KMS Overview

- Envelope encryption – please dive deeper into the data/api call flow we discussed. I think it's discussed in Ken Beer's KMS presentation.
- At the minimum you should discuss what it means that the service is integrated with KMS. I think what will help you fully grasp it

1. EBS volume encryption
2. Amazon EBS works with AWS KMS to encrypt and decrypt your EBS volumes as follows:
    1. Amazon EBS sends a GenerateDataKeyWithoutPlaintext request to AWS KMS, specifying the CMK that you chose for volume encryption.
    2. AWS KMS generates a new data key, encrypts it under the CMK that you chose for volume encryption, and sends the encrypted data key to Amazon EBS to be stored with the volume metadata.
    3. When you attach an encrypted volume to an instance, Amazon EC2 sends a Decrypt request to AWS KMS, specifying the encrypted data key.
    4. Amazon EBS sends a CreateGrant request to AWS KMS, so that it can decrypt the data key.
    5. AWS KMS decrypts the encrypted data key and sends the decrypted data key to Amazon EC2.
    6. Amazon EC2 uses the plaintext data key in hypervisor memory to encrypt disk I/O to the volume. The plaintext data key persists in memory as long as the volume is attached to the instance.

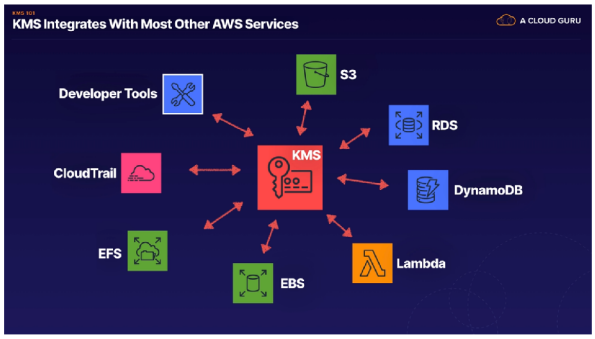
    For more information, see How Amazon Elastic Block Store (Amazon EBS) uses AWS KMS and Amazon EC2 example two in the *AWS Key Management Service Developer Guide*.
3. Amazon EBS works with AWS KMS to encrypt and decrypt your EBS volumes as follows:
    1. Amazon EBS sends a GenerateDataKeyWithoutPlaintext request to AWS KMS, specifying the CMK that you chose for volume encryption.
    2. AWS KMS generates a new data key, encrypts it under the CMK that you chose for volume encryption, and sends the encrypted data key to Amazon EBS to be stored with the volume metadata.
    3. When you attach an encrypted volume to an instance, Amazon EC2 sends a Decrypt request to AWS KMS, specifying the encrypted data key.
    4. Amazon EBS sends a CreateGrant request to AWS KMS, so that it can decrypt the data key.
    5. AWS KMS decrypts the encrypted data key and sends the decrypted data key to Amazon EC2.
    6. Amazon EC2 uses the plaintext data key in hypervisor memory to encrypt disk I/O to the volume. The plaintext data key persists in memory as long as the volume is attached to the instance.

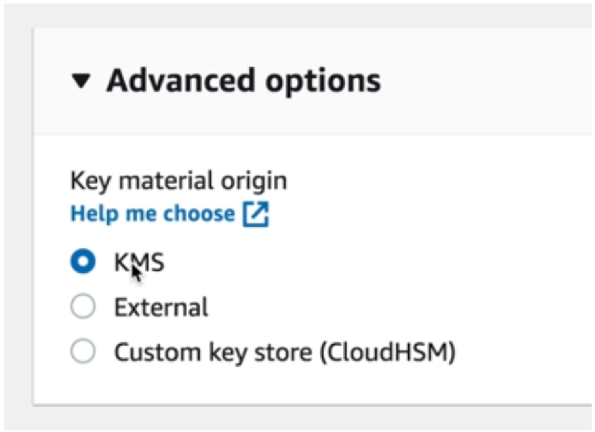
    For more information, see How Amazon Elastic Block Store (Amazon EBS) uses AWS KMS and Amazon EC2 example two in the *AWS Key Management Service Developer Guide*.
4. S3 encryption - try with default encryption setting and also without default encryption but passing the CMK name as part of PutObject call
5. Try KMS CLI directly to encrypt some data (similar to the demo you watched)

- There are no regions in Texas or Seattle: Ohio, Virginia, California, and Oregon
- Audit- I would not go to encryption context, would focus on CloudTrail. Check out CloudTrail in your account and pull up some KMS events.
- Operational assurance – I think this is an area that you may need to research a bit further, what exactly does this mean, have an example?
- Understand what an HSM is, how it's different from a typical server
- The question whether I have to use KMS? Your answer was good at the end but practice this a bit more in terms of the value KMS brings especially around seamless integration with AWS services and what it means if you use your own key management solution and don't have this seamless integration.

