SUMMIT

ONLINE

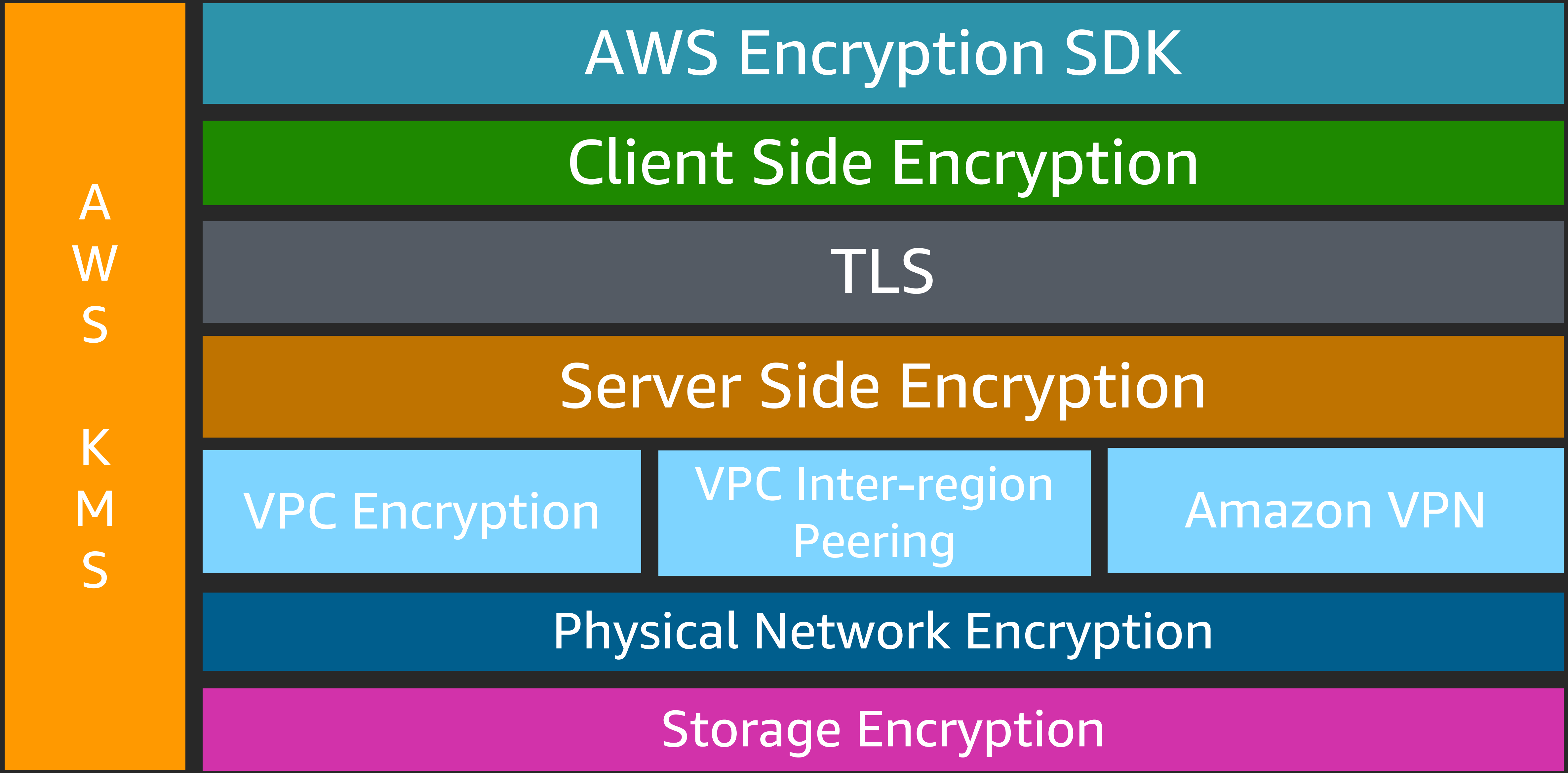
3 6

Let’s encrypt everything, really everything

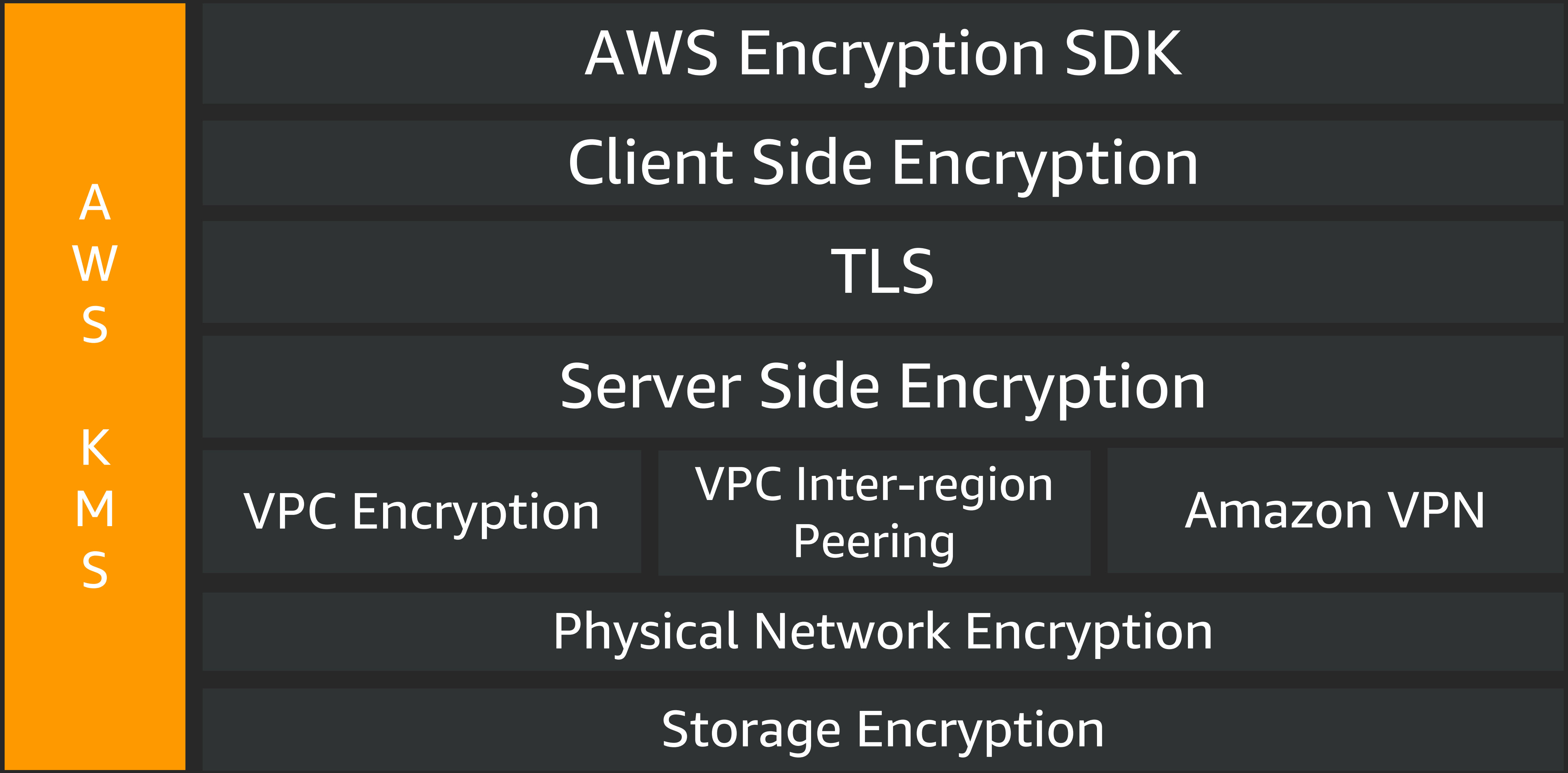
Sébastien Stormacq
Developer Advocate
Amazon Web Services

