AWS Prescriptive Guidance

Encryption best practices and features for AWS services

AWS Prescriptive Guidance: Encryption best practices and features for AWS services

Table of Contents

Encryption best practices and features for AWS services

Kurt Kumar, Amazon Web Services (AWS)

December 2022 (document history (p. 18))

Modern cybersecurity threats include the risk of a data breach, which is when an authorized person gains access to your network and steals your enterprise data. Data is a business asset unique to each organization. It can include customer information, business plans, design documents, or code. Protecting the business means protecting its data.

Measures such as firewalls can help prevent a data breach from occurring. However, data encryption can help protect your business data even after a breach occurs. It provides another layer of defense against unintended disclosure. To access encrypted data in the AWS Cloud, users need permissions to use the key to decrypt and need permissions to use the service where the data resides. Without both of these permissions, users are unable to decrypt and view the data.

Generally, there are two types of data that you can encrypt. *Data in transit* is data that is actively moving through your network, such as between network resources. *Data at rest* is data that is stationary and dormant, such as data that is in storage. Examples include block storage, object storage, databases, archives, and Internet of Things (IoT) devices. This guide discusses considerations and best practices for encrypting both types of data. It also reviews the encryption features and controls available in many AWS services so that you can implement these encryption recommendations at the service-level in your AWS Cloud environments.

Intended audience

This guide can be used by small, medium, and large organizations in both public and private sectors. Whether your organization is in the initial stages of assessing and implementing a data protection strategy or aiming to enhance existing security controls, the recommendations outlined in this guide are best suited for the following audiences:

- Executive officers who formulate policies for their enterprise, such as chief executive officers (CEOs), chief technology officers (CTOs), chief information officers (CIOs), and chief information security officers (CISOs)
- Technology officers who are responsible for setting up technical standards, such as technical vice presidents and directors
- Business stakeholders and application owners who are responsible for:
  - Assessing risk posture, data classification and protection requirements
  - Monitoring compliance with established organizational standards
- Compliance, internal audit, and governance officers who are in charge of monitoring adherence to compliance policies, including statutory and voluntary compliance regimes

About AWS cryptography services

An *encryption algorithm* is a formula or procedure that converts a plaintext message into an encrypted ciphertext. If you are new to encryption or its terminology, we recommend that you read About data encryption and Cryptography concepts before proceeding with this guide.

AWS cryptography services rely on secure, open-source encryption algorithms. These algorithms are vetted by public standards bodies and by academic research. Some AWS tools and services enforce the use of a specific algorithm. In other services, you can choose between multiple available algorithms and key lengths, or you can use the recommended defaults.

This section describes some of the algorithms that AWS tools and services support. They fall into two categories, symmetric and asymmetric, based on how their keys function:

- *Symmetric* encryption uses the same key to encrypt and decrypt the data. AWS services support Advanced Encryption Standard (AES) and Triple Data Encryption Standard (3DES or TDES), which are two widely used symmetric algorithms. For more information, see Symmetric algorithms in the *AWS cryptographic services and tools guide*.
- *Asymmetric* encryption uses a pair of keys, a public key for encryption and a private key for decryption. You can share the public key because it isn't used for decryption, but access to the private key should be highly restricted. AWS services typically support RSA and elliptic-curve cryptography (ECC) asymmetric algorithms. For more information, see Asymmetric algorithms in the *AWS cryptographic services and tools guide*.

AWS cryptographic services comply with a wide range of cryptographic security standards, so you can comply with governmental or professional regulations. For a full list of the data security standards that AWS services comply with, see AWS compliance programs. For more information about cryptographic tools and services, see AWS cryptographic services and tools.

General encryption best practices

This section provides recommendations that apply when encrypting data in the AWS Cloud. These general encryption best practices are not specific to AWS services. This section includes the following topics:

Data classification

Data classification is a process for identifying and categorizing the data in your network based on its criticality and sensitivity. It is a critical component of any cybersecurity risk management strategy because it helps you determine the appropriate protection and retention controls for the data. Data classification is a component of the security pillar in the . Categories might include *highly confidential*, *confidential*, *non-confidential*, and *public*, but the classification tiers and their names can vary from organization to organization. For more information about the data classification process, considerations, and models, see Data classification (AWS Whitepaper).

After you have classified your data, you can create an encryption strategy for your organization based on the level of protection required for each category. For example, your organization might decide that highly confidential data should use asymmetric encryption and that public data doesn't require encryption. For more information about designing an encryption strategy, see Creating an enterprise encryption strategy for data at rest. Although the technical considerations and recommendations in that guide are specific to data at rest, you can use the phased approach to create an encryption strategy for data in transit as well.

Encryption of data in transit

All data transmitted between AWS Regions over the AWS global network is automatically encrypted at the physical layer before it leaves AWS secured facilities. All traffic between Availability Zones is encrypted.

The following are general best practices when encrypting data in transit in the AWS Cloud:

- Define an organizational encryption policy for data in transit, based on your data classification, organizational requirements, and any applicable regulatory or compliance standards. We strongly recommend that you encrypt data in transit that is classified as highly confidential or confidential. Your policy might also specify encryption for other categories, such as non-confidential or public data, on an as-needed basis.
- When encrypting data in transit, we recommend using approved cryptography algorithms, block cipher modes, and key lengths, as defined in your encryption policy.
- Encrypt traffic between information assets and systems within the corporate network and AWS Cloud infrastructure by using one of the following:
  - AWS Site-to-Site VPN connections
  - A combination of AWS Site-to-Site VPN and AWS Direct Connect connections, which provides an IPsec-encrypted private connection

  - AWS Direct Connect connections that support MAC Security (MACsec) to encrypt data from corporate networks to the Amazon VPC location
- Identify access control policies for your encryption keys based on the principle of least privilege. *Least privilege* is the security best practice of granting users the minimum access they need to perform their job functions. For more information about applying least-privilege permissions, see Security best practices in IAM and Best practices for IAM policies.

