AWS S3 Data Protection

# 

# AWS S3 Data Protection

* Amazon S3 provides a S3 data protection using highly **durable** storage infrastructure designed for mission-critical and primary data storage.
* Objects are redundantly stored on multiple devices across multiple facilities in an S3 region.
* S3 PUT and PUT Object copy operations synchronously store the data across multiple facilities before returning SUCCESS.
* Once the objects are stored, S3 maintains its durability by quickly detecting and repairing any lost redundancy.
* S3 also regularly verifies the **integrity** of data stored using checksums. If  S3 detects data corruption, it is repaired using redundant data.
* In addition, S3 calculates checksums on all network traffic to detect corruption of data packets when storing or retrieving data
* Data protection against accidental overwrites and deletions can be added by enabling Versioning to preserve, retrieve and restore every version of the object stored
* S3 also provides the ability to protect data in-transit (as it travels to and from S3) and at rest (while it is stored in S3)

## S3 Data Protection

### Data in-transit

* S3 allows protection of data in-transit by enabling communication via SSL or using client-side encryption

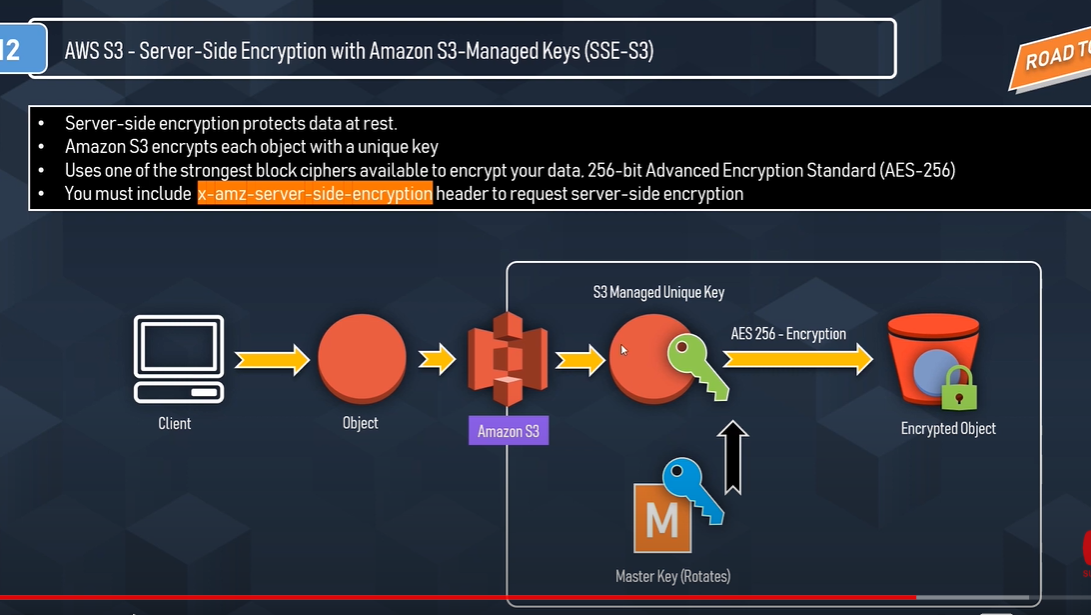
### Data at Rest

* S3 supports both client side encryption and server side encryption for protecting data at rest
* Using Server-Side Encryption, S3 encrypts the object before saving it on disks in its data centers and decrypt it when the objects are downloaded
* Using Client-Side Encryption, data is encrypted at client-side and uploaded to S3. In this case, the encryption process, the encryption keys, and related tools are managed by the user.