aws SUMMIT ONLINE

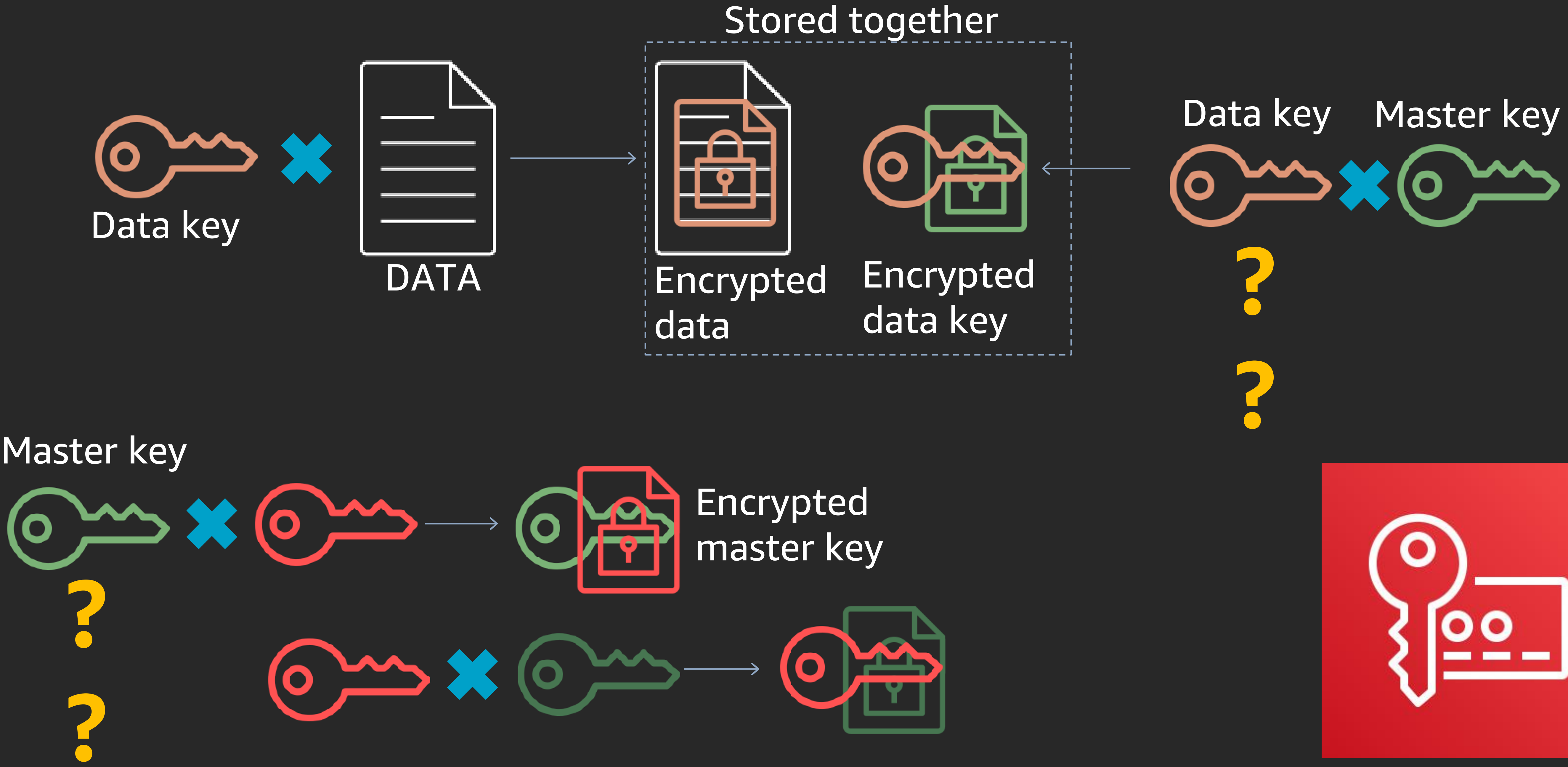
Encryption on AWS



AWS Key Management Service



Encryption primer

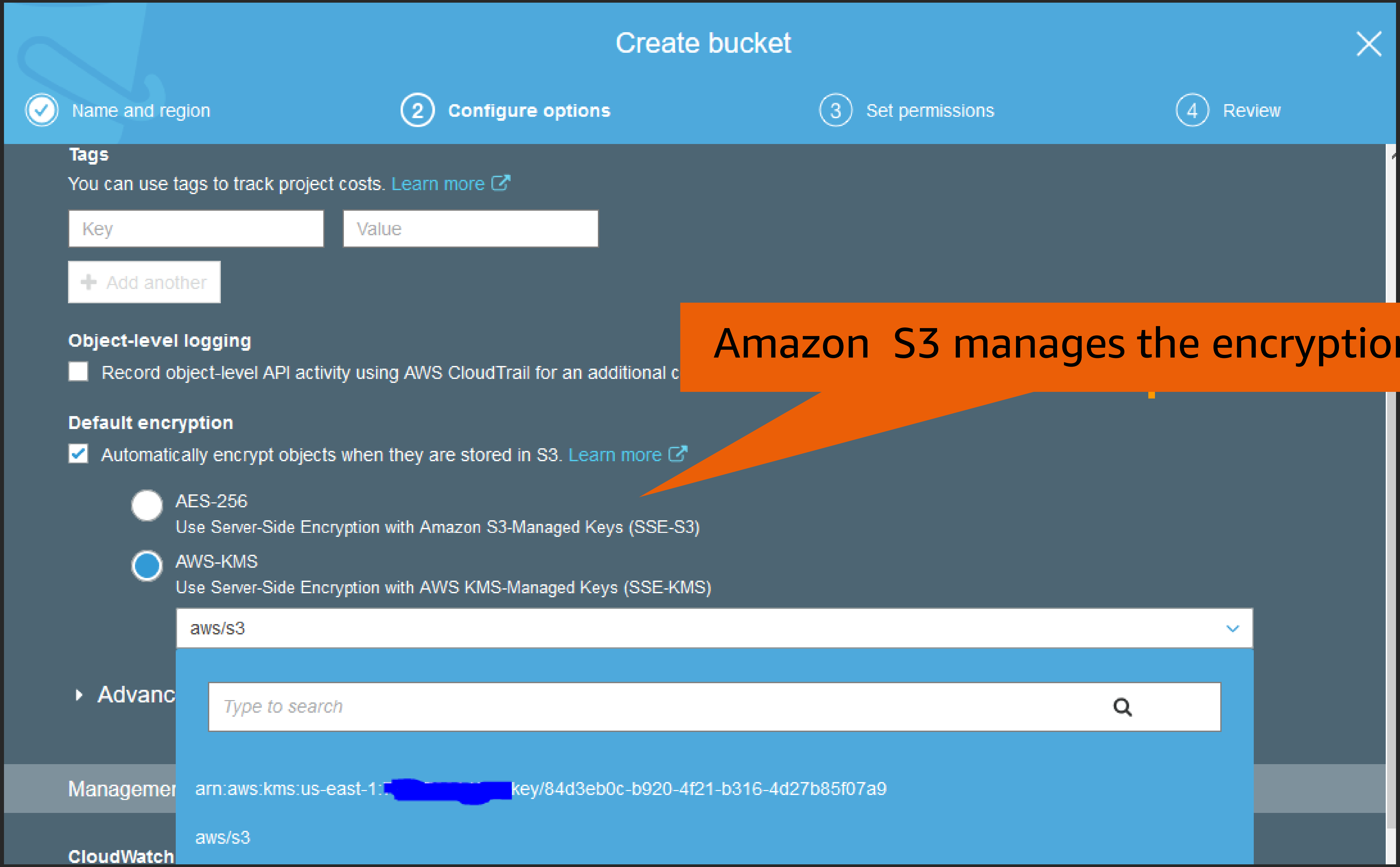


AWS Key Management Service (KMS)

Manage and Control the keys to encrypt your data

- Managed service to create, verify, rotate, delete or use your keys
- Integrated with 117+ AWS server-side encryption services
- Audit key usage with AWS CloudTrail
- FIPS 140-2 level2, SOC (1/2/3), PCI-DSS, ISO 27017/27018
- Low latency and high throughput
- High availability in the region

Simplify encryption through AWS Services integration



Create bucket

Name and region · 2 Configure options · 3 Set permissions · 4 Review

Tags

You can use tags to track project costs. Learn more

Key | Value

Add another

Object-level logging

☐ Record object-level API activity using AWS CloudTrail for an additional c

Default encryption

[✓] Automatically encrypt objects when they are stored in S3. Learn more

AES-256
Use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)

[✓ AWS-KMS]
Use Server-Side Encryption with AWS KMS-Managed Keys (SSE-KMS)

aws/s3

Type to search

arn:aws:kms:us-east-1:[redacted]key/84d3eb0c-b920-4f21-b316-4d27b85f07a9

aws/s3

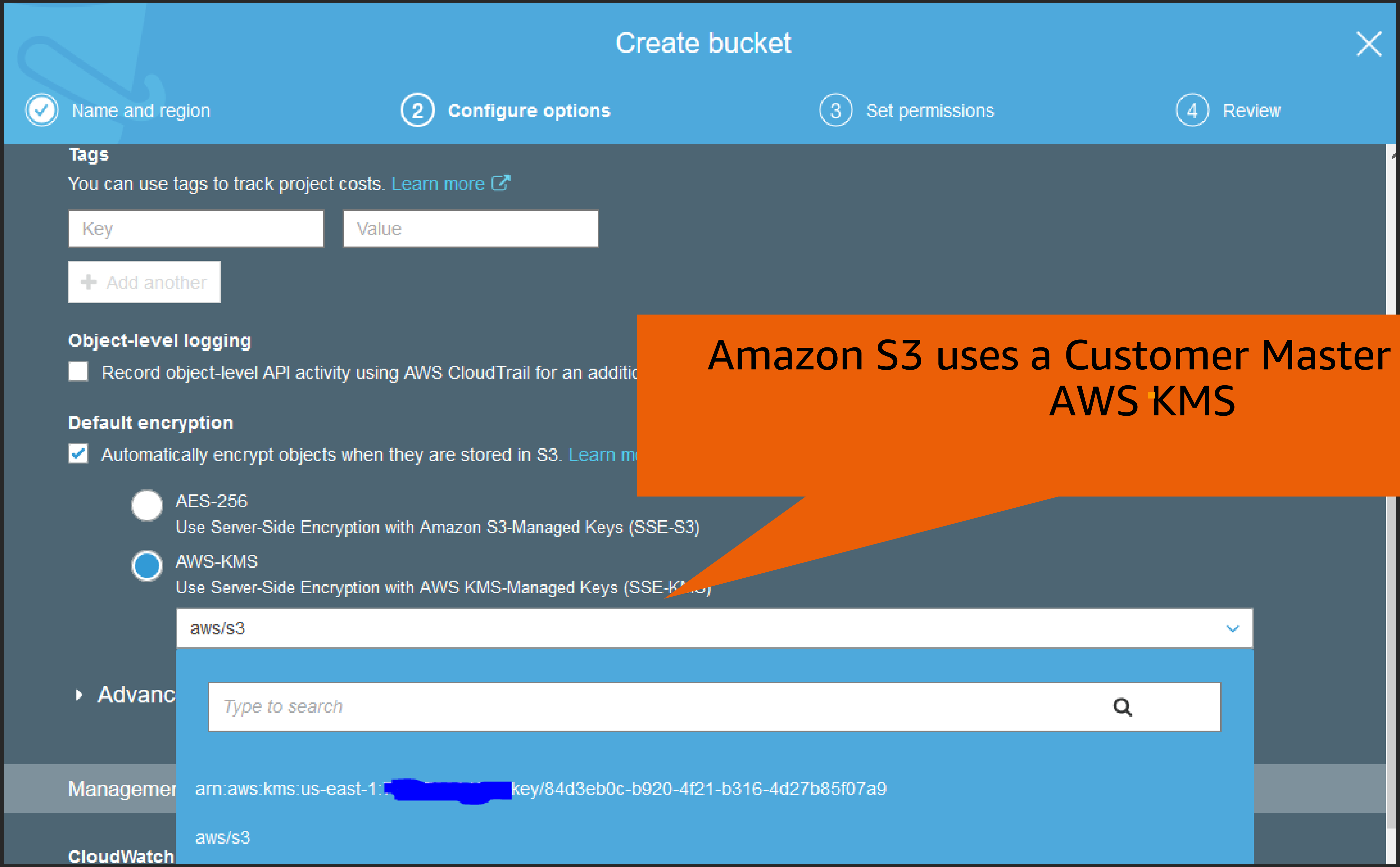
Advanc

Managemer

CloudWatch

Amazon S3 manages the encryption key

Simplify encryption through AWS Services integration



Create bucket

Name and region | 2 Configure options | 3 Set permissions | 4 Review

Tags

You can use tags to track project costs. Learn more

Key | Value

Add another

Object-level logging

☐ Record object-level API activity using AWS CloudTrail for an additic

Default encryption

☑ Automatically encrypt objects when they are stored in S3. Learn m

○ AES-256
Use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)

