBS) and Intel SGX.

Document outline

Abbreviations

AMI	Amazon Machine Image
AWS	Amazon Web Services
BYOK	see CSEK-BYOK
CAPEX	Capital Expense
CLI	Command Line Interface
CMK	Customer Managed Encryption Key
CRUD	Create, Read, Update, Delete
CSEK	Customer Supplied Encryption Key
CSEK-BYOK	Customer Supplied Encryption Key via the Bring Your Own Key principle.
CSP	Cloud Service Provider
DEK	Data Encryption Key
EBS	(Amazon) Elastic Block Store
EC2	Elastic Compute Cloud
FIPS	Federal Information Processing Standards (US)
HMAC	Hash-Based Message Authentication Code
HSM	Hardware Security Module
IaaS	Infrastructure as a Service
IaC	Infrastructure as Code
IAM	Identity and Access Management
KEK	Key Encryption Key
KMS	Key Management Service
MMK	Microsoft Managed Key (see also PMK)
NIST	National Institute of Standards and Technology (US)
NVMe	Non-Volatile Memory Express
OPEX	Operational Expense
PaaS	Platform as a Service
PMK	Platform Managed Key (in Azure also known as Microsoft Managed key)
R	R is a programming language for statistical computing
RDS	Relational Database Service
S3	Amazon Simple Storage Service
SaaS	Software as a Service
SPSS	SPSS Statistics is a statistical software suite
SSD	Solid State Drive
SSDLC	Secure Software Development Lifecycle
SSE	Server-Side Encryption
SSH	Secure SHell
SSL	Secure Socket Layer
TDE	Transparent Data Encryption
TLS	Transport Layer Security
VK	Volume Key (a DEK specifically for an EBS volume)
VM	Virtual Machine
XKS	External Key Store

Introduction

Encrypting electronic data before storing it on a medium like an SSD or harddisk can help guard against the data being viewed by unauthorised people or systems, provided that (1) these unauthorised people or systems do no have access to the encryption keys used and (2) the encryption method used is not vulnerable to attack.

This whitepaper investigates the encryption methods and key management services offered by Amazon Web Services and Microsoft Azure for encryption of data at rest and in transit. We investigate the risks against which the use of these services protects and how they can be applied by AWS and Azure users.

Before addressing the options in AWS and Azure in regards to encryption at rest and in transit, we introduce some foundational concepts, since not everybody is experienced in the term encryption and the topic of security extends more than encryption.

In this introduction chapter we will share our view on security and cloud security and the layers that together form a cloud offering. In addition we will introduce the concepts of shared responsibility as well as the role of monitoring, logging and event management to increase security.

At the end of this introduction chapter we will introduce encryption, Identity and access management and Key management.

At the end of this document you will find various appendixes where we address certain topics in a bit more detail. These topics in itself are not in focus of this whitepaper but are relevant to the nuances of improving the security of a solution or organisation.

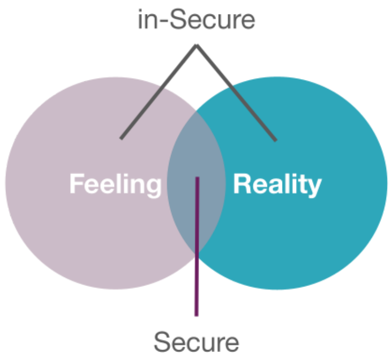
Security

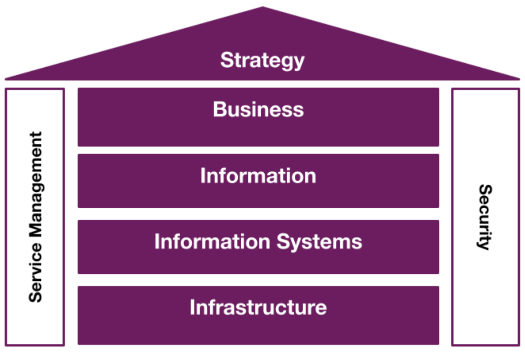
We define security as the overlap of your expectation and the reality. Your expectations are a feeling that can be made explicit in for example a design. Something is insecure when it is there and you did not expect it, and if you expect something and it is not there. See figure below.

To increase security the two circles need to be moved into each other. To do this you need to decrease the length of your feedback loop. See appendix on monitoring.

Having a shorter feedback loop will increase your resilience. Resilience is the degree in how quickly you can recover from something that happened and impacted you. This can be an unforeseen event or a foreseen event. To improve your resilience you need to steps of Identify, protect, detect, respond and recover. See append on resilience.

For the scope of your feedback loop it is important to know that security is not an IT only topic. Your business also has security as a quality attribute. See appendix on architecture scope and the appendix on reference cloud stack.





For software development and software operations there is a reference model for a more secure feedback loop. This is called a secure software development lifecycle (SSDLC). To achieve a more secure software (IT) solution you need to adhere to the SSLDC. **Encryption of data is a small part of the many steps in SSDLC.**

A good practice is to develop software according to the principle of Distributed, Immutable and Ephemeral (DIE[12]). Immutable means that you can not alter the content anymore. This helps in what you expect it to be, you expect it to be the same in reality since it can't be altered in reality. Ephemeral means that it only exists in reality for the duration of usage, which is as short as possible. This limits the time for information to wander around and get exposed to unforeseen events. Distributed means that there is no room for a single point of failure. Without a central point of access, a distributed system has a high availability to the end-user, since if one point fails there will be another point available to service the end-user. Encryption plays a role in the aspects of immutability and ephemerality.

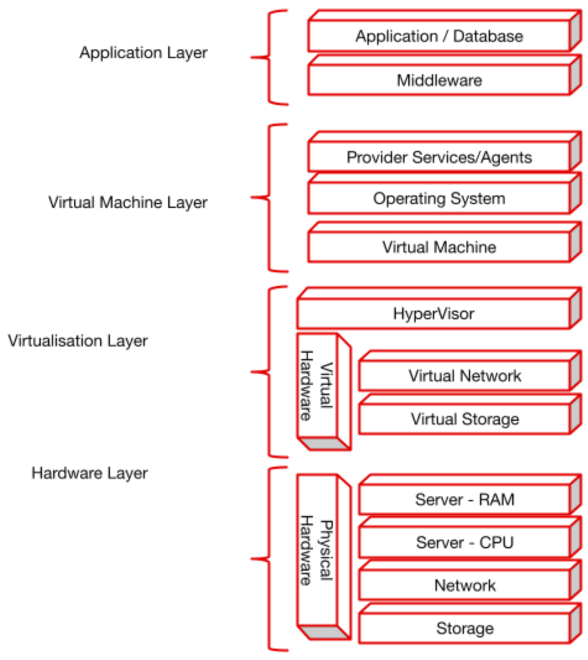
Cloud security

For an IT system to operate there are many layers of technology involved. Technology is manifested in hardware and in software.The cloud as a technology is an abstraction of how it is constructed to which function it delivers to you as a user. See Appendix Reference Cloud Stack.

Virtualisation Stack

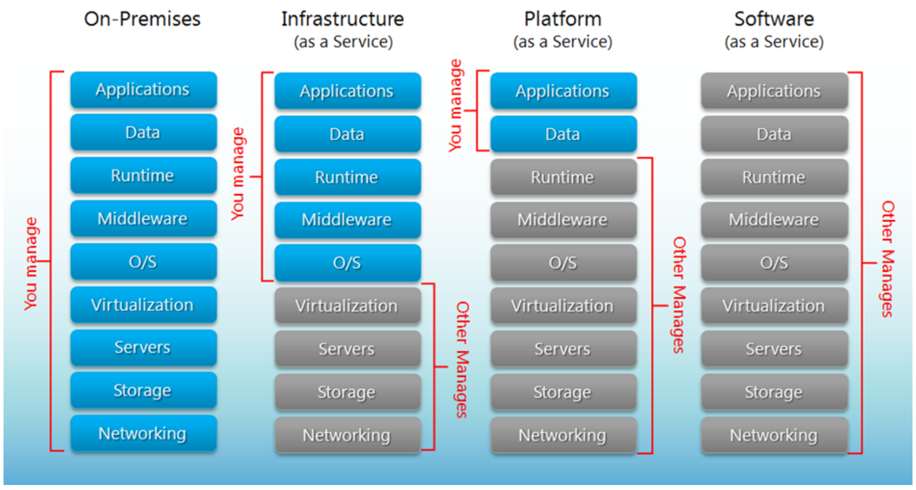
To identify the layers we use the following diagram, where at the top the service is running that you consume. Since all of these layers can be configured and are continuously improved, this stack is a collection of moving layers. This continuous movement and complexity of configuration has its impact on the security of the top service.

[12] https://www.slideshare.net/sounilyu/distributed-immutable-ephemeral-new-paradigms-for-the-next-era-of-security



Shared security

The offering of a cloud service provider is to take certain responsibilities away from you and make it their own. In the table below you see this defined for different service offerings. Do know that when you consume Infrastructure as a Service from a Cloud Service Provider, that the blue blocks are in reality a shared responsibility. Since you as a consumer are limited by the configuration that is offered by the provider, and that your configuration is executed by the services (layers) provided by the provider.



Trusting the service provider

From a security perspective this results in the statement that the services provided by the cloud service provider are for a certain degree always unknown to you and thus insecure. When you use a service provider you need to trust them to a certain extent. A way for the service providers to gain trust of the users is by (1) sharing clear and concise documentation, (2) certification of their documentation, their way of working (business processes), and the hardware, (3) audit reports that check the statements done in the certifications. It depends on your own risk profile and risk appetite which degree of trust fits your situation.

Documentation
Certification
Audit

Monitoring, Logging and event handling

To increase the overlapping intersection of the reality circle and the feeling circle, you need information. Information is data placed into context in for example a monitoring dashboard. Data is retrieved from systems via logging. Information can be translated into events that trigger action. So to increase our security (the overlap), we need to retrieve data from systems and transform this data into information. So that the user can become actionable and transfer the information into events.

Per Cloud Provider we have identified which logging features are available in regards to the creation, usage, update, and deletion of encryption keys. This excludes many other forms of data, information and events as for example the health of a Key Management Solution, data leaving the system (exfiltration) and the usage of specific cloud services.

Infrastructure-as-Code

The amount of configuration options of a cloud service are enormous. Research from the Cloud Security Alliance[13] states that 90% of the data breaches in the cloud are caused by misconfiguration. Mitigation of this risk is partly done by the application of Infrastructure-as-code (IaC). Here you capture the relevant configuration in code, so that you can use tools to check if you missed a configuration, if you are using conflicting configurations and it enables you to quickly check which configuration you are using. At present, it is considered a good practice to use automation when configuring cloud services. The use of Infrastructure-as-Code is part of this automation. Terraform is a way to describe and deliver infrastructure as code.

Encryption

---

[13] https://cloudsecurityalliance.org/artifacts/state-of-cloud-security-risk-compliance

"A court order is as ineffective at accessing encrypted data as a nuclear weapon. Only the keyholder can access that data, and that power resides exclusively with them."

- Jacob Riggs, security researcher who hacked the Dutch tax authority[14]

Encryption is the art of obfuscating a data, for example a text, in such a way nobody can read it unless they know how to de-obfuscate it.

Encryption can be considered as a translation. The oldest example of this is to add 1 character shift to every letter in a word, so APE becomes BQF. BQF, in this example, has no meaning and is therefore worthless for whoever reads it without the key of +1, while APE has meaning and therefore has value. See also appendix Encryption key life cycle and appendix Encryption.

In encryption the way to obfuscate your data is by using encryption-keys, also known as cryptographic keys. These keys are usually a string of characters, for example letters and numbers. Who has access to these keys can read your data and when changed also re-obfuscate the data. This results in the keys being an important part of making your data immutable. When you throw away the key directly after encryption, then your data is also not accessible anymore, and so contributing to the ephemeral quality of your data.

Many encryption systems are available for encrypting electronic data. AWS and Microsoft Azure use encryption services based on the same encryption systems for encrypting data at rest. Their encryption systems use symmetric encryption with a hierarchy of encryption keys.

Symmetric encryption is a form of encryption where the encryption and decryption operations use the same encryption key. Algorithms for symmetric encryption/ decryption are more efficient than asymmetric algorithms (algorithms where a distinct public and private key are used to encrypt and decrypt data). When data at rest is encrypted, the system that writes the data is generally also that system that reads it and no keys need to be distributed across untrusted connections.

Encryption and the confidentiality of the Encryption Key

"Cryptography is a tool for turning a whole swath of problems into key management problems."

- Lea Kissner, former CISO at Twitter

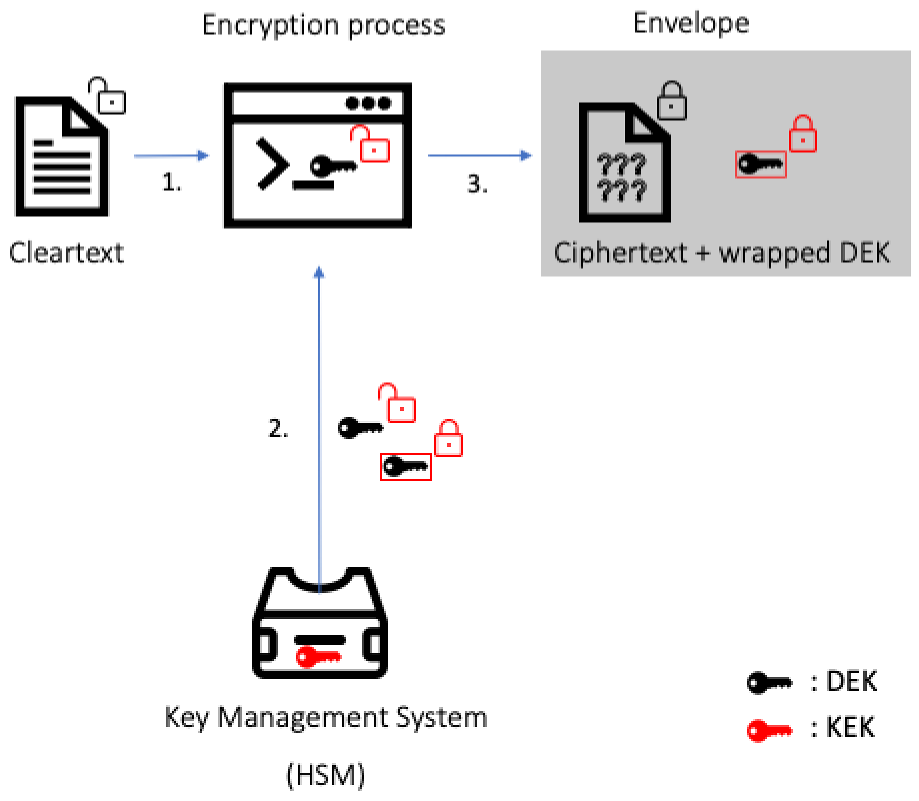
---

[14] https://jacobriggs.io/blog/posts/i-hacked-the-dutch-tax-administration-and-received-a-trophy-32.html

It is evident that whoever has access to your decryption keys has access to read your data and who has access to your encryption keys has access to write to your data. If a decryption key is compromised, all data encrypted using that key is at risk, as the attacker who obtained the decryption key can decrypt all encrypted data accessible to them using the key.

Where your data is encrypted or decrypted, that is where your key is required. The key needs to be sent to the service component that performs the encryption/decryption operations on the data and it may need to be transmitted across network connections and through software components in a virtualisation stack to arrive there. This implies that many layers are involved in the confidentiality of your key. So in order to reduce the impact of an encryption key being compromised, it is best to encrypt a limited amount of data using the same encryption key. This is where key hierarchies and envelope encryption come in.

The encryption systems used by AWS and Azure employ a key hierarchy that has Data Encryption Keys (DEKs) and Key Encryption Keys (KEKs). A DEK is used to encrypt a limited amount of data e.g. a single disk volume or a single file using an encryption algorithm. The most common algorithm for encrypting large amounts of data before storing them is the AES algorithm in XTS-[15] or GCM[16] mode. The diagram below illustrates how envelope encryption works.



[15] https://csrc.nist.gov/publications/detail/sp/800-38e/final
[16] https://csrc.nist.gov/publications/detail/sp/800-38d/final

The cleartext data and the cleartext DEK are used to generate the ciphertext. The encrypted data and the wrapped DEK are stored together. The wrapped DEK has been created within the security boundary of the key management system, using the KEK. The combination of the encrypted data and the encrypted DEK are called the envelope encrypted message.

This envelope encryption scheme requires trust in the cloud provider creating the DEK in a secure way, using it only to decrypt data when required by the customer, and not leaving traces of this key. This also demands trust on the usage of the KEK, not leaving any traces of this one-key-to-rule them all.

AWS and Microsoft have extensive security controls in place to safeguard encryption keys. These controls are designed to restrict access to key material as much as possible and to be able to trace key usage by extensive logging. Access to key material is restricted via process controls (e.g., least privilege, time-restricted access) and via technical measures in the system architecture.

These security controls are supplemented on the system level with key storage systems that operate as a vault for the KEKs. These key storage systems are tightly secured systems designed to ensure that the KEKs never leave the key storage system-boundary. Hardware-based implementations of these systems are called Hardware Security Modules (HSM). The cryptographic modules of key storage systems can be certified according to the FIPS 140-2 standard[17]. This certification states to which degree - denoted as level one to four - they provide physical evidence of tampering and are built according to specifications. The FIPS 140-2 certification is limited to the key storage devices or software, it does not cover the connected systems and services that are involved to store your data, and provide you access to your data. Also see the appendix NIST FIPS 140 Standard.

The AWS KMS and Azure Key Vault APIs allow plaintext DEKs to be returned in response to requests from users. Therefore the connection between the user and the key management system needs to be secure. The user has to be authenticated, and the connection has to be encrypted. User access is managed via AWS IAM or Azure Active Directory, respectively. In addition, key policies can be configured to further pare down access to encryption keys.

Network connections to KMS and Key Vault endpoints are encrypted using TLS. AWS and Azure require at least TLS 1.2, and suggest customers to use TLS 1.3. AWS also supports hybrid post-quantum TLS and offers FIPS 140-2 validated API endpoints.

[17] https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.140-2.pdf

AWS

Key Management in AWS

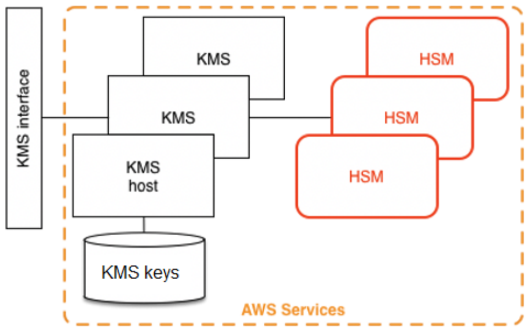
AWS Key Management Service (AWS KMS) is a managed service provided by Amazon Web Services (AWS) for creating and managing encryption keys[18]. The service allows users to create, rotate, and delete encryption keys, as well as control access to the keys. See appendix Encryption Key Lifecycle.

The KMS service can be used to encrypt and decrypt data in AWS services, as well as data stored outside of AWS. KMS can be used as a centralised way of managing your encryption keys.

There are a number of ways to interact with the KMS service:

1. AWS management console[19],
2. Command Line Interface (CLI)[20],
3. RESTful API operations[21].

The AWS KMS software utilises multiple hardware security modules (HSMs) to store your keys. The HSM's are FIPS 140-2 Level 3 certified[22] to ensure physical security, but are considered shared resources. See figure below.



https://docs.aws.amazon.com/kms/latest/cryptographic-details/intro.html

KMS keys

AWS KMS keys[23], also known as KMS keys are at the centre of the KMS service. A KMS key can be used to encrypt and decrypt data. It is also possible to generate data keys which can also be used outside of AWS[24]. KMS keys can either by symmetric encryption keys[25], asymmetric encryption and signing keys[26], or HMAC[27] keys. KMS keys are stored on persistent media in the HSMs. Outside the secure boundary of

[18] https://docs.aws.amazon.com/kms/latest/developerguide/overview.html
[19] https://docs.aws.amazon.com/awsconsolehelpdocs/latest/gsg/learn-whats-new.html
[20] https://aws.amazon.com/cli/
[21] https://docs.aws.amazon.com/general/latest/gr/aws-apis.html
[22] https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4523
[23] https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#kms_keys
[24] https://docs.aws.amazon.com/kms/latest/APIReference/API_GenerateDataKey.html
[25] https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#symmetric-cmks
[26] https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#asymmetric-keys-concept
[27] https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#hmac-key-concept

the HSM, cleartext keys are only stored in memory and only for the time required to process the cryptographic request.

