Data Access and Encryption

Cloud Security

CSE4059

Data Protection Requirements

- **Confidentiality**
  - Only authorized readers access a sensitive report
- **Integrity**
  - The report is protected from unauthorized modification or replacement
- **Availability**
  - Authorized readers can retrieve and use the report when needed

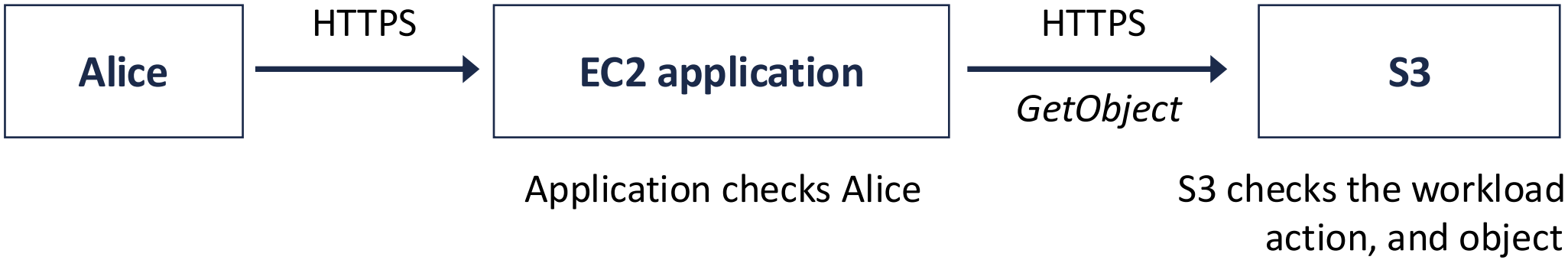
- Protect the data and its access policies, and preserve the keys and backups needed for recovery

Data Classification and Access Requirements

- Data classification assigns a sensitivity category and handling requirements
  - A public brochure and an internal report need different reader sets
- Define permitted readers, permitted changes, and retention needs for each category
  - Use those requirements to choose the object permissions and recovery controls

S3 Objects and the Report Service

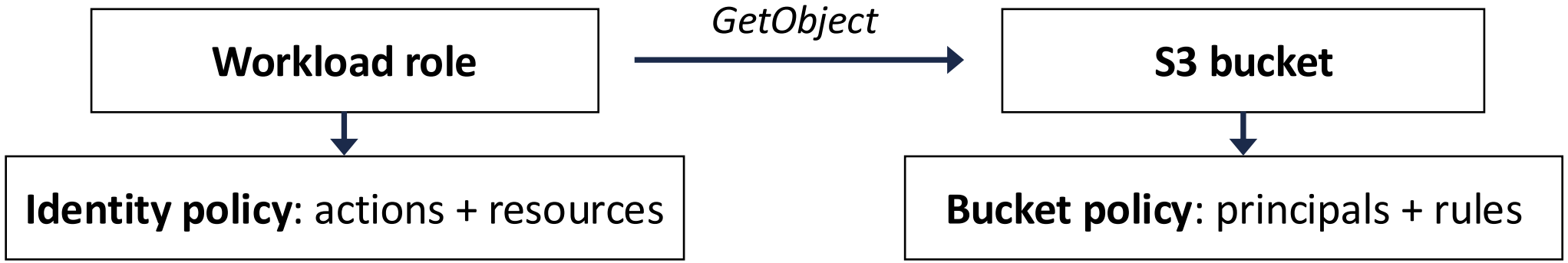
Assumption: The EC2 role and the S3 bucket belong to the same AWS account



- S3 stores objects, each containing data and metadata, in buckets
  - Bucket? cs-reports
  - Object key? reports/report-1.txt
- The application checks Alice's report permission, then calls GetObject using the EC2 role's temporary credentials
  - S3 checks the workload's permission for that object request

S3 Policies and Explicit Deny

Assumption: The EC2 role and the S3 bucket belong to the same AWS account



Both policy locations affect the same request

- An **identity policy**, attached to the role, defines its permissions for actions on resources
- A **bucket policy**, attached to the bucket, defines access rules for specified principals
- Either policy can grant the required permission, subject to other applicable restrictions
- An explicit deny that matches the request overrides an allow

S3 Actions and Resource ARNs

Operation	Permission	Example resource value
List names	s3:ListBucket	arn:aws:s3:::cs-reports → bucket ARN
Read content	s3:GetObject	arn:aws:s3:::cs-reports/reports/report-1.txt
Write content	s3:PutObject	arn:aws:s3:::cs-reports/reports/report-1.txt

- An Amazon Resource Name (ARN) identifies a resource in an AWS policy
  - A statement's Effect (Allow or Deny) specifies whether it grants or denies the action on that resource
  - Listing objects, reading their contents, and writing objects are separate permissions.