Encryption of data at rest

All AWS data storage services, such as Amazon Simple Storage Service (Amazon S3) and Amazon Elastic File System (Amazon EFS), provide options to encrypt data at rest. Encryption is performed by using the 256-bit Advanced Encryption Standard (AES-256) block cipher and AWS cryptography services, such as AWS Key Management Service (AWS KMS) or AWS CloudHSM.

You can encrypt data using client-side encryption or server-side encryption, based on factors such as data classification, the need for end-to-end encryption, or technical limitations that prevent you from using end-to-end encryption:

- *Client-side encryption* is the act of encrypting data locally before the target application or service receives it. The AWS service receives encrypted data; it does not play a role in encrypting or decrypting it. For client-side encryption, you might use AWS KMS, the AWS Encryption SDK, or other third-party encryption tools or services.
- *Server-side encryption* is the act of encrypting data at its destination, by the application or service that receives it. For server-side encryption, you might use AWS KMS for encryption of the entire storage block. You can also use other third-party encryption tools or services, such as LUKS for encrypting a Linux file system at the operating system (OS) level.

The following are general best practices when encrypting data at rest in the AWS Cloud:

- Define an organizational encryption policy for data at rest, based on your data classification, organizational requirements, and any applicable regulatory or compliance standards. For more information, see Creating an enterprise encryption strategy for data at rest. We strongly recommend that you encrypt data at rest that is classified as highly confidential or confidential. Your policy might also specify encryption for other categories, such as non-confidential or public data, on an as-needed basis.
- When encrypting data at rest, we recommend using approved cryptography algorithms, block cipher modes, and key lengths.
- Identify access control policies for your encryption keys based on the principle of least privilege.

Encryption best practices for AWS services

This section includes best practices and recommendations for specific AWS services. This section discusses the following services:

AWS CloudTrail

AWS CloudTrail helps you audit the governance, compliance, and operational and risk of your AWS account.

Consider the following encryption best practices for this service:

- CloudTrail logs should be encrypted using a customer managed AWS KMS key. Choose a KMS key that is in the same region as the S3 bucket that receives your log files. For more information, see Updating a trail to use your KMS key.
- As an additional security layer, enable log file validation for trails. This helps you determine whether a log file was modified, deleted, or unchanged after CloudTrail delivered it. For instructions, see Enabling log file integrity validation for CloudTrail.
- Use interface VPC endpoints to enable CloudTrail to communicate with resources in other VPCs without traversing the public internet. For more information, see Using AWS CloudTrail with interface VPC endpoints.
- Add an `aws:SourceArn` condition key to the KMS key policy to ensure that CloudTrail uses the KMS key only for a specific trail or trails. For more information, see Configure AWS KMS key policies for CloudTrail.

- In AWS Config, implement the cloud-trail-encryption-enabled AWS managed rule to validate and enforce log file encryption.
- If CloudTrail is configured to send notifications through Amazon Simple Notification Service (Amazon SNS) topics, add an `aws:SourceArn` (or optionally `aws:SourceAccount`) condition key to the CloudTrail policy statement to prevent unauthorized account access to the SNS topic. For more information, see Amazon SNS topic policy for CloudTrail.
- If you are using AWS Organizations, create an organization trail that logs all events for the AWS accounts in that organization. This includes the management account and all member accounts in the organization. For more information, see Creating a trail for an organization.
- Create a trail that applies to all AWS Regions where you store corporate data, to record AWS account activity in those Regions. When AWS launches a new Region, CloudTrail automatically includes the new Region and logs events in that Region.

Amazon DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service that provides fast, predictable, and scalable performance. DynamoDB encryption at rest secures data in an encrypted table—including its primary key, local and global secondary indexes, streams, global tables, backups, and DynamoDB Accelerator (DAX) clusters whenever the data is stored in durable media.

In accordance with data classification requirements, data confidentiality and integrity can be maintained by implementing server-side or client-side encryption:

For server-side encryption, when you create a new table, you can use AWS KMS keys to encrypt the table. You can use AWS owned keys, AWS managed keys, or customer managed keys. We recommend using customer managed keys because your organization has full control of the key, and because when you use this key type, the table-level encryption key, the DynamoDB table, local and global secondary indexes, and streams are all encrypted with the same key. For more information about these key types, see Customer keys and AWS keys.

> **Note**
> You can switch between an AWS owned key, AWS managed key, and customer managed key at any given time.

For client-side encryption and end-to-end protection of data, both at rest and in transit, you can use the Amazon DynamoDB Encryption Client. In addition to encryption, which protects the confidentiality of the item attribute value, DynamoDB Encryption Client signs the item. This provides integrity protection by enabling detection of unauthorized changes to the item, including adding or deleting attributes, or substituting one encrypted value for another.

Consider the following encryption best practices for this service:

- Limit permissions to disable or schedule deletion of the key to only those who need to perform these tasks. These states prevent all users and the DynamoDB service from being able to encrypt or decrypt data and to perform read and write operations on the table.
- While DynamoDB encrypts data in transit by using HTTPS by default, additional security controls are recommended. You can use any of the following options:
  - AWS Site-to-Site VPN connection using IPsec for encryption.
  - AWS Direct Connect connection to establish a private connection.
  - AWS Direct Connect connection with AWS Site-to-Site VPN connection for an IPsec-encrypted private connection.
  - If access to DynamoDB is required only from within a virtual private cloud (VPC), you can use a VPC gateway endpoint and allow only resources in the VPC to access it. This prevents the traffic from traversing the public internet.

- If you are using VPC endpoints, restrict the endpoint policies and IAM policies associated with the endpoint to only authorized users, resources, and services. For more information, see Control access to DynamoDB endpoints by using IAM policies and Control access to services using endpoint policies.
- You can implement column-level data encryption at the application level for data that requires encryption, according to your encryption policy.
- Configure DAX clusters to encrypt data at rest, such as data in cache, configuration data, and log files, at the time of setting up the cluster. You can't enable encryption at rest on an existing cluster. This server-side encryption helps protect data from unauthorized access through the underlying storage. DAX encryption at rest automatically integrates with AWS KMS for managing the single-service default key that is used to encrypt the clusters. If a service default key doesn't exist when an encrypted DAX cluster is created, AWS KMS automatically creates a new AWS managed key. For more information, see DAX encryption at rest.

