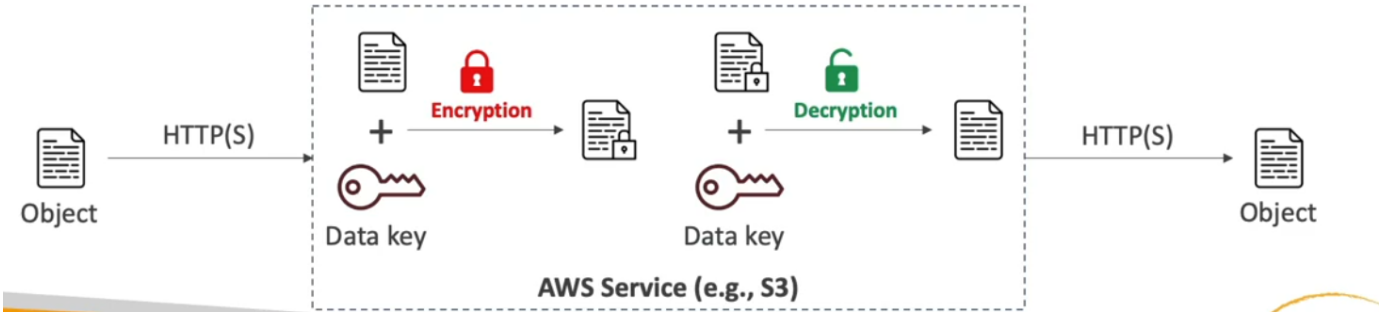
encryption

Why encryption?
Encryption in flight (TLS / SSL)

- Data is encrypted before sending and decrypted after receiving
- TLS certificates help with encryption (HTTPS)
- Encryption in flight ensures no MITM (man in the middle attack) can happen

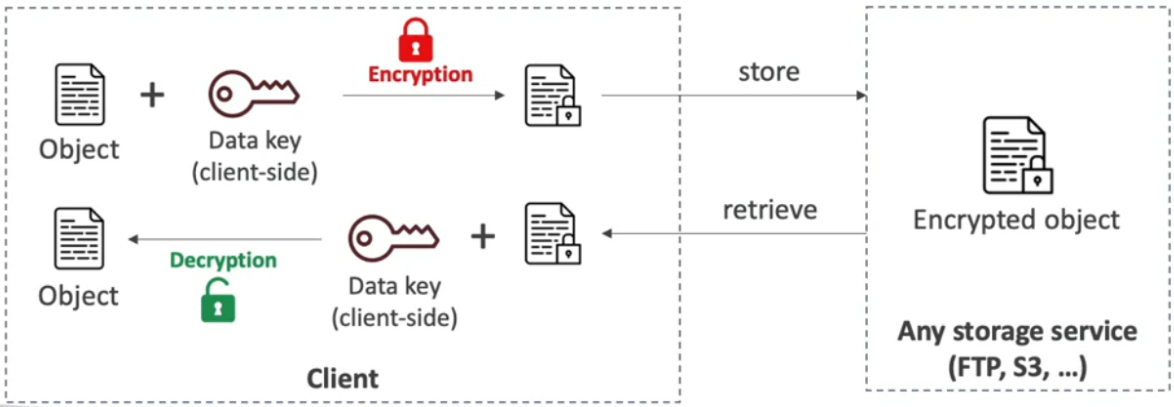
Why encryption?
Server-side encryption at rest

- Data is encrypted after being received by the server
- Data is decrypted before being sent
- It is stored in an encrypted form thanks to a key (usually a data key)
- The encryption / decryption keys must be managed somewhere, and the server must have access to it



Why encryption?
Client-side encryption

- Data is encrypted by the client and never decrypted by the server
- Data will be decrypted by a receiving client
- The server should not be able to decrypt the data
- Could leverage Envelope Encryption