S3 Block Public Access

- Block Public Access (BPA) helps prevent public exposure of S3 buckets and objects
- With all BPA settings enabled, S3 rejects new public bucket policies and blocks public access granted by existing bucket policies
- Account-level BPA applies to all buckets in that account
  - Bucket settings cannot override an account-level restriction
- The EC2 role can still read the required report when the applicable policies permit it
- Keep all BPA settings enabled for this private-report service example

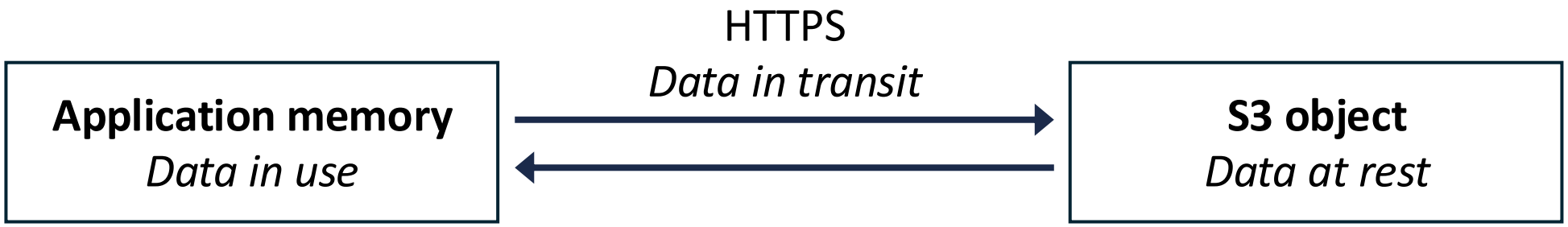
Private Access Can Still Be Too Broad

Current role policy: Allow s3:GetObject on arn:aws:s3:::cs-reports/*

Required:	**Unnecessary access**:
reports/report-1.txt	Other objects in the bucket

- BPA blocks public access but does not narrow the permissions granted to this role
- Apply least privilege by replacing the broad allowance with access to the required object

Data at Rest, in Transit, and in Use



- Data at rest
  - Stored data, such as a report object in S3
- Data in transit
  - Data moving between endpoints, such as an upload or download over HTTPS
- Data in use
  - Data being processed, such as report content in application memory

Encryption at Rest

- **Encryption** at rest protects stored data by converting plaintext into *ciphertext*

- Our example uses symmetric encryption
  - The same secret key encrypts and decrypts the data
  - The algorithm can be public, but the key must remain secret

- Encryption does not replace access control
  - Access policies must still control who can read or modify the data

- Encryption alone does not ensure recovery
  - Backups support recovery from deletion or corruption

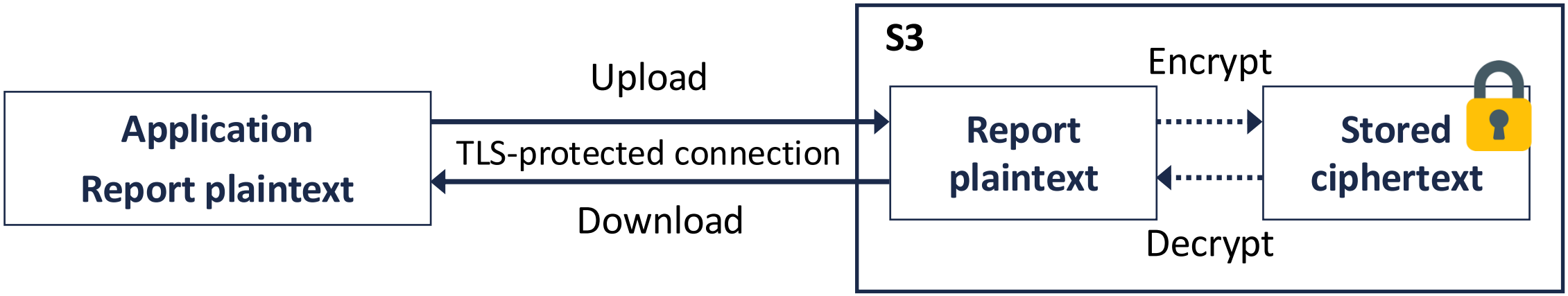
Where Does Encryption Happen?