  Note
  Customer managed keys can't be used with DAX clusters.
- Configure DAX clusters to encrypt data in transit at the time of setting up the cluster. You can't enable encryption in transit on an existing cluster. DAX uses TLS to encrypt requests and responses between the application and the cluster, and it uses the cluster's x509 certificate to authenticate the identity of the cluster. For more information, see DAX encryption in transit.
- In AWS Config, implement the dax-encryption-enabled AWS managed rule to validate and maintain encryption of DAX clusters.

Amazon Elastic Compute Cloud and Amazon Elastic Block Store

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the AWS Cloud. You can launch as many virtual servers as you need and quickly scale them up or down. Amazon Elastic Block Store (Amazon EBS) provides block-level storage volumes for use with EC2 instances.

Consider the following encryption best practices for these services:

- Tag all EBS volumes with the appropriate data classification key and value. This helps you determine and implement the appropriate security and encryption requirements, according to your policy..
- According to your encryption policy and the technical feasibility, configure encryption for data in transit between EC2 instances or between EC2 instances and your on-premises network.
- Encrypt both the boot and data EBS volumes of an EC2 instance. An encrypted EBS volume protects the following data:
  - Data at rest inside the volume
  - All data moving between the volume and the instance
  - All snapshots created from the volume
  - All volumes created from those snapshots

For more information, see How EBS encryption works.

- Enable encryption by default for EBS volumes for your account in the current Region. This enforces encryption of any new EBS volumes and snapshot copies. It has no effect on existing EBS volumes or snapshots. For more information, see Encryption by default.
- Encrypt the instance store root volume for an Amazon EC2 instance. This helps you protect configuration files and data stored with the operating system. For more information, see How to protect data at rest with Amazon EC2 instance store encryption (AWS blog post)
- In AWS Config, implement the encrypted-volumes rule to automated checks that validate and enforce appropriate encryption configurations.

Amazon Elastic Container Registry

Amazon Elastic Container Registry (Amazon ECR) is a managed container image registry service that's secure, scalable, and reliable.

Amazon ECR stores images in Amazon S3 buckets that Amazon ECR manages. Each Amazon ECR repository has an encryption configuration, which is set when the repository is created. By default, Amazon ECR uses server-side encryption with Amazon S3-managed (SSE-S3) encryption keys. For more information, see Encryption at rest (Amazon ECR documentation).

Consider the following encryption best practices for this service:

- Instead of using the default server-side encryption with Amazon S3-managed (SSE-S3) encryption keys, use customer managed KMS keys stored in AWS KMS. This key type provides the most granular control options.

  Note
  The KMS key must exist in the same AWS Region as the repository.
- Do not revoke the grants that Amazon ECR creates by default when you provision a repository. This can affect functionality, such as accessing data, encrypting new images pushed to the repository, or decrypting them when they are pulled.
- Use AWS CloudTrail to record the requests that Amazon ECR sends to AWS KMS. The log entries contain an encryption context key to make them more easily identifiable.
- Configure Amazon ECR policies to control access from specific Amazon VPC endpoints or specific VPCs. Effectively, this isolates network access to a specific Amazon ECR resource, allowing access from only the specific VPC. By establishing a virtual private network (VPN) connection with an Amazon VPC endpoint, you can encrypt data in transit.
- Amazon ECR supports resource-based policies Using these policies, you can restrict access based on the source IP address or the specific AWS service.

Amazon Elastic Container Service

Amazon Elastic Container Service (Amazon ECS) is a fast and scalable container management service that helps you run, stop, and manage containers on a cluster.

With Amazon ECS, you can encrypt data in transit by using any of the following approaches:

- Create a service mesh. Using AWS App Mesh, configure TLS connections between the deployed Envoy proxies and mesh endpoints, such as virtual nodes or virtual gateways. You can use TLS certificates from AWS Private Certificate Authority or customer-provided certificates. For more information and walkthroughs, see Enable traffic encryption between services in AWS App Mesh using AWS Certificate Manager (ACM) or customer-provided certificates (AWS blog post).
- If supported, use AWS Nitro Enclaves. AWS Nitro Enclaves is an Amazon EC2 feature that allows you to create isolated execution environments, called *enclaves*, from Amazon EC2 instances. They are designed to help protect your most sensitive data. Additionally, ACM for Nitro Enclaves allows you to use public and private SSL/TLS certificates with your web applications and web servers running on Amazon EC2 instances with AWS Nitro Enclaves. For more information, see AWS Nitro Enclaves – Isolated EC2 Environments to Process Confidential Data (AWS blog post).
- Use Server Name Indication (SNI) protocol with Application Load Balancer. You can deploy multiple applications behind a single HTTPS listener for an application load balancer. Each listener has its own TLS certificate. You can certificates provided by ACM, or you can use self-signed certificates. Both Application Load Balancer and Network Load Balancer support SNI. For more information, see

Application Load Balancers Now Support Multiple TLS Certificates with Smart Selection Using SNI (AWS blog post).

- For improved security and flexibility, use AWS Private Certificate Authority to deploy a TLS certificate with the Amazon ECS task. For more information, see Maintaining TLS all the way to your container part 2: Using AWS Private CA (AWS blog post).
- Implement mutual TLS (mTLS) in App Mesh by using Secret discovery service (Envoy) or certificates hosted in ACM (GitHub).

Consider the following encryption best practices for this service:

- Where technically feasible, for enhanced security, configure Amazon ECS interface VPC endpoints in AWS PrivateLink. Accessing these endpoints over a VPN connection encrypts data in transit.
- Store sensitive materials, such as API keys or database credentials, securely. You can store these as encrypted parameters in Parameter Store, a capability of AWS Systems Manager. However, we recommend you use AWS Secrets Manager because this service allows you to automatically rotate secrets, generate random secrets, and share secrets across AWS accounts:
- To help mitigate the risk of data leaks from environment variables, we recommend you use the AWS Secrets Manager and Config Provider for Secret Store CSI Driver (GitHub). This driver allows you to make secrets stored in Secrets Manager and parameters stored in Parameter Store appear as files mounted in Kubernetes pods.