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

aws/s3

Type to search

arn:aws:kms:us-east-1:[redacted]key/84d3eb0c-b920-4f21-b316-4d27b85f07a9

aws/s3

Advanc

Managemen

CloudWatch

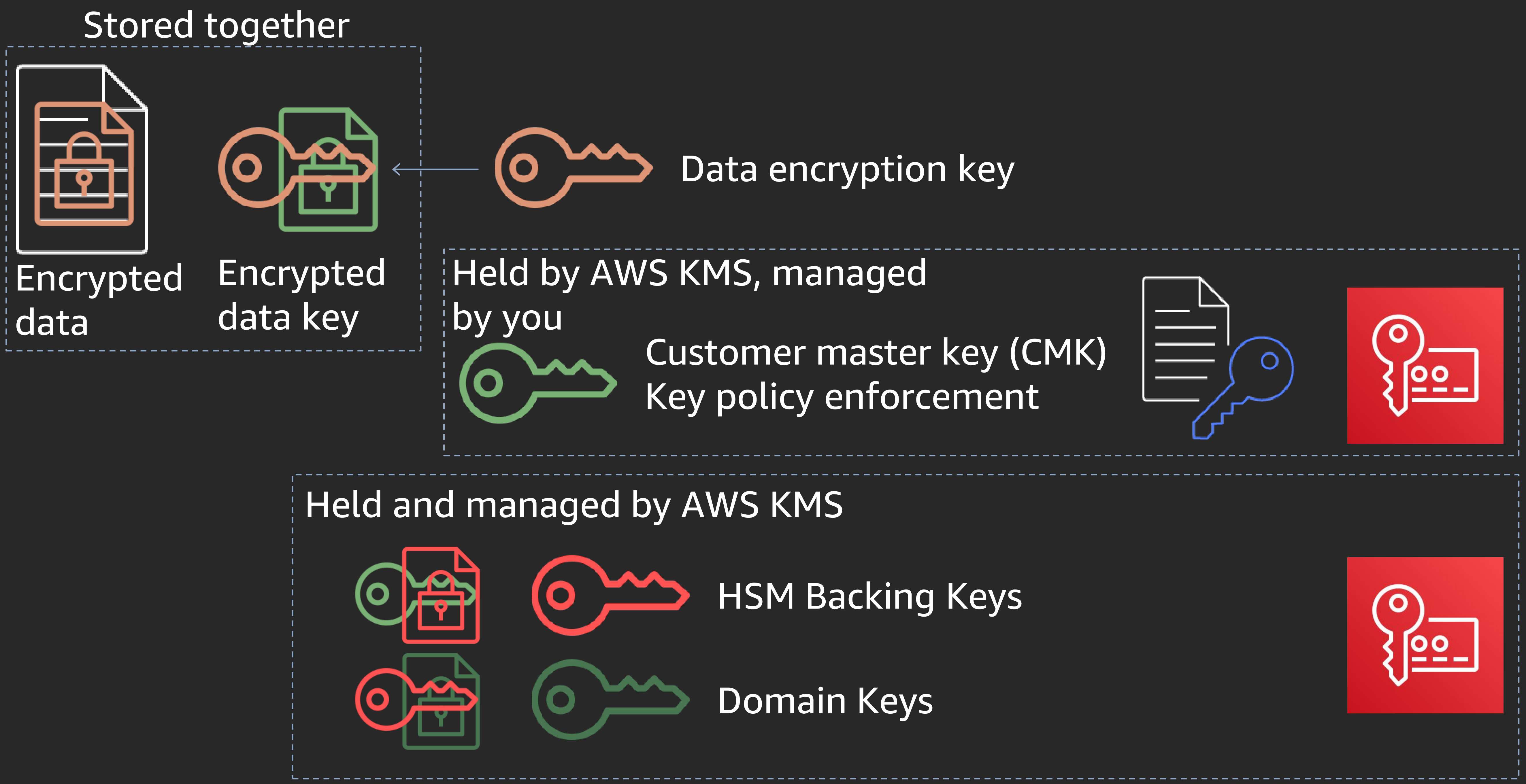
Amazon S3 uses a Customer Master Key from AWS KMS

Services integrated with AWS KMS

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

** Supports only AWS-managed KMS keys*

Key Hierarchy on AWS KMS



KMS Data Key Example (Python)

```
#
# Generate a Data Key (encoded with my Master Key in KMS)
#
key = kms.generate_data_key(KeyId=MASTER_KEY_ARN,KeySpec='AES_256')
keyPlain = key['Plaintext']
keyCipher = key['CiphertextBlob']
```

```
                          #
                          # Encode a plain text with the data key
                          #
                          obj = AES.new(keyPlain, AES.MODE_CBC, b'This is an IV123')
                          msgPlain = b'Hello world of cryptography w/managed keys'
                          msgCipher = obj.encrypt(pad(msgPlain, 16))
```

```
#
# and we decrypt our cipher text
#
obj = AES.new(keyPlain, AES.MODE_CBC, b'This is an IV123')
plainText = unpad(obj.decrypt(msgCipher), 16)
```

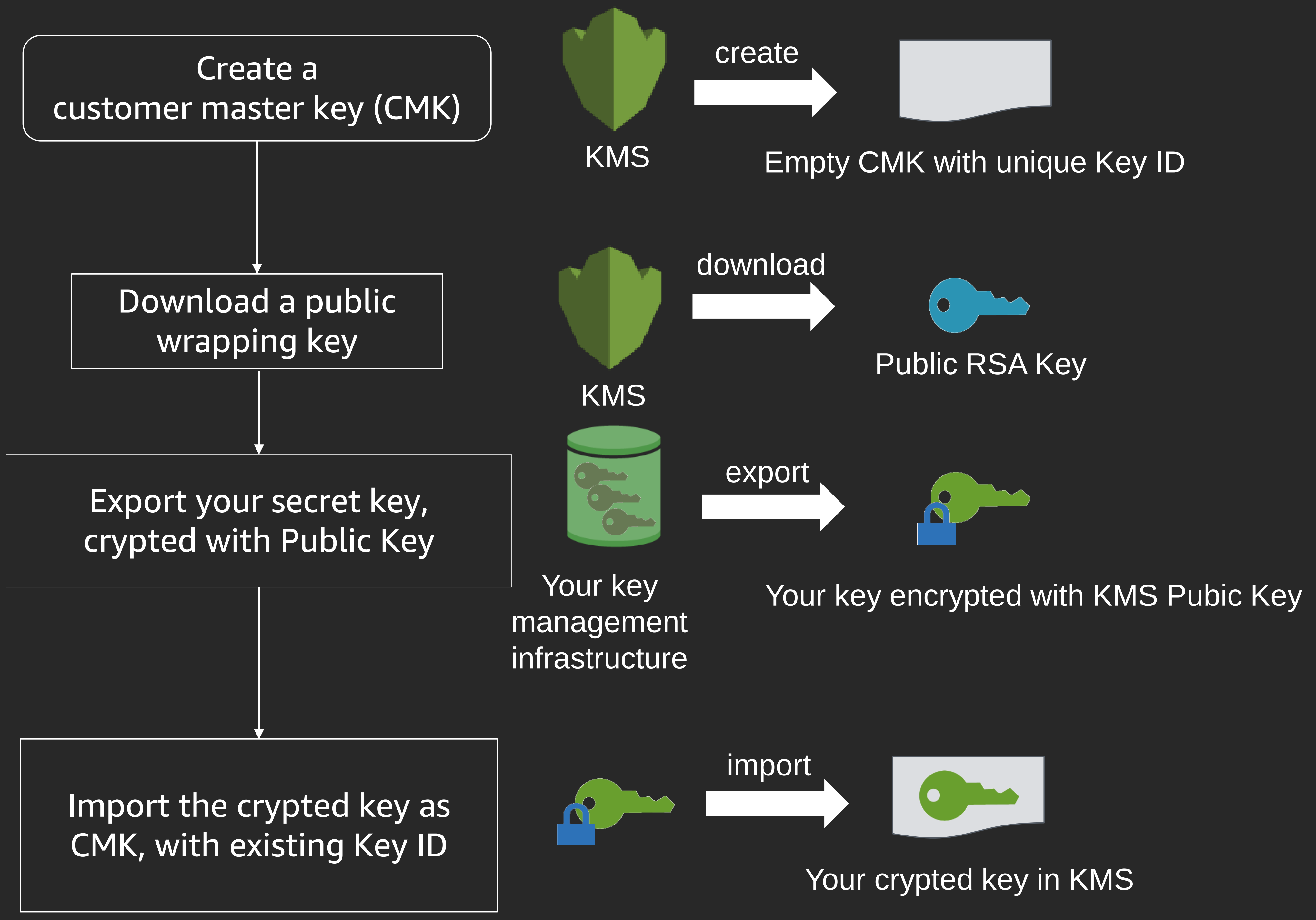
KMS Encryption Example (Python)

```
#
# Cipher a plain text object using your master key
#
ret = kms.encrypt(
   KeyId=MASTER_KEY_ARN,
   Plaintext=password
)
print ("Cipher password = %s" % base64.b64encode(ret['CiphertextBlob']))
```

```
#
# Decrypt a ciphered text
#
ret = kms.decrypt(
   CiphertextBlob=ret['CiphertextBlob’]
)
print (f"Plaintext password = {ret['Plaintext']}")
```

https://github.com/sebsto/kms-demo

Bring Your Own Key



Create a customer master key (CMK)
KMS
create
Empty CMK with unique Key ID
Download a public wrapping key
KMS
download
Public RSA Key
Export your secret key, crypted with Public Key
Your key management infrastructure
export
Your key encrypted with KMS Pubic Key
Import the crypted key as CMK, with existing Key ID
import
Your crypted key in KMS

Moneta

Goal: Have automated scalable and on demand infrastructure while keeping data privacy and security, as that is the key for financial institutions.