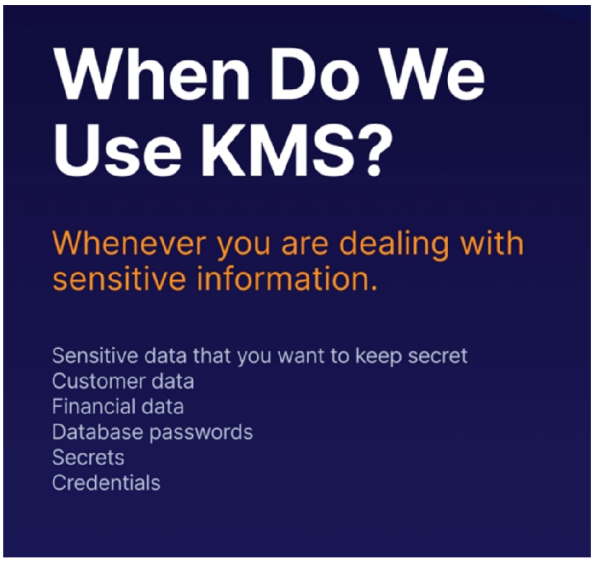
*questions for David:

1. What does it mean that its's backed by HSM (hardware security modules), then why would customers use custom key store (CloudHSM) for key material origin?

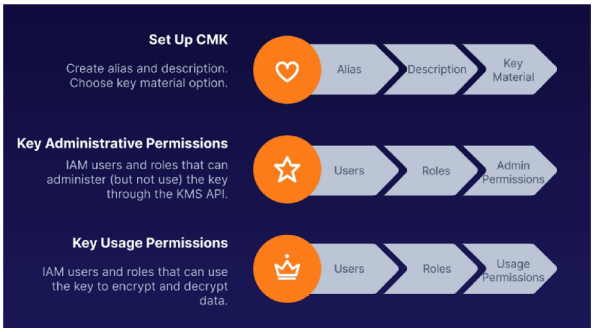




*KM: cryptographic secrets that make up the key for encryption and decryption processes
*Cloud HSM: for customers that require their own hardware security module for highly regulated compliances



CMK Review



AWS-Managed CMK

AWS-provided and AWS-managed CMK. Used on your behalf with the AWS services integrated with KMS.

Data Key

Encryption keys that you can use to encrypt data, including large amounts of data. You can use a CMK to generate, encrypt, and decrypt data keys.

Customer-Managed CMK

You create, own and manage yourself.

Exam Tips
KMS API Calls

`aws kms encrypt`
Encrypts plaintext into ciphertext by using a customer master key.

`aws kms re-encrypt`
Decrypts ciphertext and then re-encrypts it entirely within AWS KMS (e.g. when you change the CMK or manually rotate the CMK).

`aws kms decrypt`
Decrypts ciphertext that was encrypted by an AWS KMS customer master key (CMK).

`aws kms enable-key-rotation`
Enables automatic key rotation every 365 days.

`aws kms generate-data-key`

Uses the CMK to generate a data key to encrypt data > 4KB.

EXPLORING ENVELOPE ENCRYPTION

What Is Envelope Encryption?

A process for encrypting your data. It applies to files > 4 KB.

Why Envelope Encryption?

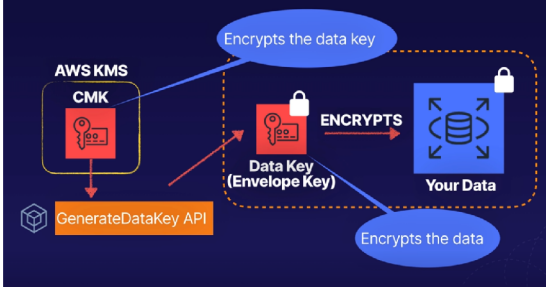
Why not encrypt the data using the CMK directly?

Network
When you encrypt data directly with KMS it must be transferred over the network.

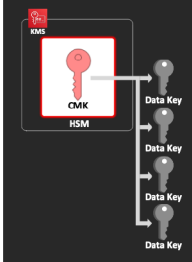
Performance
With envelope encryption, only the data key goes over the network, not your data.

Benefits
The data key is used locally in your application or AWS service, avoiding the need to transfer large amounts of data to KMS.