  Note
  AWS Fargate is not supported.
- If users or applications in your data center or an external third party on the web are making direct HTTPS API requests to AWS services, sign those requests with temporary security credentials obtained from AWS Security Token Service (AWS STS).

Amazon Elastic File System

Amazon Elastic File System (Amazon EFS) helps you create and configure shared file systems in the AWS Cloud.

Consider the following encryption best practices for this service:

- In AWS Config, implement the efs-encrypted-check AWS managed rule. This rule checks if Amazon EFS is configured to encrypt the file data using AWS KMS.
- Enforce encryption for Amazon EFS file systems by creating an Amazon CloudWatch alarm that monitors CloudTrail logs for `CreateFileSystem` events and triggers an alarm if an unencrypted file system is created. For more information, see Walkthrough: Enforcing Encryption on an Amazon EFS File System at Rest.
- Mount the file system by using the EFS mount helper. This sets up and maintains a TLS 1.2 tunnel between the client and the Amazon EFS service and routes all Network File System (NFS) traffic over this encrypted tunnel. The following command implements the use of TLS for in-transit encryption.

```
sudo mount -t efs  -o tls file-system-id:/ /mnt/efs
```

  For more information, see Using EFS mount helper to mount EFS file systems.
- Using AWS PrivateLink, implement interface VPC endpoints to establish a private connection between VPCs and the Amazon EFS API. Data in transit over the VPN connection to and from the endpoint is encrypted. For more information, see Access an AWS service using an interface VPC endpoint.
- Use the `elasticfilesystem:Encrypted` condition key in IAM identity-based policies to prevent users from creating EFS file systems that aren't encrypted. For more information, see Using IAM to enforce creating encrypted file systems.

- KMS keys used for EFS encryption should be configured for least-privilege access by using resource-based key policies.
- Use the `aws:SecureTransport` condition key in the EFS file system policy to enforce use of TLS for NFS clients when connecting to an EFS file system. For more information, see Encryption of data in transit in *Encrypting File Data with Amazon Elastic File System* (AWS Whitepaper).

Amazon Elastic Kubernetes Service

Amazon Elastic Kubernetes Service (Amazon EKS) helps you run Kubernetes on AWS without needing to install or maintain your own Kubernetes control plane or nodes. In Kubernetes, *secrets* help you manage sensitive information, such as user certificates, passwords, or API keys. By default, these secrets are stored unencrypted in the API server's underlying data store, which is called etcd. Any user with API access or access to `etcd` can retrieve or modify a secret. Additionally, anyone who is authorized to create a pod in a namespace can use that access to read any secret in that namespace. You can encrypt these secrets at rest in Amazon EKS by using AWS KMS keys, either AWS managed keys or customer managed keys. An alternative approach to using `etcd` is using AWS Secrets and Config Provider (ASCP) (GitHub repository). ASCP integrates with IAM and resource-based policies to limit and restrict access to secrets only within specific Kubernetes pods inside a cluster.

You can use the following AWS storage services with Kubernetes:

- For Amazon Elastic Block Store (Amazon EBS), you can use the in-tree storage driver or the Amazon EBS CSI driver. Both include parameters for encrypting volumes and supplying a customer managed key.
- For Amazon Elastic File System (Amazon EFS), you can use the Amazon EFS CSI driver with support for both dynamic and static provisioning.

Consider the following encryption best practices for this service:

- For Amazon EFS, configure encryption in transit by adding the `tls` parameter to `mountOptions` in the Amazon EKS persistent volume. For more information, see Data encryption and secrets management (Amazon EKS Best Practices Guide).
- If you are using `etcd`, which stores secret objects unencrypted by default, do the following to help protect secrets:
  - Encrypt secret data at rest (Kubernetes documentation).
  - Enable or configure authorization through role-based access control (RBAC) rules that restrict reading and writing the secret. Restrict permissions to create new secrets or replace existing ones. For more information, see Authorization overview (Kubernetes documentation).
  - If you are defining multiple containers in a pod and only one of those containers needs access to a secret, define the volume mount so that the other containers do not have access to that secret. Secrets that are mounted as volumes are instantiated as `tmpfs` volumes and are automatically removed from the node when the pod is deleted. You could also use environment variables, but we don't recommend this approach because the values of environment variables can appear in logs. For more information, see Secrets (Kubernetes documentation).
  - When possible, avoid granting access to `watch` and `list` requests for secrets within a namespace. In the Kubernetes API, these requests are powerful because they allow the client to inspect the values of every secret in that namespace.
  - Allow only cluster administrators to access `etcd`, including read-only access.
  - If there are multiple `etcd` instances, ensure `etcd` is using TLS for communication between `etcd` peers.
- If you are using ASCP, do the following to help protect secrets:
  - Use IAM roles for service accounts to limit secret access to only authorized pods.

  - Enable encryption of Kubernetes secrets by using the AWS Encryption Provider (GitHub repository) to implement envelope encryption with a customer managed KMS key.
- Create an Amazon CloudWatch metrics filter and alarm to send alerts for administrator-specified operations, such as secret deletion or use of a secret version in the waiting period to be deleted. For more information, see Creating an alarm based on anomaly detection.

AWS Encryption SDK

The AWS Encryption SDK is an open-source, client-side encryption library. It uses industry standards and best practices to support implementation and interoperability in several programming languages. AWS Encryption SDK encrypts data by using a secure, authenticated, symmetric key algorithm and offers default implementation that adheres to cryptography best practices. For more information, see Supported algorithm suites in the AWS Encryption SDK.