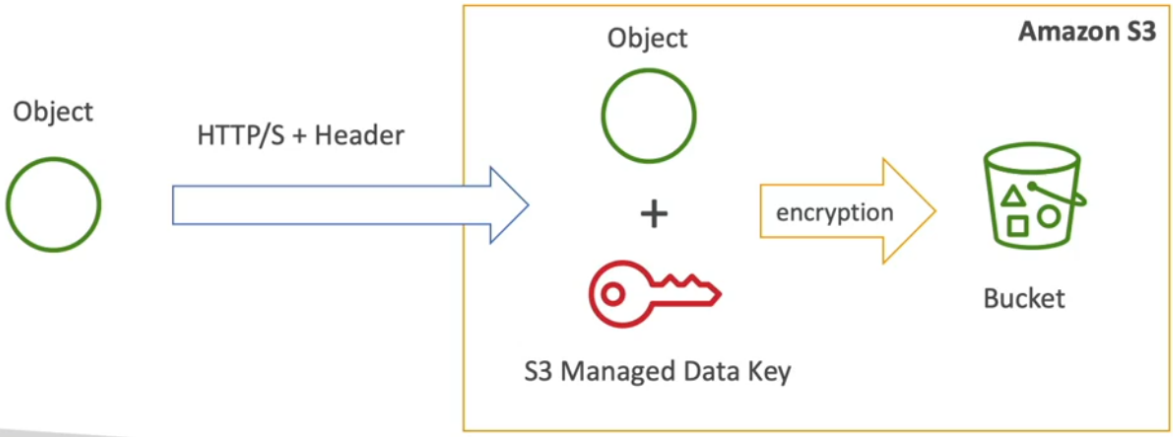
S3 Encryption for Objects

- There are 4 methods of encrypting objects in S3
  - SSE-S3: encrypts S3 objects using keys handled & managed by AWS
  - SSE-KMS: leverage AWS Key Management Service to manage encryption keys
  - SSE-C: when you want to manage your own encryption keys
  - Client Side Encryption

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

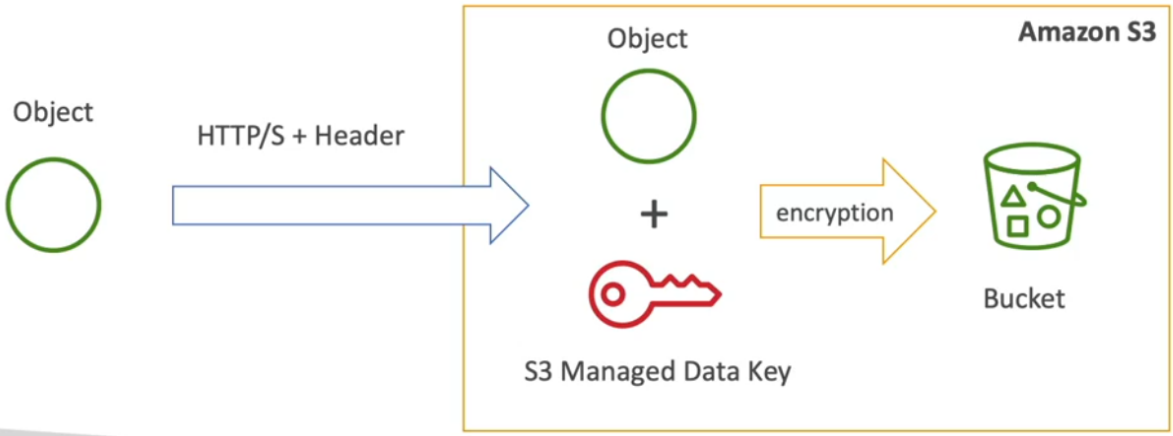
SSE-S3

- 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"



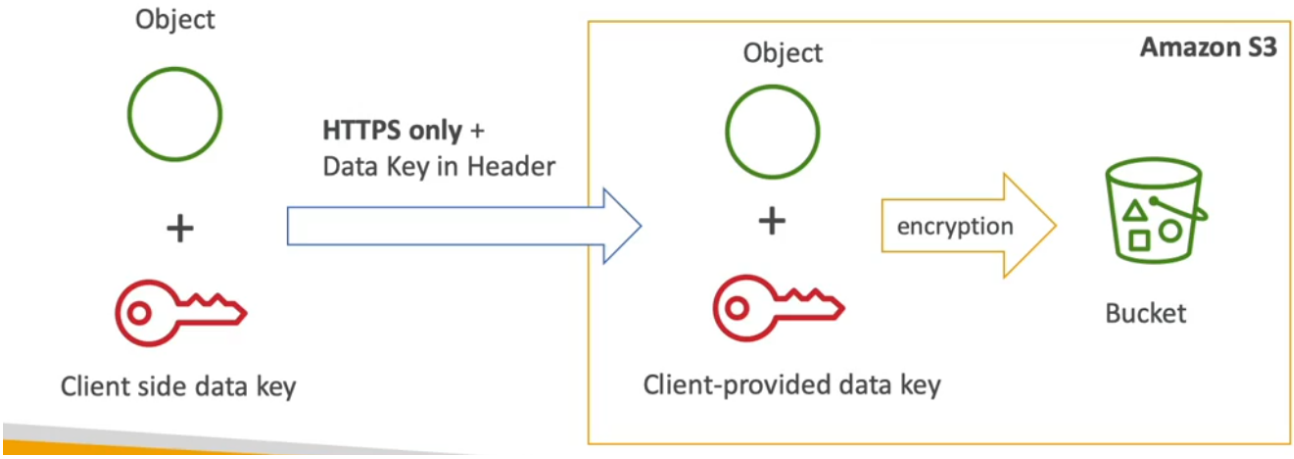
SSE-S3

- 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"