Here, the client is the application calling S3

Comparison	Server-side encryption	Client-side encryption
Who encrypts and decrypts the report?	S3	The client application
Data uploaded over HTTPS	Report plaintext	Encrypted report
Data returned by S3 over HTTPS	Report plaintext	Encrypted report

- If S3 cannot access the client's decryption key, it cannot recover the original plaintext
- Our report-service example uses server-side encryption

TLS and Storage Encryption

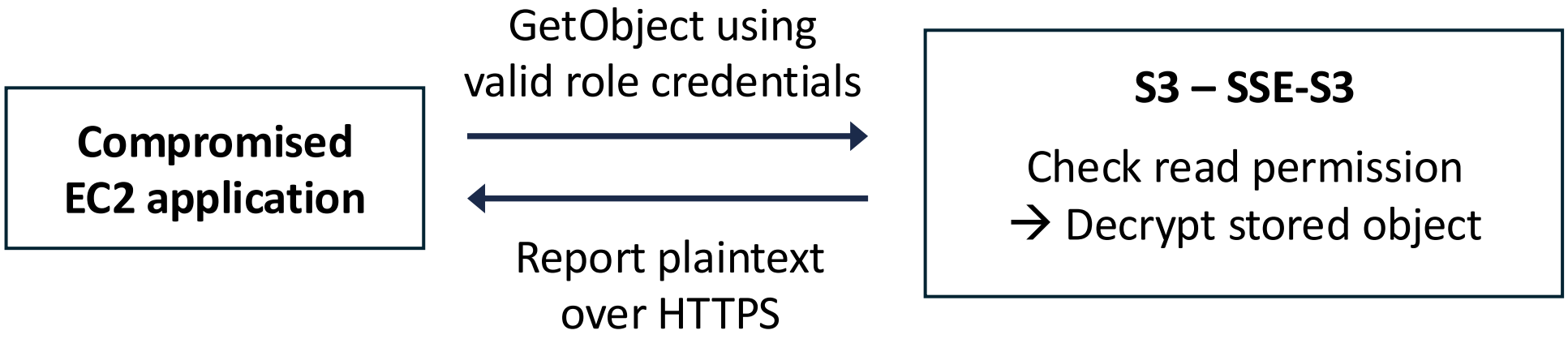


- TLS protects data in transit between the application and S3
- Server-side encryption protects data at rest in S3
  - During a download, S3 decrypts the stored object and sends its content through TLS
- S3 still checks whether the caller is authorized to access the object

SSE-S3: S3-Managed Keys

- SSE-S3 is Server-Side Encryption (SSE) with keys managed by Amazon S3
- New uploads use SSE-S3 by default unless another server-side encryption option is configured
- S3 manages the encryption keys, so applications do not need to generate or supply them
- Customers still control object access through IAM and bucket policies

A Compromised Workload Can Read Plaintext



- SSE-S3 does not prevent an attacker from misusing a compromised workload's valid read permissions
- Revoking the role's read access blocks new GetObject requests after the change takes effect
- Revocation does not erase plaintext that has already been downloaded

AWS Key Management Service (KMS)

- KMS manages cryptographic keys and provides API operations for using them
- With a customer managed KMS key, you control who may use it and manage its lifecycle, including enabling, disabling, and scheduling deletion
- A key ARN identifies a KMS key
  - it is not the secret key material
- Applications and AWS services request operations from KMS without receiving that KMS key's secret key material