Consider the following best practices for this service:

- Adhere to all of the recommendations in Best practices for the AWS Encryption SDK.
- Select one or more wrapping keys to help protect your data keys. For more information, see Select wrapping keys.
- Pass the `KeyId` parameter to the ReEncrypt operation to help prevent use of an untrusted KMS key. For more information, see Improved client-side encryption: Explicit KeyIds and key commitment (AWS blog post).
- When using AWS Encryption SDK with AWS KMS, use local `KeyId` filtering. For more information, see Improved client-side encryption: Explicit KeyIds and key commitment (AWS blog post).
- For applications with large volumes of traffic requiring encryption or decryption, or if your account is exceeding AWS KMS request quotas, you can use the data key caching feature of the AWS Encryption SDK. Note the following best practices for data key caching:
  - Configure cache security thresholds to limit how long each cached data key is used and how much data is protected under each data key. For recommendations when configuring these thresholds, see Setting cache security thresholds.
  - Limit the local cache to the smallest number of data keys necessary to achieve the performance improvements for your specific application use case. For instructions and an example of configuring limits for the local cache, see Using data key caching: Step-by-step.

  For more information, see AWS Encryption SDK: How to Decide if Data Key Caching Is Right for Your Application (AWS blog post).

AWS Key Management Service

AWS Key Management Service (AWS KMS) helps you create and control cryptographic keys to help protect your data. AWS KMS integrates with most other AWS services that can encrypt your data. For a complete list, see AWS services integrated with AWS KMS. AWS KMS also integrates with AWS CloudTrail to log use of your KMS keys for auditing, regulatory, and compliance needs.

KMS keys are the primary resource in AWS KMS, and they are logical representations of a cryptographic key. There are three primary types of KMS keys:

- Customer managed keys are KMS keys that you create.
- AWS managed keys are KMS keys that AWS services create in your account, on your behalf.
- AWS owned keys are KMS keys that an AWS service owns and manages, for use in multiple AWS accounts.

For more information about these key types, see Customer keys and AWS keys.

In the AWS Cloud, policies are used to control who can access resources and services. For example, in AWS Identity and Access Management (IAM), *identity-based policies* define permissions for users, user groups, or roles, and *resource-based policies* attach to a resource, such as an S3 bucket, and define which principals are allowed access, supported actions, and any other conditions that must be met. Similar to IAM policies, AWS KMS uses key policies to control access to a KMS key. Each KMS key must have a key policy, and each key can have only one key policy. Note the following when defining policies that allow or deny access to KMS keys:

- You can control the key policy for customer managed keys, but you can't directly control the key policy for AWS managed keys or for AWS owned keys.
- Key policies allow for granting granular access to AWS KMS API calls within an AWS account. Unless the key policy explicitly allows it, you cannot use IAM policies to allow access to a KMS key. Without permission from the key policy, IAM policies that allow permissions have no effect. For more information, see Allow IAM policies to allow access to the KMS key.
- You can use an IAM policy to deny access to a customer managed key without corresponding permission from the key policy.
- When designing key policies and IAM policies for multi-Region keys, consider the following:
  - Key policies are not shared properties of multi-Region keys and are not copied or synchronized among related multi-Region keys.
  - When a multi-Region key is created using the `CreateKey` and `ReplicateKey` actions, the default key policy is applied unless a key policy is specified in the request.
  - You can implement condition keys, such as aws:RequestedRegion, to limit permissions to a particular AWS Region.
  - You can use grants to allow permissions to a multi-Region primary key or replica key. However, a single grant cannot be used to allow permissions to multiple KMS keys, even if they are related multi-Region keys.

When using AWS KMS and creating key policies, consider the following encryption best practices and other security best practices:

- Adhere to the recommendations in the following resources for AWS KMS best practices:
  - AWS Key Management Service best practices (AWS Whitepaper)
  - Best practices for AWS KMS grants (AWS KMS documentation)
  - Best practices for IAM policies (AWS KMS documentation)
- In accordance with the separation of duties best practice, maintain separate identities for those who administer keys and those who use them:
  - Administrator roles that create and delete keys should not have the ability to use the key.
  - Some services may only need to encrypt data and should not be granted the ability to decrypt the data using the key.
- Key policies should always follow a model of least privilege. Do not use `kms:*` for actions in IAM or key policies because this gives the principal permissions to both administer and use the key.
- Limit the use of customer managed keys to specific AWS services by using the kms:ViaService condition key within the key policy.
- If you have a choice between key types, customer managed keys are preferred because they provide the most granular control options, including the following:
  - Managing authentication and access control
  - Enabling and disabling keys
  - Rotating AWS KMS keys
  - Tagging keys

  - Creating aliases
  - Deleting AWS KMS keys
- AWS KMS administrative and modification permissions must be explicitly denied to unapproved principals and AWS KMS modification permissions should not exist in an allow statement for any unauthorized principals. For more information, see Actions, resources, and condition keys for AWS Key Management Service.
- In order to detect unauthorized usage of KMS keys, in AWS Config, implement the iam-customer-policy-blocked-kms-actions and iam-inline-policy-blocked-kms-actions rules. This prevents principals from using the AWS KMS decryption actions on all resources.
- Implement service control policies (SCPs) in AWS Organizations to prevent unauthorized users or roles from deleting KMS keys, either directly as a command or through the console. For more information, see Using SCPs as preventative controls (AWS blog post).
- Log AWS KMS API calls in CloudTrail log. This records the relevant event attributes, such as what requests were made, the source IP address from which the request was made, and who made the request. For more information, see Logging AWS KMS API calls with AWS CloudTrail.
- If you use encryption context, it shouldn't contain any sensitive information. CloudTrail stores the encryption context in plaintext JSON files, which can be viewed by anyone with access to the S3 bucket containing the information.
- When monitoring usage of customer managed keys, configure events to notify you if specific actions are detected, such as key creation, updates to customer managed key policies, or import of key material are detected. It's also recommended that you implement automated responses, such as an AWS Lambda function that disables the key or performs any other incident response actions as dictated by your organizational policies.
- Multi-Region keys are recommended for specific scenarios, such as compliance, disaster recovery, or backups. The security properties of multi-Region keys are significantly different than single-Region keys. The following recommendations apply when authorizing the creation, management, and use of multi-Region keys:
  - Allow principals to replicate a multi-Region key only into AWS Regions that require them.
  - Give permission for multi-Region keys only to principals who need them and only for tasks that require them.