A KMS key is a logical representation of a cryptographic key. A KMS key contains metadata about the key (Key ID, key spec, creation date, key state, …), and most importantly a reference to the actual key material that is used when a cryptographic operation is performed with the KMS key.

AWS Managed Keys

AWS Managed keys[28] are KMS keys that are created in your account, managed and used on your behalf by an AWS service which is integrated with AWS KMS[29]. As a user, you only have the permissions to view the AWS managed keys, view their key policies, and audit their use in AWS CloudTrail logs.

However, you cannot change any properties of AWS managed keys. AWS is responsible for key rotation, change to key policies, scheduling keys for deletion, etc.. Furthermore, the AWS managed keys cannot be directly used in cryptographic operations. Using these keys is easy and straightforward, but you do hand in a level of control and therefore increase the trust in AWS.

The AWS Managed keys are only used as Key Encryption Keys (KEKs) to wrap Data Encryption Keys (DEKs) used by AWS services to encrypt data. See also appendix Encryption intro.

Creating AWS Managed Keys is free of charge, there is no monthly fee. There is however a pay per use fee. This fee applies to each API request to the KMS key, and is priced per 10,000 requests[30].

AWS Owned Keys

An AWS owned key[31] is a KMS key of a specific AWS service. The AWS service owns and manages the key. It is possible that an AWS service is using the same key in the data encryption while servicing multiple AWS accounts.

Since these keys are not owned by your AWS account, but by AWS, you don't have any control over them and cannot monitor their usage. From a security perspective the lack of insight and lack of control makes it advisable to use customer managed keys wherever you can.

The pricing of AWS Owned keys is the only reason to use these keys, since AWS owned keys are free of charge. There is no monthly fee and also no usage fee. Additionally AWS owned keys do not count against the AWS quotas in your account.

Customer Managed Keys

In KMS, it is also possible to create your own encryption keys, these keys are called customer managed keys[32]. As the name suggests, you are responsible for creating and managing the encryption keys, and you have full control over these keys. This

[28] https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#aws-managed-cmk
[29] https://aws.amazon.com/kms/features/#AWS_Service_Integration
[30] https://aws.amazon.com/kms/pricing/
[31] https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#aws-owned-cmk
[32] https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#key-mgmt

includes establishing and maintaining key policies, Identity and Access Management (IAM) policies, key grants, enabling and disabling the keys, rotating the cryptographic material, scheduling the keys for deletion, and other management tasks. There is still a level of shared responsibility[33]. AWS is still responsible for the underlying hardware and maintenance of the HSMs. But since you have more control over the lifecycle of the key as well as the access control of the key, a lower level of trust in AWS is required.

There are two ways in which you can create your own keys, by either letting AWS generate the key material for you, or importing your own key material in the form of *bring your own key (BYOK)[34].*

For creating Customer Managed keys there is a fee of $1 per key per month. This applies to both AWS generated keys as well as imported key material (*BYOK).* Additionally, the usage of the key is also billed (see AWS Managed Keys pricing).

AWS Generated Keys

This is the easiest way to create a customer managed key in AWS KMS. You can create KMS keys through either the AWS management console or by using the AWS KMS API[35]. The key material will be generated in AWS KMS hardware security modules (HSMs), certified to FIPS 140-2 Level 3[36]. The key material for KMS keys never leaves AWS KMS unencrypted, and it is not possible to extract, export, view or manage this key material. It is also not possible to directly delete the key material, you must delete the entire KMS key.

Bring Your Own Key (BYOK)

In AWS KMS It is also possible to import your own key material into the KMS service. This concept is called *Bring your own key (BYOK)[37].* First, you need to generate an encryption key yourself, this can be done in a HSM you control, in another cloud, or any other way you can generate an encryption key. Then, in AWS you create an empty KMS key with no key material. Finally, you import your key material into the KMS key.

The BYOK feature is useful if there are certain compliance or internal policies that request demonstration of control over the encryption key generation process, such as provable key entropy. There are however a number of things to keep in mind when importing your own key material:

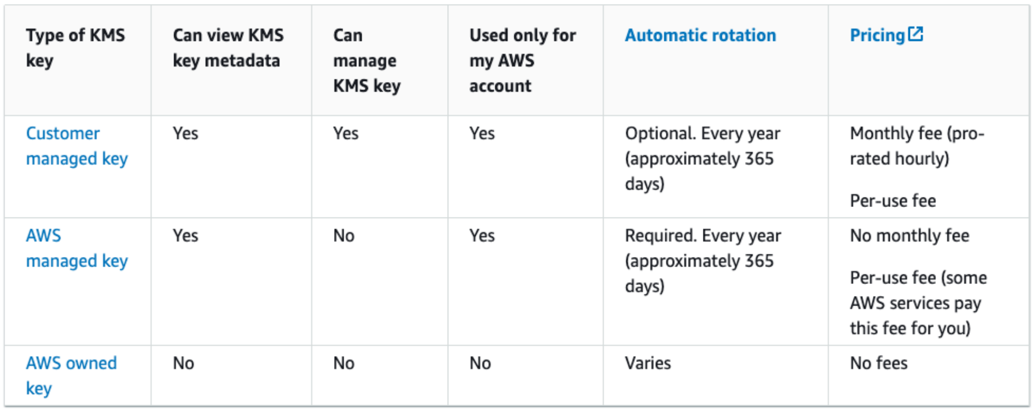
- Imported key material is only supported for symmetric encryption KMS keys.
- It is not possible to change the key material once it is associated with a KMS key. In order to use the new key material, you need to create a new KMS key. Therefore it is not possible to automatically rotate the key material as we discuss in the key rotation section.

[33] https://aws.amazon.com/compliance/shared-responsibility-model/
[34] https://docs.aws.amazon.com/kms/latest/developerguide/importing-keys.html
[35] https://docs.aws.amazon.com/kms/latest/developerguide/create-keys.html
[36] https://csrc.nist.gov/projects/cryptographic-module-validation-program/certificate/4523
[37] https://aws.amazon.com/blogs/security/demystifying-kms-keys-operations-bring-your-own-key-byok-custom-key-store-and-ciphertext-portability/

For the full list of considerations see the documentation page about imported key material[16].

The customer managed AWS keys, be it AWS generated or BYOK, can be used in two ways: direct cryptographic operations via the AWS SDK or through AWS services that are integrated with AWS KMS[38].

In the table below, the three types of KMS keys are summarised.



Type of KMS key	Can view KMS key metadata	Can manage KMS key	Used only for my AWS account	Automatic rotation	Pricing
Customer managed key	Yes	Yes	Yes	Optional. Every year (approximately 365 days)	Monthly fee (pro-rated hourly) Per-use fee
AWS managed key	Yes	No	Yes	Required. Every year (approximately 365 days)	No monthly fee Per-use fee (some AWS services pay this fee for you)
AWS owned key	No	No	No	Varies	No fees

https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#key-mgmt

AWS CloudHSM

AWS CloudHSM[39] provides customers a dedicated, single-tenant HSM domain, a partition of a hardware security module (HSM). The underlying hardware may be shared by multiple AWS customers, but the HSM domain is only accessible to you. CloudHSM requires you to configure and maintain the cloud-based HSM. This gives you the ability to create users and manage their permissions. This leads to increased control over the HSM, since for the regular KMS service this is all managed by AWS. On the flip side however, it leads to an increased burden on the customer. Additionally, CloudHSM comes with an hourly fee per HSM, if you use the CloudHSM generated keys in KMS, you additionally pay the monthly fee for the keys as well as the pay per use fee.

It is not possible to use AWS CloudHSM directly to encrypt the data stored by AWS services, including Amazon EBS, Amazon S3, Amazon EFS, Amazon FSx for Windows File Server, and Amazon RDS. AWS services use the AWS KMS APIs for cryptographic operations on DEKs. AWS CloudHSM does not support those APIs; instead you can use PKCS #11, JCE provider, CNG provider, KSP provider to manage keys in CloudHSM.

AWS CloudHSM is relevant to encrypting your data at rest in the AWS cloud, because it can be used as a so-called KMS Custom Key Store.

[38] https://docs.aws.amazon.com/kms/latest/developerguide/service-integration.html
[39] https://docs.aws.amazon.com/cloudhsm/latest/userguide/introduction.html

AWS KMS Custom Key Store

A *key store* is a secure location for storing cryptographic keys. The default key store in AWS KMS also supports methods for generating and managing the keys that it stores. The default key store consists of FIPS 140-3 certified HSMs. Customers who require more control over the HSMs than offered by AWS KMS's default key stores with Customer Managed Keys, can create custom key stores[40].

AWS KMS supports two types of custom key stores.

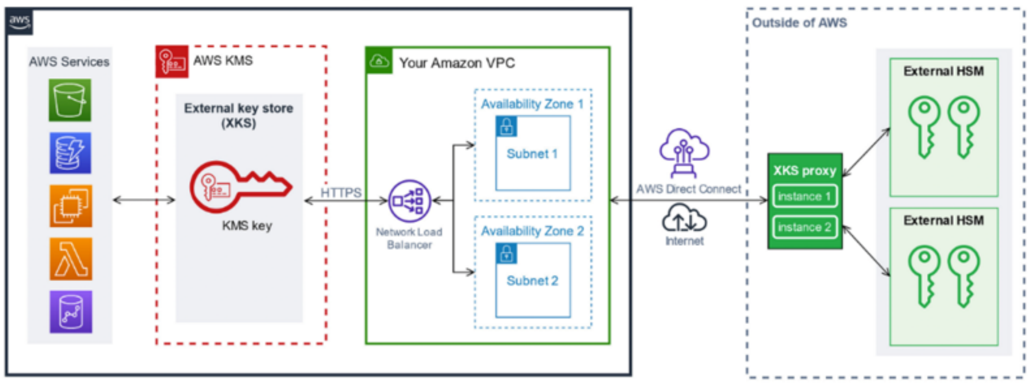
1. An AWS CloudHSM key store is an AWS KMS custom key store backed by an AWS CloudHSM cluster. When you create a KMS key in your AWS CloudHSM key store, AWS KMS generates a 256-bit, persistent, non-exportable Advanced Encryption Standard (AES) symmetric key in the associated AWS CloudHSM cluster. This key material never leaves your AWS CloudHSM clusters unencrypted. When you use a KMS key in a AWS CloudHSM key store, the cryptographic operations are performed in the HSMs in the CloudHSM cluster.
2. An External key store is an AWS KMS custom key store backed by an external key manager outside of AWS that you own and control. When you use a KMS key in your external key store, all encryption and decryption operations are performed by your external key manager using your cryptographic keys. External key stores are designed to support a variety of external key managers from different vendors.

External Key Store

The final category of AWS KMS keys are external keys, these are keys that are generated and managed outside of AWS (e.g. on-premises HSM) and then used within your AWS infrastructure. In AWS this is possible through the External Key Store (XKS)[41] feature of KMS.

When using an XKS, AWS KMS forwards API calls to your HSM. The key material never leaves the key store that you control. The connection between the AWS cloud and your own HSM is established through the XKS proxy[42]. Once the proxy and the key store are configured correctly, you can create a corresponding external key store resource in KMS. You have to create the encryption key in your own key store, then you can map that key to the external key store resource in KMS. Now, every time that that KMS key is used within AWS, the request will be forwarded through the proxy to your key store.

[40] https://docs.aws.amazon.com/kms/latest/developerguide/custom-key-store-overview.html
[41] https://docs.aws.amazon.com/kms/latest/developerguide/create-xks-keystore.html
[42] https://docs.aws.amazon.com/kms/latest/developerguide/keystore-external.html#concept-xks-proxy



https://aws.amazon.com/blogs/aws/announcing-aws-kms-external-key-store-xks/

Using XKS gives you complete control over the encryption keys. Since they never leave your key store, you can configure strict access policies and perform additional monitoring on your key store to make sure that the keys are only used as intended.

XKS can be useful in certain use cases, especially if there are strict regulatory or compliance requirements. However, configuring and maintaining your own key store requires very specific knowledge, time, and can become quite expensive. You are responsible for purchasing the key management solution as well as keeping it running. Additionally, you still pay a monthly fee for the KMS key and the KMS usage fees. Furthermore, you become responsible for the availability of the key store. If the key store goes down, you won't be able to decrypt anything in AWS. Depending on the geographic distance between the HSM and the nearest AWS Point of Presence (POP), latency can become a factor as well. All in all it requires a significant investment from your side and complicates the use of encryption keys a lot. So unless you have a very good reason to use this, don't.

KMS Logging and Monitoring

Monitoring AWS KMS is essential to gain insights into the availability, state, and usage of your KMS keys. Every call made to an AWS KMS API is captured as an event in a AWS CloudTrail logs[43]. This includes all create, read, update and delete (CRUD) operations related to KMS keys, as well as the actual key usage[44]. If KMS keys are shared across multiple accounts, then cross-account API calls are recorded in CloudTrail logs of both accounts.

CloudTrail logs can also be stored and monitored in Amazon CloudWatch Logs[45]. The CloudWatch service provides a centralised place where all logs can be stored and monitored. Additionally, Amazon EventBridge[46] can be used for alerting when important events occur in the life cycle of KMS keys[47]. The following events will generate an EvenBridge alert:

- The key material of a KMS key was automatically rotated.

[43] https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-user-guide.html
[44] https://docs.aws.amazon.com/kms/latest/developerguide/logging-using-cloudtrail.html
[45] https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html
[46] https://docs.aws.amazon.com/eventbridge/latest/userguide/eb-what-is.html
[47] https://docs.aws.amazon.com/kms/latest/developerguide/kms-events.html

- The imported key material of a KMS key has expired.
- A KMS key that has been scheduled for deletion has been deleted.

KMS Best Practices

A policy is an object that, when associated with an identity or resource, defines their permissions. There are two types of policies in AWS, identity-based or resource-based, for KMS there are key policies and IAM policies respectively.

Key Policies

A key policy is a resource policy for AWS KMS keys[48]. Key policies are the primary way to control access to KMS keys. Each KMS key must have exactly one key policy. In a policy you define statements to determine who has permission to use the KMS key and how it can be used. By default, no AWS principal, including the account root user or the key creator, has any permissions to a KMS key unless they are explicitly allowed, and never denied, in a key policy, IAM policy, or key grant.

When creating new KMS keys you can specify the key policy. If no key policy is provided, KMS creates a default one for you. The key policy will differ depending on how the KMS key has been created. When a KMS key is created programmatically (through the AWS KMS API[49]) a default key policy is created that give the AWS account that owns the KMS key permission to use IAM policies to allow access to all KMS operations on the KMS key[50]. When the KMS key is created through the AWS Management Console, the key policy begins with the statement that allows access to the AWS account and enables IAM policies. Then the console also adds a key administrators statement, a key users statement, and a statement that allows principals to use the KMS key with other AWS services.

When AWS owned keys or AWS managed keys are used, you don't have any control over the key policies. For customer managed keys however, you create and maintain the key policy as needed. When configuring the key policies, it is advised to follow the least privileged principle, meaning only give the permissions that are required. To achieve this with KMS key policies, you can include the kms:EncryptionContext:context-key[51] or kms:EncryptionContextKeys[52] condition keys in the policy that allows principals to call cryptographic KMS key operations.

Identity and Access Management (IAM)

IAM policies are optional for KMS keys, but provide an additional layer of access control for KMS keys[53]. Unlike key policies, IAM policies can control access to multiple KMS keys and provide permissions for the operations of several AWS services. Additionally, IAM policies are useful for controlling access to certain operations that

[48] https://docs.aws.amazon.com/kms/latest/developerguide/key-policies.html
[49] https://docs.aws.amazon.com/kms/latest/APIReference/Welcome.html
[50] https://docs.aws.amazon.com/kms/latest/developerguide/key-policy-default.html#key-policy-default-allow-root-enable-iam
[51] https://docs.aws.amazon.com/kms/latest/developerguide/conditions-kms.html#conditions-kms-encryption-context
[52] https://docs.aws.amazon.com/kms/latest/developerguide/conditions-kms.html#conditions-kms-encryption-context-keys
[53] https://docs.aws.amazon.com/kms/latest/developerguide/iam-policies.html

cannot be controlled by a key policy. An example would be the CreateKey[54] operation, since it cannot be controlled by a key policy because it does not refer to a single KMS Key.

For IAM policies it is also advised to adhere to the principle of least privilege, only assign the required permissions. At the foundation, it is advised to provide permissions in key policies as much as possible. This is because key policies only apply to one specific key, while an IAM policy can be applied to multiple keys. Furthermore, it is advised to restrict the ability to create keys to only those that actually require that permission. As a best practice, also specify the key ARN in the resource block of the IAM policy to which the permissions apply. Finally, avoid wildcards ("*") in IAM policies as much as possible, since it can impact more keys than intended.

Key Rotation

Cryptographic best practices discourage extensive reuse of encryption keys and therefore advise to rotate encryption keys on a regular basis. Rotating encryption keys in AWS can be done in two primary ways and can vary between the types of key[55]. First, you can manually rotate the KMS keys by creating new (customer-managed) KMS keys and then change the applications or aliases to use the new keys[56]. AWS also offers the functionality to automatically rotate key material in a transparent and easy manner. When you enable automatic key rotation, AWS KMS generates new cryptographic material for the KMS key. New encryption operations will from now on be performed with the new key material. However, data that was previously encrypted with the old key material, will not be re-encrypted. Therefore, AWS KMS will maintain a versioned history of the KMS key, which can be used to decrypt data that was encrypted with previous versions of the KMS key. Therefore, you can continue using the original KMS key as before, while AWS handles all the versions of the key material in the background. Re-encrypting previously encrypted data with the new KMS key cannot be done automatically, an example on how to do this is provided in the rotate data key for EBS section. Automatic rotation is currently only supported for symmetric KMS keys with AWS generated key material. For asymmetric KMS keys, HMAC KMS keys, KMS keys in custom keys stores (XKS), or KMS keys with imported key material (BYOK), only manual rotation is supported.

Automatic Rotation

AWS managed KMS keys are automatically rotated every year (approximately 365 days), this cannot be disabled or changed. When using AWS owned keys, it is also not possible to change the rotation period. The key rotation strategy however is determined by the AWS service that creates and manages the key and therefore can vary between services. For customer managed keys, automatic rotation is disabled

[54] https://docs.aws.amazon.com/kms/latest/APIReference/API_CreateKey.html
[55] https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html
[56] https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html#rotate-keys-manually

by default, but can be turned on for keys that have AWS generated key material. The rotation period then also becomes 1 year. If for certain reasons you need to rotate keys more frequently, you can resort to manual rotation as described above.

Cloud Service Encryption

In this section we dive deeper into how encryption works for several services in AWS.

VMs and Block Storage (EBS)

To deploy virtual machines, AWS offers the Amazon Elastic Compute Cloud (Amazon EC2)[57] service. EC2 provides horizontally and vertically scalable computing capacity and you can deploy as many or as few virtual servers as needed. The virtual machines deployed on EC2 are known as *instances.* EC2 supports a variety of operating system environments, including numerous Linux distributions and Windows Server. These operating systems are provided by AWS in the form of Amazon Machine Images (AMIs)[58], and are supported and maintained by AWS. AMIs are required to launch an EC2 instance, it is also possible to launch multiple EC2 instances from a single AMI. Customers can also create their own AMIs.

AWS offers a wide range of storage services[59], for EC2 instances both temporary (ephemeral) and persistent volumes are available. In this section, we primarily focus on the Amazon Elastic Block Store (EBS) service, and the EC2 instance store. When creating a new EC2 instance, you need to create a *root device volume.* This volume will contain the image used to boot the instance. Previously, these root device volumes were backed by instance stores, however, more recently EBS has become the advised service to use for root device volumes since it launches faster and provides persistent storage[60]. Next to the root device volume, you can attach multiple EBS volumes to the EC2 instance for application and data storage.

EBS provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices, and are attached through the network. These volumes can be mounted as devices on EC2 instances. EBS volumes that are attached to an instance are exposed as storage volumes that persist independently from the life of the instance. Therefore EBS volumes are the recommended way of storing data for EC2 instances. It is possible to create a file system on top of the EBS volumes, or use them in any other way you would use a block device (such as a hard drive). EBS volumes can be detached from one instance and attached to another. Some volume types allow it to be attached to multiple instances at the same time[61].