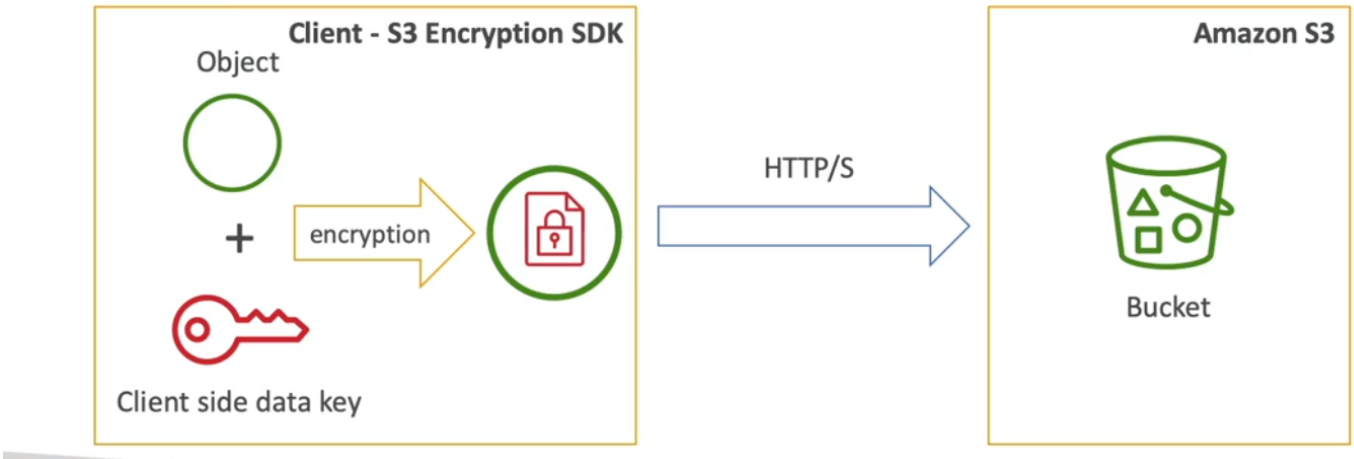
SSE-C

- 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 provided 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
- Clients must decrypt data themselves when retrieving from 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

AWS KMS (Key Management Service)

- Anytime you hear "encryption" for an AWS service, it's most likely KMS
- Easy way to control access to your data, AWS manages keys for us
- Fully integrated with IAM for authorization
- Seamlessly integrated into:
  - Amazon EBS: encrypt volumes
  - Amazon S3: Server side encryption of objects
  - Amazon Redshift: encryption of data
  - Amazon RDS: encryption of data
  - Amazon SSM: Parameter store
  - Etc…
- But you can also use the CLI / SDK