AWS Lambda

AWS Lambda is a compute service that helps you run code without needing to provision or manage servers. For securing your environment variables, you can use server-side encryption to protect your data at rest and client-side encryption to protect your data in transit.

Consider the following encryption best practices for this service:

- Lambda always provides server-side encryption at rest with an AWS KMS key. By default, Lambda uses an AWS managed key. We recommend you use a customer managed key because you have full control over the key, including management, rotation, and auditing.
- For data in transit that requires encryption, enable helpers, which ensures that environment variables are encrypted client-side for protection in transit by using the preferred KMS key. For more information, see *Security in transit* in Securing environment variables.
- Lambda function environment variables that hold sensitive or critical data should be encrypted in transit to help protect the data that is dynamically passed to the functions (usually access information) from unauthorized access.
- To prevent a user from viewing environment variables, add a statement to the user's permissions in the IAM policy or to the key policy that denies access to the default key, a customer managed key, or all keys. For more information, see Using AWS Lambda environment variables.

Amazon Relational Database Service

Amazon Relational Database Service (Amazon RDS) helps you set up, operate, and scale a relational database (DB) in the AWS Cloud. Data that is encrypted at rest includes the underlying storage for the DB instances, its automated backups, read replicas, and snapshots.

The following are the approaches you can use to encrypt data at rest in RDS DB instances:

- You can encrypt Amazon RDS DB instances with AWS KMS keys, either an AWS managed key or a customer managed key. For more information, see AWS Key Management Service (p. 11) in this guide.
- Amazon RDS for Oracle and Amazon RDS for SQL Server support encrypting DB instances with Transparent Data Encryption (TDE). For more information, see Oracle Transparent Data Encryption or Support for Transparent Data Encryption in SQL Server.

You can use both TDE and KMS keys to encrypt DB instances. However, this can slightly affect the performance of your database, and you must manage these keys separately.

The following are the approaches you can use to encrypt data in transit to or from RDS DB instances:

- For an Amazon RDS DB instance running MariaDB, Microsoft SQL Server, MySQL, Oracle, or PostgreSQL, you can use SSL to encrypt the connection. For more information, see Using SSL/TLS to encrypt a connection to a DB instance.
- Amazon RDS for Oracle also supports Oracle native network encryption (NNE), which encrypts data as it moves to and from a DB instance. NNE and SSL encryption cannot be used simultaneously. For more information, see Oracle native network encryption.

Consider the following encryption best practices for this service:

- When connecting to Amazon RDS for SQL Server or Amazon RDS for PostgreSQL DB instances in order to process, store, or transmit data that requires encryption, use the RDS Transport Encryption feature to encrypt the connection. You can implement this by setting the `rds.force_ssl` parameter to `1` in the parameter group. For more information, see Working with parameter groups. Amazon RDS for Oracle uses Oracle database native network encryption.
- Customer managed keys for RDS DB instance encryption should be used solely for that purpose and not used with any other AWS services.
- Before encrypting an RDS DB instance, establish KMS key requirements. The key used by the instance cannot be changed later. For example, in your encryption policy, define use and management standards for AWS managed keys or customer managed keys, based on your business requirements.
- It is strongly recommended that you enable backups for encrypted RDS DB instances. Amazon RDS can lose access to the KMS key for a DB instance, such as when the KMS key isn't enabled or when RDS access to a KMS key is revoked. If this occurs, the encrypted DB instance goes into a recoverable state for seven days. If the DB instance does not regain access to the key after seven days, the database becomes terminally inaccessible and must be restored from a backup. For more information, see Encrypting a DB instance.
- If a read replica and its encrypted DB instance are in the same AWS Region, you must use the same KMS key to encrypt both.
- In AWS Config, implement the rds-storage-encrypted AWS managed rule to validate and enforce encryption for RDS DB instances and the rds-snapshots-encrypted rule to validate and enforce encryption for RDS database snapshots.

AWS Secrets Manager

AWS Secrets Manager helps you replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically. Secrets Manager integrates with AWS KMS to encrypt every version of every secret value with a unique data key that is protected by an AWS KMS key. This integration protects stored secrets with encryption keys that never leave AWS KMS unencrypted. You can also define custom permissions on the KMS key to audit the operations that generate, encrypt, and decrypt the data keys that protect stored secrets. For more information, see Secret encryption and decryption in AWS Secrets Manager.

Consider the following encryption best practices for this service:

- In the key policy, use the kms:ViaService condition key to limit use the of the key to only requests from Secrets Manager by assigning the value `secretsmanager.<region>.amazonaws.com`.
- For additional security, based on business requirements, use keys or values in the Secrets Manager encryption context as a condition for using the KMS key by creating:
  - A string condition operator in an IAM or key policy
  - A grant constraint in a grant
- In AWS Config, implement the secretsmanager-using-cmk AWS managed rule to verify all secrets in Secrets Manager are encrypted with an AWS managed KMS key or a customer managed KMS key.
- To ensure secrets comply with defined rotation policies, implement the following AWS Config rules:
  - secretsmanager-rotation-enabled-check – Checks whether rotation is configured for secrets stored in Secrets Manager.
  - secretsmanager-scheduled-rotation-success-check – Checks whether secrets were successfully rotated. AWS Config also checks if the last rotated date falls within the configured rotation frequency.
  - secretsmanager-secret-periodic-rotation – Checks whether secrets were rotated within the specified number of days.
  - secretsmanager-secret-unused – Checks whether secrets were accessed within the specified number of days.
- Use AWS CloudTrail to record all API calls for Secrets Manager and non-API events, such as rotation start, rotation success, rotation failures, and scheduled deletion of secrets. For more information, see Logging AWS Secrets Manager events with AWS CloudTrail.
- Use Amazon CloudWatch Events to configure alerts for some Secrets Manager operations, such as deleting secrets, rotating secrets, or trying use a secret that is scheduled for deletion. You can choose which operations trigger an alert. The alert can be an SNS topic that sends an email or text message to subscribers, or it can be a Lambda function that logs the details of the operation for later review.

Amazon Simple Storage Service

Amazon Simple Storage Service (Amazon S3) is a cloud-based object storage service that helps you store, protect, and retrieve any amount of data.

