

Amazon S3 Introduction

Notes by Christina Gryś

- Amazon S3 is one of the main building blocks of AWS
- It's advertised as "infinitely scaling" storage
- Many websites use Amazon S3 as a backbone
- Many AWS services use Amazon S3 as an integration as well

Amazon S3 - Buckets

- Amazon S3 allows people to store objects (files) in "buckets" (directories)
- Buckets must have a globally unique name
- Buckets are defined at the region level
 - **Naming convention:**
 - No uppercase
 - No underscore
 - 3-63 characters long
 - Not an IP
 - Must start with lowercase letter or number

Amazon S3 - Objects

- Objects (files) have a Key
- The **key** is the FULL path:
 - s3://my-bucket/my_file.txt
 - s3://my-bucket/my_folder/another_folder/my_file.txt
- The key is composed of **prefix** + **object name**
 - s3://my-bucket/my_folder/another_folder/my_file.txt
- There's no concept of "directories" within buckets
- (although the UI will trick you to think otherwise)
- Just keys with very long names that contain slashes ("/")
- Object values are the content of the body:
 - Max Object Size is 5TB (5000GB)
 - If uploading more than 5GB, must use "multi-part upload"
- Metadata (list of text key / value pairs – system or user metadata)
- Tags (Unicode key / value pair – up to 10) – useful for security / lifecycle
- Version ID (if versioning is enabled)

Amazon S3 – Versioning

- You can version your files in Amazon S3
- It is **enabled** at the bucket level
- Same key overwrite will increment the “version”: 1,2,3...
- It is best practice to version your buckets
 - Protect against unintended deletes (ability to restore a version)
 - Easy roll back to previous version
- Notes:
 - Any file that is not versioned prior to enabling versioning will have version “**null**”
 - **Suspending** versioning does not delete the previous versions.

S3 Encryption for Objects

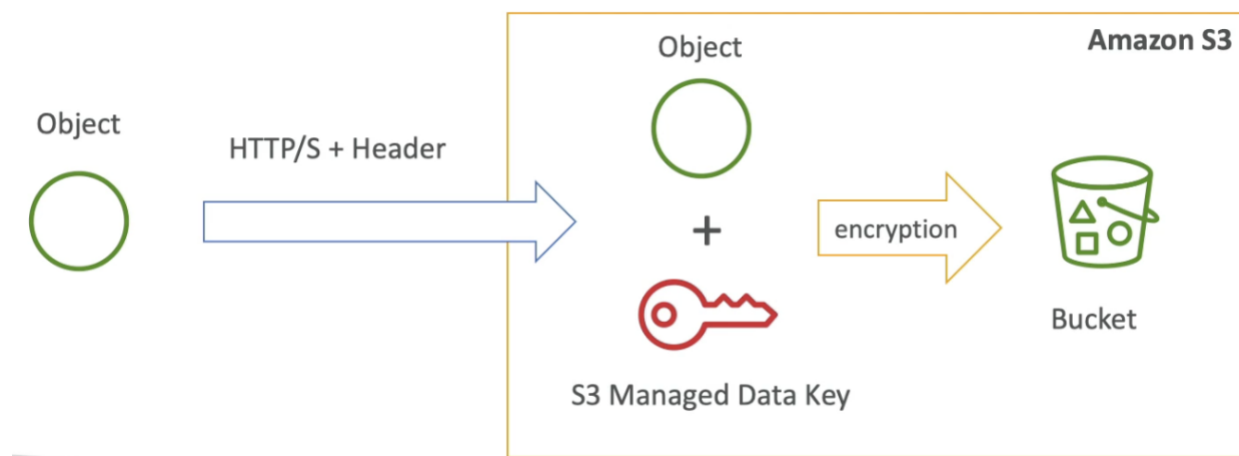
There are 4 methods of encrypting objects in S3

1. **SSE-S3**: encrypts S3 objects using keys handled & managed by AWS
2. **SSE-KMS**: leverage AWS Key Management Service to manage encryption keys
3. **SSE-C**: when you want to manage your own encryption keys
4. **Client Side Encryption**

It's important to understand which ones are adapted to which situation for the exam