EC2 instance store provides the instances with temporary block level storage. The storage is located on disks that are physically attached to the host computer. Instance stores are well suited for temporary storage of information that changes

[57] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html
[58] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html
[59] https://aws.amazon.com/products/storage/
[60] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/RootDeviceStorage.html
[61] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volumes-multi.html

frequently, such as buffers, caches, and other temporary files. Instance stores can only be specified at instance launch, and cannot be detached from one instance and attached to another. More detailed information about the EC2 instance store volume lifecycle is available in the EC2 documentation[62].

In order to protect your data that is stored in the block device volumes (e.g., EC2 instance store, EBS) and processed by the EC2 instances it is advised to apply encryption. We distinguish between three ways of encrypting your data, encryption at rest, encryption in transit, and encryption in use. The first is applied to EBS volumes as well as instance stores, the second is relevant when data is moving between the storage service to the virtual machine, and the last one is relevant for EC2 instances that process the data.

Encryption at rest

Encryption at rest is the key protection against a data breach. “At rest” means that the data is stored somewhere and is not moving through the network. Encrypting the data makes sure that even if the storage device is breached the data is not readable to the attacker. In other words, the plaintext data is converted to a non-readable encoded format, called ciphertext before it is stored on the persistent storage medium. This is achieved through the use of the AES-256-XTS encryption system[63], the most common industry standard for encryption for encrypting data at rest[64].

Encryption at rest for EBS

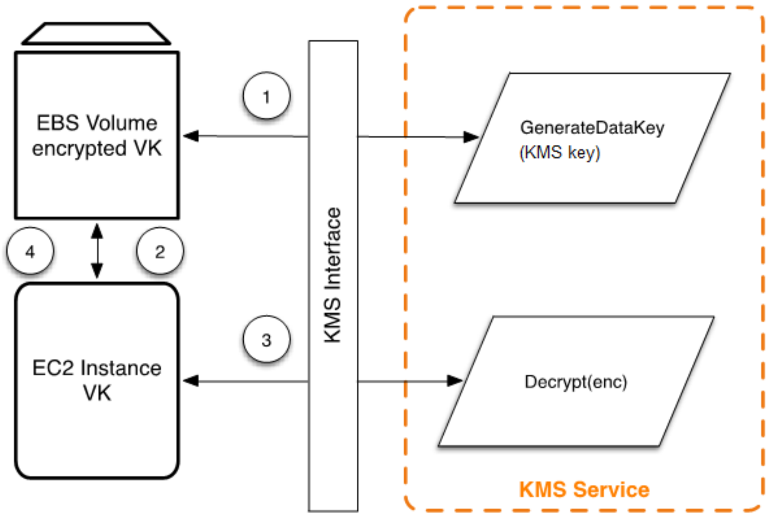
EBS volumes can be encrypted at rest. When creating an EBS volume, you can specify a KMS encryption key that should be used for the encryption. Then, AWS KMS will generate a new AES-256 data encryption key (DEK) which will be used to encrypt the data on the volume. The data key is then encrypted with the specified KMS key which is used as key encryption key (KEK) and stored in the EBS service. In the AWS documentation the data encryption key is sometimes also addressed as the *Volume Key (VK).* This technique is called *envelope encryption.* For modern EC2 instance families, i.e., instance families that use the AWS Nitro system, the encryption of EBS volumes occurs on the Nitro controller connected to the hypervisor hosts that run the EC2 instances to which the volumes are attached. The Nitro controller keeps the DEK in memory while the volume is attached[65]. Older EC2 instance families that predate the Nitro system may store data encryption keys in the hypervisor host's memory. It is recommended to use these instance families only for compatibility reasons. DEKs stored in Nitro controllers cannot be accessed by AWS[66].

The KEK never leaves the HSMs used in the KMS service, whether the EBS volume is attached to a modern (Nitro-based) or older host.

[62] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html#instance-store-lifetime
[63] https://docs.aws.amazon.com/kms/latest/cryptographic-details/ebs-volume-encryption.html
[64] http://csrc.nist.gov/publications/nistpubs/800-38E/nist-sp-800-38E.pdf
[65] https://docs.aws.amazon.com/whitepapers/latest/security-design-of-aws-nitro-system/putting-the-pieces-together-ebs-volume-attachment.html
[66] See Article 96 in the AWS Service terms at https://aws.amazon.com/service-terms/ or https://docs.aws.amazon.com/whitepapers/latest/security-design-of-aws-nitro-system/no-aws-operator-access.html.

EBS offers the possibility to automatically encrypt all EBS volumes on the account level. This can prevent the creation of unencrypted EBS volumes[67].

EBS encryption is available for all EBS volume types (HDD- and SSD-backed), current and previous generation EC2 instance types and does not affect IOPS performance. Storage latency is minimally higher than for non-encrypted volumes[68].



https://docs.aws.amazon.com/kms/latest/cryptographic-details/ebs-volume-encryption.html

Rotate data key

It can become necessary to rotate the data key which is used to encrypt the EBS volume, for example when a data key is compromised. Rotating the data key used to encrypt the EBS volume is however not as straightforward as rotating a KMS key, and requires a number of actions[69]:

1. Create a snapshot from the encrypted EBS volume of which you want to rotate the data key.
2. Create a new encrypted volume from the encrypted snapshot, it is very important to specify another KMS key in this step. (If you use the same KMS key as before, the data key will not be rotated).
3. Now the EBS volume is encrypted with a new data key and you can attach it to the EC2 instances again.

Encryption at rest for Instance Stores

The way instance stores are encrypted at rest depends on which instance store volume is chosen[70]. Non-volatile memory express (NVMe) solid state drives (SSD) are by default encrypted using an XTS-AES-256 cipher, implemented on a hardware

[67] https://aws.amazon.com/premiumsupport/knowledge-center/ebs-automatic-encryption/
[68] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html#ebs-encryption-requirements
[69] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html
[70] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/data-protection.html

module on the instance. The keys that are used to encrypt the data that is written to the locally-attached NVMe storage devices are per-customer, and per volume. The keys are generated by, and only live on, the hardware module on the server. The customer has no control over these encryption keys and cannot specify their own keys to be used. The encryption keys are destroyed when the instance is stopped or terminated and can never be recovered.

For non-NVMe instance store volumes, the process is more involved. Since AWS is actively advising not to use Instance sores for EC2 we don't go into detail on how to encrypt non-NVMe instance store volumes, but the solution relies on the Linux native dm-crypt[71] infrastructure. More information can be found in an AWS blog[72].

Encryption in transit

Encryption of the data that is written to the volume storage is performed on the servers that host the EC2 instance, on the dedicated AWS Nitro controller[73] to be more specific. Therefore, all I/O operations between the EC2 instance and the EBS volume are encrypted, so all data in transit is also encrypted.

Encryption in use

Our investigation does not cover encryption of data in use, i.e. data stored in RAM. A brief introduction to the topic of confidential computing can be found in Appendix - Confidential Compute.

Object Storage (S3)

Amazon Simple Storage Service (Amazon S3)[74] is an object storage service with a focus on scalability, data availability, security, and performance. S3 serves a wide variety of use cases ranging from data lakes, websites, (mobile) applications, backup and restore, archiving, and many more. S3 provides its users with management features to organise and secure the stored data , to meet business, organisational, and compliance requirements. Data objects are stored in so-called *S3 Buckets*. An *object* is a file and any metadata that describes the file, the *bucket* is a container for objects.

S3 offers a range of storage classes depending on the use case. Mission-critical production data can be stored in S3 Standard, while cost saving are possible for data that is accessed less frequently[75].

To store data in S3, you first need to create a bucket, configure the bucket name and AWS regions. Note that bucket names need to be globally unique. Once the bucket is created, data can be uploaded to the bucket as objects. Each object will have a *key* (or *key name)*, which will be the unique identifier for that object within that specific bucket.

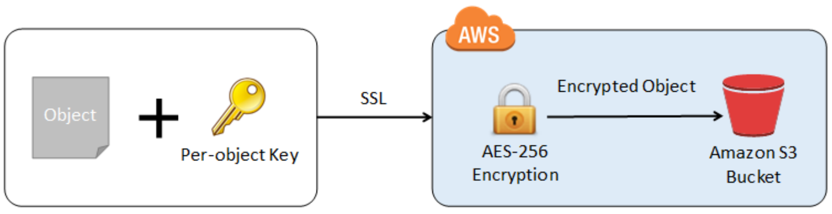
[71] https://wiki.archlinux.org/title/dm-crypt
[72] https://aws.amazon.com/blogs/security/how-to-protect-data-at-rest-with-amazon-ec2-instance-store-encryption/
[73] https://docs.aws.amazon.com/whitepapers/latest/security-design-of-aws-nitro-system/security-design-of-aws-nitro-system.html?did=wp_card&trk=wp_card
[74] https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html
[75] https://aws.amazon.com/s3/storage-classes/

Encryption at rest

In order to encrypt the stored objects, S3 offers a number of encryption options that can be split up into two main categories, server-side and client-side encryption.

Server-side Encryption

Server-side encryption (SSE) means that the data is encrypted at its destination by the application or service that receives it, in this case S3. The data is being encrypted by S3 as it is written to the disks in the datacenter and decrypted when the data is accessed. Since S3 handles the cryptographic operations for you, as long as you are authenticated and have the right permissions, you don't notice the difference between encrypted and unencrypted objects. S3 offers a number of server-side encryption options, of which only one at a time can be applied to an object. S3 server-side encryption is applied on the data object itself, therefore the metadata is not encrypted. S3 SSE uses 256-bit Advanced Encryption Standard key (AES-256), in the Galois/Counter Mode of Operation (AES-GCM)[76]. For SSE-S3 and SSE-KMS AWS uses the *envelope encryption* principle. Meaning the data is encrypted with a data encryption key (DEK), and the KMS key is only used as the key encryption key (KEK). Encrypted data keys are stored with the S3 objects. The S3 service decrypts an object by first decrypting the KEK, and then using the cleartext DEK to decrypt the object.



https://aws.amazon.com/blogs/aws/s3-encryption-with-your-keys/

SSE-S3

When using server-side encryption with Amazon S3 managed keys (SSE-S3), each object is encrypted with a unique key, then the key itself is additionally encrypted with an AWS managed root key. By default, all new objects uploaded to an S3 bucket will be encrypted with SSE-S3, unless another encryption method is specified[77]. Using SSE-S3 is relatively simple and requires no additional effort, however, since it is using AWS managed keys you have no control over the key lifecycle or other security configuration like IAM and key policies.

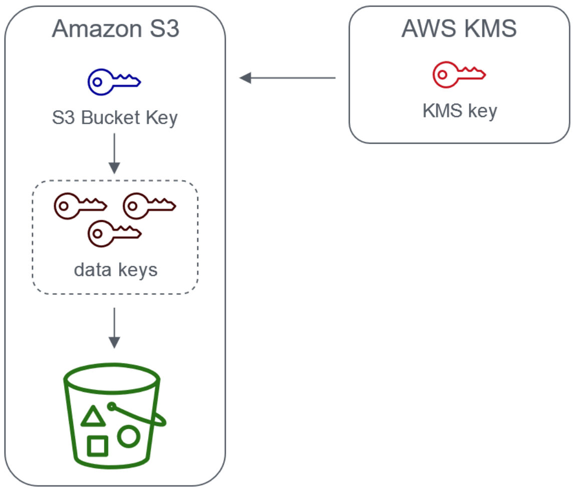
SSE-KMS

Server-side encryption is also possible with AWS KMS keys (SSE-KMS) and is similar to SSE-S3 but uses customer managed KMS keys. Data keys are generated inside KMS HSMs and the KMS key never leaves the HSM in cleartext. The advantages of

[76] https://csrc.nist.rip/groups/ST/toolkit/BCM/documents/proposedmodes/gcm/gcm-spec.pdf
[77] https://aws.amazon.com/blogs/aws/amazon-s3-encrypts-new-objects-by-default/

using customer managed KMS keys are the increased amount of control over the key lifecycle, the key policies, as well as the fact that KMS key use is logged in CloudTrail. Key logging is covered in KMS Logging and Monitoring.

When using SSE-KMS there is also the option to use a bucket key to reduce KMS costs[78]. When a bucket key is enabled for an S3 bucket, AWS generates a short-lived bucket-level key from AWS KMS and then temporarily stores it in S3. Data keys will be generated from this bucket key and used to encrypt the data. This reduces the number of requests to the KMS service and therefore reduces cost. The trade-off is that bucket keys are effectively KEKs that leave the secure boundary of the KMS HSMs. The FIPS 140-2 validation that applies to KMS keys does not apply to bucket keys,



Server-side encryption with AWS Key Management service using an S3 Bucket Key

https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-key.html

SSE-C

It is also possible to use server-side encryption with a customer-provided key (SSE-C)[79]. In order to use this method, you need to provide an encryption key together with the object when you upload it to S3. You need to generate and maintain the key outside of AWS. The encryption key that is provided is not persisted by S3, it only lives in memory as long as the cryptographic operation is performed, then the key is removed from memory. S3 will however generate a randomly salted Hash-based Message Authentication Code (HMAC) value of the encryption key to validate future requests. It is not possible to derive the encryption key from the HMAC.

Since the encryption key is not persisted by S3, you need to provide the key every time you want to retrieve the object. The encryption key is then validated against the HMAC, and if it matches, the object is decrypted and returned.

The main benefit of using SSE-C is that AWS does not persist the encryption key. They do however have access to the encryption key every time you provide it, therefore there is still a level of trust. Using SSE-C does however require you to maintain the

[78] https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-key.html
[79] https://docs.aws.amazon.com/AmazonS3/latest/userguide/ServerSideEncryptionCustomerKeys.html

encryption keys yourself and provide it every time you access the objects. If you lose the encryption key, there will be no other way of decrypting the data.

There are no additional fees for using SSE-C.

Default server-side encryption

In S3 It is possible to set default encryption behaviour on the bucket level[80]. Meaning that every new object that is uploaded will be encrypted with that specific server-side encryption method and KMS key. The KMS key can be either an AWS managed one (SSE-S3) or a customer managed KMS key (SSE-KMS). By default all new objects in S3 will already be encrypted with SSE-S3, with default encryption you can override this and use your own KMS key.

S3 Bucket Policy to enforce encryption

When default encryption is not an option for a specific bucket, it is still possible to enforce encryption of uploaded objects. This can be achieved by adding a bucket policy to the bucket that requires all new objects to be encrypted[81], you can also specify what type of server-side encryption should be used.

DSSE-KMS

Amazon S3 offers Dual-layer server-side encryption with keys stored in AWS KMS[82]. This provides an extra layer of security, by encrypting data objects twice. Each layer of encryption uses a different implementation of the 256-bit AES-GCM algorithm. DSSE-KMS uses AWS Key Management Service (KMS) to generate data keys, allowing customers to control their customer managed keys for both encryption layers by setting permissions per key and specifying key rotation schedules.

For DSSE-KMS, in addition to the charges for AWS KMS, you pay an additional per-gigabyte encryption fee for the second layer of encryption and decryption of data.

From a risk perspective, DSSE-KMS does not fundamentally change the risk of data being accessed by or through the cloud provider. It does offer additional protection against algorithmic weaknesses or an encryption key being compromised. The most common use case for DSSE-KMS is under regulatory regimes that require double encryption.

Client-side Encryption

In contrast to server-side encryption, client-side encryption[83] requires you to encrypt the data locally before uploading it to an S3 bucket. In this approach, S3 only receives the encrypted data and stores it, it does not play a role in the encryption or decryption of it. To encrypt the data locally you can use the AWS Encryption SDK[84], and you can either use an encryption key stored in AWS KMS or a key that is stored

[80] https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-encryption.html
[81] https://aws.amazon.com/blogs/security/how-to-prevent-uploads-of-unencrypted-objects-to-amazon-s3/
[82] https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingDSSEncryption.html
[83] https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingClientSideEncryption.html
[84] https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/introduction.html

within your application. In the former, all other considerations with regards to KMS encryption keys are still applicable. For the latter however, it is possible to completely shield off the data from the cloud provider as well. Since it is possible to generate an encryption outside of AWS and perform the encryption locally. Therefore, AWS will have no way of possibly decrypting the data. This however leads to an additional burden on the client, since you have to generate and maintain the keys yourself, if the key is lost, there is no way of decrypting the data.

Encryption in transit

Users interact with the S3 service directly via a REST API, or via applications that utilise the REST API such as the AWS Console for S3. The REST API uses HTTP or HTTPS as the underlying protocol. HTTPS is enforced by configuring the `aws:SecureTransport` condition in S3 bucket policies[85].

At the time of writing, AWS allows clients to make TLS 1.0 and 1.1 connections to S3. The process of updating the TLS configuration to a minimum level of TLS 1.2 is scheduled to be completed before the end of 2023[86]. Customers who have clients that use older TLS protocol versions are warned via their Personal Health Dashboard.

Amazon S3 allows customers to replicate the stored data between AWS cloud regions. AWS automatically replicates data between multiple data centres within a single cloud region. Whenever data is transmitted between AWS data centres, whether within a single region or across distinct regions, the traffic is encrypted[87].

Customers can enforce the use of TLS 1.2 or higher by setting conditions in S3 bucket policies[88].

Managed Databases (RDS)

Amazon Relational Database Service (RDS)[89] is a service that enables customers to set up, operate, and scale a relational database in AWS. RDS is a managed service and falls under the AWS shared responsibility model[90]. Specifically for RDS this means that AWS is responsible for the maintenance and patching of the operating system and database engine. While the customer is responsible for data protection, network connectivity and user management.

RDS supports a wide range of database engines:

- MariaDB
- Microsoft SQL server
- MySQL
- Oracle
- PostgreSQL

[85] https://docs.aws.amazon.com/AmazonS3/latest/userguide/security-best-practices.html
[86] https://aws.amazon.com/blogs/storage/enforcing-encryption-in-transit-with-tls1-2-or-higher-with-amazon-s3/
[87] https://docs.aws.amazon.com/whitepapers/latest/logical-separation/encrypting-data-at-rest-and--in-transit.html
[88] https://aws.amazon.com/blogs/storage/enforcing-encryption-in-transit-with-tls1-2-or-higher-with-amazon-s3/
[89] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html
[90] https://aws.amazon.com/compliance/shared-responsibility-model/

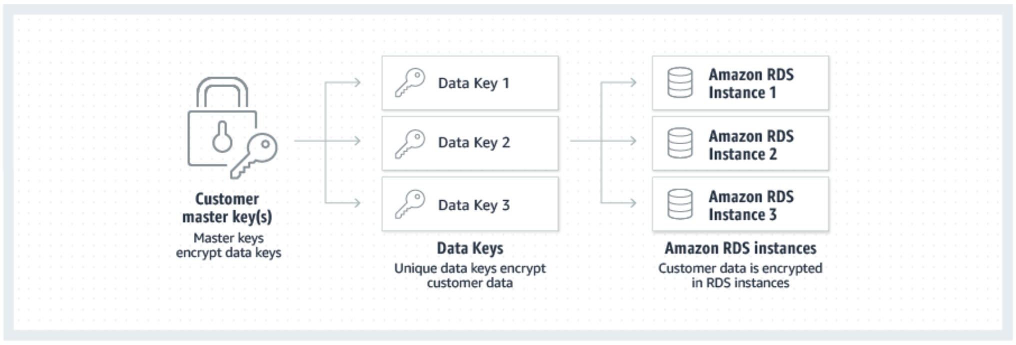
Each database engine has its own supported features, and each version of the engine can include specific features[91].

Encryption at rest

RDS uses EBS volumes for database and log storage. Therefore encryption at rest of RDS instances relies on the same concepts as encryption of EBS volumes. Encryption at rest includes encryption of the underlying storage of the database instances, its automated backups, read replicas, and snapshots[92]. Encryption at rest is handled by the RDS service on the server side, and is both transparent to the database engine as well as the client interacting with the database. Meaning no additional configuration is needed and both don't need to perform the encryption or decryption operations. The same principle of *envelope encryption* is applied, a data encryption key (DEK) is generated for each RDS instance and protected by the key encryption key (KEK) which is stored in KMS.

Encryption of an RDS instance can only be enabled on creation of the instance. You need to specify a KMS key that should be used for the encryption. To encrypt an existing RDS instance, you first need to create a snapshot from the existing instance. Then, when restoring a new instance from the snapshot, you can enable encryption and specify a KMS key.