For server-side encryption in Amazon S3, there are three options:

- Server-side encryption with Amazon S3-managed encryption keys (SSE-S3)
- Server-side encryption with AWS Key Management Service (SSE-KMS)
- Server-side encryption with customer-provided encryption keys (SSE-C)

If server-side encryption is used to encrypt an object at the time of upload, add the `x-amz-server-side-encryption` header to the request to tell Amazon S3 to encrypt the object using SSE-S3, SSE-

KMS, or SSE-C. The following are the possible values for the `x-amz-server-side-encryption` header:

- `AES256`, which tells Amazon S3 to use Amazon S3 managed keys.
- `aws:kms`, which tells Amazon S3 to use AWS KMS managed keys.
- Setting value as `True` or `False` for SSE-C

For more information, see *Defense-in-depth requirement 1: Data must be encrypted at rest and during transit* in How to Use Bucket Policies and Apply Defense-in-Depth to Help Secure Your Amazon S3 Data (AWS blog post).

For client-side encryption in Amazon S3, there are two options:

- A key stored in AWS KMS
- A key that is stored within the application

Consider the following encryption best practices for this service:

- In AWS Config, implement the s3-bucket-server-side-encryption-enabled AWS managed rule to validate and enforce S3 bucket encryption.
- Deploy an Amazon S3 bucket policy that validates that all objects being uploaded are encrypted using the `s3:x-amz-server-side-encryption` condition. For more information, see the example bucket policy in Protecting data using SSE-S3 and the instructions in Adding a bucket policy.
- Allow only encrypted connections over HTTPS (TLS) by using the `aws:SecureTransport` condition on S3 bucket policies. For more information, see What S3 bucket policy should I use to comply with the AWS Config rule s3-bucket-ssl-requests-only?
- In AWS Config, implement the s3-bucket-ssl-requests-only AWS managed rule to require requests to use SSL.
- Use a customer managed key when you need to grant cross-account access to Amazon S3 objects. Configure the key policy to allow access from another AWS account.

Amazon Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) helps you launch AWS resources into a virtual network that you've defined. This virtual network resembles a traditional network that you'd operate in your own data center, with the benefits of using the scalable infrastructure of AWS.

Consider the following encryption best practices for this service:

- Encrypt traffic between information assets and systems within the corporate network and VPCs by using one of the following:
  - AWS Site-to-Site VPN connections
  - A combination of AWS Site-to-Site VPN and AWS Direct Connect connections, which provides an IPsec-encrypted private connection
  - AWS Direct Connect connections that support MAC Security (MACsec) to encrypt data from corporate networks to the AWS Direct Connect location
- Use VPC endpoints in AWS PrivateLink to privately connect your VPCs to supported AWS services without using an internet gateway. You can use AWS Direct Connect or AWS VPN services to establish this connection Traffic between your VPC and the other service does not leave the AWS network. For more information, see Access AWS services through AWS PrivateLink.
- Configure security group rules that allow traffic only from ports associated with secure protocols, such as HTTPS over TCP/443. Periodically audit security groups and their rules.

Resources

- Creating an enterprise encryption strategy for data at rest
- Security best practices for AWS Key Management Service
- How AWS services use AWS KMS

Document history

The following table describes significant changes to this guide. If you want to be notified about future updates, you can subscribe to an RSS feed.

Change	Description	Date
Initial publication (p. 18)	—	December 2, 2022

AWS Prescriptive Guidance glossary

The following are commonly used terms in strategies, guides, and patterns provided by AWS Prescriptive Guidance. To suggest entries, please use the **Provide feedback** link at the end of the glossary.

Security terms

attribute-based access control (ABAC)

The practice of creating fine-grained permissions based on user attributes, such as department, job role, and team name. For more information, see ABAC for AWS in the AWS Identity and Access Management (IAM) documentation.

asymmetric encryption

An encryption algorithm that uses a pair of keys, a public key for encryption and a private key for decryption. You can share the public key because it isn't used for decryption, but access to the private key should be highly restricted.

behavior graph

A unified, interactive view of resource behavior and interactions over time. You can use a behavior graph with Amazon Detective to examine failed logon attempts, suspicious API calls, and similar actions. For more information, see Data in a behavior graph in the Detective documentation.

client-side encryption

Encryption of data locally, before the target AWS service receives it.

conformance pack

A collection of AWS Config rules and remediation actions that you can assemble to customize your compliance and security checks. You can deploy a conformance pack as a single entity in an AWS account and Region, or across an organization, by using a YAML template. For more information, see Conformance packs in the AWS Config documentation.

data at rest

Data that is stationary in your network, such as data that is in storage.

data classification

A process for identifying and categorizing the data in your network based on its criticality and sensitivity. It is a critical component of any cybersecurity risk management strategy because it helps you determine the appropriate protection and retention controls for the data. Data classification is a component of the security pillar in the AWS Well-Architected Framework. For more information, see Data classification.

data in transit

Data that is actively moving through your network, such as between network resources.

defense-in-depth

An information security approach in which a series of security mechanisms and controls are thoughtfully layered throughout a computer network to protect the confidentiality, integrity, and availability of the network and the data within. When you adopt this strategy on AWS, you add multiple controls at different layers of the AWS Organizations structure to help secure resources.

delegated administrator

In AWS Organizations, a compatible service can register an AWS member account to administer the organization's accounts and manage permissions for that service. This account is called the *delegated administrator* for that service. For more information and a list of compatible services, see Services that work with AWS Organizations in the AWS Organizations documentation.

detective control

A security control that is designed to detect, log, and alert after an event has occurred. These controls are a second line of defense, alerting you to security events that bypassed the preventative controls in place. For more information, see Detective controls in *Implementing security controls on AWS*.

encryption key

A cryptographic string of randomized bits that is generated by an encryption algorithm. Keys can vary in length, and each key is designed to be unpredictable and unique.