## Server-Side Encryption

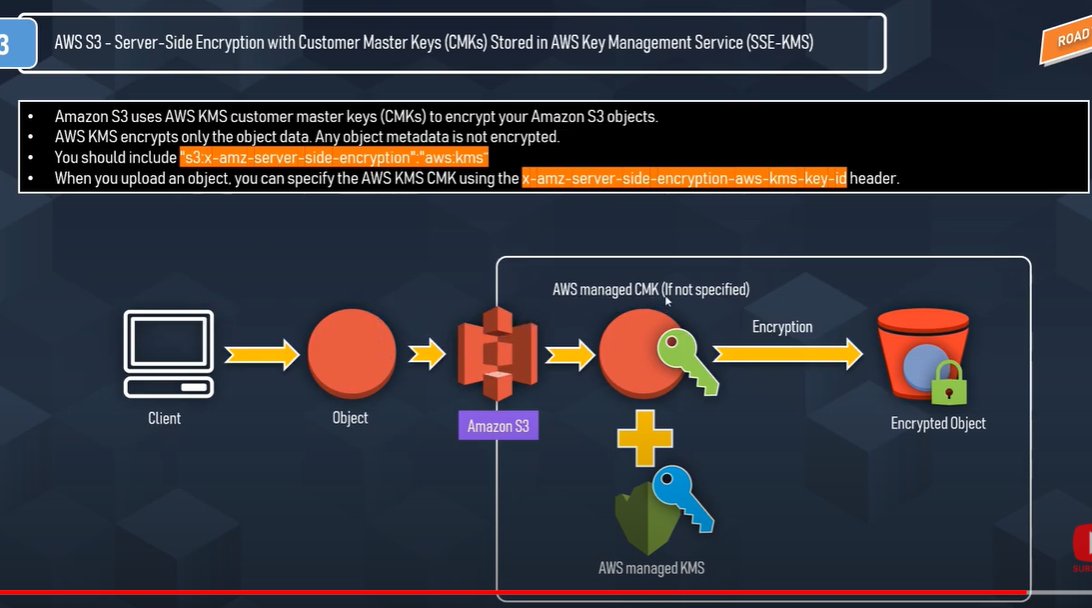
* Server-side encryption is about data encryption at rest
* Server-side encryption encrypts only the object data. Any object metadata is not encrypted.
* S3 handles the encryption (as it writes to disks) and decryption (when objects are accessed) of the data objects
* There is no difference in the access mechanism for both encrypted or unencrypted objects and is handled transparently by S3

### Server-Side Encryption with S3-Managed Keys (SSE-S3)



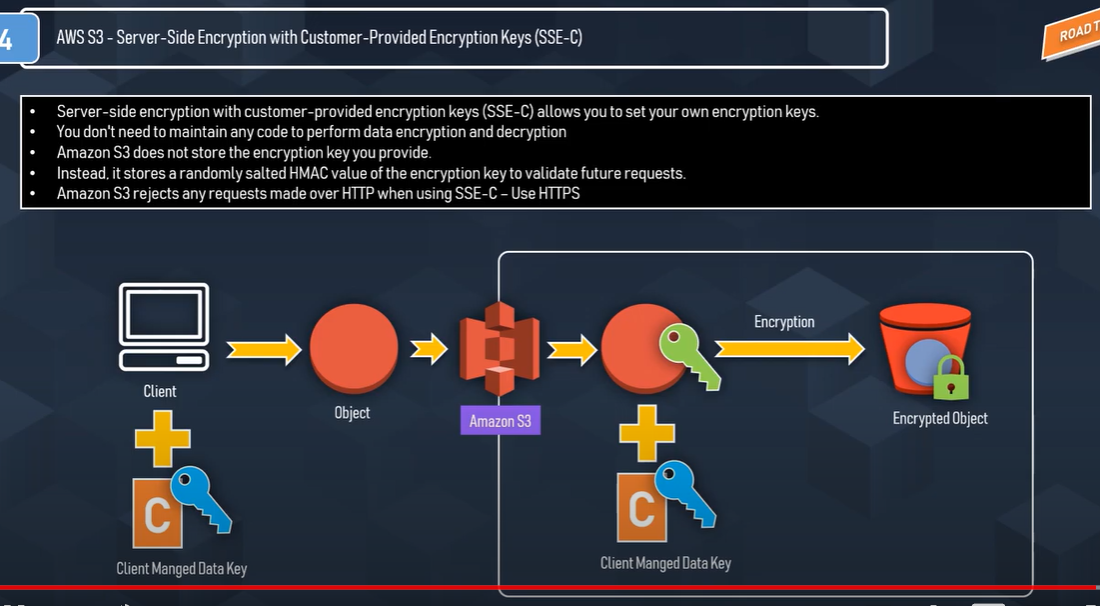
* Each object is encrypted with a unique data key employing strong multi-factor encryption.
* SSE-S3 encrypts the data key with a master key that is regularly rotated.
* S3 server-side encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt the data.
* Whether or not objects are encrypted with SSE-S3 can’t be enforced when they are uploaded using pre-signed URLs, because the only way server-side encryption can be specified is through the AWS Management Console or through an HTTP request header.