A Bank in the Cloud!

- 1 Million Czech consumer and business customers.
- €400M revenue.
- 3200 employees and 190 branch offices.

Managing their user access and event logging is crucial

- Removed generic access for their users and have enabled MFA.
- Use of CloudTrail and VPC flow logs for all accounts. With all data being stored in a central account.

Data Protection

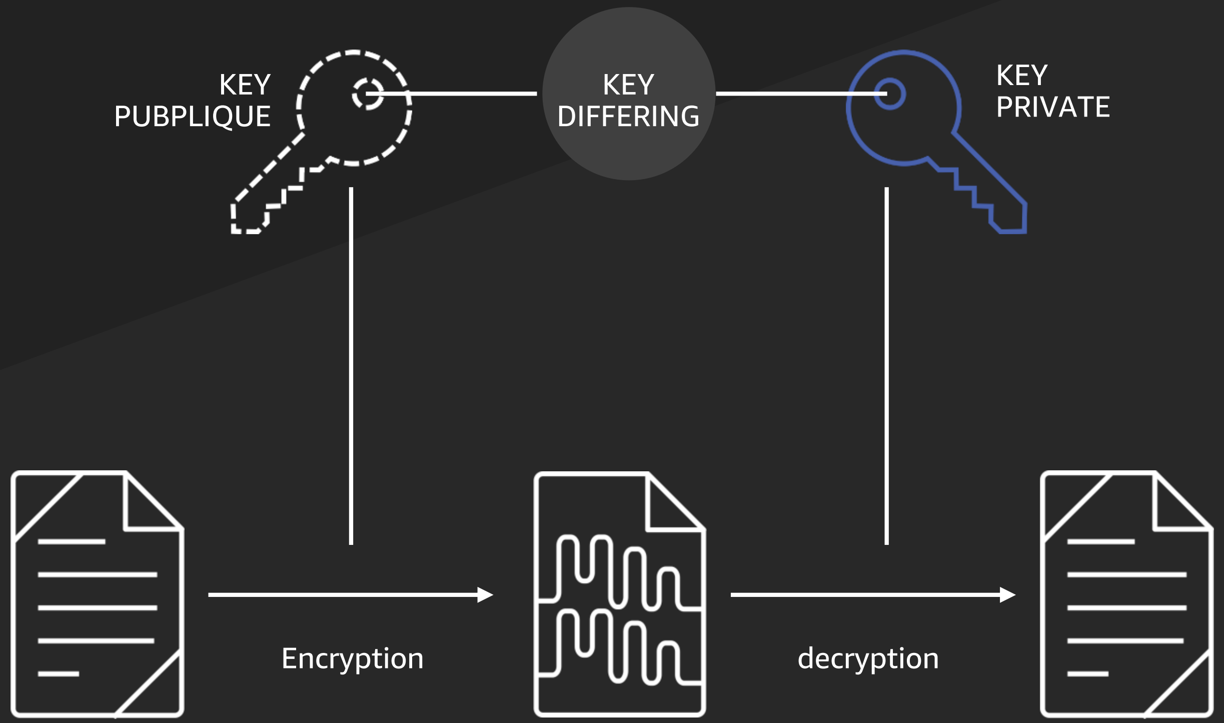
- Encrypted data at rest and in transit by their own keys.
- Keys generated by them are imported into KMS

New

Asymmetric keys management

- Digital signature using
  RSA keys or Elliptic Curve (ECC)

- Encryption using
  RSA asymmetric keys.



KMS Signatures Example (Python)

```
#
# Sign a piece of text
#
print('Signing a simple text ‘)
response = kms.sign(
    KeyId=SIGNATURE_KEY_ARN,
    Message=MESSAGE_TO_SIGN,
    MessageType='RAW’,
    SigningAlgorithm='RSASSA_PSS_SHA_256'
)

signature = response['Signature']
```

```
#
# Verify signature
#
response = kms.verify(
    KeyId=SIGNATURE_KEY_ARN,
    Message=MESSAGE_TO_SIGN,
    MessageType='RAW’,
    Signature=signature,
    SigningAlgorithm='RSASSA_PSS_SHA_256'
)

if response['SignatureValid'] == True:
        print('Signature is valid')
```

What else can AWS KMS do besides key management?

AWS KMS gives you an additional mechanism for access control and data protection

AWS KMS integration with AWS services provides you with a robust set of audit records for data events

AWS KMS provides operational assurance for you to answer critical questions from stakeholders

AWS CloudHSM

AWS CloudHSM Delivers

Single-tenant FIPS 140-2 Level 3 validated HSMs in your VPC

Zero config high-availability and one-click cluster expansion

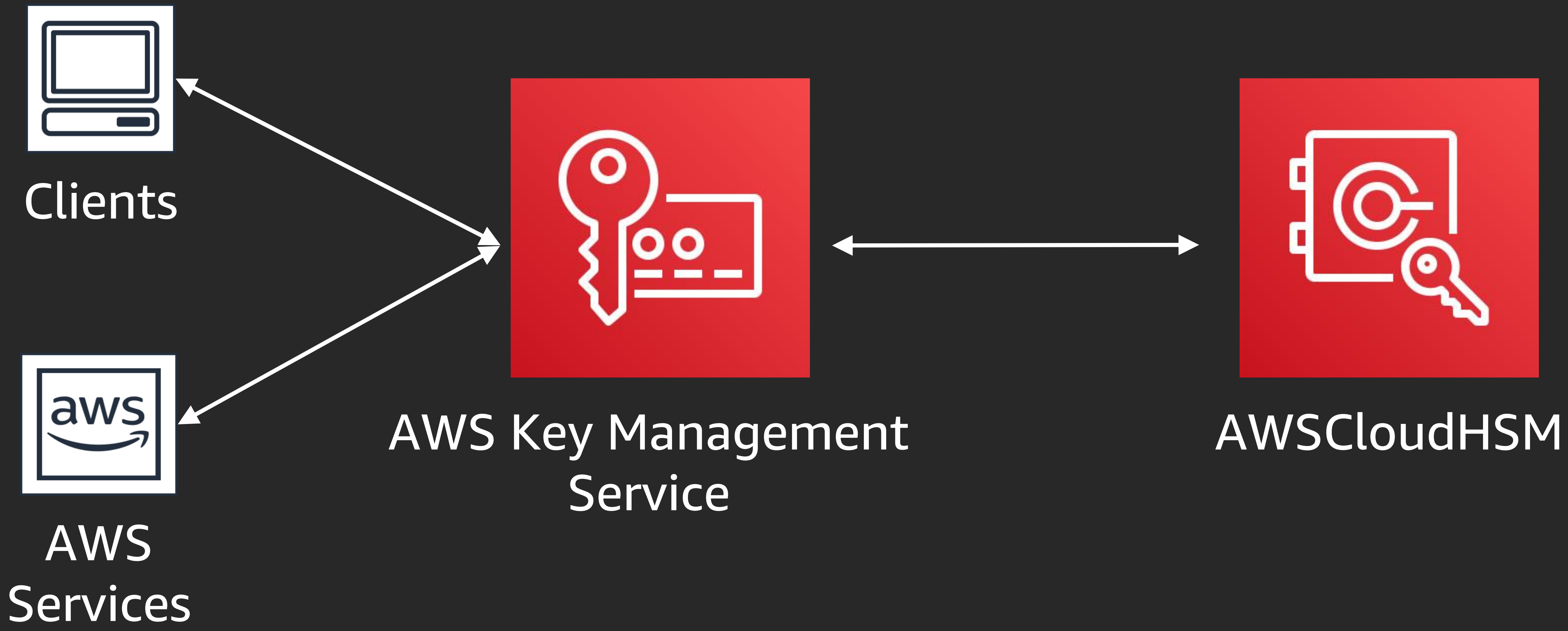
Audit logs and HSM metrics to Amazon CloudWatch