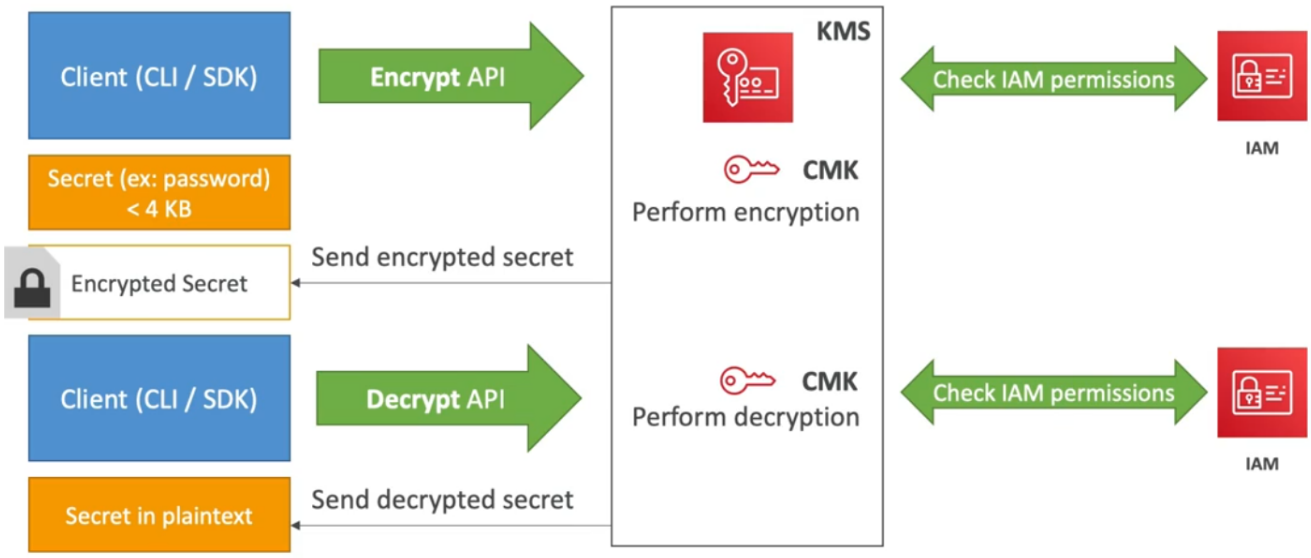
AWS KMS 101

- Anytime you need to share sensitive information… use KMS
  - Database passwords
  - Credentials to external service
  - Private Key of SSL certificates
- The value in KMS is that the CMK used to encrypt data can never be retrieved by the user, and the CMK can be rotated for extra security
- **Never ever store your secrets in plaintext, especially in your code!**
- Encrypted secrets can be stored in the code / environment variables
- **KMS can only help in encrypting up to 4KB of data per call**
- If data > 4 KB, use envelope encryption
- To give access to KMS to someone:
  - Make sure the Key Policy allows the user

AWS KMS (Key Management Service)

- Able to fully manage the keys & policies:
  - Create
  - Rotation policies
  - Disable
  - Enable
- Able to audit key usage (using CloudTrail)
- Three types of Customer Master Keys (CMK):
  - AWS Managed Service Default CMK: **free**
  - User Keys created in KMS: **$1 / month**
  - User Keys imported (must be 256-bit symmetric key): **$1 / month**
- + pay for API call to KMS ($0.03 / 10000 calls)

How does KMS work?
API – Encrypt and Decrypt



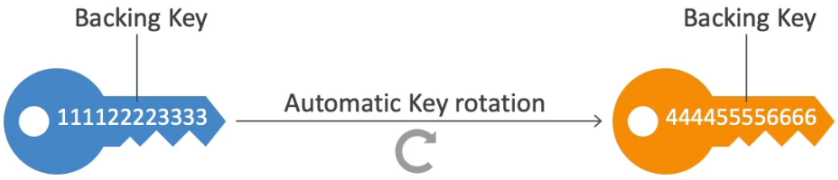
Encryption in AWS Services

- Requires migration (through Snapshot / Backup):
  - EBS Volumes
  - RDS databases
  - ElastiCache
  - EFS network file system

- In-place encryption:
  - S3

KMS Automatic Key Rotation

- For Customer-managed CMK (not AWS managed CMK)
- If enabled: automatic key rotation happens every 1 year
- Previous key is kept active so you can decrypt old data
- New Key has the same CMK ID (only the backing key is changed)

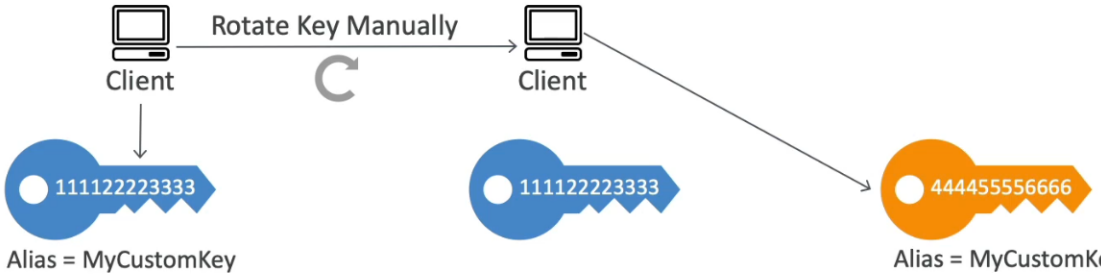


CMK ID = 1234abcd-12ab-34cd-56ef-1234567890ab

CMK ID = 1234abcd-12ab-34cd-56ef-1234567890ab

KMS Manual Key Rotation

- When you want to rotate key every 90 days, 180 days, etc…
- New Key has a different CMK ID
- Keep the previous key active so you can decrypt old data
- Better to use aliases in this case (to hide the change of key for the application)
- Good solution to rotate CMK that are not eligible for automatic rotation (like asymmetric CMK)



CMK ID = 1234abcd-12ab-34cd-56ef-1234567890ab

CMK ID = 1234abcd-12ab-34cd-56ef-1234567890ab

CMK ID = 0978dcba-09fe-65ba-ab0987654321

KMS Alias Updating

- Better to use aliases in this case (to hide the change of key for the application)