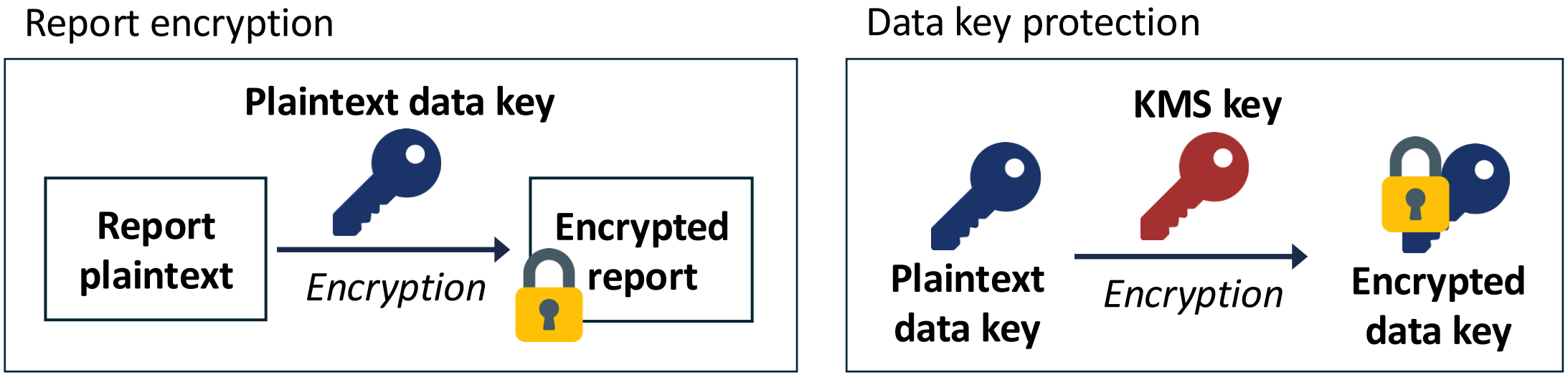
SSE-S3 and SSE-KMS: Key Management

SSE-KMS: server-side encryption with AWS KMS keys

Encryption option	Who controls the key?
SSE-S3	S3 manages the encryption keys
SSE-KMS + AWS managed key	AWS controls the KMS key policy and lifecycle
SSE-KMS + customer managed key	You control the KMS key policy and lifecycle

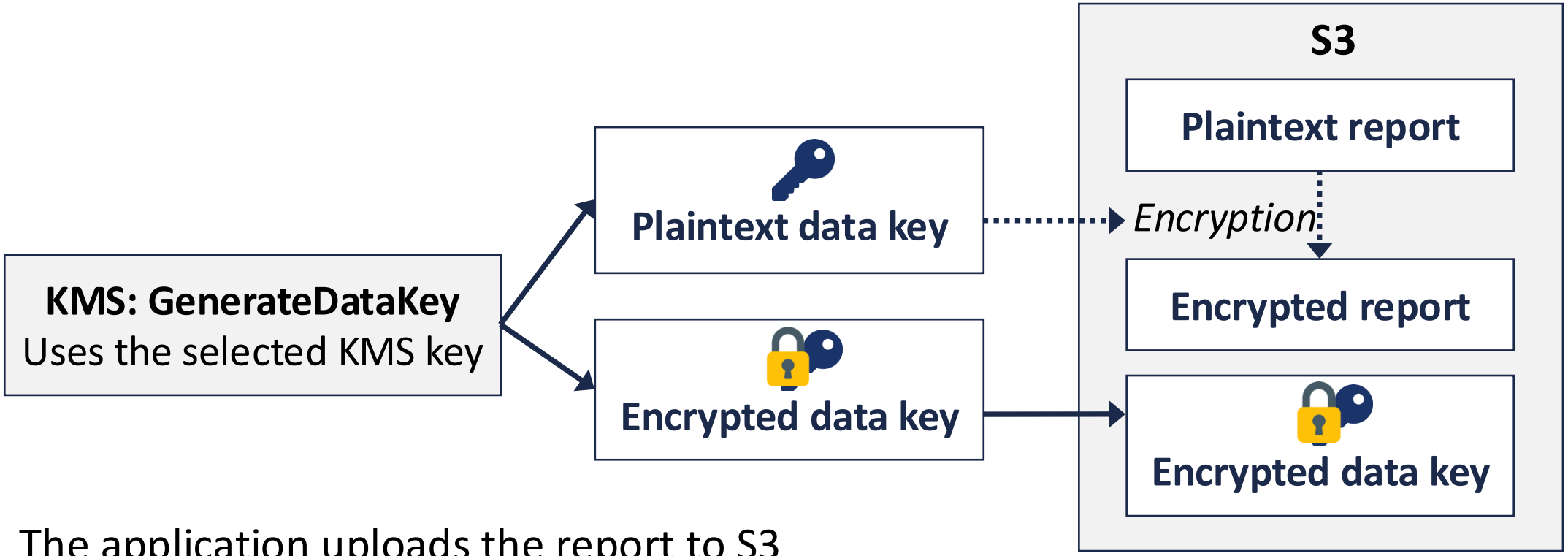
- All three options still require S3 object access permissions
- The following examples use SSE-KMS with a customer managed key