RDS uses KMS keys to encrypt the database, which you can specify when configuring the RDS instance.



https://aws.amazon.com/rds/features/security/

Transparent Data Encryption (TDE)

RDS also supports encrypting an Oracle or SQL server database instance with Transparent Data encryption (TDE). TDE automatically encrypts data before it is written to storage, and automatically decrypts data when the data is read from storage. TDE for SQL Server[93] provides a two-tier key management architecture. A certificate is generated from the database master key and is used to protect the data encryption keys (envelope encryption). The database encryption key actually performs the encryption and decryption of the data on the database. RDS handles

[91] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.RDSFeaturesRegionsDBEngines.grids.html
[92] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.Encryption.html
[93] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Appendix.SQLServer.Options.TDE.html

and manages the database master key as well as the TDE certificate. For full details about TDE, please see the Microsoft documentation[94].

Encryption in transit

To protect data in transit, you can use Secure Socket Layer (SSL) or Transport Layer Security (TLS) from your application to encrypt the connection to a database instance[95]. SSL/TLS connections provide an additional layer of security by encrypting the data as it moves between the client and a database instance. Additionally, it is also possible to use a server certificate to ensure that the connection is being made to an RDS instance. This is done by checking the server certificate that is automatically installed on all database instances provisioned in RDS. Implementing SSL/TLS for RDS can vary between different database engines, therefore always check the AWS documentation[96].

SMB File Shares (FSx for Windows File Server)

Amazon FSx for Windows File Server[97] provides managed Microsoft Windows file servers. FSx has native support for Windows file system features and for the Server Message Block (SMB) protocol to access file storage over a network. FSx for Windows File Server consists of two primary resources, file systems and backups. The file system is where you actually store and access files and folders. A file system consists of one or more Windows file servers and storage volumes.

Encryption at rest

By default, all FSx file systems are encrypted at rest with keys managed in AWS KMS[98]. Data is always automatically encrypted before it is written to the file system, and automatically decrypted as it is being read. Both are handled transparently by the FSx service, so no additional modification of applications is needed.

FSx encryption can be configured to either use an AWS managed KMS key, or a customer managed KMS key.

Encryption in transit

Since FSx for Windows File Server supports the SMB protocol, encryption in transit can be easily configured too. The only requirement is that the file shares are mapped on a compute instance that supports SMB protocol 3.0 or newer[99]. FSx for Windows File Server will then automatically encrypt data in transit using SMB encryption when you access the file system. SMB encryption relies on AES-128-GCM or AES-128-CCM as the encryption algorithms. To ensure that data in transit is always

[94] https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/transparent-data-encryption?redirectedfrom=MSDN&view=sql-server-ver16
[95] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.SSL.html
[96] https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.SSL.html
[97] https://docs.aws.amazon.com/fsx/latest/WindowsGuide/what-is.html
[98] https://docs.aws.amazon.com/fsx/latest/WindowsGuide/encryption-at-rest.html
[99] https://docs.aws.amazon.com/fsx/latest/WindowsGuide/encryption-in-transit.html

encrypted, it is possible to limit access to the file system only to clients that support SMB encryption[100].

NFS File Shares (Amazon Elastic File System)

Amazon Elastic File System or Amazon EFS provides managed NFS file shares in a pay-as-you-grow model. Customers can create file shares and pay only for the amount of data stored. The underlying file systems grow elastically with the amount of data stored; the customer does not have to manage file system capacity.

Amazon EFS offers NFS access to file shares from Amazon EC2 VMs, containers (with Amazon Elastic Kubernetes Service, Amazon Elastic Container Service, including Fargate), AWS Lambda, and on-premises servers. EFS supports the NFS 4 protocol, allowing file shares to be mounted in Linux-based environments; Windows clients are not supported.

Encryption at rest

EFS supports encryption of data at rest. By default, EFS file systems are not encrypted. Customers must choose whether to enable encryption at rest when creating EFS file shares. Once created, the encryption setting of a file share cannot be changed.

In an encrypted EFS file system, data and metadata are automatically encrypted before being written to the file system. Similarly, as data and metadata are read, they are automatically decrypted before being presented to the application. These processes are handled transparently by Amazon EFS.

Amazon EFS integrates with AWS Key Management Service (AWS KMS) for key management. EFS metadata is encrypted using an AWS managed KMS key. Customers have a choice of using an AWS managed key or a customer managed key for encrypting data.

Encryption is done using an AES-256 algorithm.

Encryption in transit

Enabling encryption of data in transit for your Amazon EFS file system is done by enabling Transport Layer Security (TLS) when you mount your file system using the Amazon EFS mount helper[101].

When encryption of data in transit is declared as a mount option for your Amazon EFS file system, the mount helper initialises a client stunnel process. Stunnel is an open source multipurpose network relay. The client stunnel process listens on a local port for inbound traffic, and the mount helper redirects Network File System (NFS) client traffic to this local port. The mount helper uses TLS version 1.2 to communicate with your file system.

[100] https://docs.aws.amazon.com/fsx/latest/WindowsGuide/manage-encrypt-in-transit.html
[101] https://docs.aws.amazon.com/efs/latest/ug/efs-mount-helper.html

Customers who cannot or do not want to use the EFS mount helper, can still enable encryption in transit by installing, configuring and running stunnel on the client computer[102].

[102] https://docs.aws.amazon.com/efs/latest/ug/encryption-in-transit.html#how-encrypt-transit

Azure

Key Management

Encryption models

The data encryption (encryption at rest) models in Azure split into two main groups: "Client-side Encryption" and "Server-side Encryption"[103].

The Client-side Encryption model encrypts data independent of Azure services. Azure services, as a result, can not see decrypted data which reduces the offered cloud functionality. Likewise, customers manage and store keys on-premises (or in other secure stores).

The Server-side Encryption model encrypts data using Azure services. Azure services, as a result, perform encryption and decryption operations which requires access to the encryption key. The encryption key store is chosen by the customer and could be Platform-managed (default), Customer-managed using Azure Key Vault or Customer-provided using customer-controlled hardware.

The available encryption models differ per Azure service[104]. Using Server-side encryption with a customer-controlled key store, for example, is currently not applicable to any Azure service and limited to certain Microsoft 365 applications. Alternatively, customers could use Client-side Encryption to integrate their customer-controlled key store with their applications.

Key types

Platform-managed keys

Platform-managed keys (PMK) are encryption keys that are generated, stored, and managed entirely by Azure. Customers do not interact with PMKs. The keys used for Azure Data Encryption-at-Rest, for instance, are PMKs by default[105].

PMKs are available for Server-side Encryption using Azure Services and supported by basically any Azure service[106]. Note that the Azure documentation refers to these keys as Microsoft-managed keys (MMK)[107] or Service-managed keys (SMK)[108].

PMKs are stored in a Microsoft-managed key store[109].

Customer-managed keys

Customer-managed keys (CMK) are encryption keys that can be read, created, imported (also known as Bring Your Own Key - BYOK), deleted, updated, and/or administered by customers.

[103] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models
[104] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models#supporting-services
[105] https://learn.microsoft.com/en-us/azure/security/fundamentals/key-management
[106] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models#supporting-services
[107] https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption#about-encryption-key-management
[108] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-atrest#encrypted-compute
[109] https://learn.microsoft.com/en-us/azure/azure-government/azure-secure-isolation-guidance#storage-service-encryption

CMKs are available for Server-side Encryption using Azure services and supported by a broad set of Azure services using Azure Key Vault Standard or Key Vault Premium. A limited set of Azure services supports Key Vault Managed HSM. Azure SQL Database, for example, does not support Key Vault Managed HSM, yet does support Key Vault Standard and Key Vault Premium[110].

CMKs are stored in a customer-managed Key Vault Standard, Key Vault Premium or Key Vault Managed HSM (hardware security module) instance[111].

Key access controls are configured using Azure role-based access controls (Azure RBAC)[112]. Key Vault Standard and Premium additionally support Access Policies. This is a legacy feature and not recommended by Microsoft[113].

Customer-provided keys

Customer-provided keys (CPK) are encryption keys that can be read, created, deleted, updated, and/or administered by customers.

CPKs are available for Client-side encryption and supported by a small set of Azure services such as Azure Blob Storage and Azure SQL (Server) Database[114]. Note that the Azure documentation refers to these keys as Client-managed keys[115] or Customer-managed keys in customer-controlled hardware[116].

CPKs are stored in a customer-controlled key store outside of Azure. Therefore, this model is also known as Host Your Own Key[117].

Comparison

	Platform-managed keys	Customer-managed keys	Customer-provided keys
Alternative names	- Microsoft-managed key (MMK) - Service-managed key (SMK)		- Client-managed key - Customer-managed keys in customer-controlled hardware
Encryption models	Server-side encryption	Client- + server-side encryption	Client-side encryption
Supported Azure services[118]	All	Most	Few

[110] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models#supporting-services
[111] https://learn.microsoft.com/en-us/azure/security/fundamentals/key-management
[112] https://learn.microsoft.com/en-us/azure/key-vault/general/rbac-access-policy
[113] https://learn.microsoft.com/en-us/azure/key-vault/general/rbac-access-policy#data-plane-access-control-recommendation
[114] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models#supporting-services
[115] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models#supporting-services
[116] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models#server-side-encryption-using-customer-managed-keys-in-customer-controlled-hardware
[117] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models#server-side-encryption-using-customer-managed-keys-in-customer-controlled-hardware
[118] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models#supporting-services

Key storage

Azure offers several options for storing and managing your keys in the cloud, including Microsoft key stores, Azure Key Vault, Azure Managed HSM, Azure Dedicated HSM, and Azure Payment HSM. These options differ in terms of their FIPS compliance level, management overhead, and intended applications[119].

Azure Key Vault and Azure Managed HSM can be used to manage encryption keys for server-side encryption. The ensuing sections focus on these two solutions and briefly mentions Microsoft key stores, Azure Dedicated HSM and Azure Payments HSM.

Microsoft key stores

For Platform-Managed Keys, Azure implements cryptographic key management through the use of internal key stores. Each Azure service uses FIPS 140 approved algorithms[120]. PMKs for the services addressed in this document are stored in an internal key store, certified at FIPS 140-2 Level 1.

Azure ensures that internal key stores contain the approved trust anchors, including certificates with visibility external to Azure and certificates related to the internal operations of services[121].

Azure Key Vault

Azure Key Vault is one of several key management services in Azure. Key Vault is used to manage secrets, certificates and encryption keys[122]. It is integrated with many Azure services to support server-side encryption[123]. Client-side encryption is supported through the Key Vault API[124].

The service is offered in three tiers:

- Key Vault Standard is a multi-tenant cloud key management service. Stored keys are software-protected using industry-standard algorithms and key lengths.
- Key Vault Premium is a multi-tenant HSM offering. Stored keys are hardware-protected using nCipher nShield HSMs certified by NIST as being FIPS 140-2 compliant; relevant NIST certificate number is 2643[125,126].
- Key Vault Managed HSM is a single-tenant HSM offering. This offering is further described in the section Azure Managed HSM.

Azure Key Vault Standard and Premium contents are replicated within the region and to a secondary region at least 150 miles away, but within the same geography to maintain high durability of your keys and secrets[127]. Within Europe, Key Vault

[119] https://learn.microsoft.com/en-us/azure/security/fundamentals/key-management#azure-key-management-services
[120] https://learn.microsoft.com/en-us/azure/compliance/offerings/offering-fips-140-2#azure-and-fips-140
[121] Azure Commercial - System Security Plan (2022), SC-12 p. 905. https://servicetrust.microsoft.com/DocumentPage/3a8e0db5-9bc6-4837-856f-6788dc1ae950
[122] https://learn.microsoft.com/en-us/azure/key-vault/general/overview
[123] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models#supporting-services
[124] https://learn.microsoft.com/en-us/rest/api/keyvault/
[125] https://learn.microsoft.com/en-us/azure/key-vault/general/overview#securely-store-secrets-and-keys
[126] Azure Commercial - System Security Plan (2022), SC-12 p. 910. https://servicetrust.microsoft.com/DocumentPage/3a8e0db5-9bc6-4837-856f-6788dc1ae950
[127] https://learn.microsoft.com/en-us/azure/key-vault/general/disaster-recovery-guidance

contents are replicated between the North Europe (Ireland) and West Europe (The Netherlands) Azure regions[128].

Azure Managed HSM

Azure Managed HSM is one of several key management services in Azure. Managed HSM is a fully managed, highly available, single-tenant, FIPS 140-2 Level 3 validated HSM as a service fronted by a service that exposes crypto functionality through the Key Vault API[129]. Note that the service is also known as Azure Key Vault Managed HSM.

With Azure Managed HSM customers control the cryptographic root of trust of the security domain. This ensures that Microsoft does not have access to your cryptographic key material on the managed HSM[130].

The security domain, essentially, is an encrypted blob file that contains artefacts such as the HSM backup, user credentials, the signing key, and the data encryption key unique to your managed HSM. When initialising the service, the security domain is first generated in the managed HSM hardware (Marvell LiquidSecurity HSM adapters[131]), and the service software enclaves (trusted execution environments built on Intel Software Guard Extension (SGX)[132]). Second, the security domain is initialised and encrypted using 3 or more customer-provided public RSA keys using Shamir's Secret Sharing Algorithm[133]. Finally, to prevent lockout of the security domain, customers are required to manage the security domain quorum by distributing security domain encryption keys to different persons and storing keys in an offline fashion such as a USB drive or offline HSM[134].

High availability is provided through an HSM cluster managed by Microsoft. Each HSM cluster consists of multiple HSM partitions. If the hardware fails, member partitions for your HSM cluster will be automatically migrated to healthy nodes[135].

Azure Dedicated HSM

Azure Dedicated HSM is a FIPS 140-2 Level 3 validated single-tenant bare metal HSM offering that lets customers lease a general-purpose HSM appliance that resides in Microsoft datacenters. The customer has complete ownership over the Thales Luna 7 HSM device[136] and is responsible for patching and updating the firmware when required. Microsoft has no permissions on the device or access to the key material[137].

Azure Dedicated HSM is used for client-side encryption, independent of Azure services, through PKCS#11, JCE/JCA, and KSP/CNG APIs.

[128] https://learn.microsoft.com/en-us/azure/reliability/cross-region-replication-azure
[129] https://learn.microsoft.com/en-us/azure/key-vault/managed-hsm/overview#fully-managed-highly-available-single-tenant-hsm-as-a-service
[130] https://learn.microsoft.com/en-us/azure/key-vault/managed-hsm/security-domain
[131] https://learn.microsoft.com/en-us/azure/key-vault/managed-hsm/overview#access-control-enhanced-data-protection--compliance
[132] https://learn.microsoft.com/en-us/azure/key-vault/managed-hsm/mhsm-control-data#how-does-azure-key-vault-managed-hsm-protect-your-keys
[133] https://learn.microsoft.com/en-us/azure/key-vault/managed-hsm/security-domain#downloading-the-encrypted-security-domain
[134] https://learn.microsoft.com/en-us/azure/key-vault/managed-hsm/security-domain#establishing-a-security-domain-quorum
[135] https://learn.microsoft.com/en-us/azure/key-vault/managed-hsm/overview#fully-managed-highly-available-single-tenant-hsm-as-a-service
[136] https://learn.microsoft.com/en-us/azure/dedicated-hsm/overview
[137] https://learn.microsoft.com/en-us/azure/security/fundamentals/key-management-choose#learn-more-about-azure-key-management-solutions

High availability requires multiple HSM instances in multiple availability zones or regions. The HSM instances are grouped using Thales HA Group[138].

Azure Payments HSM

Azure Payment HSM is a FIPS 140-2 Level 3, PCI HSM v3, validated single-tenant bare metal HSM offering that lets customers lease a payment HSM appliance in Microsoft datacenters for payments operations, including payment processing, payment credential issuing, securing keys and authentication data, and sensitive data protection. The service is PCI DSS, PCI 3DS, and PCI PIN compliant[139].

Azure Payments HSM is independent of Azure services.

Comparison

	Microsoft key store	Azure Key Vault Standard	Azure Key Vault Premium	Azure Managed HSM	Azure Dedicated HSM	Azure Payments HSM
Encryption models	Server-side encryption	Client- + server-side encryption	Client- + server-side encryption	Client- + server-side encryption	Client-side encryption	Client-side encryption
Supported Azure services[140]	All	Most	Most	Many	Few	Few
Key types	n/a	Asymmetric keys, Secrets, Certs	Asymmetric keys, Secrets, Certs	Asymmetric/Symmetric keys	Asymmetric/Symmetric keys, Certs	Local Primary Key
Key protection	n/a	Software	Software or Multi-tenant HSM	Single-tenant HSM	Single-tenant HSM	Single-tenant HSM
Key sovereignty[141]	No	No	No	Yes	Yes	Yes
Compliance	n/a	FIPS 140-2 level 1	FIPS 140-2 level 2	FIPS 140-2 level 3	FIPS 140-2 level 3	FIPS 140-2 level 3, PCI HSM v3
Patching and maintenance	Microsoft	Microsoft	Microsoft	Microsoft	Customer	Customer
High	Microsoft	Microsoft	Microsoft	Shared	Customer	Customer

[138] https://learn.microsoft.com/en-us/azure/dedicated-hsm/high-availability
[139] https://learn.microsoft.com/en-us/azure/security/fundamentals/key-management-choose#learn-more-about-azure-key-management-solutions
[140] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-models#supporting-services
[141] https://learn.microsoft.com/en-us/azure/security/fundamentals/key-management-choose#compare-other-customer-requirements

	Microsoft key store	Azure Key Vault Standard	Azure Key Vault Premium	Azure Managed HSM	Azure Dedicated HSM	Azure Payments HSM
availability / disaster recovery						
Budget[142]	n/a	$	$$	$$$	$$$$	$$$$

The Microsoft key store, Azure Key Vault and Azure Managed HSM are the key management services relevant to encryption at rest; the Microsoft key store for PMKs and the others for CMKs. The NBA maturity model for information security[143] contains a control on cryptographic key management (SM.10). At maturity level 3 - the target that the Dutch education sector has committed to - a formal documented policy and procedure for key lifecycle management must be in place, from key generation, distribution and use, to revocation, archiving and destruction. This suggests as a baseline that educational institutions use CMKs, preferably stored in an HSM. We recommend that at minimum Azure Key Vault Premium be used with HSM-backed keys.

Key Logging and Monitoring

Each key store provides a different degree of logging and monitoring information. Microsoft key store, for example, provides no logging information, whereas Azure Dedicated HSM provides device level diagnostics such as temperature or power supply status. Likewise, the customer responsibilities differ between these key stores, hence the different degree of information to support the customer in managing their key store.

For Microsoft Key Store, no logs and metrics are available, as the service is fully operated by Microsoft.

For Azure Key Vault, logs are provided at the Vault, Key, Secret and Certificate level[144]. These logs include allowed and denied access requests, and hence aid in troubleshooting as well as detecting abuse. Azure Key Vault logs integrate with monitoring services such as Log Analytics. Therefore, it is possible to correlate the logs with protected resource activity logs for more advanced log analysis.

Azure Key Vault, additionally, provides metrics to track Vault availability, Vault capacity and API latency[145]. These metrics are integrated with monitoring service Azure Monitor and retained for 90 days. Additionally, metrics can be exported for long-term storage.

For Azure Managed HSM, logs are provided for REST API requests, Security domain operations, Role management operations, Backup and restore operations and Key

[142] https://learn.microsoft.com/en-us/azure/security/fundamentals/key-management-choose#compare-other-customer-requirements
[143] https://www.nba.nl/intern-en-overheidsaccountants/volwassenheidsmodel-informatiebeveiliging/
[144] https://learn.microsoft.com/en-us/azure/key-vault/general/logging?tabs=Vault#operation-names-table
[145] https://learn.microsoft.com/en-us/azure/key-vault/general/monitor-key-vault-reference#key-vault-metrics

operations[146]. These logs include allowed and denied access requests, and hence aid in troubleshooting as well as detecting abuse. Azure Managed HSM logs integrate with monitoring services such as Log Analytics. Therefore, it is possible to correlate the logs with protected resource activity logs for more advanced log analysis.

Azure Managed HSM, additionally, provides metrics to track HSM availability and API latency[147]. These metrics are integrated with monitoring service Azure Monitor and retained for 90 days. Additionally, metrics can be exported for long-term storage.