Durability of backups

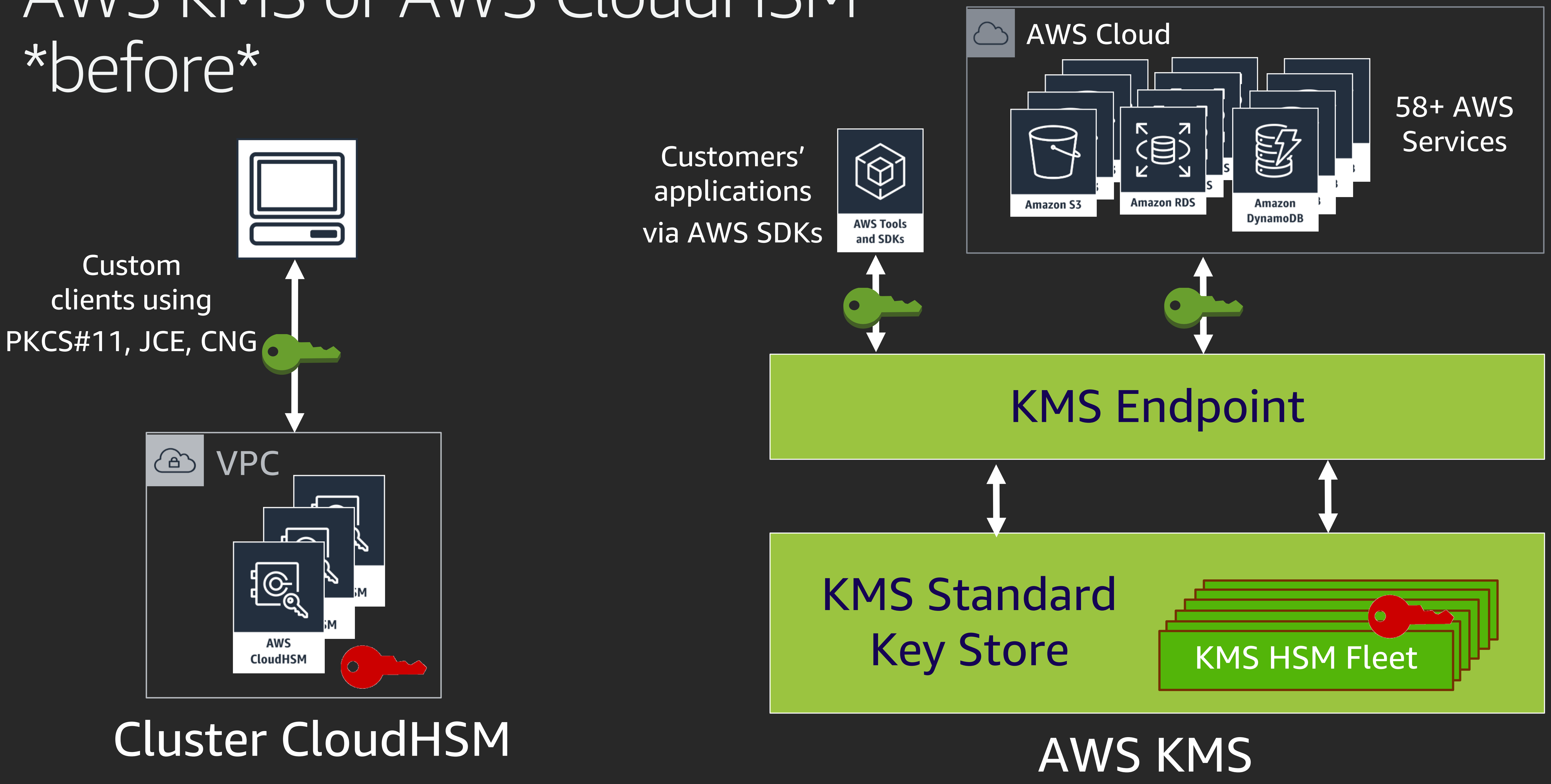
Features and performance

- ~1100 SSL offload operations per second
- AES at speed of your network
- Capacity of storage : ~3500 keys
- Scale up up to 32 HSMS per cluster
- Up to 32000 RSA signature/verification in a VPC cluster
- Scale down to zero, restore from backup