A KMS Key Protects a Data Key



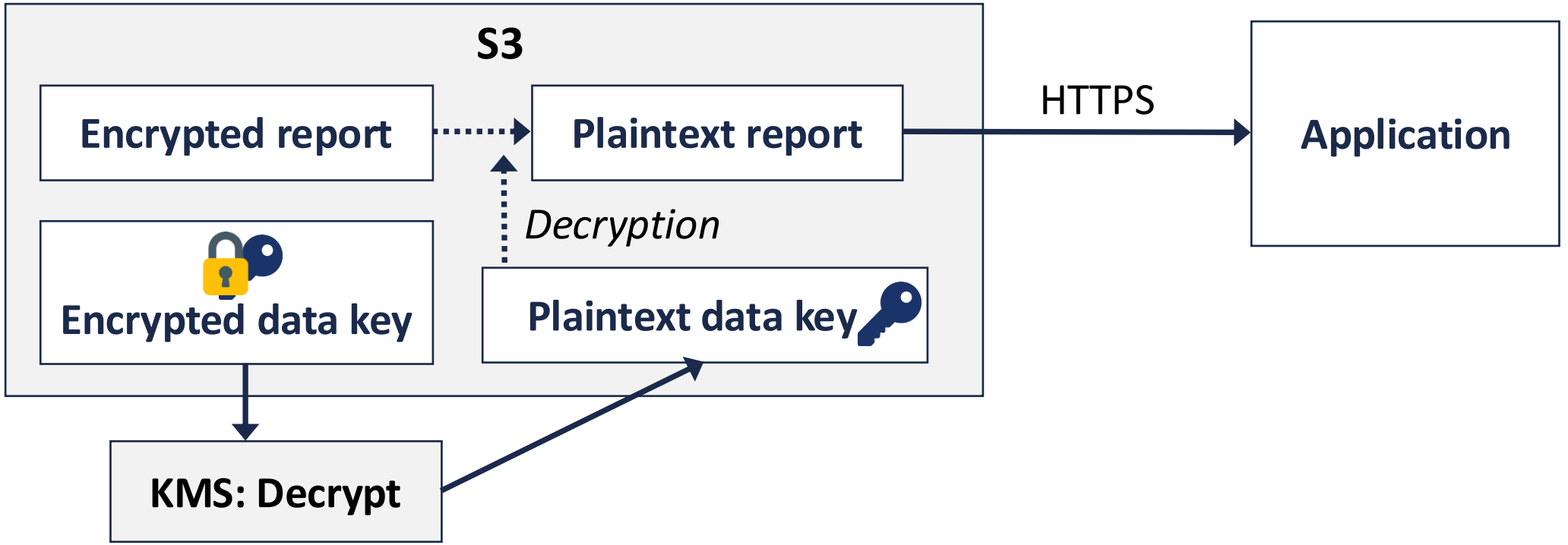
- A symmetric data key encrypts the report, and a KMS key protects the data key
  - This is called ***envelope encryption***
- S3 handles report encryption without sending the full report to KMS or receiving the KMS key's secret material
- S3 stores the encrypted data key with the encrypted report
  - S3 removes the plaintext data key from memory after encrypting the report

Envelope Encryption: Writing



- The application uploads the report to S3
- S3 requests a data key from KMS
- KMS returns plaintext and encrypted copies of the same new data key
- S3 removes the plaintext data key from memory after encrypting the report

Envelope Encryption: Reading



- On a GetObject request, S3 asks KMS to recover the original data key
- S3 returns the report plaintext to the application over HTTPS, without the data key
- After decryption, S3 removes the plaintext data key from memory

SSE-KMS: Required Permissions

- The application role needs permissions on both the S3 object and the KMS key

Operation	Permission on the S3 object	Permission on the KMS key
Upload with PutObject	s3:PutObject	kms:GenerateDataKey
Download with GetObject	s3:GetObject	kms:Decrypt

- S3 performs the KMS operation on behalf of the application
- The application calls S3; it does not need to call KMS directly in this workflow

KMS Key Policies and IAM Policies

- A key policy is attached to a KMS key and controls who can use or administer it
- It can grant permissions directly or enable IAM policies to grant them

Example: Manage the EC2 role's S3 and KMS permissions through IAM

The EC2 role and the KMS key belong to the same AWS account

Policy	KMS permission setting in this example
KMS key policy	Enables IAM policies in this account to grant access to the key
IAM identity policy	Allows kms:Decrypt on this specific KMS key