For Azure Dedicated HSM and Azure Payments HSM, customers are responsible for logging and monitoring using the SNMP ports of the devices[148,149] and integrating the logs with Azure monitoring services. Note, however, that because customers fully control the device, all logs of the device are available for logging and monitoring, including temperature and power supply data.

Key Management Best Practices

When using Azure Key Vault for key encryption, it is key to protect the encryption keys and secrets using strict access controls, hardware-backed vaults, frequent key rotation and alerting to respond to incidents as soon as possible. An extensive checklist is provided below:

1. Lock down access to your subscription, resource group, and key vaults (role-based access control (RBAC))[150].
2. Create access policies for every vault.
3. Use the principle of least privilege access to control access.
4. Enable Multi-factor authentication[151] in your organisation.
5. Enable Logging[152] and alerting[153] in Azure Key Vault to monitor how and when keys are accessed by whom.
6. Enable Azure Key Vault purge protection to be able to restore a key within 90 days after deletion.
7. Using the HSM-backed keys for FIPS 140-2 Level 2 compliance[154].
8. Rotate encryption keys periodically[155].
9. Use separate Key Vaults per application per environment per region[156].

Summary

The data encryption (encryption at rest) models in Azure consist of two main groups: "Client-side Encryption" and "Server-side Encryption". Client-side encryption

[146] https://learn.microsoft.com/en-us/azure/key-vault/managed-hsm/logging
[147] https://learn.microsoft.com/en-us/azure/azure-monitor/reference/supported-metrics/microsoft-keyvault-managedhsms-metrics
[148] https://learn.microsoft.com/en-us/azure/dedicated-hsm/monitoring#customer-monitoring
[149] https://learn.microsoft.com/en-us/azure/payment-hsm/faq#how-do-i-monitor-payshield-10k-
[150] https://learn.microsoft.com/en-us/azure/key-vault/general/best-practices
[151] https://learn.microsoft.com/en-us/azure/key-vault/general/security-features
[152] https://learn.microsoft.com/en-us/azure/key-vault/general/logging
[153] https://learn.microsoft.com/en-us/azure/key-vault/general/alert
[154] https://learn.microsoft.com/en-us/azure/key-vault/keys/about-keys
[155] https://learn.microsoft.com/en-us/azure/key-vault/keys/how-to-configure-key-rotation
[156] https://learn.microsoft.com/en-us/azure/key-vault/general/best-practices#use-separate-key-vaults

encrypts data independent of Azure services, whereas Server-side encryption uses Azure services to encrypt data.

By default Azure services encrypt data with a Platform-managed key (PMK): a key managed by Microsoft. Most Azure services, however, support Server-side encryption using Customer-managed keys (CMKs): keys managed by customers, and stored in Azure Key Vault-compatible key stores (Azure Key Vault, Azure Managed HSM). A few Azure services, additionally, support Client-side encryption using Customer-provided keys (CPKs): keys managed by customers, and stored in customer controlled key stores (outside of Azure).

While the confidentiality increases by using CMKs and CPKs, the availability becomes a shared responsibility between the customer and Microsoft. First and foremost, because the customer is responsible for key lifecycle management. Additionally, because the customer is responsible for the availability and reliability of the customer-controlled key store.

Key store lifecycle management challenges differ per key store. Azure Key Vault is fully-managed by Microsoft. Azure Managed HSM is managed by Microsoft, yet disaster recovery is a customer responsibility. Azure Dedicated HSM and Azure Payments HSM, finally, are entirely managed by the customer.

Access to Azure components by Microsoft personnel

Customer VMs and VMs backing Azure services are technically accessible by Azure staff. When Azure staff needs to access customer VMs for support reasons, the access is granted temporarily and requires appropriate approvals[157]. Customer approval is brought into the process via Customer Lockbox[158]. In response to questions about Lockbox, Microsoft provided the following statement:

> "Lockbox provides customers the ability to review and approve or reject requests from Microsoft engineering to access their data with minimal exceptions. For example, a major service outage requiring immediate attention to recover or restore services in an unexpected or unpredictable scenario would not trigger a customer Lockbox request. These "break glass" events are rare and, in most instances, do not require any access to customer data to resolve. Customer Lockbox requests are also not triggered by external legal demands for data, which are also exceedingly rare. For more information about how Microsoft protects your data and responds to government requests, please visit the Microsoft Trust Center."

[157] See "Access to Customer Virtual Machines by Azure Personnel" in https://servicetrust.microsoft.com/DocumentPage/1204614a-cefb-4e8e-8c4b-e3abfd1ddb9e (available to Microsoft customers after login)

[158] https://learn.microsoft.com/en-us/azure/security/fundamentals/customer-lockbox-overview%23general-availability&sa=D&source=docs&ust=1693227618981930&usg=AOvVaw0JVomgmhpzNR-MWb9vKpNv

While exceptional, the use of Azure VMs leaves a residual risk of data being accessed by or through Microsoft without the customer being made aware, irrespective of encryption settings.

Cloud Service Encryption

VMs and Block Storage (Disks)

Azure Virtual Machines (VMs) provide scalable and flexible deployment options for virtual machines. Azure offers multiple instance sizes, disk types and options to control availability and security.

Storage is attached to VMs using Azure Disks: virtual block-level storage volumes[159] powered by Azure Storage[160]. Azure Disks are Managed or Unmanaged. With Managed Disks, Azure manages the Azure Storage Account hosting the disk. Consequently, Azure manages the performance and availability of the disk. With Unmanaged Disks, customers bear the former responsibilities. Unmanaged disks, however, are being deprecated. The functionality is retired per September 30, 2025[161]. Azure customers are recommended to use Managed disks whenever possible.

Azure Disks attached to Azure VMs have three main roles: OS disk, data disk, and temporary disk[162]. Each VM has one OS disk containing the boot volume of a pre-installed OS, which was selected when the VM was created[163]. Optionally, a VM has one or more data disks to store application data, or any other data you need to persist[164]. Most VM series, additionally, contain a temporary disk - which is not a managed disk - that provides short-term storage for applications and processes such as page files, swap files and/or SQL Server tempdb. Data on the temporary disk may be lost during a maintenance event, when you redeploy a VM, or when you stop the VM. During a successful standard reboot of the VM, data on the temporary disk will persist[165].

Azure Disks performance can be improved by enabling host caching. Host caching works by bringing storage closer to the VM that can be written or read to quickly. The amount of storage that is available to the VM for host caching is in the documentation. For example, you can see the Standard_D8s_v3 comes with 200 GiB of cache storage[166].

Encryption at rest

Disks are always encrypted at rest using Azure Disk Storage Service Encryption (SSE)[167]. SSE can be controlled and extended using the disk encryption options:

[159] https://learn.microsoft.com/en-us/azure/virtual-machines/managed-disks-overview
[160] https://learn.microsoft.com/en-us/azure/virtual-machines/managed-disks-overview#disk-allocation-and-performance
[161] https://learn.microsoft.com/en-us/azure/virtual-machines/unmanaged-disks-deprecation
[162] https://learn.microsoft.com/en-us/azure/virtual-machines/managed-disks-overview#disk-roles
[163] https://learn.microsoft.com/en-us/azure/virtual-machines/managed-disks-overview#temporary-disk
[164] https://learn.microsoft.com/en-us/azure/virtual-machines/managed-disks-overview#data-disk
[165] https://learn.microsoft.com/en-us/azure/virtual-machines/managed-disks-overview#temporary-disk
[166] https://learn.microsoft.com/en-us/azure/virtual-machines/disks-performance#virtual-machine-uncached-vs-cached-limits
[167] https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption

Azure Disk Encryption (ADE), Encryption at host and Confidential disk encryption[168]. Note that it is not possible to combine ADE and SSE with customer-managed keys[169].

Each disk encryption option covers a different set of infrastructure layers and disk roles. ADE, for example, uses OS-level encryption to encrypt OS-, Data-, Temporary- and Cache disks. SSE, on the other hand, uses Azure Storage Encryption to encrypt OS- and Data disk data when it is persisted to the Azure Storage service, but does not encrypt Temporary and Cache disks.

Azure Disk Storage Service Encryption

Azure Disk Storage Service Encryption (SSE)[170] automatically encrypts data stored on Azure managed disks (OS and data disks) at rest when persisting it to Azure Storage service[171]. Due to the server-side encryption, the content flows unencrypted from the VM to the Azure Storage backend[172].

The disk encryption keys are protected (wrapped) by a Platform-Managed Key stored in Microsoft Key Store or Customer-Managed Key stored in a customer-managed Key Vault. When using customer-managed keys, the storage service uses the user-assigned managed identity and Key Vault key as configured by the Disk Encryption Set. Since both resources are customer-controlled, customers are able to deny access, disable and/or revoke keys and audit key usage to ensure that only managed disks or other authorised resources are accessing the encryption key[173].

Disks encrypted with SSE can be double encrypted: once at the Storage Service-layer using a customer-managed key and once at the Storage Infrastructure-layer using a platform-managed key[174]. Note that double encryption is not available for several disk types, including, Azure Ultra Disks and Premium SSD v2 disks[175].

Azure Disk Encryption

Azure Disk Encryption (ADE)[176] uses OS-level encryption to encrypt OS-, Data-, Temporary- and Cache disks[177][178] using BitLocker[179] on Windows and dm-crypt[180] on Linux. The disk encryption keys and secrets are created by Azure and stored in a customer-managed Key Vault[181]. Optionally, these secrets are protected (wrapped) by a key encryption key stored in (another) Key Vault[182].

Disks encrypted with ADE are always double encrypted: once at the VM-layer using a customer-managed key and once at the Storage service-layer using a platform-managed key. Due to the VM-layer encryption, content flows encrypted

[168] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption-overview
[169] https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-faq#how-is-azure-disk-encryption-different-from-storage-server-side-encryption-with-customer-managed-key-and-when-should-i-use-each-solution-
[170] https://learn.microsoft.com/en-us/azure/azure-government/azure-secure-isolation-guidance#server-side-encryption-for-managed-disks
[171] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption
[172] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption-overview#comparison
[173] https://learn.microsoft.com/en-us/azure/azure-government/azure-secure-isolation-guidance#storage-service-encryption
[174] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption#double-encryption-at-rest
[175] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption#restrictions-2
[176] https://learn.microsoft.com/en-us/azure/azure-government/azure-secure-isolation-guidance#azure-disk-encryption
[177] https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-faq#what-is-azure-disk-encryption-for-windows-vms-
[178] https://learn.microsoft.com/en-us/azure/virtual-machines/linux/disk-encryption-faq#what-is-azure-disk-encryption-for-linux-vms-
[179] https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-faq#what-is-azure-disk-encryption-for-windows-vms-
[180] https://learn.microsoft.com/en-us/azure/virtual-machines/linux/disk-encryption-faq#what-is-azure-disk-encryption-for-linux-vms-
[181] https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-key-vault
[182] https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-key-vault#set-up-a-key-encryption-key-kek

from the VM to the Azure Storage backend, thereby, providing end-to-end encryption with a customer-managed key[183].

Since ADE requires VM capacity, ADE is not available on Basic, A-series VMs, or on virtual machines with a less than 2 GB of memory[184]. ADE, additionally, requires a supported operating system[185,186]:

- Windows 8 and later
- Windows Server 2008R2 and later
- Windows 10 Enterprise multi-session and later
- Ubuntu 18.04 LTS, 20.04 LTS, 22.04 LTS
- CentOS 6.8, 7.4, 7.5, 7.6, 7.7, 7.9, 7.9, 8.1, 8.2, 8.3, 8.4
- RHEL 6.7, 6.8, 7.4, 7.5, 7.6, 7.7, 7.9, 7.9, 8.1, 8.2, 8.3, 8.4, 8.5, 8.6, 8.7
- Oracle Linux 8.5, 8.6
- openSUSE 42.3
- SLES 12-SP3, 12-SP4

Note that ADE is not available for several disk types, including, Azure ultra disks, Premium SSD v2 disks and disks configured as distributed file systems[187,188].

Encryption at host

Encryption at host[189] automatically encrypts OS-, Data-, Temporary- and Cache disks at rest using VM Host-level (Hypervisor-layer) encryption. Due to the Hypervisor-layer encryption, content flows encrypted from the VM host to the Azure Storage backend.

Encryption at host enhances Azure Disk Storage Service Encryption (SSE) by adding Temporary- and Cache-disk encryption and offering end-to-end encryption. Likewise, Encryption at host uses the same means[190] and restrictions[191] to protect disk encryption keys and support double encryption. Note, however, that temporary- and ephemeral OS disks are encrypted at rest with platform-managed keys[192].

Confidential disk encryption

Confidential disk encryption[193] automatically encrypts OS- and Cache disks at rest using VM-level encryption. Due to the VM-level encryption, content flows encrypted from the VM to the Azure Storage backend, thereby, providing end-to-end encryption.

Confidential disk encryption enhances Azure Disk Storage Service Encryption (SSE) by sealing disk encryption keys to the vTPM[194] to ensure that data is locked until specific hardware or software conditions are met[195]. Confidential disk encryption,

[183] https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-faq#what-is-azure-disk-encryption-for-windows-vms-
[184] https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-overview#supported-vms
[185] https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-overview#supported-vms-and-operating-systems
[186] https://learn.microsoft.com/en-us/azure/virtual-machines/linux/disk-encryption-overview#supported-vms-and-operating-systems
[187] https://learn.microsoft.com/en-us/azure/virtual-machines/windows/disk-encryption-windows#unsupported-scenarios
[188] https://learn.microsoft.com/en-us/azure/virtual-machines/linux/disk-encryption-linux#unsupported-scenarios
[189] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption#encryption-at-host---end-to-end-encryption-for-your-vm-data
[190] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption#encryption-at-host---end-to-end-encryption-for-your-vm-data
[191] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption#restrictions-1
[192] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption#encryption-at-host---end-to-end-encryption-for-your-vm-data
[193] https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-vm-overview#confidential-os-disk-encryption
[194] https://learn.microsoft.com/en-us/windows/security/information-protection/tpm/tpm-fundamentals
[195] https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-vm-overview#attestation-and-tpm

therefore, is only available on confidential computing infrastructure[196] and on operating systems supporting generation 2 VMs[197,198].

As the scope of Confidential disk encryption is limited to OS- and Cache disks, Confidential disk encryption can be combined with SSE or Encryption at host to encrypt Temporary- and Data disks[199].

Comparison

	Azure Disk Encryption	**Azure Storage Service Encryption (SSE)**	**Encryption at Host**	**Confidential disk encryption (OS disk only)**
Encryption at rest (OS and data disks)	✅	✅	✅	✅
Temp disk encryption	✅	❌	✅	❌
Encryption of caches	✅	❌	✅	✅
Data flows encrypted between Compute and Storage	✅	❌	✅	✅
Customer control of keys	✅ When configured with KEK	✅ When configured with DES	✅ When configured with DES	✅ When configured with DES
HSM-backed key support	Azure Key Vault Premium	Azure Key Vault Premium and Managed HSM	Azure Key Vault Premium and Managed HSM	Azure Key Vault Premium and Managed HSM
Does not use your VM's CPU	❌	✅	✅	❌
Works for custom images	❌ Does not work for custom Linux images	✅	✅	✅
Enhanced Key Protection	❌	❌	❌	✅
Microsoft Defender for Cloud disk encryption status	Healthy	Unhealthy	Healthy	Not applicable

Source: https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption-overview#comparison

Encryption in transit

Azure Customer traffic is encrypted using IEEE 802.1AE MAC Security Standards (also known as MACsec) whenever traffic moves between datacenters - outside physical boundaries not controlled by Microsoft (or on behalf of Microsoft). This MACsec encryption is on by default for all Azure traffic travelling within a region or between regions, and no action is required on customers' part to enable[200].

Within the Azure datacenter, traffic between the VM and the Storage service is unencrypted. To ensure end-to-end encryption, the disk encryption can be applied

[196] https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-vm-overview#size-support
[197] https://learn.microsoft.com/en-us/azure/confidential-computing/confidential-vm-overview#os-support
[198] https://learn.microsoft.com/en-us/azure/virtual-machines/generation-2
[199] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption-overview
[200] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-overview#data-link-layer-encryption-in-azure

at the VM- or VM Host-level using Azure Disk Encryption or Encryption at host respectively[201]. Note, however, that Azure Disk Encryption and/or Encryption at host are not available for all Managed Disk types. SSD Premium v2 disks and Azure Ultra Disks cannot be protected using Encryption at Host or Azure Disk Encryption.

Object Storage (Blob Storage)

Azure Blob Storage is Microsoft's object storage solution for the cloud. Blob Storage is optimised for storing massive amounts of unstructured data. Unstructured data is data that doesn't adhere to a particular data model or definition, such as text or binary data[202].

Objects stored in Blob Storage are called Blobs[203]. Blobs are organised using Containers and associated with a Storage Account. This Storage account - a unique namespace for your Azure Storage data[204] - defines the overall performance and redundancy options of your data[205]. Premium Blob Storage, for example, offers low latency and high throughput. Similarly, Zone-redundant storage (ZRS) ensures availability by replicating data across three Azure availability zones[206].

Azure Blob Storage supports many more features[207]. This report, however, solely focuses on the available encryption options.

Encryption at rest

Objects are always encrypted at rest using Azure Storage Service Encryption (SSE). SSE can be controlled and extended using Customer-managed- / Customer-provided keys, Infrastructure encryption, Encryption scopes[208] and Client-side encryption.

Azure Storage Service Encryption (SSE)

Data in Azure Storage is encrypted and decrypted transparently using 256-bit AES encryption. Azure Storage encryption is enabled for all storage accounts, and cannot be disabled[209].

Azure Storage Service Encryption (SSE) uses the Storage Account Account Encryption Key (AEK) to derive a unique Data Encryption Key (DEK) for each block of data[210]. The AEK is generated when the Storage Account is created, and protected using envelope encryption by a Platform-managed-[211] or Customer-managed key[212,213]. The DEK derivation metadata is stored alongside the encrypted data, and regenerated when needed[214].

[201] https://learn.microsoft.com/en-us/azure/virtual-machines/disk-encryption-overview#comparison
[202] https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-overview
[203] https://learn.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction#blob-storage-resources
[204] https://learn.microsoft.com/en-us/azure/storage/common/storage-account-overview
[205] https://learn.microsoft.com/en-us/azure/storage/common/storage-account-overview#types-of-storage-accounts
[206] https://learn.microsoft.com/en-us/azure/storage/common/storage-redundancy#zone-redundant-storage
[207] https://learn.microsoft.com/en-us/azure/storage/blobs/storage-feature-support-in-storage-accounts
[208] https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption
[209] https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption#about-azure-storage-service-side-encryption
[210] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-atrest#envelope-encryption-with-a-key-hierarchy
[211] https://learn.microsoft.com/en-us/azure/azure-government/azure-secure-isolation-guidance#storage-service-encryption
[212] https://learn.microsoft.com/en-us/azure/storage/common/customer-managed-keys-overview#enable-customer-managed-keys-for-a-storage-account
[213] https://learn.microsoft.com/en-us/azure/storage/common/customer-managed-keys-overview#about-customer-managed-keys
[214] https://learn.microsoft.com/en-us/events/ignite-2016/brk3211, slide 39

Customer-managed keys

The Storage Account Account Encryption Key (AEK) can be protected using Customer-managed keys stored in Azure Key Vault or Managed HSM. The key vault or managed HSM that stores the key must have both soft delete and purge protection enabled. SSE supports RSA and RSA-HSM keys of sizes 2048, 3072 and 4096[215]. Note that a role assignment is required to allow Azure Storage to use the Key Vault key. This role assignment grants the Managed identity assigned to the Storage Account permissions to read/wrap/unwrap the Customer-managed key[216].

Customer-provided keys

Clients making requests against Azure Blob storage can provide an AES-256 encryption key to encrypt that Blob on a write operation. Subsequent requests to read or write to the blob must include the same key[217].

When a client application provides an encryption key on the request, Azure Storage performs encryption and decryption transparently while reading and writing blob data. Azure Storage writes an SHA-256 hash of the encryption key alongside the blob's contents. The hash is used to verify that all subsequent operations against the blob use the same encryption key[218].

Infrastructure encryption

Objects encrypted with SSE can be double encrypted: once at the Storage Service-layer using a platform- or customer-managed key and once at the Storage Infrastructure-layer using a platform-managed key. Double encryption of Azure Storage data protects against a scenario where one of the encryption algorithms or keys may be compromised. In this scenario, the additional layer of encryption continues to protect your data[219].