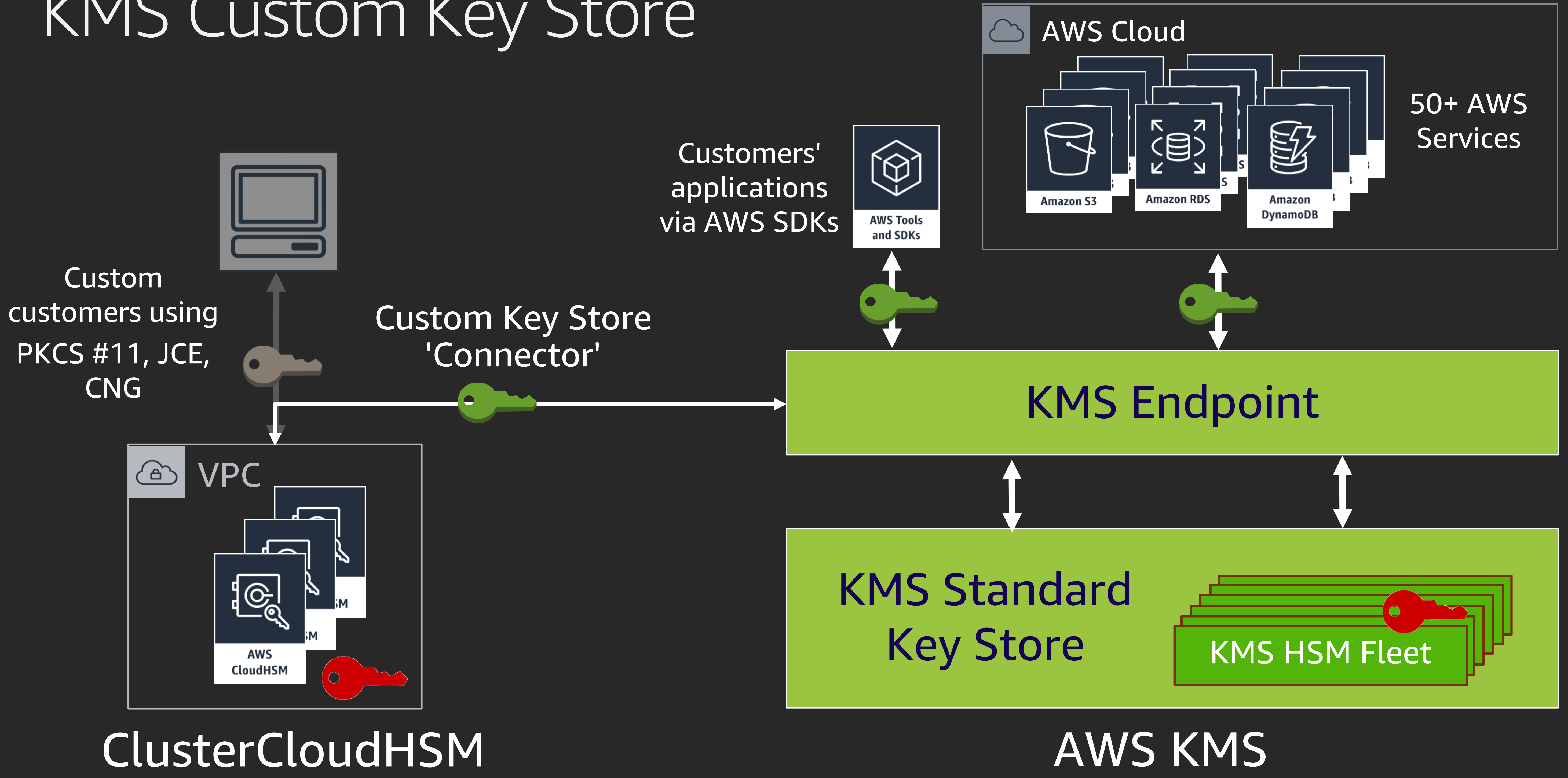
AWS KMS custom key store



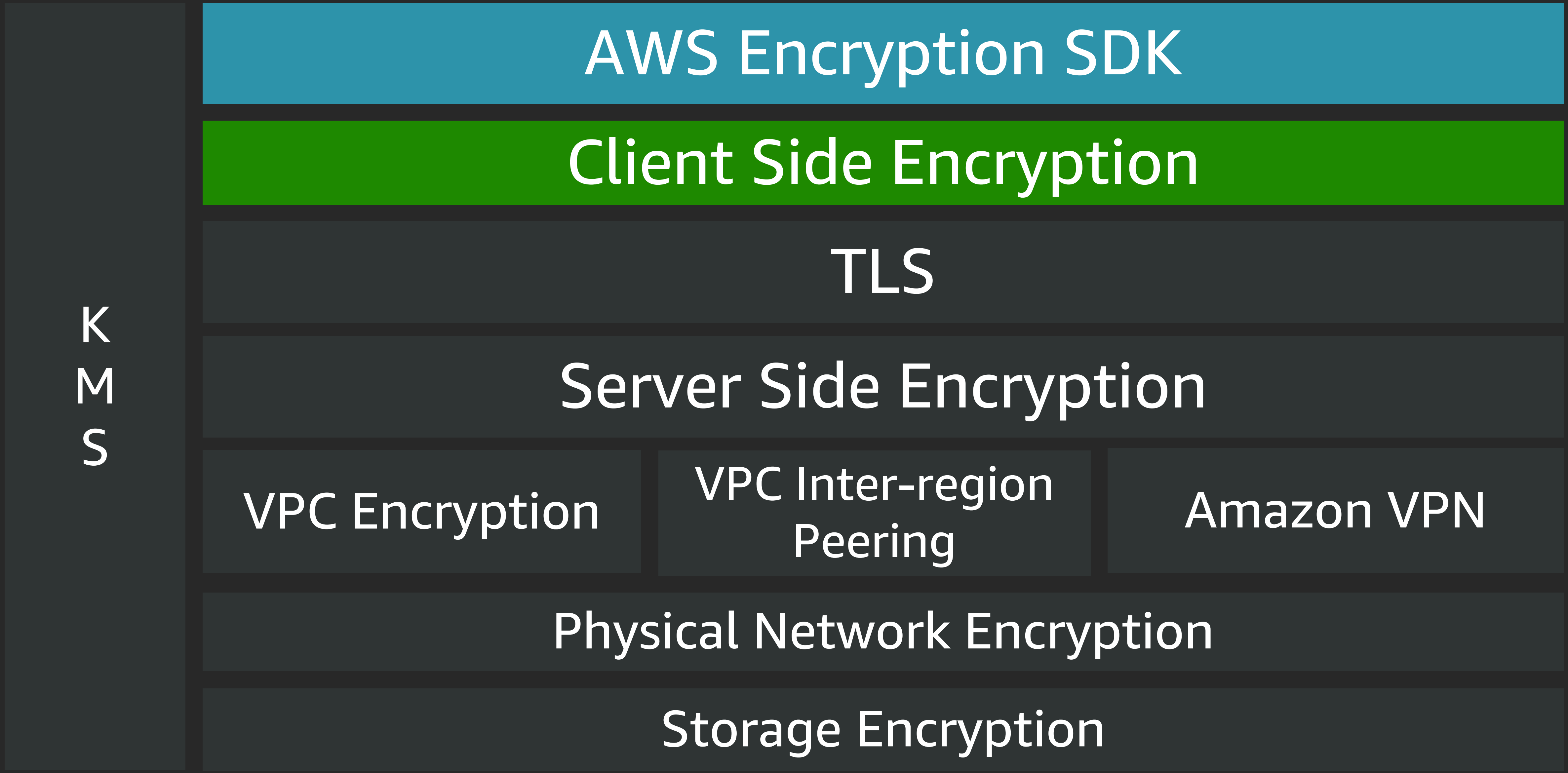
AWS KMS or AWS CloudHSM *before*



KMS Custom Key Store



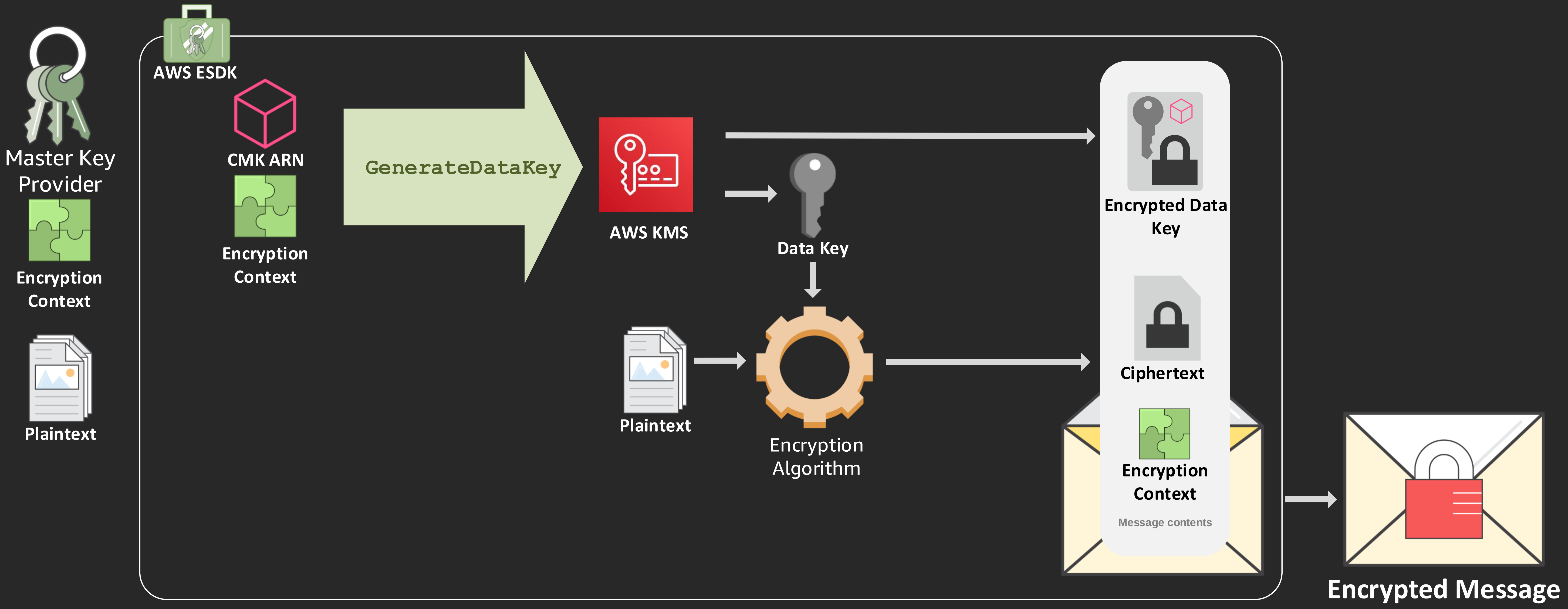
Client Side Encryption



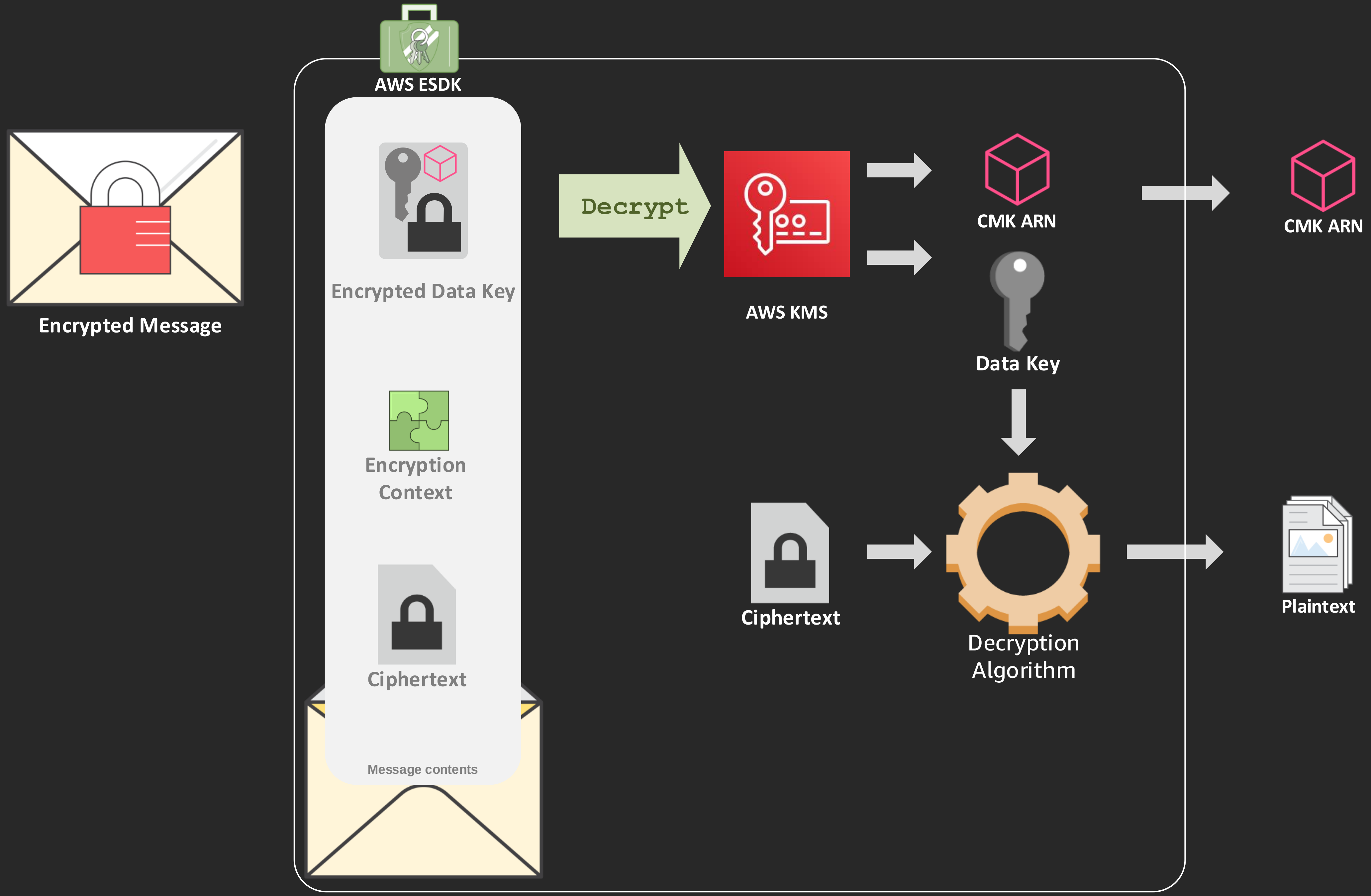
AWS Encryption SDK

- APIs and **data format** for the encryption
- Interface simplified with AWS KMS for the **encryption in envelope**
- **Open-Source**, open-specification, Apache 2.0
- Multi Languages
  - **AWS Encryption SDK for Java**
  - **AWS Encryption SDK for Python**
  - **AWS Encryption SDK for C**
  - **AWS Encryption SDK for JavaScript and Node.js**
- **Multiple** master keys and **data key caching** built in

Encrypting with the AWS Encryption SDK



Decrypting with the AWS Encryption SDK



https://github.com/aws/aws-encryption-sdk-python

Encryption SDK Examples (Python)

```python
#
# The MKP object contains reference to master keys
#
mkp = aws_encryption_sdk.KMSMasterKeyProvider(
   key_ids=[MASTER_KEY_ARN]
)
encryption_context = {"data_type": "example", "classification": "public"}

#
# Let's encrypt the plaintext data
#
ciphertext, encryptor_header = aws_encryption_sdk.encrypt(
   source=plaintext,
   key_provider=mkp,
   encryption_context=encryption_context
)
```

Encryption SDK Examples (Python)

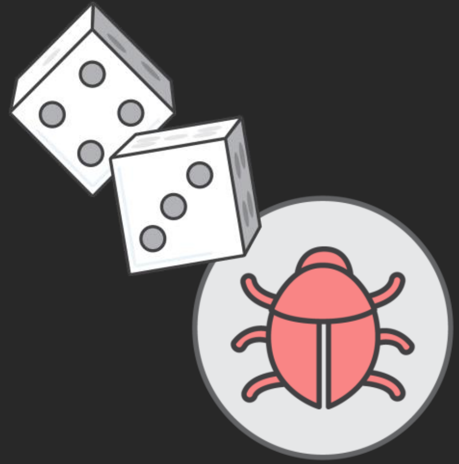
```python
#
# Let's decrypt the ciphertext data
#
decrypted_plaintext, decryptor_header = aws_encryption_sdk.decrypt(
   source=ciphertext,
   key_provider=mkp
)

print(decrypted_plaintext.decode("utf-8"))
```

AWS Secret Manager

AWS Secrets Manager

- Lifecycle management of secrets such as database credentials and API keys
- Controlling access to secrets is as important as controlling access to data
- Integrated with AWS IAM and other services
- Manipulating secrets by a large number of people creates risks and vulnerabilities.
- Automated secret rotation ensures security and availability
- Integration with Amazon RDS for updating client and database server credentials