- An applicable explicit deny overrides an allow

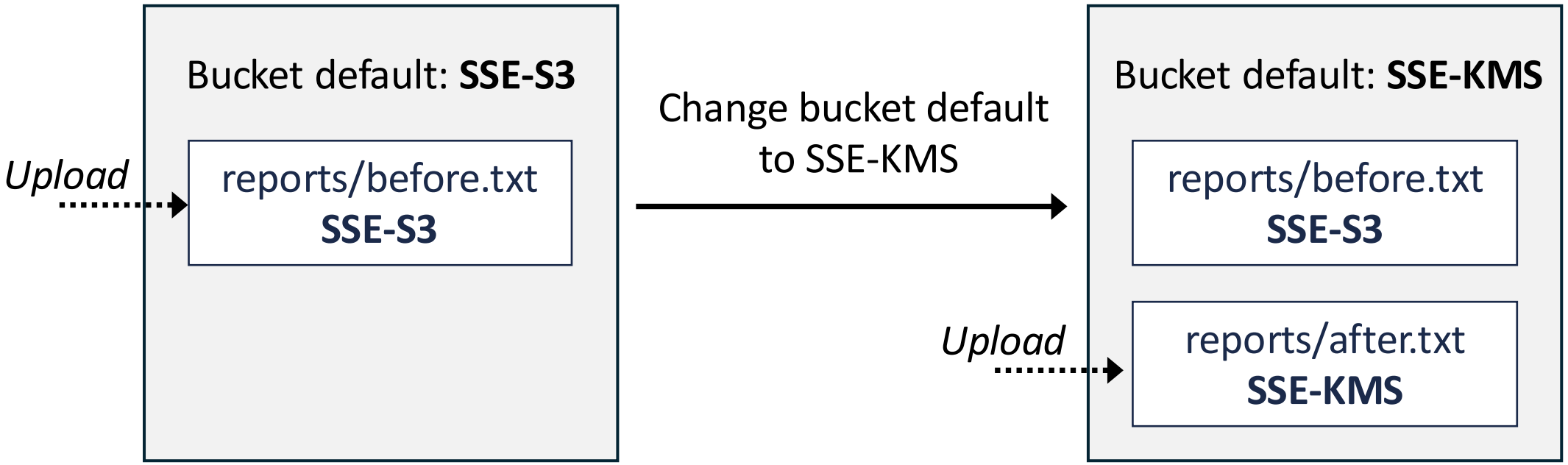
Key Use and Key Administration

Permission type	Purpose	Example actions
Key use	Perform cryptographic operations	kms:GenerateDataKey kms:Decrypt
Key administration	Manage key state or policy	kms:DisableKey kms:PutKeyPolicy

- The report-reading role needs kms:Decrypt, but does not need key-management permissions
- Management permissions do not automatically grant permission to decrypt
- A principal that can change the key policy may be able to grant itself key-use permissions

Bucket Defaults and Existing Objects

- The bucket default applies when an upload does not specify encryption settings
- Changing the default does not change the encryption of existing objects
- Objects in the same bucket can therefore use different encryption settings



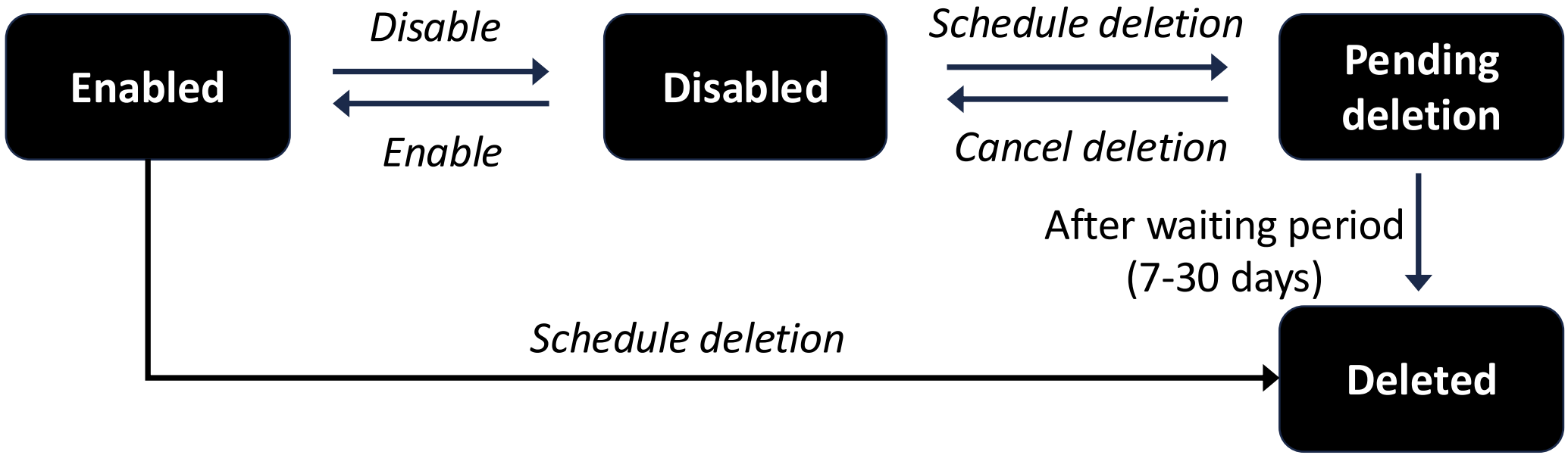
Verify Each Object's Encryption

- **HeadObject** is an S3 API operation that retrieves an object's metadata, including its encryption settings, without downloading its content