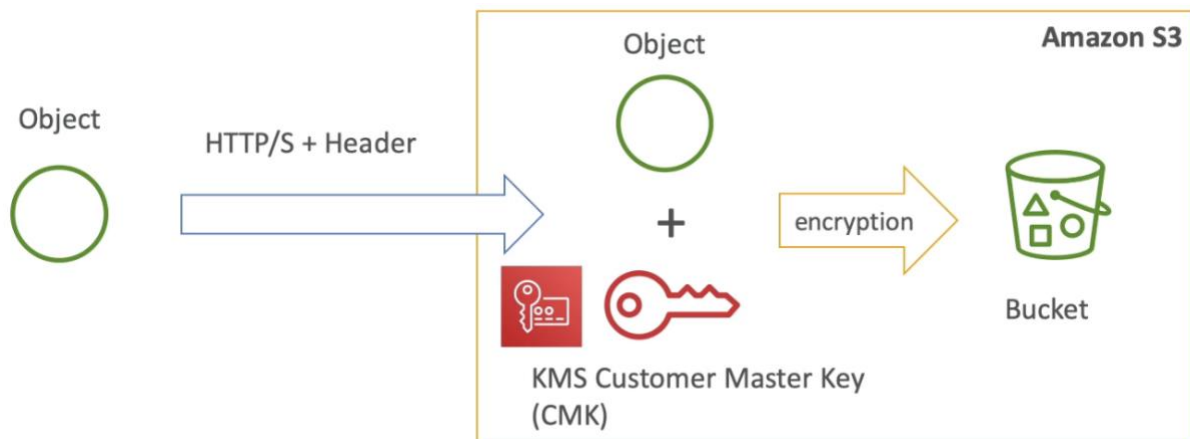
SSE-S3

- SSE-S3: encryption using keys handled & managed by Amazon S3
- Object is encrypted server side
- AES-256 encryption type
- Must set header: **“x-amz-server-side-encryption”:“AES256”**



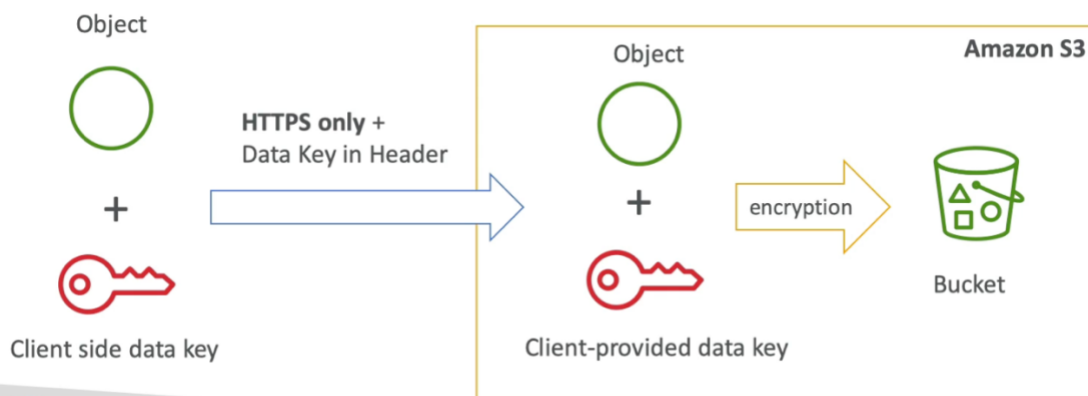
SSE-KMS

- SSE-KMS: encryption using keys handled & managed by KMS
- KMS Advantages: user control + audit trail
- Object is encrypted server side
- Must set header: **"x-amz-server-side-encryption": "aws:kms"**