AWS Secrets Manager

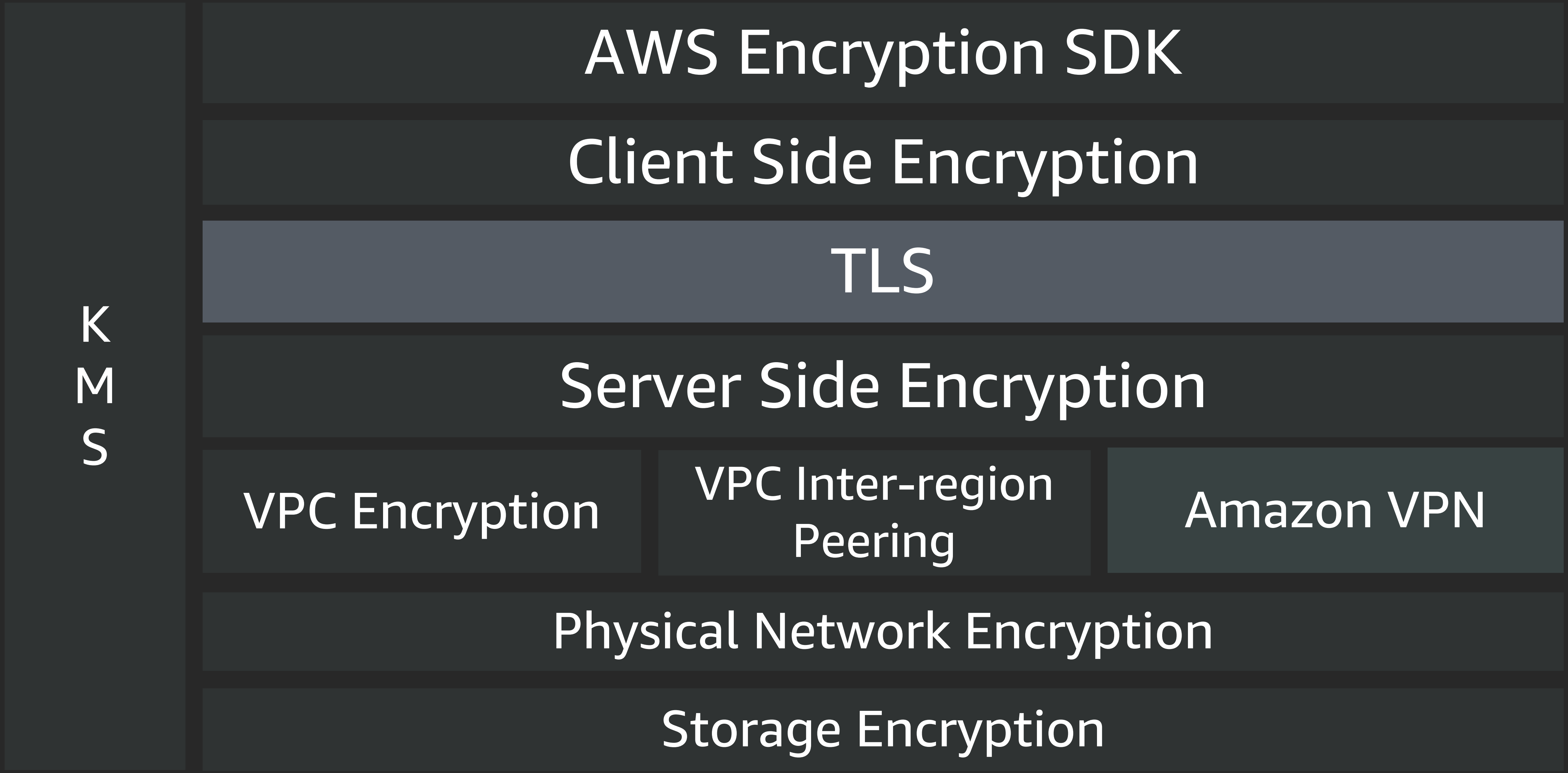
Retrieve and manage secrets such as database credentials and API keys throughout their life cycle

Automatic rotation of secrets

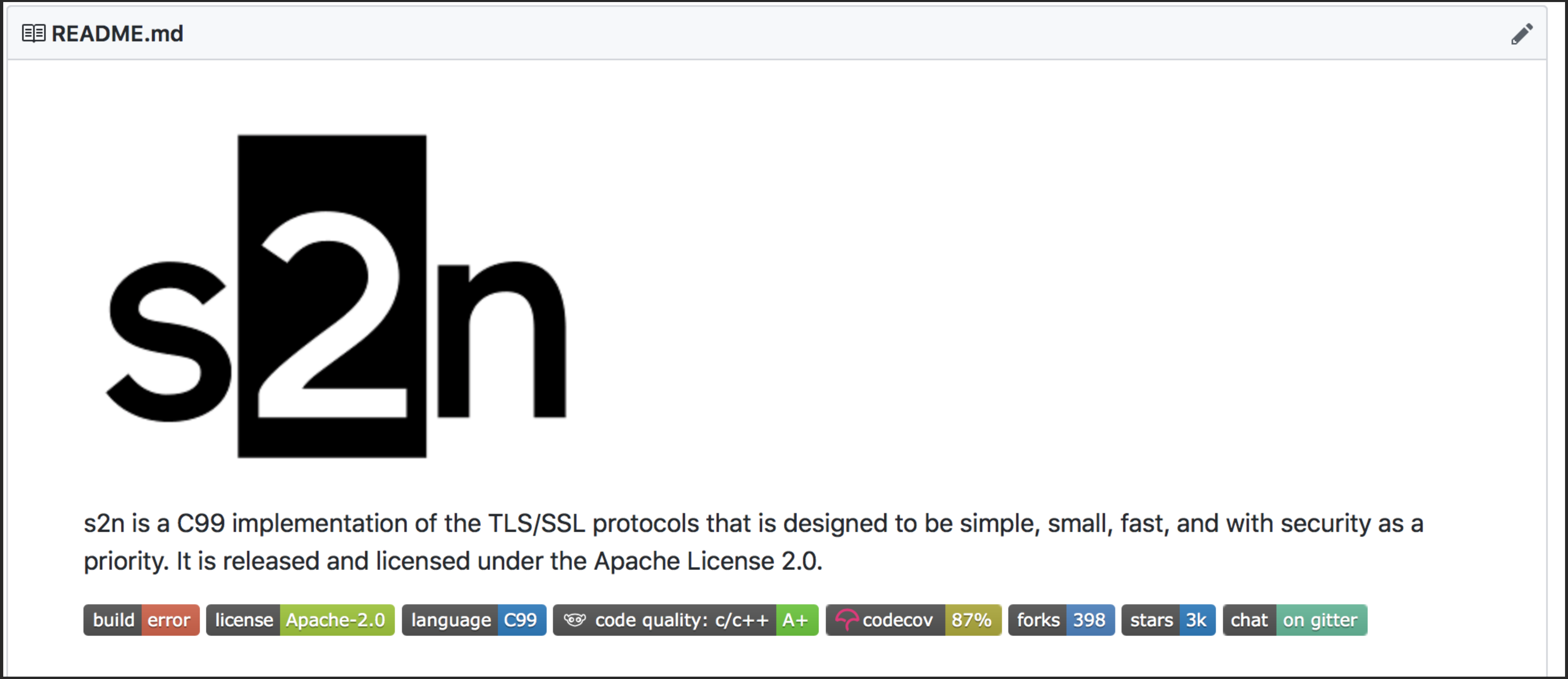
Endpoint VPC supports policies

In-scope for SOC, HIPAA, PCI, and ISO

In Transit Encryption



TLS



README.md

s2n is a C99 implementation of the TLS/SSL protocols that is designed to be simple, small, fast, and with security as a priority. It is released and licensed under the Apache License 2.0.

AWS Certificate Manager

Certificate Manager provision, manage, deploy and update SSL/TLS certificates on AWS Cloud