Object	ServerSideEncryption	SSEKMSKeyId
reports/before.txt	AES256 – SSE-S3	Not returned
reports/after.txt	aws:kms – SSE-KMS	Selected KMS key ARN

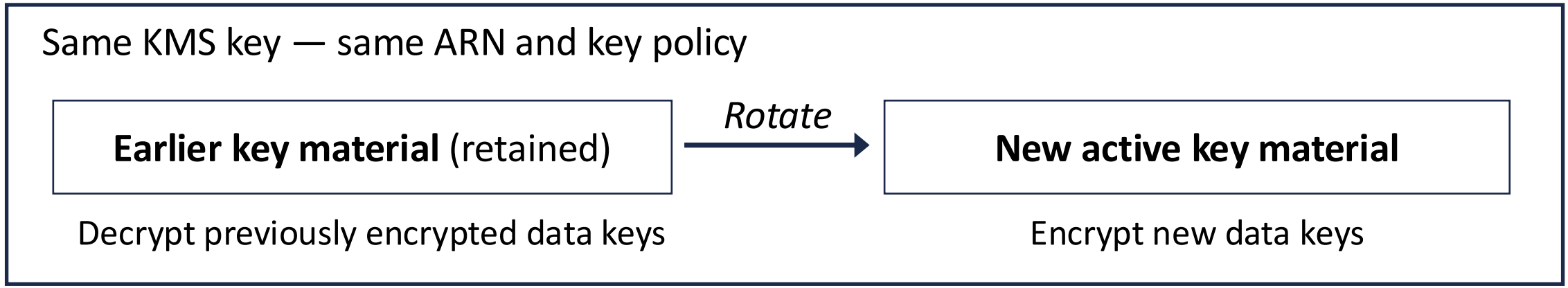
- Viewing metadata does not prove that the application can read the report

KMS Key State and Data Availability



- Reading an SSE-KMS report requires both access permissions and a usable KMS key
- Disabled keys and keys pending deletion cannot perform cryptographic operations
- After permanent deletion, KMS can no longer recover the data keys protected by that key

KMS Key Rotation and Existing Data



- Rotation introduces new key material (the secret value used for cryptographic operations) within the same KMS key
- It limits how long the same key material is used for new encryption and helps meet security policy requirements
- When decrypting a data key, KMS automatically selects the key material originally used to encrypt it
- Rotation does not change existing data keys or re-encrypt stored reports

Recovery Needs Data, Keys, and Permission

All three are required to recover an SSE-KMS report:

Requirement	What must be available
Data	An intact copy of the required report version
KMS key	The KMS key protecting that copy, usable for decryption
Permissions	A role allowed to read the object and use the KMS key

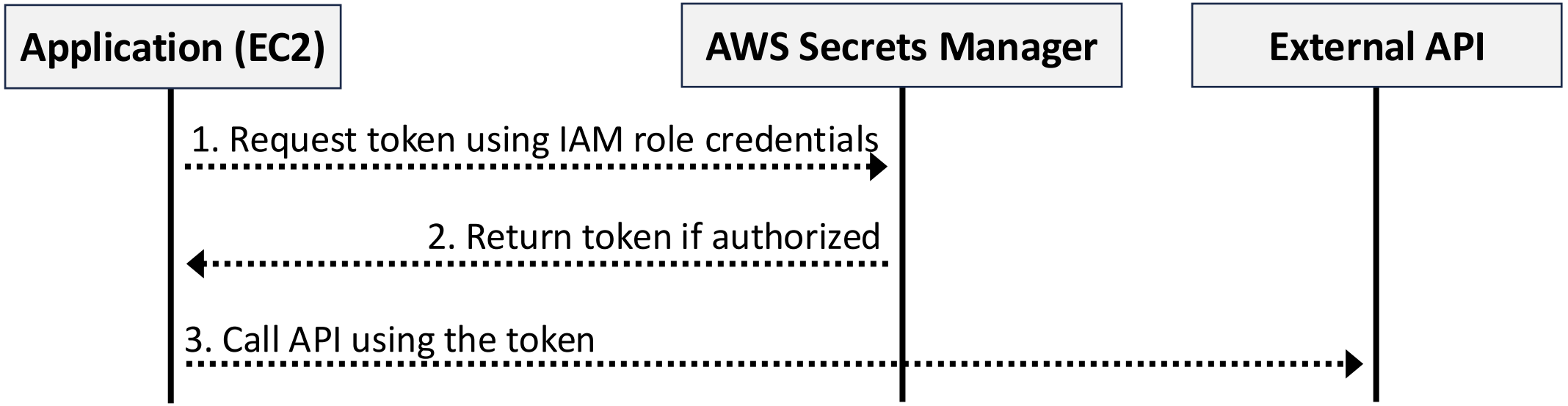
- S3 Versioning can preserve earlier object versions to help recover from accidental overwrites or deletions
- Keep the required KMS keys for as long as the encrypted copies must remain recoverable
- Test recovery using the IAM role that will perform the actual recovery