### Server-Side Encryption with AWS KMS-Managed Keys (SSE-KMS)



* SSE-KMS is similar to SSE-S3, but it uses AWS Key management Services (KMS) which provides additional benefits along with additional charges
  + KMS is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud.
  + KMS uses customer master keys (CMKs) to encrypt the S3 objects.
  + Master key is never made available
  + KMS enables you to centrally create encryption keys, define the policies that control how keys can be used
  + Allows audit use of key usage to prove they are being used correctly, by inspecting logs in AWS CloudTrail
  + Allows keys to temporarily disabled and re-enabled
  + Allows keys to be rotated regularly
  + Security controls in AWS KMS can help meet encryption-related compliance requirements.
* SSE-KMS enables separate permissions for the use of an envelope key (that is, a key that protects the data’s encryption key) that provides added protection against unauthorized access of the objects in S3.
* SSE-KMS provides the option to create and manage encryption keys yourself, or use a default customer master key (CMK) that is unique to you, the service you’re using, and the region you’re working in.
* Creating and Managing CMK gives more flexibility, including the ability to create, rotate, disable, and define access controls, and to audit the encryption keys used to protect the data.
* Data keys used to encrypt your data are also encrypted and stored alongside the data they protect and are unique to each object
* **Process flow**
  + An application or AWS service client requests an encryption key to encrypt data and passes a reference to a master key under the account.
  + Client requests are authenticated based on whether they have access to use the master key.
  + A new data encryption key is created, and a copy of it is encrypted under the master key.
  + Both the data key and encrypted data key are returned to the client.
  + Data key is used to encrypt customer data and then deleted as soon as is practical.
  + Encrypted data key is stored for later use and sent back to AWS KMS when the source data needs to be decrypted.

### Server-Side Encryption with Customer-Provided Keys (SSE-C)



* Encryption keys can be managed and provided by the Customer and S3 manages the encryption, as it writes to disks, and decryption, when you access the objects
* When you upload an object, the encryption key is provided as a part of the request and S3 uses that encryption key to apply AES-256 encryption to the data and removes the encryption key from memory.
* When you download an object, the same encryption key should be provided as a part of the request. S3 first verifies the encryption key and if matches decrypts the object before returning back to you
* As each object and each object’s version can be encrypted with a different key, you are responsible for maintaining the mapping between the object and the encryption key used.
* **SSE-C request must be done through HTTPS** **and S3 will reject any requests made over HTTP when using SSE-C.**
* For security considerations, AWS recommends to consider any key sent erroneously using HTTP to be compromised and discarded or rotated
* S3 does not store the encryption key provided. Instead, it stores a randomly salted HMAC value of the encryption key which can be used to validate future requests. The salted HMAC value cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object. That means, if you lose the encryption key, you lose the object.

## Client-Side Encryption

Client-side encryption refers to encrypting data before sending it to Amazon S3 and decrypting the data after downloading it

### AWS KMS-managed Customer Master Key (CMK)

* Customer can maintain the encryption CMK with AWS KMS and can provide the CMK id to the client to encrypt the data
* Uploading Object
  + AWS S3 encryption client first sends a request to AWS KMS for the key to encrypt the object data
  + AWS KMS returns a randomly generated data encryption key with 2 versions a plain text version for encrypting the data and cipher blob to be uploaded with the object as object metadata
  + Client obtains a unique data encryption key for each object it uploads.
  + AWS S3 encryption client uploads the encrypted data and the cipher blob with object metadata
* Download Object
  + AWS Client first downloads the encrypted object from S3 along with the cipher blob version of the data encryption key stored as object metadata
  + AWS Client then sends the cipher blob to AWS KMS to get the plain text version of the same, so that it can decrypt the object data.

### Client-Side master key

* Encryption master keys are completely maintained at Client-side
* Uploading Object
  + S3 encryption client ( for e.g. AmazonS3EncryptionClient *in the AWS SDK for Java*) locally generates randomly a one-time-use symmetric key (also known as a data encryption key or data key).
  + Client encrypts the data encryption key using the customer provided master key
  + Client uses this data encryption key to encrypt the data of a single S3 object (for each object, the client generates a separate data key).
  + Client then uploads the encrypted data to Amazon S3 and also saves the encrypted data key and its material description  as object metadata (x-amz-meta-x-amz-key) in Amazon S3 by default
* Downloading Object
  + Client first downloads the encrypted object from Amazon S3 along with the object metadata.
  + Using the material description in the metadata, the client first determines which master key to use to decrypt the encrypted data key.
  + Using that master key, the client decrypts the data key and uses it to decrypt the object
* Client-side master keys and your unencrypted data are never sent to AWS
* If the master key is lost the data cannot be decrypted