endpoint service

A service that you can host in a virtual private cloud (VPC) to share with other users. You can create an endpoint service with AWS PrivateLink and grant permissions to other AWS accounts or to IAM principals. These accounts or principals can connect to your endpoint service privately by creating interface VPC endpoints. For more information, see Create an endpoint service in the Amazon VPC documentation.

envelope encryption

The process of encrypting an encryption key with another encryption key. For more information, see Envelope encryption in the AWS Key Management Service (AWS KMS) documentation.

geographic restrictions (geo blocking)

In Amazon CloudFront, an option to prevent users in specific countries from accessing content distributions. You can use an allow list or block list to specify approved and banned countries. For more information, see Restricting the geographic distribution of your content in the CloudFront documentation.

guardrail

A high-level rule that helps govern resources, policies, and compliance across organizational units (OUs). *Preventive guardrails* enforce policies to ensure alignment to compliance standards. They are implemented by using service control policies and IAM permissions boundaries. *Detective guardrails* detect policy violations and compliance issues, and generate alerts for remediation. They are implemented by using AWS Config, AWS Security Hub, Amazon GuardDuty, AWS Trusted Advisor, Amazon Inspector, and custom AWS Lambda checks.

identity-based policy

A policy attached to one or more IAM principals that defines their permissions within the AWS Cloud environment.

inbound (ingress) VPC

In an AWS multi-account architecture, a VPC that accepts, inspects, and routes network connections from outside an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

inspection VPC

In an AWS multi-account architecture, a centralized VPC that manages inspections of network traffic between VPCs (in the same or different AWS Regions), the internet, and on-premises networks. The AWS Security Reference Architecture recommends setting up your Network account with inbound,

outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

least privilege

The security best practice of granting the minimum permissions required to perform a task. For more information, see Apply least-privilege permissions in the IAM documentation.

member account

All AWS accounts other than the management account that are part of an organization in AWS Organizations. An account can be a member of only one organization at a time.

organization trail

A trail that's created by AWS CloudTrail that logs all events for all AWS accounts in an organization in AWS Organizations. This trail is created in each AWS account that's part of the organization and tracks the activity in each account. For more information, see Creating a trail for an organization in the CloudTrail documentation.

outbound (egress) VPC

In an AWS multi-account architecture, a VPC that handles network connections that are initiated from within an application. The AWS Security Reference Architecture recommends setting up your Network account with inbound, outbound, and inspection VPCs to protect the two-way interface between your application and the broader internet.

origin access control (OAC)

In CloudFront, an enhanced option for restricting access to secure your Amazon Simple Storage Service (Amazon S3) content. OAC supports all S3 buckets in all AWS Regions, server-side encryption with AWS KMS (SSE-KMS), and dynamic `PUT` and `DELETE` requests to the S3 bucket.

origin access identity (OAI)

In CloudFront, an option for restricting access to secure your Amazon S3 content. When you use OAI, CloudFront creates a principal that Amazon S3 can authenticate with. Authenticated principals can access content in an S3 bucket only through a specific CloudFront distribution. See also OAC (p. 21), which provides more granular and enhanced access control.

permissions boundary

An IAM management policy that is attached to IAM principals to set the maximum permissions that the user or role can have. For more information, see Permissions boundaries in the IAM documentation.

policy

An object that can define permissions (see identity-based policy (p. 20)), specify access conditions (see resource-based policy (p. 22)), or define the maximum permissions for all accounts in an organization in AWS Organizations (see service control policy (p. 22)).

preventative control

A security control that is designed to prevent an event from occurring. These controls are a first line of defense to help prevent unauthorized access or unwanted changes to your network. For more information, see Preventative controls in *Implementing security controls on AWS*.

principal

An entity in AWS that can perform actions and access resources. This entity is typically a root user for an AWS account, an IAM role, or a user. For more information, see *Principal* in Roles terms and concepts in the IAM documentation.

ransomware

A malicious software that is designed to block access to a computer system or data until a payment is made.

resource-based policy

A policy attached to a resource, such as an Amazon S3 bucket, an endpoint, or an encryption key. This type of policy specifies which principals are allowed access, supported actions, and any other conditions that must be met.

responsive control

A security control that is designed to drive remediation of adverse events or deviations from your security baseline. For more information, see Responsive controls in *Implementing security controls on AWS*.

SAML 2.0

An open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log into the AWS Management Console or call the AWS API operations without you having to create user in IAM for everyone in your organization. For more information about SAML 2.0-based federation, see About SAML 2.0-based federation in the IAM documentation.

security control

A technical or administrative guardrail that prevents, detects, or reduces the ability of a threat actor to exploit a security vulnerability. There are three primary types of security controls: preventative (p. 21), detective (p. 20), and responsive (p. 22).

security hardening

The process of reducing the attack surface to make it more resistant to attacks. This can include actions such as removing resources that are no longer needed, implementing the security best practice of granting least privilege, or deactivating unnecessary features in configuration files.

security information and event management (SIEM) system

Tools and services that combine security information management (SIM) and security event management (SEM) systems. A SIEM system collects, monitors, and analyzes data from servers, networks, devices, and other sources to detect threats and security breaches, and to generate alerts.

server-side encryption

Encryption of data at its destination, by the AWS service that receives it.

service control policy (SCP)

A policy that provides centralized control over permissions for all accounts in an organization in AWS Organizations. SCPs define guardrails or set limits on actions that an administrator can delegate to users or roles. You can use SCPs as allow lists or deny lists, to specify which services or actions are permitted or prohibited. For more information, see Service control policies in the AWS Organizations documentation.

shared responsibility model

A model describing the responsibility you share with AWS for cloud security and compliance. AWS is responsible for security *of* the cloud, whereas you are responsible for security *in* the cloud. For more information, see Shared responsibility model.

symmetric encryption

An encryption algorithm that uses the same key to encrypt and decrypt the data.

trusted access

Granting permissions to a service that you specify to perform tasks in your organization in AWS Organizations and in its accounts on your behalf. The trusted service creates a service-linked role in each account, when that role is needed, to perform management tasks for you. For more information, see Using AWS Organizations with other AWS services in the AWS Organizations documentation.

workload

A collection of resources and code that delivers business value, such as a customer-facing application or backend process.