Protecting Application Secrets

- Application secrets are sensitive values
  - ex) database passwords and API tokens
- Passwords and tokens authenticate the application to other services; data keys encrypt and decrypt reports
- Hard-coded secrets can be exposed through source code or shared configuration files
- For AWS access from EC2, use temporary IAM role credentials instead of hard-coded long-term access keys

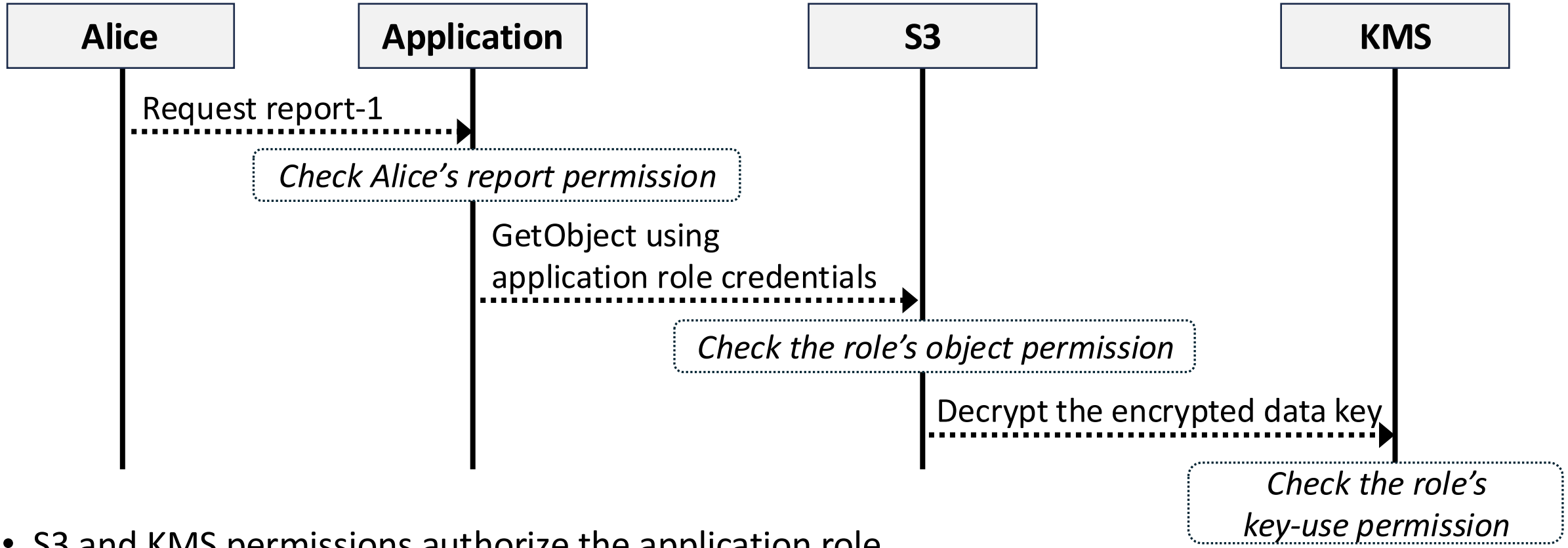
Retrieving Secrets with AWS Secrets Manager

Example: An external API token is already stored in Secrets Manager



- AWS Secrets Manager stores application secrets and controls who can retrieve them
- The application receives the plaintext secret over HTTPS
  - Do not write the secret to logs or diagnostic output
- When an API token is replaced, update Secrets Manager with the new valid token

Authorization Checks for a Report Read



- S3 and KMS permissions authorize the application role
  - The application must separately check whether Alice may receive the requested report
- A private bucket and encrypted storage do not prevent unauthorized disclosure if the application serves a report to the wrong user

Summary

- Application, S3, and KMS access checks serve different purposes
- Data keys encrypt reports and KMS keys protect data keys
- Bucket default changes and key rotation do not re-encrypt existing objects
- Recovery requires data, usable keys, and permissions
- Protect application secrets and plaintext during use