## AWS Certification Exam Practice Questions

1. A company is storing data on Amazon Simple Storage Service (S3). The company’s security policy mandates that data is encrypted at rest. Which of the following methods can achieve this? Choose 3 answers
   1. **Use Amazon S3 server-side encryption with AWS Key Management Service managed keys**
   2. **Use Amazon S3 server-side encryption with customer-provided keys**
   3. Use Amazon S3 server-side encryption with EC2 key pair.
   4. Use Amazon S3 bucket policies to restrict access to the data at rest.
   5. **Encrypt the data on the client-side before ingesting to Amazon S3 using their own master key**
   6. Use SSL to encrypt the data while in transit to Amazon S3.
2. A user has enabled versioning on an S3 bucket. The user is using server side encryption for data at Rest. If the user is supplying his own keys for encryption (SSE-C) which of the below mentioned statements is true?
   1. The user should use the same encryption key for all versions of the same object
   2. **It is possible to have different encryption keys for different versions of the same object**
   3. AWS S3 does not allow the user to upload his own keys for server side encryption
   4. The SSE-C does not work when versioning is enabled
3. A storage admin wants to encrypt all the objects stored in S3 using server side encryption. The user does not want to use the AES 256 encryption key provided by S3. How can the user achieve this?
   1. The admin should upload his secret key to the AWS console and let S3 decrypt the objects
   2. The admin should use CLI or API to upload the encryption key to the S3 bucket. When making a call to the S3 API mention the encryption key URL in each request
   3. S3 does not support client supplied encryption keys for server side encryption
   4. **The admin should send the keys and encryption algorithm with each API call**
4. A user has enabled versioning on an S3 bucket. The user is using server side encryption for data at rest. If the user is supplying his own keys for encryption (SSE-C), what is recommended to the user for the purpose of security?
   1. User should not use his own security key as it is not secure
   2. Configure S3 to rotate the user’s encryption key at regular intervals
   3. Configure S3 to store the user’s keys securely with SSL
   4. **Keep rotating the encryption key manually at the client side**
5. A system admin is planning to encrypt all objects being uploaded to S3 from an application. The system admin does not want to implement his own encryption algorithm; instead he is planning to use server side encryption by supplying his own key (SSE-C.. Which parameter is not required while making a call for SSE-C?
   1. **x-amz-server-side-encryption-customer-key-AES-256**
   2. x-amz-server-side-encryption-customer-key
   3. x-amz-server-side-encryption-customer-algorithm
   4. x-amz-server-side-encryption-customer-key-MD5
6. A customer is leveraging Amazon Simple Storage Service in eu-west-1 to store static content for a web-based property. The customer is storing objects using the Standard Storage class. Where are the customers objects replicated?
   1. A single facility in eu-west-1 and a single facility in eu-central-1
   2. A single facility in eu-west-1 and a single facility in us-east-1
   3. **Multiple facilities in eu-west-1**
   4. A single facility in eu-west-1
7. You are designing a personal document-archiving solution for your global enterprise with thousands of employee. Each employee has potentially gigabytes of data to be backed up in this archiving solution. The solution will be exposed to he employees as an application, where they can just drag and drop their files to the archiving system. Employees can retrieve their archives through a web interface. The corporate network has high bandwidth AWS DirectConnect connectivity to AWS. You have regulatory requirements that all data needs to be encrypted before being uploaded to the cloud. How do you implement this in a highly available and cost efficient way?
   1. Manage encryption keys on-premise in an encrypted relational database. Set up an on-premises server with sufficient storage to temporarily store files and then upload them to Amazon S3, providing a client-side master key. (Storing temporary increases cost and not a high availability option)
   2. Manage encryption keys in a Hardware Security Module(HSM) appliance on-premise server with sufficient storage to temporarily store, encrypt, and upload files directly into amazon Glacier. (Not cost effective)
   3. **Manage encryption keys in amazon Key Management Service (KMS), upload to amazon simple storage service (s3) with client-side encryption using a KMS customer master key ID and configure Amazon S3 lifecycle policies to store each object using the amazon glacier storage tier.** (with CSE-KMS the encryption happens at client side before the object is upload to S3 and KMS is cost effective as well)
   4. Manage encryption keys in an AWS CloudHSM appliance. Encrypt files prior to uploading on the employee desktop and then upload directly into amazon glacier (Not cost effective)
8. A user has enabled server side encryption with S3. The user downloads the encrypted object from S3. How can the user decrypt it?
   1. S3 does not support server side encryption
   2. S3 provides a server side key to decrypt the object
   3. The user needs to decrypt the object using their own private key
   4. **S3 manages encryption and decryption automatically**
9. When uploading an object, what request header can be explicitly specified in a request to Amazon S3 to encrypt object data when saved on the server side?
   1. x-amz-storage-class
   2. Content-MD5
   3. x-amz-security-token
   4. **x-amz-server-side-encryption**