Encryption scopes

By default, a storage account is encrypted with a key scoped to the entire Storage Account. With Encryption Scopes, Containers or Blobs can be encrypted using a key scoped to the Encryption Scope, while using a single Blob Storage resource[220]. The Encryption Scope key supports the same encryption options as Azure Storage Service Encryption (SSE): Platform-managed- or Customer-managed keys and Infrastructure encryption.

When uploading a Blob with a specific Encryption Scope[221], the Encryption Scope overrides the default Storage Account Encryption[222]. Additionally, Storage Account Containers can be configured to enforce a specific Encryption Scope. This ensures

[215] https://learn.microsoft.com/en-us/azure/storage/common/customer-managed-keys-overview#key-vault-requirements

[216] https://learn.microsoft.com/en-us/azure/storage/common/customer-managed-keys-configure-existing-account#choose-a-managed-identity-to-authorize-access-to-the-key-vault

[217] https://learn.microsoft.com/en-us/azure/storage/blobs/encryption-customer-provided-keys

[218] https://learn.microsoft.com/en-us/azure/storage/blobs/encryption-customer-provided-keys#encrypting-read-and-write-operations

[219] https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption#doubly-encrypt-data-with-infrastructure-encryption

[220] https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption#about-encryption-key-management

[221] https://learn.microsoft.com/en-us/rest/api/storageservices/put-blob?tabs=azure-ad#request-headers-all-blob-types

[222] https://learn.microsoft.com/en-us/azure/storage/blobs/encryption-scope-overview#encryption-scopes-for-containers-and-blobs

that all data within that Container is encrypted using the encryption settings of the Encryption Scope.

Client-side encryption

Regardless of Storage Service Encryption (SSE), Microsoft provides Azure Blob Storage libraries to support encrypting data within client applications before uploading to Azure Storage, and decrypting data while downloading to the client[223].

By using client-side encryption, customers can use Customer-provided keys and prevent exchanging key material with Azure altogether. Logically, this entails that data can no longer be processed using solely Azure services.

Encryption in transit

Objects in Blob Storage can be accessed from anywhere in the world via HTTP or HTTPS. Clients can also securely connect to Blob Storage by using SSH File Transfer Protocol (SFTP) and mount Blob Storage containers by using the Network File System (NFS) 3.0 protocol[224].

Note that HTTP access is rejected by default for new Storage Accounts due to the Secure transfer option[225]. When secure transfer is required, a call to an Azure Storage REST API operation must be made over HTTPS. HTTPS traffic requires a minimum version TLS 1.2 by default, although TLS 1.0 and TLS 1.1 are still supported for backward compatibility[226].

Similarly, note that NFS access must originate from an Azure VNet. By itself, this doesn’t encrypt the traffic, but this limits the exposure to the customers network. Second, note that any other tool used to secure data including account key authorization, Azure Active Directory (AD) security, and access control lists (ACLs) are not yet supported in accounts that have the NFS 3.0 protocol support enabled on them[227].

File Shares (Azure Files)

Managed file shares are available through Azure Blob Storage (NFSv3), Azure Files (SMB, NFSv4.1) and Azure NetApp Files (SMB, NFSv3, NFSv4.1)[228]. Azure Blob Storage and Azure Files are built on top of Azure Storage. Azure NetApp Files is built on NetApp's bare metal with ONTAP storage OS running inside the Azure datacenter[229]. This section is scoped to encryption options of the Azure Files service.

Azure Files is deployed in two main ways: by directly mounting the serverless Azure file shares or by caching Azure file shares on-premises using Azure File Sync[230]. This

[223] https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption#client-side-encryption-for-blobs-and-queues
[224] https://learn.microsoft.com/en-us/azure/storage/common/storage-introduction#blob-storage
[225] https://learn.microsoft.com/en-us/azure/storage/common/storage-require-secure-transfer
[226] https://learn.microsoft.com/en-us/azure/storage/common/transport-layer-security-configure-minimum-version
[227] https://learn.microsoft.com/en-us/azure/storage/blobs/network-file-system-protocol-support#network-security
[228] https://learn.microsoft.com/en-us/azure/storage/common/nfs-comparison#comparison
[229] https://learn.microsoft.com/en-us/azure/storage/files/storage-files-netapp-comparison
[230] https://learn.microsoft.com/en-us/azure/storage/files/storage-files-planning

report addresses deployment considerations for deploying an Azure file share to be directly mounted by an on-premises or cloud client.

Azure Files deploys File shares on top of Azure Storage Accounts - a unique namespace for your Azure Storage data[231]. Azure Files offers four different tiers of storage - premium, transaction optimised, hot, and cool - to allow you to tailor your shares to the performance and price requirements of your use case. The premium tier is built on top of FileStorage storage accounts backed by SSD-disks and the other tiers are built on top of General Purpose version 2 (GPv2) storage accounts backed by HDD-disks[232]. All tiers offer SMB-based storage. NFS-based storage, on the other hand, requires the Premium tier.

Azure Files supports many more features. This report, however, solely focuses on the available encryption options.

Encryption at rest

File Shares are always encrypted at rest using Azure Storage Service Encryption (SSE). SSE can be controlled and extended using Customer-managed keys and Infrastructure encryption[233]. SSE applies regardless of the file server protocol.

Azure Storage Service Encryption (SSE)

Data in Azure Storage is encrypted and decrypted transparently using 256-bit AES encryption. Azure Storage encryption is enabled for all storage accounts, and cannot be disabled[234].

Azure Storage Service Encryption (SSE) uses the Storage Account Account Encryption Key (AEK) to derive a unique Data Encryption Key (DEK) for each block of data[235]. The AEK is generated when the Storage Account is created, and protected using envelope encryption by a Platform-managed-[236] or Customer-managed key[237,238]. The DEK derivation metadata is stored alongside the encrypted data, and regenerated when needed[239].

Customer-managed keys

The Storage Account Account Encryption Key (AEK) can be protected using Customer-managed keys stored in Azure Key Vault or Managed HSM. The key vault or managed HSM that stores the key must have both soft delete and purge protection enabled. SSE supports RSA and RSA-HSM keys of sizes 2048, 3072 and 4096[240]. Note that a role assignment is required to allow Azure Storage to use the

[231] https://learn.microsoft.com/en-us/azure/storage/common/storage-account-overview
[232] https://learn.microsoft.com/en-us/azure/storage/files/storage-files-planning#storage-tiers
[233] https://learn.microsoft.com/en-us/azure/storage/files/storage-files-planning#encryption-at-rest
[234] https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption#about-azure-storage-service-side-encryption
[235] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-atrest#envelope-encryption-with-a-key-hierarchy
[236] https://learn.microsoft.com/en-us/azure/azure-government/azure-secure-isolation-guidance#storage-service-encryption
[237] https://learn.microsoft.com/en-us/azure/storage/common/customer-managed-keys-overview#enable-customer-managed-keys-for-a-storage-account
[238] https://learn.microsoft.com/en-us/azure/storage/common/customer-managed-keys-overview#about-customer-managed-keys
[239] https://learn.microsoft.com/en-us/events/ignite-2016/brk3211, slide 39
[240] https://learn.microsoft.com/en-us/azure/storage/common/customer-managed-keys-overview#key-vault-requirements

Key Vault key. This role assignment grants the Managed identity assigned to the Storage Account permissions to read/wrap/unwrap the Customer-managed key[241].

Infrastructure encryption

File Shares encrypted with SSE can be double encrypted: once at the Storage Service-layer using a platform-/customer-managed key and once at the Storage Infrastructure-layer using a platform-managed key. Double encryption of Azure Storage data protects against a scenario where one of the encryption algorithms or keys may be compromised. In this scenario, the additional layer of encryption continues to protect your data[242].

Encryption in transit

File shares are accessible using Server Message Block (SMB) protocol, Network File System (NFS) protocol, and Azure Files REST API (HTTP/S)[243]. By default, encryption in transit is enforced by the Secure transfer option for SMB and Files REST API access[244]. The supported SMB channel encryption algorithms are AES-256-GCM, AES-128-GCM, and AES-128-CCM[245]. Note that the Secure transfer option does not apply to NFS connections, since Azure Files does not yet support encryption-in-transit for NFS[246].

In addition to the Secure transfer option, Azure Files support Azure VNet integration. In fact, NFS file shares requires Azure VNet integration to control access to file shares[247], since user authentication is not supported for NFS file shares. For Azure VNet integrated traffic, VNet specific transit encryption applies. Thus, Azure customer traffic is encrypted using IEEE 802.1AE MAC Security Standards (also known as MACsec) whenever traffic moves between datacenters - outside physical boundaries not controlled by Microsoft (or on behalf of Microsoft). This MACsec encryption is on by default for all Azure traffic travelling within a region or between regions, and no action is required on customers' part to enable[248]. Within the Azure datacenter, traffic between the source resource and the File share is unencrypted.

Managed databases

Azure offers managed Microsoft SQL (MSSQL)-, PostgreSQL-, MySQL- and MariaDB databases. The MSSQL databases are offered as Azure SQL services, whereas the PostgreSQL, MySQL and MariaDB databases are offered as Azure Database services. Note that Azure Cosmos DB also offers a PostgreSQL interface to run a globally distributed database platform[249]. As this deployment option is not relevant to the research use case, Azure Cosmos DB is ignored.

[241] https://learn.microsoft.com/en-us/azure/storage/common/customer-managed-keys-configure-existing-account#choose-a-managed-identity-to-authorize-access-to-the-key-vault
[242] https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption#doubly-encrypt-data-with-infrastructure-encryption
[243] https://learn.microsoft.com/en-us/azure/storage/files/storage-files-introduction
[244] https://learn.microsoft.com/en-us/azure/storage/common/storage-require-secure-transfer
[245] https://learn.microsoft.com/en-us/azure/storage/files/files-smb-protocol?tabs=azure-portal#smb-security-settings
[246] https://learn.microsoft.com/en-us/azure/storage/files/files-nfs-protocol#support-for-azure-storage-features
[247] https://learn.microsoft.com/en-us/azure/storage/files/storage-files-networking-overview#public-endpoint-firewall-settings
[248] https://learn.microsoft.com/en-us/azure/security/fundamentals/encryption-overview#data-link-layer-encryption-in-azure
[249] https://learn.microsoft.com/en-us/azure/cosmos-db/distributed-relational

Azure SQL family (Azure SQL Database, Azure SQL Managed Instance)

The Azure SQL family offers a range of Microsoft SQL Server database deployment options from edge to cloud[250]. The platform-as-a-service options include Azure SQL Database and Azure SQL Managed Instance. Functionally, the former provides a database on a logical server, whereas the latter provides a server instance to provide a greater compatibility to Microsoft SQL Server features[251].

Encryption at rest

Microsoft SQL Server provides the following mechanisms for encryption: Transact-SQL functions, Asymmetric keys, Symmetric keys, Certificates, Transparent Data Encryption (TDE) and Always Encrypted[252]. The Azure SQL family, by default, uses TDE for server-side encryption of data at rest.

TDE encrypts an entire database using that symmetric key called the database encryption key. The database encryption key is protected by the TDE protector, other keys or certificates which are protected either by the database master key or by an asymmetric key stored in an Extensible Key Management (EKM) module such as Azure Key Vault or Azure Managed HSM[253].

For Azure SQL Database, the TDE protector is set at the logical server[254] level and is inherited by all databases associated with that logical server. For Azure SQL Managed Instance, the TDE protector is set at the instance level and it is inherited by all encrypted databases on that instance[255]. Note that TDE cannot be used to encrypt system databases, such as the master database, in Azure SQL Database and Azure SQL Managed Instance. The master database contains objects that are needed to perform TDE operations on user databases[256].

In Azure SQL, TDE is enabled by default using a built-in server certificate as TDE protector. The built-in server certificate is unique for each server and the encryption algorithm used is AES 256[257]. Customer-managed TDE uses a customer-managed asymmetric key, which is stored in a customer-managed Azure Key Vault or Azure Managed HSM and never leaves the key vault, as TDE protector. Note that Azure SQL needs to be granted permissions to the customer-owned key vault to decrypt and encrypt the DEK[258].

Always Encrypted

Always Encrypted is a feature designed to protect sensitive data, such as credit card numbers, stored in Azure SQL database columns. With Always Encrypted, sensitive data is encrypted inside client applications without revealing the encryption keys to the Database Engine. This provides a separation between those who own the data

[250] https://azure.microsoft.com/en-us/products/azure-sql
[251] https://learn.microsoft.com/en-us/azure/azure-sql/database/features-comparison
[252] https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/encryption-hierarchy?view=sql-server-ver16#encryption-mechanisms
[253] https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/encryption-hierarchy?view=sql-server-ver16#transparent-data-encryption
[254] https://learn.microsoft.com/en-us/azure/azure-sql/database/logical-servers
[255] https://learn.microsoft.com/en-us/azure/azure-sql/database/transparent-data-encryption-tde-overview
[256] https://learn.microsoft.com/en-us/azure/azure-sql/database/transparent-data-encryption-tde-overview
[257] https://learn.microsoft.com/en-us/azure/azure-sql/database/transparent-data-encryption-tde-overview#service-managed-transparent-data-encryption
[258] https://learn.microsoft.com/en-us/azure/azure-sql/database/transparent-data-encryption-tde-overview#customer-managed-transparent-data-encryption---bring-your-own-key

and can view it, and those who manage the data but should have no access - on-premises database administrators, cloud database operators, or other high-privileged unauthorised users[259].

Note that, because Azure SQL database is unable to read the data in encrypted columns, Azure SQL is limited in its ability to apply queries. A where clause, for example, needs to match an encrypted value (assuming that the encrypted value is deterministic), hence querying all smaller or larger values is not possible[260].

Always Encrypted with secure enclaves

Always Encrypted with secure enclaves addresses the query limitations of Always Encrypted by allowing some computations on plaintext data inside a secure enclave on the server side. Please note that this requires hosting the keys in a key store available to Azure SQL: Windows Certificate Store or Key Vault[261].

A secure enclave is a protected region of memory within the Database Engine process, which appears as an opaque box to the rest of the Database Engine and other processes on the hosting machine. There's no way to view any data or code inside the enclave from the outside, even with a debugger. These properties make the secure enclave a trusted execution environment that can safely access cryptographic keys and sensitive data in plaintext, without compromising data confidentiality[262].

Always Encrypted with secure enclaves supports the following enclave technologies (or enclave types):

- Virtualization-based Security (VBS) enclaves (also known as Virtual Secure Mode, or VSM enclaves) - a software-based technology that relies on Windows hypervisor and doesn't require any special hardware. At the time of writing, VBS enclaves in Azure SQL Database are in preview.
- Intel Software Guard Extensions (Intel SGX) enclaves - a hardware-based trusted execution environment technology.

In Azure SQL Database, a database can use either an Intel SGX enclave or a VBS enclave, depending on the hardware the database is configured to run on. In Azure SQL Managed Instance, Always Encrypted using secure enclaves is not yet available[263].

Encryption in transit

Azure SQL secures customer data by encrypting data in transit with Transport Layer Security (TLS 1.2).

Azure SQL enforces encryption (SSL/TLS) at all times for all connections. This ensures all data is encrypted "in transit" between the client and server irrespective of the setting of Encrypt or TrustServerCertificate in the connection string[264].

[259] https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine
[260] https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine#limitations
[261] https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-enclaves-provision-keys
[262] https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-enclaves
[263] https://learn.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-enclaves#supported-enclave-technologies
[264] https://learn.microsoft.com/en-us/azure/azure-sql/database/security-overview?view=azuresql#transport-layer-security-encryption-in-transit

Azure Databases family (Azure Databases for PostgreSQL, MySQL, MariaDB)

The Azure Databases family offers a range of open source databases such as PostgreSQL, MySQL and MariaDB. The databases are offered as a managed service, built on top of common Azure infrastructure such as Azure VMs and Azure Storage[265,266,267].

Encryption at rest

Azure Databases database data and backup files are always encrypted at-rest using Azure Storage Service encryption. Temporary files created while running queries are encrypted for PostgreSQL[268] and MySQL[269]. For MariaDB the temporary files created while running queries are not encrypted[270].

Data in Azure Storage is encrypted and decrypted transparently using 256-bit AES encryption[271]. By default, Azure Storage Service encryption, uses a Platform-managed key associated with the Azure Databases service. Customer-managed keys are supported for PostgreSQL[272] and MySQL[273]. Note that MariaDB only supports Platform-managed keys.

Encryption in transit

All the encryption in transit for database related resources is protected by SSL/TLS (1.0, 1.1, 1.2+) and is enforced by default[274,275,276]. For PostgreSQL and MySQL the default minimum requirement is TLS 1.2. For MariaDB the default minimum requirement is SSL.

265 https://learn.microsoft.com/en-us/azure/postgresql/flexible-server/overview#overview
266 https://learn.microsoft.com/en-us/azure/mysql/flexible-server/overview#overview
267 https://learn.microsoft.com/en-us/azure/mariadb/concepts-high-availability#components-in-azure-database-for-mariadb
268 https://learn.microsoft.com/en-us/azure/postgresql/flexible-server/concepts-security#information-protection-and-encryption
269 https://learn.microsoft.com/en-us/azure/mysql/single-server/concepts-security#at-rest
270 https://learn.microsoft.com/en-us/azure/mariadb/concepts-security#at-rest
271 https://learn.microsoft.com/en-us/azure/storage/common/storage-service-encryption#about-azure-storage-service-side-encryption
272 https://learn.microsoft.com/en-us/azure/postgresql/flexible-server/concepts-data-encryption
273 https://learn.microsoft.com/en-us/azure/mysql/flexible-server/concepts-customer-managed-key
274 https://learn.microsoft.com/en-us/azure/postgresql/flexible-server/concepts-networking#tls-and-ssl
275 https://learn.microsoft.com/en-us/azure/mysql/flexible-server/concepts-networking#tls-and-ssl
276 https://learn.microsoft.com/en-us/azure/mariadb/concepts-security#in-transit

Conclusion / Final remarks / Open questions

The most important question to be answered is who has access to your data given a certain encryption setting. When you encrypt stored data with a strong encryption system, it is rendered inaccessible to anyone who can access the storage media but who does not have access to the encryption/decryption keys. Our deep dive into the encryption options provided by AWS and Azure shows that they both implement strong encryption systems.

Data that has been encrypted is accessible to actors who have access to the encryption/decryption keys used. Considering the encryption systems used by AWS and Azure, there are two risks: a compromise of Data Encryption Keys (DEKs), and a compromise of Key Encryption Keys (KEKs) - the keys used to protect Data Encryption Keys.

When using server-side encryption the DEKs are used in the storage service. AWS and Azure have many security controls in place that restrict access to their production services, including the services investigated in this paper. Their customers have access to audit reports for, amongst others, ISO27001, SOC2, C5, and other information security standards, allowing them to ascertain whether these security controls are appropriate for the data processed. We note that the security controls are mostly process-oriented. With a few exceptions, there are technical means available to the cloud providers to access customer data, whether encrypted or not. The process controls indicate that in most cases the customer has the opportunity to pre-approve such data access (via Azure Customer Lockbox), or can see in log files that data has been accessed by the provider. But there is no guarantee that all such access is logged in a logging system available to the customer.

The only option available to customers that do not want *data* encryption keys available to the cloud provider is to use client side encryption. In that case, the customer encrypts the data before it is sent to the cloud without transmitting the key. This approach severely limits the cloud storage and processing services that can be used. The only services that support this model are the object storage services - Amazon S3 and Azure Blob Storage - and SQL Server Always Encrypted. Azure SQL not only offers Always Encrypted, but also Always Encrypted with Secure Enclaves, broadening the set of use cases supported. Additionally, when using client side encryption, the customer has to implement a secure and scalable key management system to safeguard the data encryption keys.

When customers use client side encryption the cloud providers have no technical means to read the customer data. We conclude with a high degree of confidence that this is also the case with server-side encryption for data on Amazon EBS virtual disks used with Amazon EC2 virtual or bare-metal machines that run on Nitro hosts (all current generation EC2 instances).

AWS and Azure offer key management services to securely store Key Encryption Keys. They offer two types of KEKs: provider managed keys (PMKs) and customer managed keys (CMKs). Key access and the key lifecycle of PMKs are managed by the provider. Customers cannot disable PMKs, nor can they access key usage logs. Therefore we recommend that customers use CMKs: then the customer can manage the key lifecycle and can access logs that show when keys are used and by whom. The key storage systems backing the key management services are FIPS 140-2 validated. Azure Key Vault Premium with HSM-backed keys and AWS KMS store KEKs (and other cryptographics material) in cryptographic modules that give a high level of assurance that the key material is accessible only to authorized users.