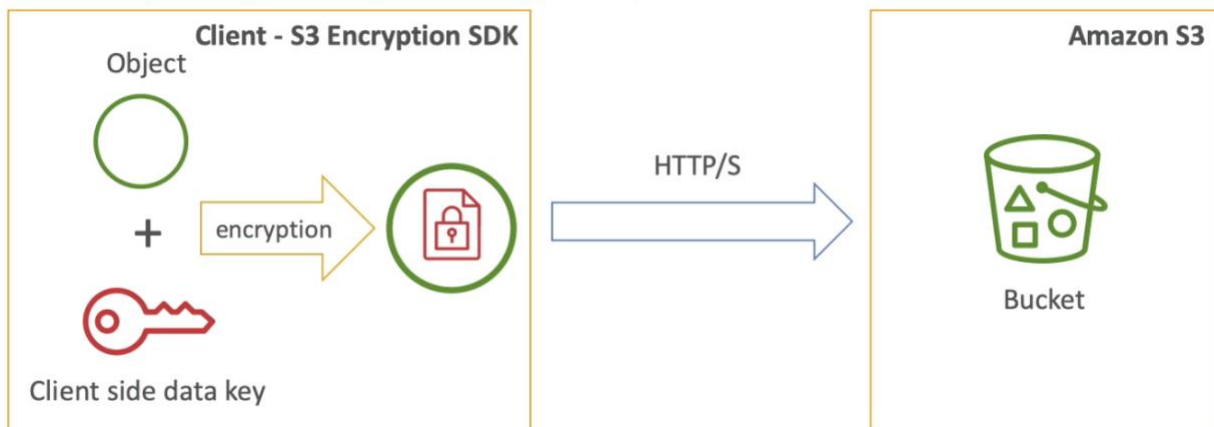
SSE-C

- SSE-C: server-side encryption using data keys fully managed by the customer outside of AWS
- Amazon S3 does not store the encryption key you provide
- HTTPS must be used
- Encryption key must provide in HTTP headers, for every HTTP request made



Client Side Encryption

- Client library such as the Amazon S3 Encryption Client
- Clients must encrypt data themselves before sending to S3
- Customer fully manages the keys and encryption cycle



Encryption in Transit (SSL/TLS)

- Amazon S3 exposes:
 - HTTP endpoint: non encrypted
 - HTTPS endpoint: encryption in flight
- You're free to use the endpoint you want, but HTTPS is recommended
- Most clients would use the HTTPS endpoint by default
- HTTPS is mandatory for SSE-C
- Encryption in flight is also called SSL / TLS

S3 Security

- **User based**
 - IAM policies – which API calls should be allowed for a specific user from IAM console
- **Resource Based**
 - Bucket Policies – bucket wide rules from the S3 console – allows cross account
 - Object Access Control List (ACL) – finer grain
 - Bucket Access Control List (ACL) – less common
- **Note:** an IAM principal can access an S3 object if
 - The user IAM permissions allow it OR the resource policy **ALLOWS** it
 - AND there's no explicit DENY

S3 - Bucket Policies

- JSON based policies

- Resources: buckets and objects
- Actions: Set of API to Allow or Deny
- Effect: Allow / Deny
- Principal: The account or user to apply the policy to:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicRead",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::examplebucket/*"
      ]
    }
  ]
}
```

- Use S3 bucket for policy to:
 - Grant public access to the bucket
 - Force objects to be encrypted at upload
 - Grant access to another account (Cross Account)

Bucket settings for Block Public Access

- Block public access to buckets and objects granted through
- *New* access control lists (ACLs)
- *Any* access control lists (ACLs)
- *New* public bucket or access point policies
- Block public and cross-account access to buckets and objects through any public bucket or access point policies
- **These settings were created to prevent company data leaks**
- If you know your bucket should never be public, leave these on
- Can be set at the account level

S3 Security – Other

- Networking:
 - Supports VPC Endpoints (for instances in VPC without www internet)
- Logging and Audit:

- S3 Access Logs can be stored in other S3 bucket
- API calls can be logged in AWS CloudTrail
- User Security:
 - MFA Delete: MFA (multi-factor authentication) can be required in versioned buckets to delete objects
 - Pre-Signed URLs: URLs that are valid only for a limited time (ex: premium video service for logged in users)

S3 Websites

- S3 can host static websites and have them accessible on the www
- The website URL will be:
 - <bucket-name>.s3-website-<AWS>.amazonaws.com
 - OR
 - <bucket-name>.s3-website.<AWS-region>.amazonaws.com
- If you get a 403 (Forbidden) error, make sure the bucket policy allows public reads!

S3 CORS

- An origin is a scheme (protocol), host (domain) and port
- E.g.: <https://www.example.com> (implied port is 443 for HTTPS, 80 for HTTP)
- CORS means Cross- Origin Resource Sharing
- Web Browser based mechanism to allow requests to other origins while visiting the main origin
- Same origin: <http://example.com/app1> & <http://example.com/app2>
- Different origins: <http://www.Example.com> & <http://other.example.com>
- The requests won't be fulfilled unless the other origin allows for the requests, using CORS Headers (ex: Access-Control-Allow-Origin)

CORS – Diagram



S3 CORS

If a client does a cross-origin request on our S3 bucket, we need to enable the correct CORS headers

It's a popular exam question

You can allow for a specific origin or for * (all origins)