THE ENVELOPE ENCRYPTION PROCESS



Customer master keys and data keys



Customer master keys (CMKs)

- Represent the top of a key hierarchy
- Unique ARN
- Non-exportable
- Generated on HSM or imported by the customer
- Region-specific, globally unique

Data keys

- Used to encrypt your data
- Protected by CMKs
- Generated (or decrypted) by AWS KMS
- Do NOT write these to durable storage

Key policies vs. key grants

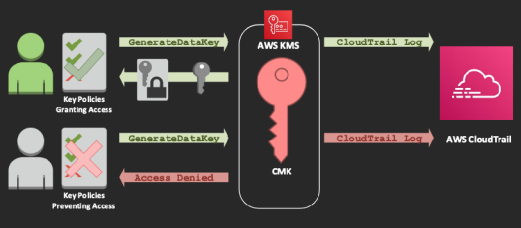
Key policies

- **One policy per key**
- Big blob of JSON
- Maximum size **32 KB**
- Can be **difficult to modify programmatically**

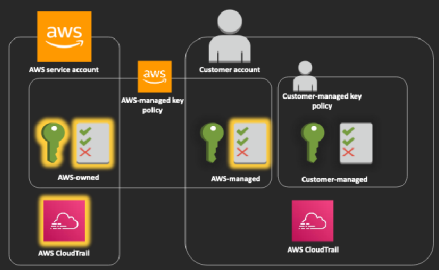
Key grants

- **Multiple grants per key** (up to 10,000 per key)
- **Granular** (least privilege) control for usage of CMKs
- Built for **programmatic access**
- Most appropriate for **temporary access** (easy to retire)
- Allows **delegation of permissions** to additional grantees while limited to scope of original grant

AWS KMS authorization



Customer master key ownership models

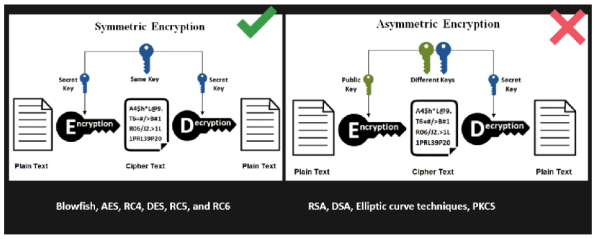


Cross-account/Cross-region considerations

- AWS KMS **CMKs are region-specific**
- To replicate, you will need to do a **cross-region call**
- Choosing how to use keys from different regions depends on your threat model and your latency requirements
- Most customers choose to **encrypt data in the source region** with keys from all destination regions

Integrated services

Alexa for Business*	Amazon Lex	AWS CloudTrail
Amazon Athena	Amazon Lightsail*	AWS CodeBuild
Amazon Aurora	Amazon Managed Streaming for Kafka (MSK)	AWS CodeCommit*
Amazon CloudWatch Logs	Amazon MQ	AWS CodeDeploy
Amazon Comprehend*	Amazon Neptune	AWS CodePipeline
Amazon Connect	Amazon Personalize	AWS Database Migration Service
Amazon DocumentDB	Amazon Redshift	AWS Glue
Amazon DynamoDB*	Amazon Relational Database Service (RDS)	AWS Lambda
Amazon DynamoDB Accelerator	Amazon S3	AWS Secrets Manager
Amazon FBS	Amazon SageMaker	AWS Systems Manager
Amazon EFS	Amazon Simple Email Service (SES)	AWS Snowball
Amazon Elastic Transcoder	Amazon Simple Notification Service (SNS)	AWS Snowball Edge
Amazon Elasticsearch Service	Amazon Simple Queue Service (SQS)	AWS Snowmobile
Amazon EMR	Amazon Translate	AWS Storage Gateway
Amazon FSx for Windows File Server	Amazon WorkMail	AWS X-Ray
Amazon S3 Glacier	Amazon WorkSpaces	
Amazon Kinesis Data Firehose	AWS Backup	
Amazon Kinesis Data Streams	AWS Certificate Manager*	
Amazon Kinesis Video Streams	AWS Cloud9*	



Type of CMK	Can view CMK metadata	Can manage CMK	Used only for my AWS account	Automatic rotation
Customer managed CMK	Yes	Yes	Yes	Optional. Every 365 days (1 year).
AWS managed CMK	Yes	No	Yes	Required. Every 1095 days (3 years).
AWS owned CMK	No	No	No	Varies

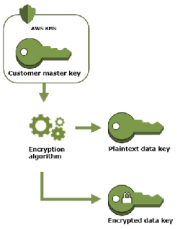
Encrypt data with a data key

AWS KMS cannot use a data key to encrypt data. But you can use the data key outside of KMS, such as by using OpenSSL or a cryptographic library like the AWS Encryption SDK.

After using the plaintext data key to encrypt data, remove it from memory as soon as possible. You can safely store the encrypted data key with the encrypted data so it is available to decrypt the data.

Create a data key

To create a data key, call the GenerateDataKey operation. AWS KMS uses the CMK that you specify to generate a data key. The operation returns a plaintext copy of the data key and a copy of the data key encrypted under the CMK. The following image shows this operation.

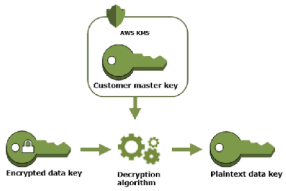


AWS KMS also supports the GenerateDataKeyWithoutPlaintext operation, which returns only an encrypted data key. When you need to use the data key, ask AWS KMS to decrypt it.