When configuring key management in the cloud, the importance of configuring access to KEKs is paramount, because anybody with access to KEKs, can in principle access all data that has been encrypted with a DEK that is protected with these KEKs.

The use of Bring Your Own Key (BYOK) or Hold Your Own Key (HYOK) does not significantly change the risks because the data encryption keys need to be accessible to the cloud services that perform the cryptographic operations. The table below summarises the service encryption options, for data at rest encryption. The table specifies the scope of data encryption, the supported Key Encryption Key (KEK) types and supported Key management services per encryption option.

Table: Who knows what - trust model - managed service vs supported key management.

Service	**Key Encryption Key (KEK) type**	**Data encryption scope**	**Key Encryption Key (KEK) management**
Block storage: Azure managed disk			
Azure disk storage service encryption	PMK or CMK	Disk	Microsoft Key Store, Key Vault, Managed HSM
Azure disk storage service - infrastructure encryption	PMK	Disk	Microsoft Key Store
Azure Disk Encryption	CMK	Disk	Key Vault
Encryption at host	PMK or CMK	Disk	Microsoft Key Store, Key Vault, Managed HSM
Confidential disk encryption	PMK or CMK	Disk	Microsoft Key Store, Key Vault, Managed HSM
Object storage: Azure Blob Storage			
Azure Storage Service encryption	PMK or CMK	Storage account	Microsoft Key Store, Key Vault Managed HSM
Azure Storage Service - infrastructure encryption	PMK	Storage account	Microsoft Key Store
Encryption Scope	PMK or CMK	Container or Blob	Microsoft Key Store, Key Vault Managed HSM
Customer-provided keys	CPK	Blob	Customer key store
Client-side encryption	CPK	Blob	Customer key store
File shares: Azure Files			

Service	Key Encryption Key (KEK) type	Data encryption scope	Key Encryption Key (KEK) management
Azure Storage Service encryption	PMK or CMK	File Share	Microsoft Key Store, Key Vault Managed HSM
Azure Storage Service - infrastructure encryption	PMK	File share	Microsoft Key Store
Managed databases: Azure SQL			
Transparent Data Encryption (TDE)	PMK or CMK	SQL Server Instance or Logical server	Microsoft Key Store, Key Vault, Managed HSM
Always encrypted	CMK or CPK	Database column	Key Vault, Managed HSM, Customer key store
Always encrypted with secure enclaves	CMK or CPK	Database column	Windows Certificate Store, Key Vault
Managed databases: Azure Databases for PostgreSQL & MySQL			
Azure Storage Service encryption	PMK or CMK	Database	Microsoft Key Store, Key Vault, Managed HSM
Managed databases: Azure Databases for MariaDB			
Azure Storage Service encryption	PMK	Database	Microsoft Key Store

Service	Key Encryption Key (KEK) type	Data encryption scope	Key Encryption Key (KEK) management
Block storage: AWS EBS Volumes			
EBS Volume Encryption	AWS KMS (AWS managed, customer managed, external)	EBS Volume	AWS KMS / On-prem (XKS)
Instance Store (NVMe SSD)	AWS EC2 Managed on the server that hosts EC2 instances	Instance store (block volume)	AWS on the server that hosts the EC2 instances
Object Storage: AWS S3			
SSE-S3	AWS managed KMS	Object	AWS KMS
SSE-KMS	Customer managed KMS	Object	AWS KMS, On-prem (XKS)
SSE-KMS Bucket Key	AWS S3 Managed	Object	AWS S3
SSE-C	Customer Supplied	Object	On-prem
Managed Databases: RDS			
RDS	AWS KMS (AWS managed, customer managed, external)	RDS instance disk volumes	AWS KMS / On-prem (XKS)
File share: FSx for Windows File server			

Service	Key Encryption Key (KEK) type	Data encryption scope	Key Encryption Key (KEK) management
FSx for Windows at Rest	AWS KMS (AWS managed, customer managed, external)	File share instance	AWS KMS / On-prem (XKS)
FSx for Windows in Transit	FSx for Windows managed	File share instance	FSx for Windows (SMB)

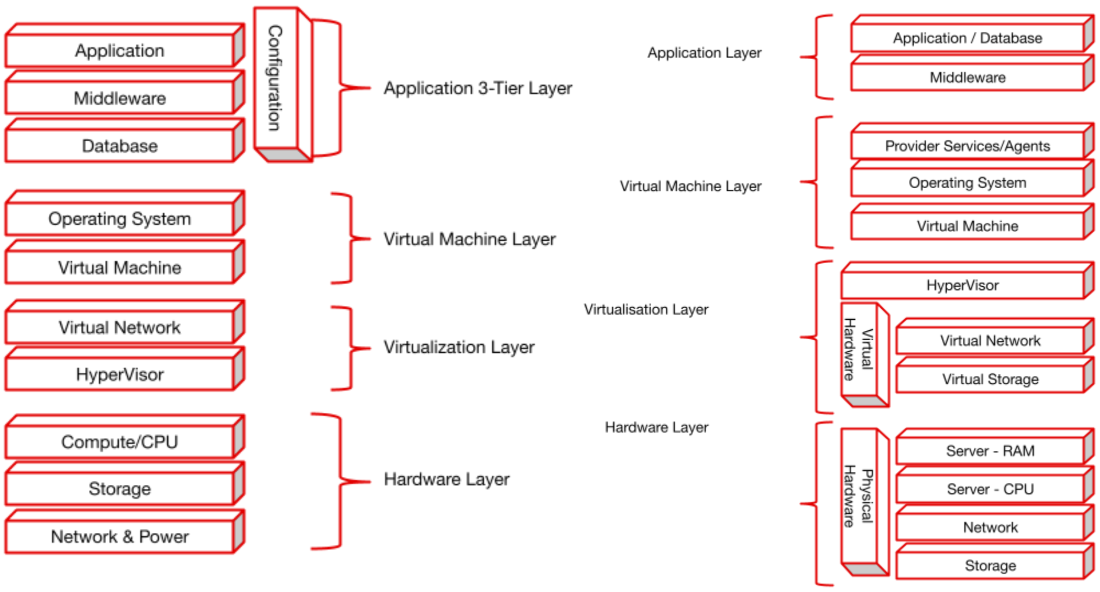
Over the course of the project we found that AWS and Azure launched a number of features related to encryption and key management. This emphasises the importance of cloud users in continuing to learn about evolving technology, assessing it, and applying it to their environment to improve their information protection. We found that on the topic of encryption of data at rest the AWS documentation is more clear and consistent than Azure documentation. AWS, for example, has defined more explicit and detailed security controls about cryptography in their security controls framework, which is reflected in the details found in the ISO 27001, SOC 2 and C5 reports.

Appendices

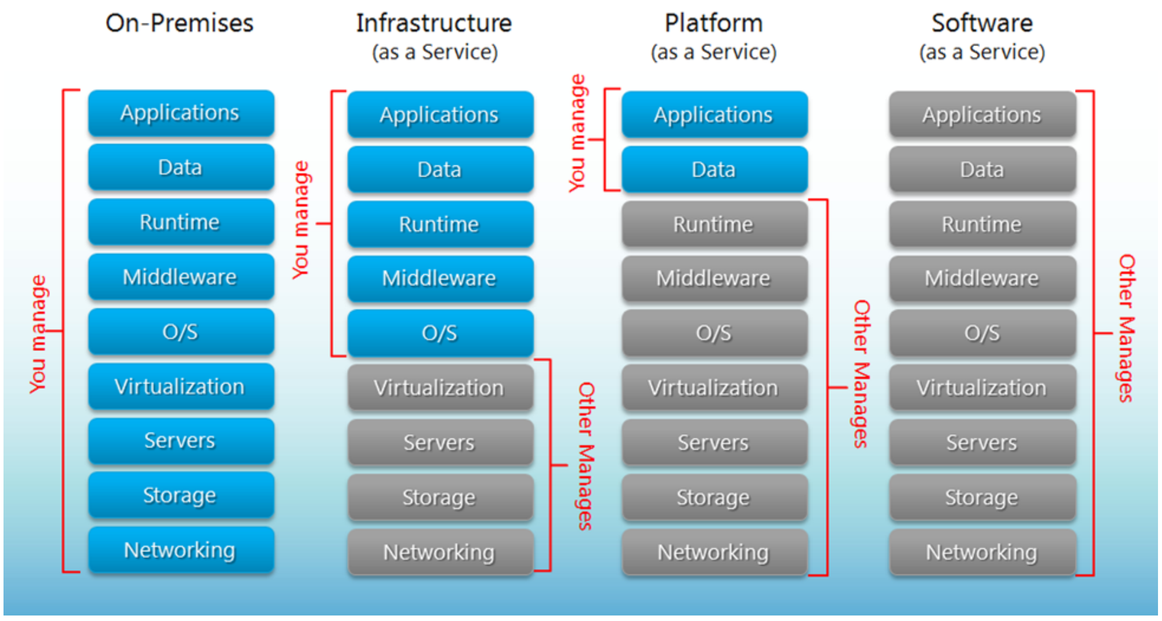
For certain topics we provide elaboration and some extra nuance in the appendix.
Below the list of available Apendices.

Appendix - Reference Cloud Stack

We use the following reference model when analysing the functionality of a Cloud Service Provider. Every service is an application that has a certain configuration and is running on a certain virtual and physical services. The reference model is based on the OSI Layer model.



Cloud Service Providers differentiates their services according the following “as a Service” model[277].



[277] Muratore, L., Lennox, B., & Tsagarakis, N. G. (2018, October). Xbotcloud: A scalable cloud computing infrastructure for xbot powered robots. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 1-9). IEEE. https://www.researchgate.net/publication/327700356

Appendix - Virtualization architecture

Cloud providers leverage server virtualization in the provision of many of their services, most prominently their VM services. They do this because it allows a single physical computer to run multiple operating system instances at the same time.

The architecture of the virtualization system used may impact security and performance of the services running atop. This appendix serves to explain the concepts at a high level, excluding many details.

Traditional virtualization

A virtualization system implements translation, emulation, and restriction functions that allow it to provide one or more virtualized operating systems (“guest VMs", "guests" or "instances") with virtual representations of their own self-contained hardware (“virtual machines” or “VMs”) on a single computer ("virtualization host").

One of the primary benefits of virtualization is the ability to make efficient use of a single powerful server by dividing its resources among multiple virtual machines each of which is allocated an amount of resources which is optimal for its assigned tasks.

The core component responsible for managing the lifecycle and operation of guest VMs in a virtualization system is called the virtual machine monitor (VMM), or hypervisor. A majority of the machine instructions performed by guest VMs run natively on the virtualization host's physical CPU(s) without any involvement of the hypervisor. For example, when a guest increments or decrements a value, it can communicate directly with the CPU hardware of the system to issue the machine code instructions.

There are some classes of sensitive or privileged instructions, such as reading or writing from CPU control registers, that a guest should not be allowed to run directly on the CPU hardware. Allowing guest VMs to run these instructions directly on the physical hardware would endanger the stability and isolation of the system as a whole. When a guest tries to issue sensitive or privileged instruction to the CPU, the instruction is redirected to the hypervisor. The hypervisor emulates a permitted result for the instruction, and then returns control back to the guest as if the instruction had been performed on the physical CPU directly.

A hypervisor itself is a relatively simple piece of software. However, a virtualization host requires more than the core functionality of a VMM to provide guest VMs with access to I/O devices such as network interfaces, storage drives, and input peripherals. To provide these features, hosts rely on additional software components called device models. Device models communicate with the system’s shared physical I/O hardware and emulate the state and behaviour of one or more unique virtual device interfaces exposed to guest VMs.

Hypervisors typically employ a general-purpose operating system to interface with a variety of system hardware, run device models, and run other management software for the virtualization system. This operating system is commonly implemented as a special privileged virtual machine. The Xen project calls this privileged VM the system's dom0, and Microsoft Hyper-V calls it the system's root partition or root VM.

AWS employed Xen as the hypervisor for EC2, and used Amazon Linux as the operating system for dom0. AWS has moved away from using a hypervisor system based on Xen with the introduction of the AWS Nitro system. Microsoft Azure uses the traditional architecture as described here[278].

We continue in the next section with a brief discussion of the security benefits of the AWS Nitro architecture.

The AWS Nitro system

The AWS Nitro system and its security design are described in an AWS whitepaper[279]. This appendix summarises the key components; the reader is referred to the aforementioned whitepaper for a detailed description.

The Nitro system is a combination of purpose-built server designs, data processors, system management components, and specialised firmware that jointly provide the underlying platform for all Amazon EC2 instances launched since the beginning of 2018. There are three key components to the Nitro system:

- Nitro cards - dedicated data processors designed, built and tested by an Amazon subsidiary that allow AWS to move key virtualization functionality off the EC2 hypervisor hosts that run EC2 instances. Nitro cards for storage and network I/O perform crypto operations. the data encryption keys used for data-at-rest encryption reside in the Nitro storage card, not in the hypervisor hosts's memory. As a result, the risk of encryption keys being compromised as a result of a hypervisor vulnerability are minimised.
- The Nitro system provides a hardware-based root of trust using the Nitro security chip, allowing AWS to cryptographically measure and validate the system starting from the boot process.
- The Nitro hypervisor is a lightweight hypervisor that manages memory and CPU allocation to VMs on the host. The design does not require a dom0 or root partition to mediate VMs accessing I/O devices.

A security benefit of the Nitro architecture is that no AWS operators need to access the virtualization host. There is no mechanism for any system or person to log in to EC2 Nitro hosts, access the memory of EC2 instances, or access any customer data stored on local encrypted instance storage or remote encrypted EBS volumes. If any AWS operator, including those with the highest privileges, needs to do maintenance work on an EC2 server, they can only use a limited set of authenticated, authorised, logged, and audited administrative APIs. None of these APIs provide an operator the ability to access customer data on the EC2 server. Because these are designed and tested technical restrictions built into the Nitro System itself, no AWS operator can bypass these controls and protections.

[278] Microsoft customers with access to the Microsoft Service Trust portal (https://servicetrust.microsoft.com/) can refer to the Azure system description in the Azure Dynamics 365 + Online Services SOC2 Type II + C5 + CSA Star Report for more details.
[279] https://docs.aws.amazon.com/whitepapers/latest/security-design-of-aws-nitro-system/security-design-of-aws-nitro-system.html.

The Nitro system design has been independently reviewed by the NCC group[280] (note that the implementation has not been reviewed as part of that research).

[280] https://research.nccgroup.com/2023/05/03/public-report-aws-nitro-system-api-security-claims/

Appendix - Confidential Compute

Confidential Compute is a technology that aims to provide confidentiality for data in the CPU and in the memory of a computer, by utilising a hardware-based, attested Trusted Execution Environment (TEE). The Confidential Computing Consortium[281] (CCC) is a project community at the linux foundation of hardware vendors, cloud providers and software developers.

When CPU's are part of a cloud service offering the application data goes through multiple abstraction layers such as the hypervisor. See image below from Kohlbrenner et al. (2020)[282] that illustrates the difference between hardware manufacturers and how to isolate the data in process to the Cloud Service Provider[283].

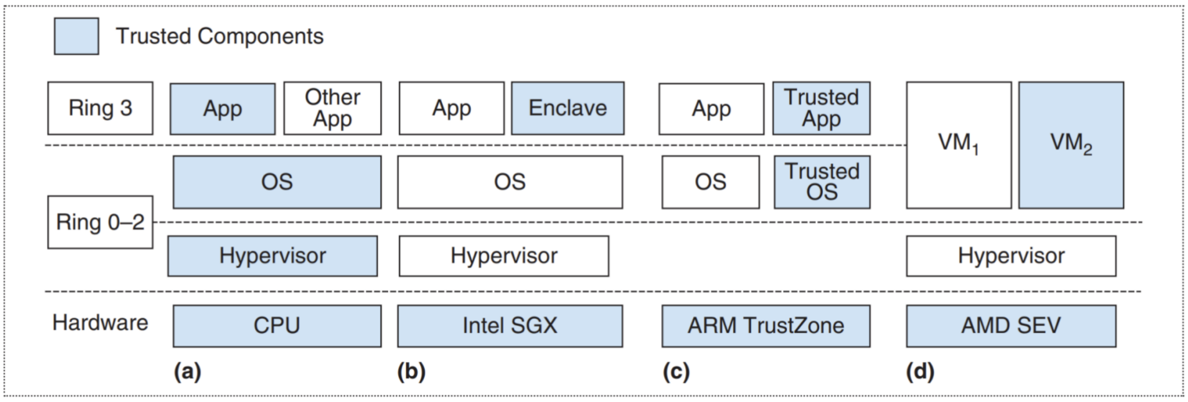


Figure 1. The schematic and trust stack for (a) a legacy stack, (b) Intel SGX, (c) ARM TrustZone, and (d) AMD SEV. The shaded components are trusted.

In the Intel solution the "sensitive" code needs to be decoupled from the regular code. The "sensitive" code then is executed (CPU, RAM) in a secure enclave. This solution requires software to be made compatible with this technology.

In the AMD solution the CPU and RAM are considered part of the enclave. Software does not have to be changed to run on this technology. Contents of the entire VM are protected from the underlying hypervisor.

The AMD solution is supported by Azure, GCP and AWS. The intel solution is available on Azure.

AWS has decided to take a different approach[284] to confidential compute in the form of AWS Nitro Enclaves service upon the Nitro-based EC2 instances[285]. The proposition of AWS Nitro Enclaves[286] is: "Enclaves are separate, hardened, and highly-constrained virtual machines". Nitro Enclaves do not share CPU cores or RAM with regular EC2 instances to improve confidentiality of the data processed in enclaves. The Nitro enclave is relatively new and therefore there is not yet much (security) research[287] available for providing insight in the possible vulnerabilities and threats mitigated. This is a topic for future research. AWS emphasises the fact that

[281] https://confidentialcomputing.io/
[282] Kohlbrenner, D., Shinde, S., Lee, D., Asanovic, K., and Song, D. (2020). Building Open Trusted Execution Environments. IEEE Security & Privacy, 18(5):47–56. Publisher: IEEE.
[283] https://github.com/binxio/gcp-conf-compute-analysis/releases/download/v0.0.0/main.pdf
[284] https://aws.amazon.com/blogs/security/confidential-computing-an-aws-perspective/
[285] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html#ec2-nitro-instances
[286] https://docs.aws.amazon.com/enclaves/latest/user/nitro-enclave.html
[287] https://scholar.google.com/scholar?q=aws+nitro+enclave on 2023-02-17 at 14:55 CET resulted in 79 results

the use of Nitro Enclaves aim to protect data against not just the cloud provider - the Nitro system already offers that protection - but also against the customer's own administrators or users[288].

Microsoft offers enclaves using Virtualization-based Security (VBS). VBS uses hardware virtualization and the Windows hypervisor to create an isolated, constrained virtual environments[289]. VBS enclaves are currently included in Azure services such as Azure SQL. Researching possible vulnerabilities and threats of VBS is a topic for future research.

[288] https://aws.amazon.com/ec2/nitro/nitro-enclaves/faqs/
[289] https://learn.microsoft.com/en-us/windows-hardware/design/device-experiences/oem-vbs

Appendix - Encryption Key LifeCycle

Cryptography

Cryptography is the practice and study of secure communication[290]. Communication is secure when a third party can not read the private messages that are exchanged and stored. Private messages can be sent to yourself, to one other party or to multiple other parties.

Concepts relevant to a private or secure message are[291]: Data confidentiality, data integrity, authentication and nonrepudiation. Confidentiality addresses if other people are allowed to read the information and based on which terms and conditions. Integrity addresses the means needed to change the information. Authentication addresses how to determine who is trying to access the information and how to provide the terms and policies needed to gain access. nonrepudiation addresses that all parties are explicitly knowledgeable about the success of sending, receiving and reading the information that is exchanged.

Cryptographic keys

Cryptographic keys are usually a string of characters (letters & numbers) that are being used to encrypt and decrypt data[292]. Encryption can be considered as a translation. The oldest example of this is to add 1 character shift to every letter in a word, so APE becomes BQF. BQF, in this example, has no meaning and is therefore worthless for whoever reads it without the key of +1, while APE has meaning and therefore has value.