Decrypt data with a data key

To decrypt your data, pass the encrypted data key to the Decrypt operation. AWS KMS uses your CMK to decrypt the data key and then it returns the plaintext data key. Use the plaintext data key to decrypt your data and then remove the plaintext data key from memory as soon as possible.

The following diagram shows how to use the Decrypt operation to decrypt an encrypted data key.



How EBS encryption works

You can encrypt both the boot and data volumes of an EC2 instance. When you create an encrypted EBS volume and attach it to a supported instance type, the following types of data are encrypted:

- 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

Amazon EBS works with AWS KMS to encrypt and decrypt your EBS volumes as follows:

1. Amazon EBS sends a GenerateDataKeyWithoutPlaintext request to AWS KMS, specifying the CMK that you chose for volume encryption.
2. AWS KMS generates a new data key, encrypts it under the CMK that you chose for volume encryption, and sends the encrypted data key to Amazon EBS to be stored with the volume metadata.
3. When you attach an encrypted volume to an instance, Amazon EC2 sends a Decrypt request to AWS KMS, specifying the encrypted data key.
4. Amazon EBS sends a CreateGrant request to AWS KMS, so that it can decrypt the data key.
5. AWS KMS decrypts the encrypted data key and sends the decrypted data key to Amazon EC2.
6. Amazon EC2 uses the plaintext data key in hypervisor memory to encrypt disk I/O to the volume. The plaintext data key persists in memory as long as the volume is attached to the instance.

For more information, see How Amazon Elastic Block Store (Amazon EBS) uses AWS KMS and Amazon EC2 example two in the *AWS Key Management Service Developer Guide*.

Ilya's feedbacks

- HSMs – needs a crisp, short definition around a device that performs crypto operations (encrypt/decrypt), temper protected hardware with security controls that prevent compromise of loss of keys.

A hardware security module (HSM) is a hardware device designed with the security of your data and cryptographic key material in mind. It is tamper-resistant hardware that prevents unauthorized users from attempting to pry open the device, plug any extra devices in to access data or keys such as subtokens, or damage the outside housing. If any such interference occurs, the device wipes all information stored so that unauthorized parties do not gain access to your data or cryptographic key material. A high-availability (HA) setup could be beneficial because, with multiple HSMs kept in different data centers and all data synced between them, the loss of one HSM does not mean the loss of your data.

What is a Hardware Security Module?

The hardware security module (HSM) is a special "trusted" network computer performing a variety of cryptographic operations: key management, key exchange, encryption etc.

It seems to be obvious that cryptographic operations must be performed in a trusted environment. When I say trusted, I mean "no viruses, no malware, no exploit, no unauthorized access." An HSM is trusted because it:

1. Is built on top of specialized hardware. The hardware is well-tested and certified in special laboratories.
2. Has a security-focused OS.
3. Has limited access via a network interface that is strictly controlled by internal rules.
4. Actively hides and protects cryptographic material.

- Envelope encryption – as we discussed, please prepare to whiteboard envelope encryption with two use cases (1) plaintext data on EC2 with CLI directly to KMS (2) integrated use case with S3. This should include data flow and storage location of the data keys (i.e. encrypted data keys are not stored in KMS)
- Better short messaging on FIPS if asked

The Federal Information Processing Standard (FIPS) Publication 140-2 is a US and Canadian government standard that specifies the security requirements for cryptographic modules that protect sensitive information.

- Difference between durability and availability–your final explanation of the two was good. Let's discuss this more on Tuesday.

https://aws.amazon.com/kms/features/#Custom_Key_Store

- There are no guarantees in life! We never guarantee anything J. When we discuss durability and even availability the focus should be on “design principles”- our services are designed for high durability and high availability but never guaranteed.
- KMS is a **regional** service, not global. Keys are not replicated across regions.
- In regards to using my own key management solution instead of KMS–the two key values are (1) **out of the box integration with 50+ services** which customers would have to build themselves if they are using their own key management service (2) **cost of the infrastructure such as the EC2 instances** would likely make it more expensive as compared to KMS which you only pay for management and use of keys but not compute costs
- General earn trust concern–as we discussed previously please don’t guess. Try to avoid “as far as i know”. It’s OK to defer for a follow up. Similarly, I would not go into details around what you have researched–focus on an answer to a customer that they can do something with. If you are still researching or unsure, just say that you will need to get back to them but don’t go into the details of your research and any confusing or seemingly contradictory text you may have come across.

What is the advantage of REST API?

One of the key advantages of REST APIs is that they provide a great deal of **flexibility**. Data is not tied to resources or methods, so REST can handle multiple types of calls, return different data formats and even change structurally with the correct implementation of hypermedia.