It can be that you use the same key for both the encryption and the decryption, and it is possible that you use two separate keys. Who has access to the key(s) can gain access to the information. The whole design and implementation of the number of keys, which algorithm, which processes etc you use is called your cryptosystem[293].

Cryptographic key management

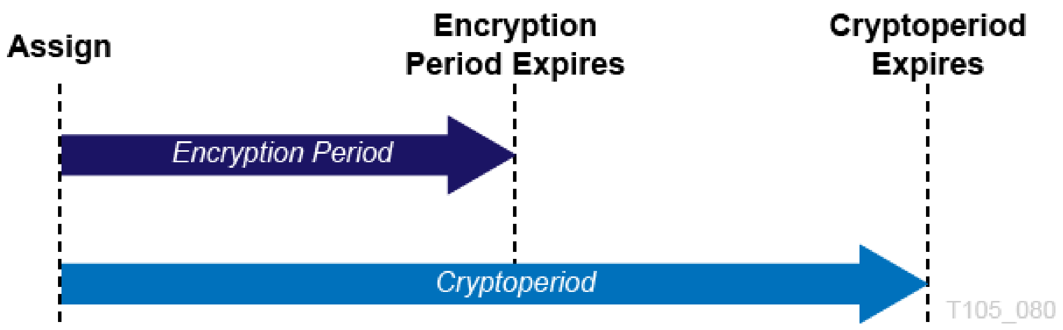
Cryptographic key management[294] refers to the management of the Creation, Access, Update and Deletion (CRUD) of keys. Access to keys is relevant to users (Personal Accounts) and systems (non-Personal Accounts). The creation, update and delete of certain keys is provided by a collection of software and/ or hardware.

It speaks for itself that the access to the key management system provides access to all the secrets, and focus point when accessing the strength of the encryption in the solution.

[290] https://en.wikipedia.org/wiki/Cryptography
[291] https://en.wikipedia.org/wiki/Information_security
[292] https://en.wikipedia.org/wiki/Key_(cryptography)
[293] https://en.wikipedia.org/wiki/Cryptosystem
[294] https://en.wikipedia.org/wiki/Key_management

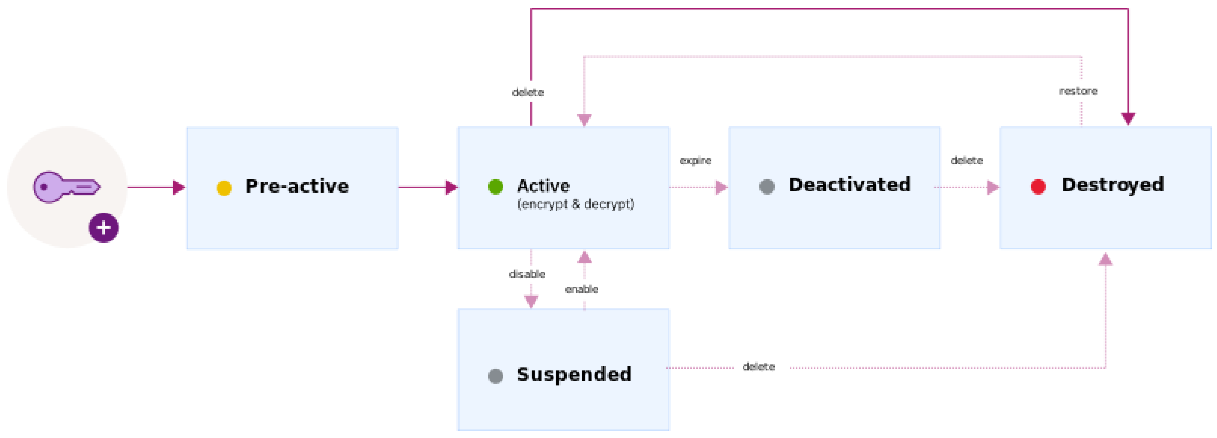
Key management life cycle periods

Below an image from Oracle[295] providing a view on the key management life cycle, and defining three periods which are important in the life cycle.



Key management life cycle states

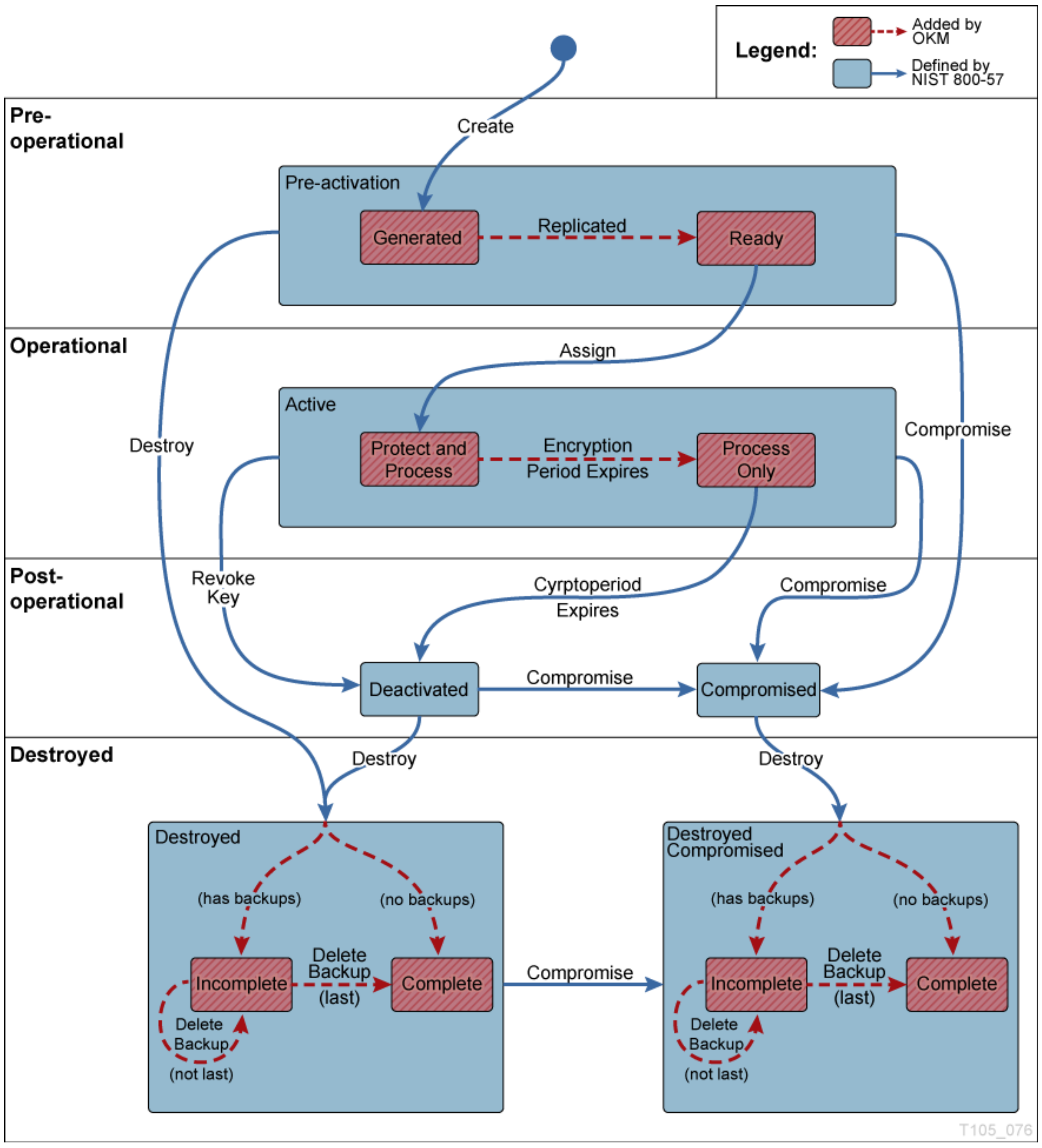
Below an image from IBM[296] providing a view on Key management life cycle and the states of the keys.



[295] https://docs.oracle.com/en/storage/storage-software/oracle-key-manager/3/okmag/key-lifecycles.htm
[296] https://cloud.ibm.com/docs/key-protect?topic=key-protect-key-states

Key management life cycle states

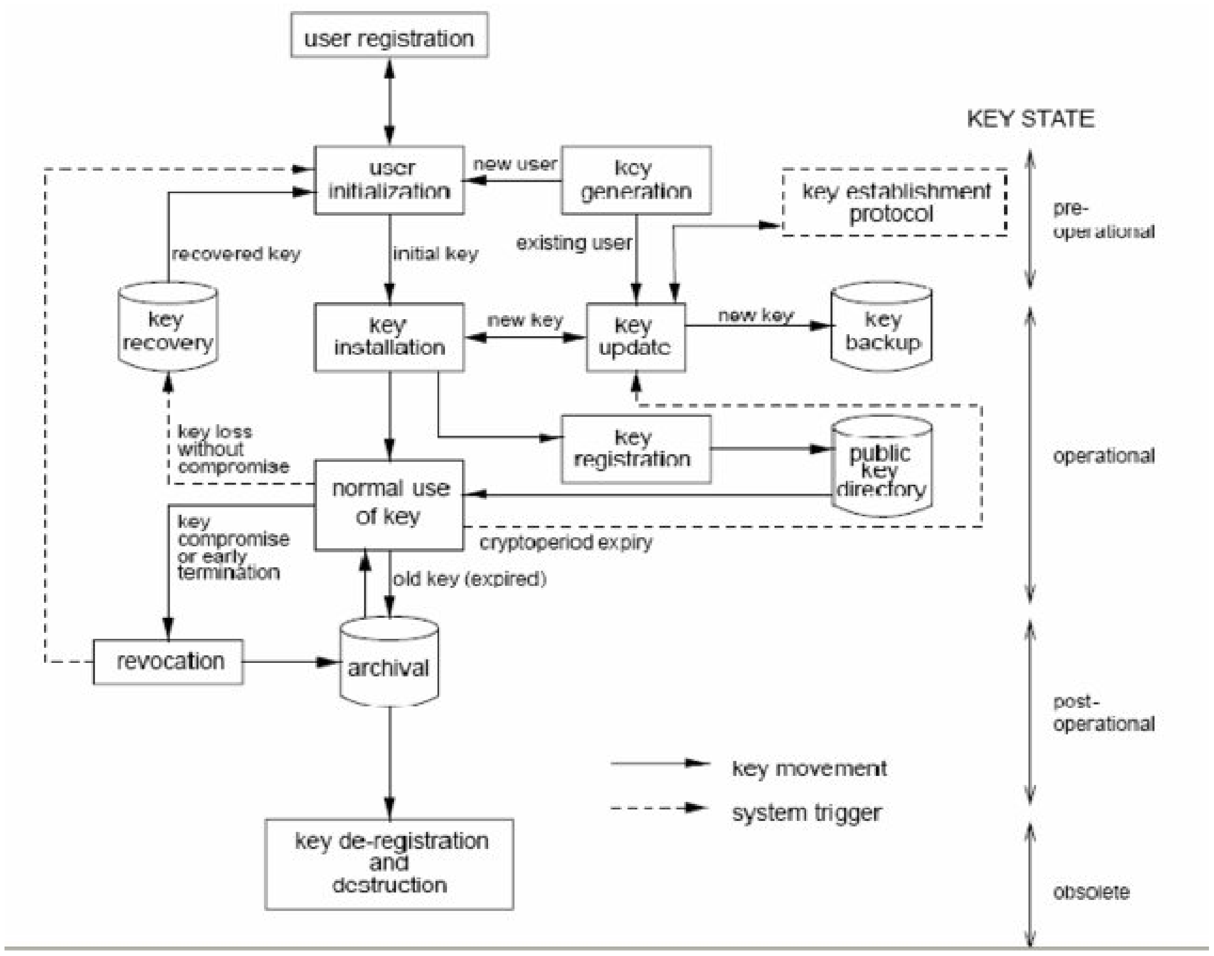
Below an Image from Oracle[297] with another view on the key management life cycle and the states of the keys.



[297] https://docs.oracle.com/en/storage/storage-software/oracle-key-manager/3/okmag/key-lifecycles.htm

Key management life cycle

image from Bardis, N., Doukas, N., & Ntaikos, K. (2008). A new approach to the secret key management lifecycle for military applications. Transactions on Computers, 7(12), 2011-2021.[298]



[298] https://www.researchgate.net/publication/342376506_A_New_Approach_of_Secret_Key_Management_Lifecycle_for_Military_Applications

Appendix - NIST FIPS 140 Standard

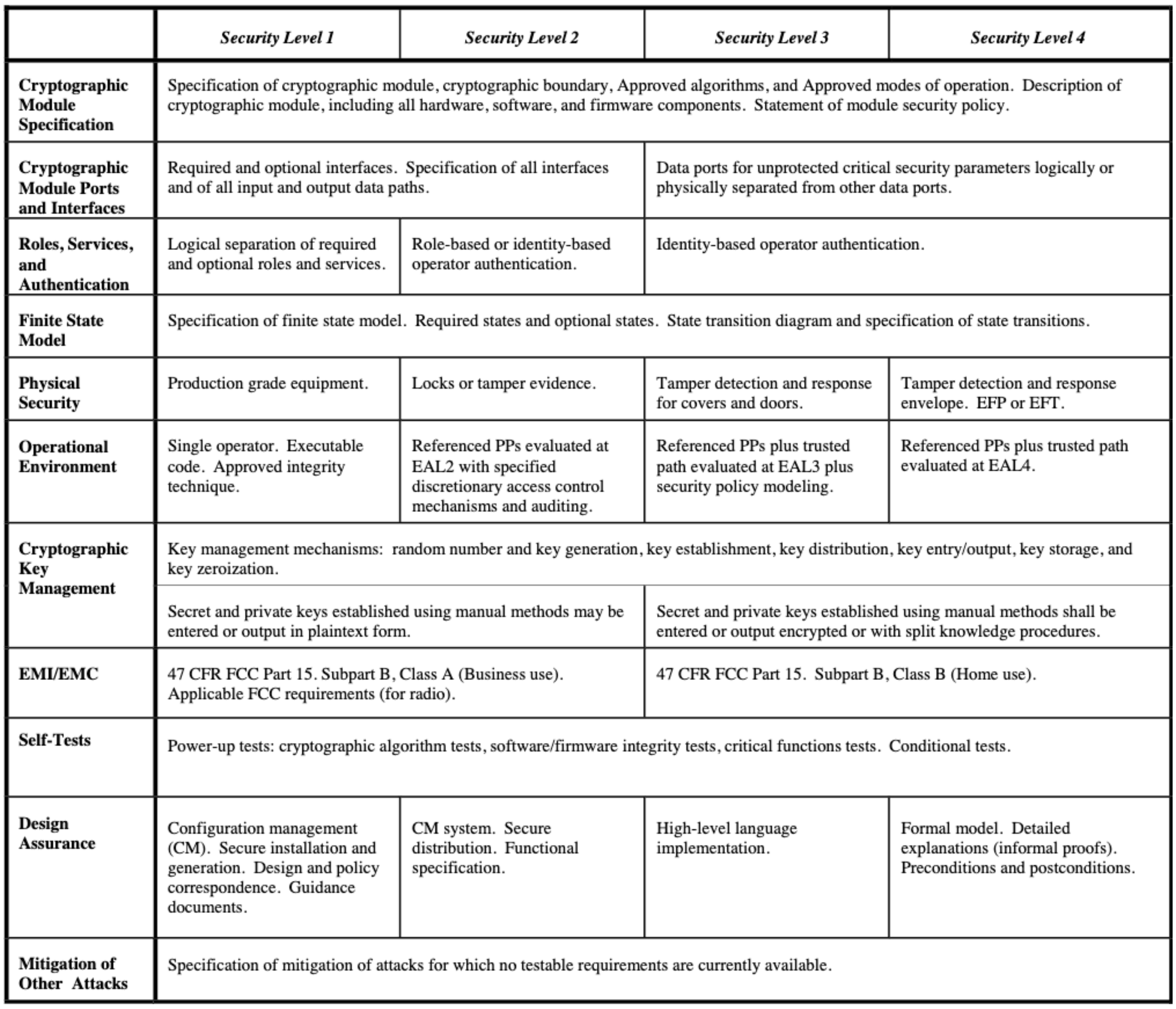
The Federal Information Process Standards[299] (FIPS) are computer security standards issued by the U.S. Government via the National Institute of standard and Technology (NIST). The NIST standards are publicly available[300].

The FIPS 140 computer security standard specifies requirements for cryptographic modules in hardware and software. These requirements address the build of the service and not the usage of the service. The result is that a FIPS 140 certified device can be considered secure in build and for example proves the presence of physical tamper evidence[301].

The FIPS 140 standard has 3 versions. FIPS 140-1, the oldest version, has been deprecated. FIPS 140-2 is still active, but no new cryptographic modules are accepted for validation. FIPS 140-3 is the latest version against which new cryptographic modules are validated. At the time of writing, no cryptographic modules have been validated against version 3 of the FIPS 140 standard.

The FIPS 140-2 standard distinguishes four levels of certification. The higher the level the higher the requirements on physical security and access controls. The table below provides an overview of the requirements for these levels.[302]

[299] https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.140-2.pdf
[300] https://csrc.nist.gov/publications
[301] https://en.wikipedia.org/wiki/FIPS_140
[302] https://csrc.nist.gov/csrc/media/publications/fips/140/2/final/documents/fips1402.pdf, p.12.



	Security Level 1	*Security Level 2*	*Security Level 3*	*Security Level 4*
Cryptographic Module Specification	Specification of cryptographic module, cryptographic boundary, Approved algorithms, and Approved modes of operation. Description of cryptographic module, including all hardware, software, and firmware components. Statement of module security policy.			
Cryptographic Module Ports and Interfaces	Required and optional interfaces. Specification of all interfaces and of all input and output data paths.		Data ports for unprotected critical security parameters logically or physically separated from other data ports.	
Roles, Services, and Authentication	Logical separation of required and optional roles and services.	Role-based or identity-based operator authentication.	Identity-based operator authentication.	
Finite State Model	Specification of finite state model. Required states and optional states. State transition diagram and specification of state transitions.			
Physical Security	Production grade equipment.	Locks or tamper evidence.	Tamper detection and response for covers and doors.	Tamper detection and response envelope. EFP or EFT.
Operational Environment	Single operator. Executable code. Approved integrity technique.	Referenced PPs evaluated at EAL2 with specified discretionary access control mechanisms and auditing.	Referenced PPs plus trusted path evaluated at EAL3 plus security policy modeling.	Referenced PPs plus trusted path evaluated at EAL4.
Cryptographic Key Management	Key management mechanisms: random number and key generation, key establishment, key distribution, key entry/output, key storage, and key zeroization.			
	Secret and private keys established using manual methods may be entered or output in plaintext form.		Secret and private keys established using manual methods shall be entered or output encrypted or with split knowledge procedures.	
EMI/EMC	47 CFR FCC Part 15. Subpart B, Class A (Business use). Applicable FCC requirements (for radio).		47 CFR FCC Part 15. Subpart B, Class B (Home use).	
Self-Tests	Power-up tests: cryptographic algorithm tests, software/firmware integrity tests, critical functions tests. Conditional tests.			
Design Assurance	Configuration management (CM). Secure installation and generation. Design and policy correspondence. Guidance documents.	CM system. Secure distribution. Functional specification.	High-level language implementation.	Formal model. Detailed explanations (informal proofs). Preconditions and postconditions.
Mitigation of Other Attacks	Specification of mitigation of attacks for which no testable requirements are currently available.			

Note that the requirements for Levels 1 and 2 allow for secret and private keys to be entered or output in plaintext form, while Levels 3 and 4 allow this only with a split knowledge procedure, meaning that malicious insider attacks require multiple actors to collude in order to be successful. In addition, Levels 3 and 4 require identity-based operator authentication, while at Level 2 role-based authentication suffices. At Level 1, the only requirement is that there be role separation.

On the NIST website[303] “Cryptographic Module Validation Program” you can find the FIPS 140 certification reports (e.g. Google Titan Security Chip[304], AWS Key Management Service HSM[305] and AWS Nitro Card Security Engine[306].

[303] https://csrc.nist.gov/Projects/cryptographic-module-validation-program/validated-modules
[304] https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp3382.pdf
[305] https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp4523.pdf
[306] https://csrc.nist.gov/CSRC/media/projects/cryptographic-module-validation-program/documents/security-policies/140sp3739.pdf