Authentication

AWS Certificate Manager Benefits

Free to use public SSL/TLS certificates on AWS Services

Centralized Management

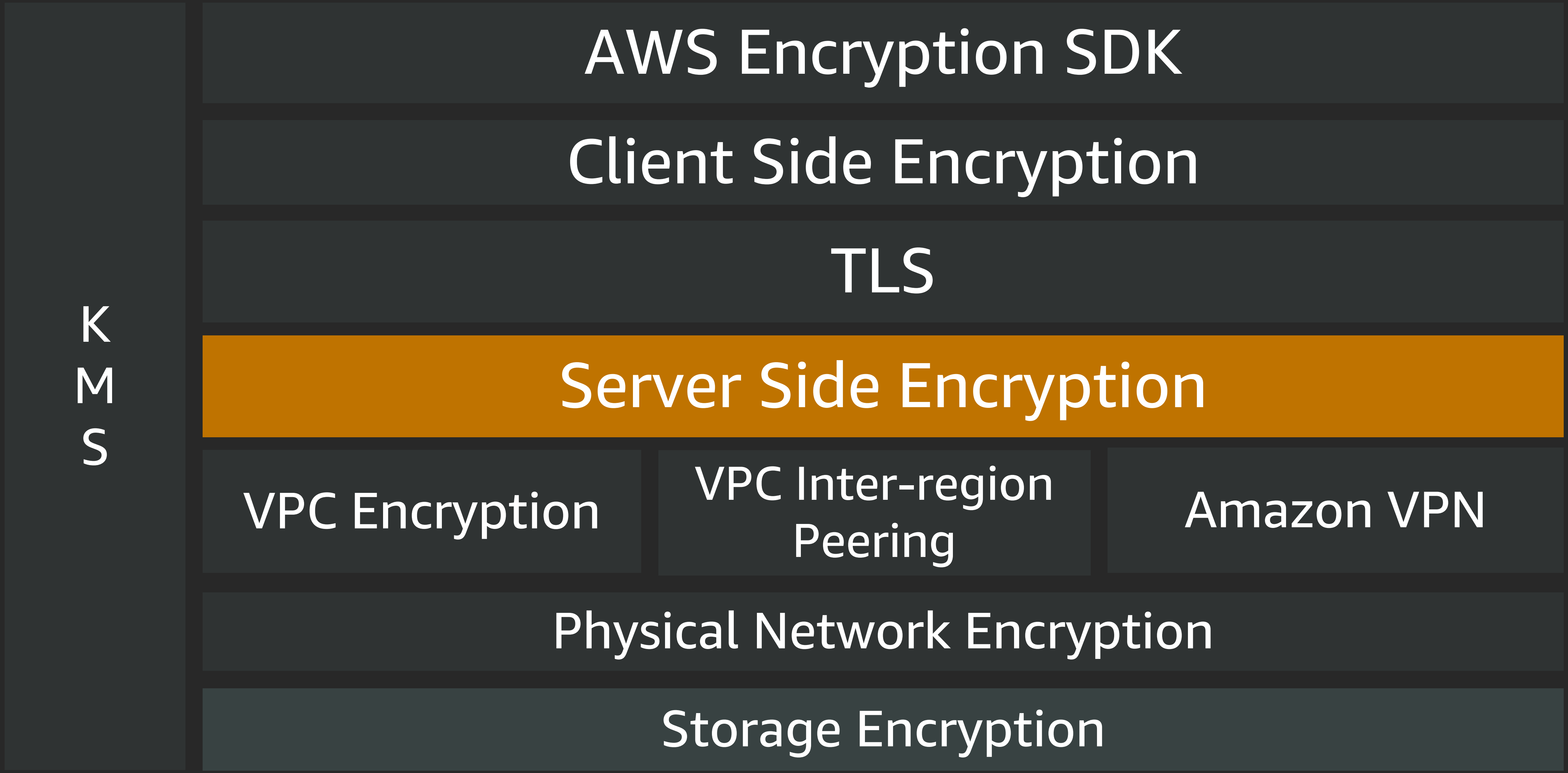
Private Certification Authority

Agility

Private certificate

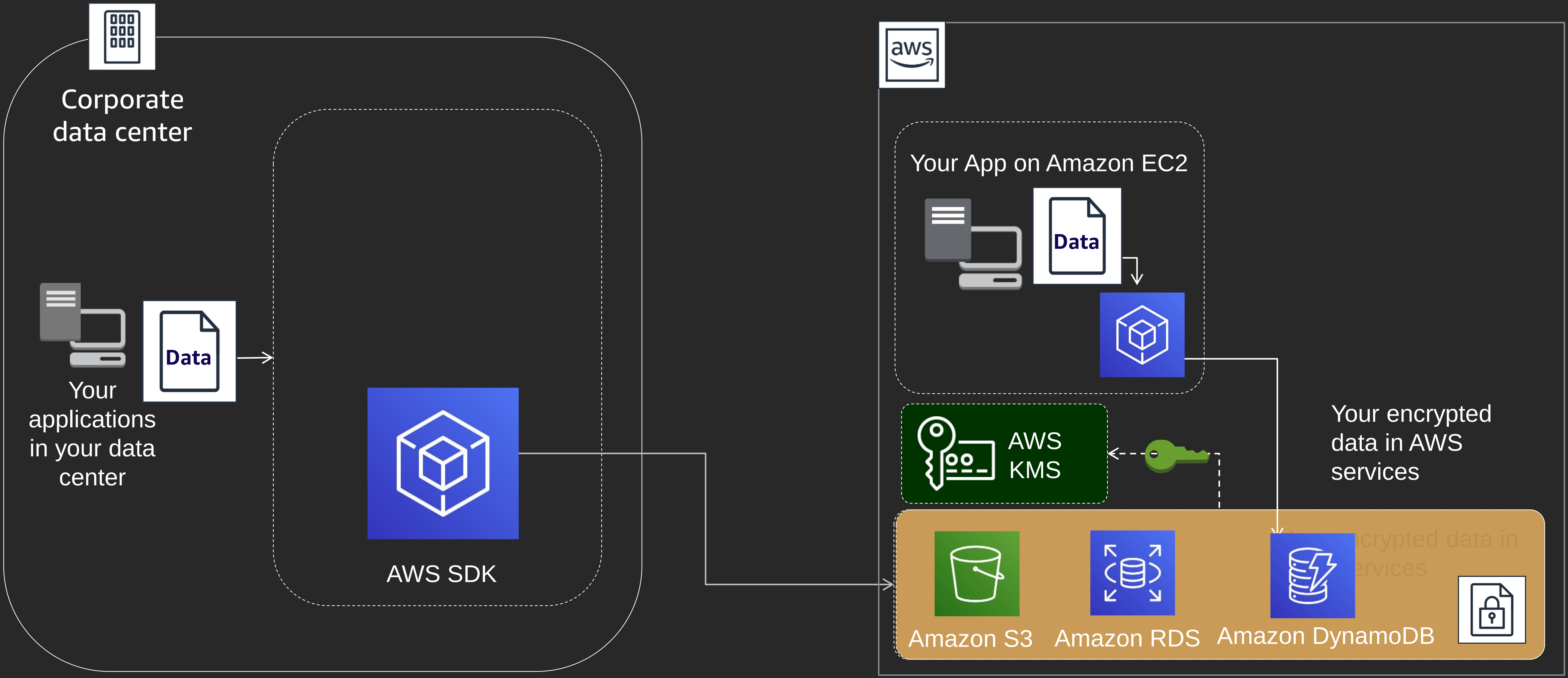
On demand pricing

Server-side Encryption

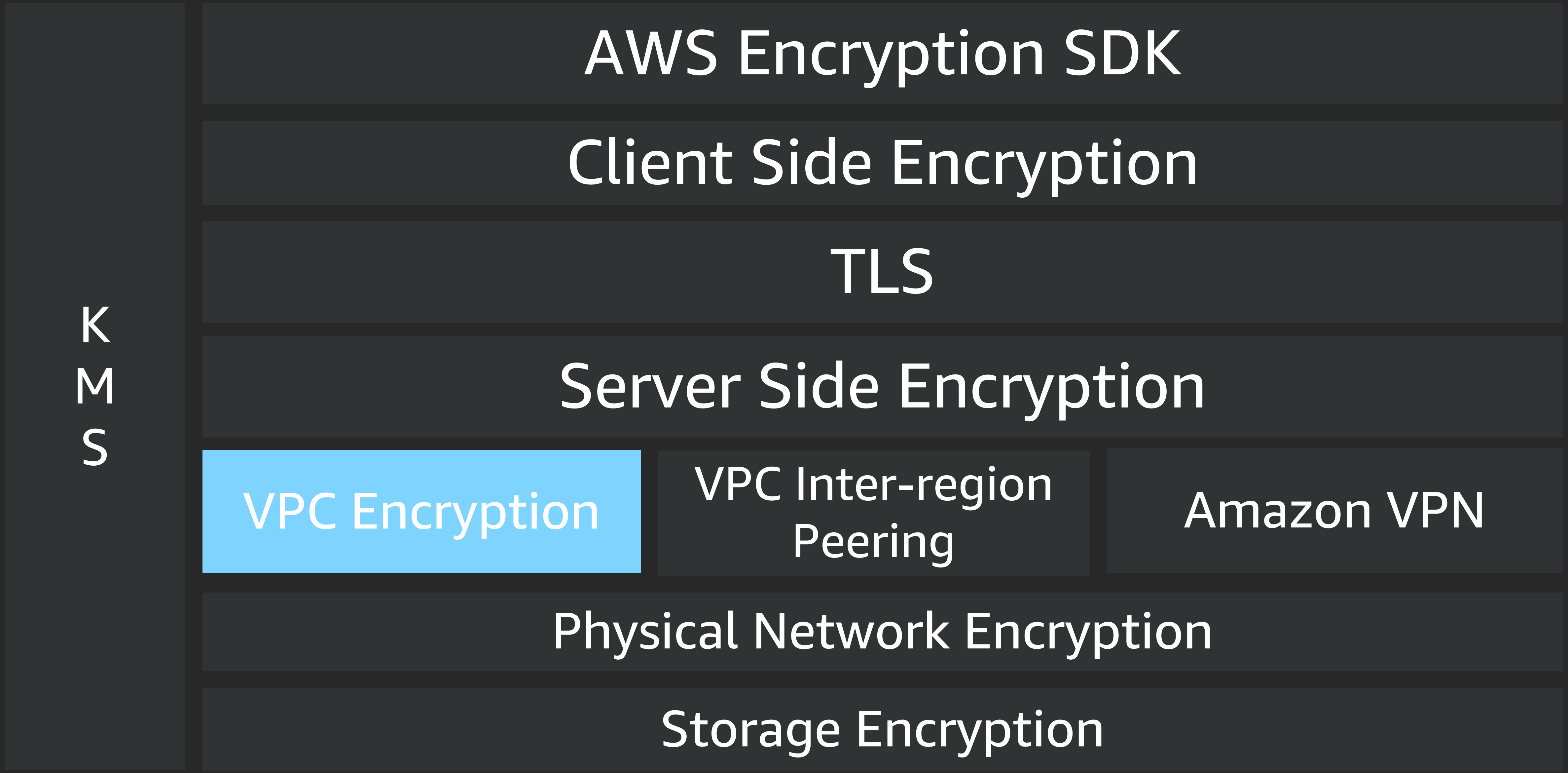


Server side Encryption

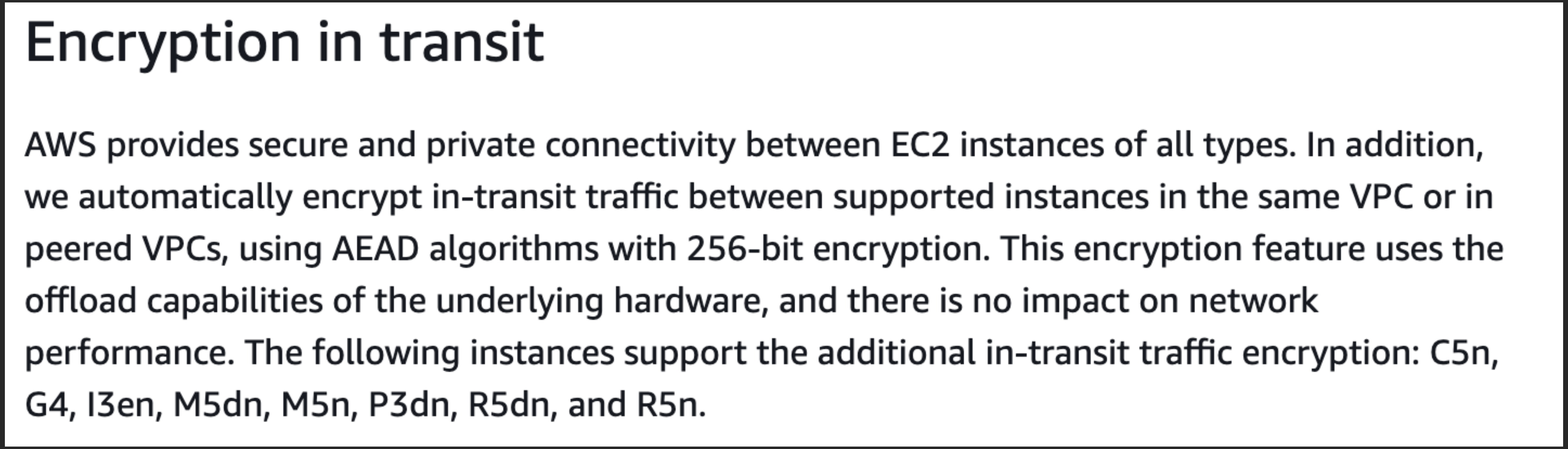
AWS Services encrypt data as soon as the service receive them



VPC Encryption



VPC encryption

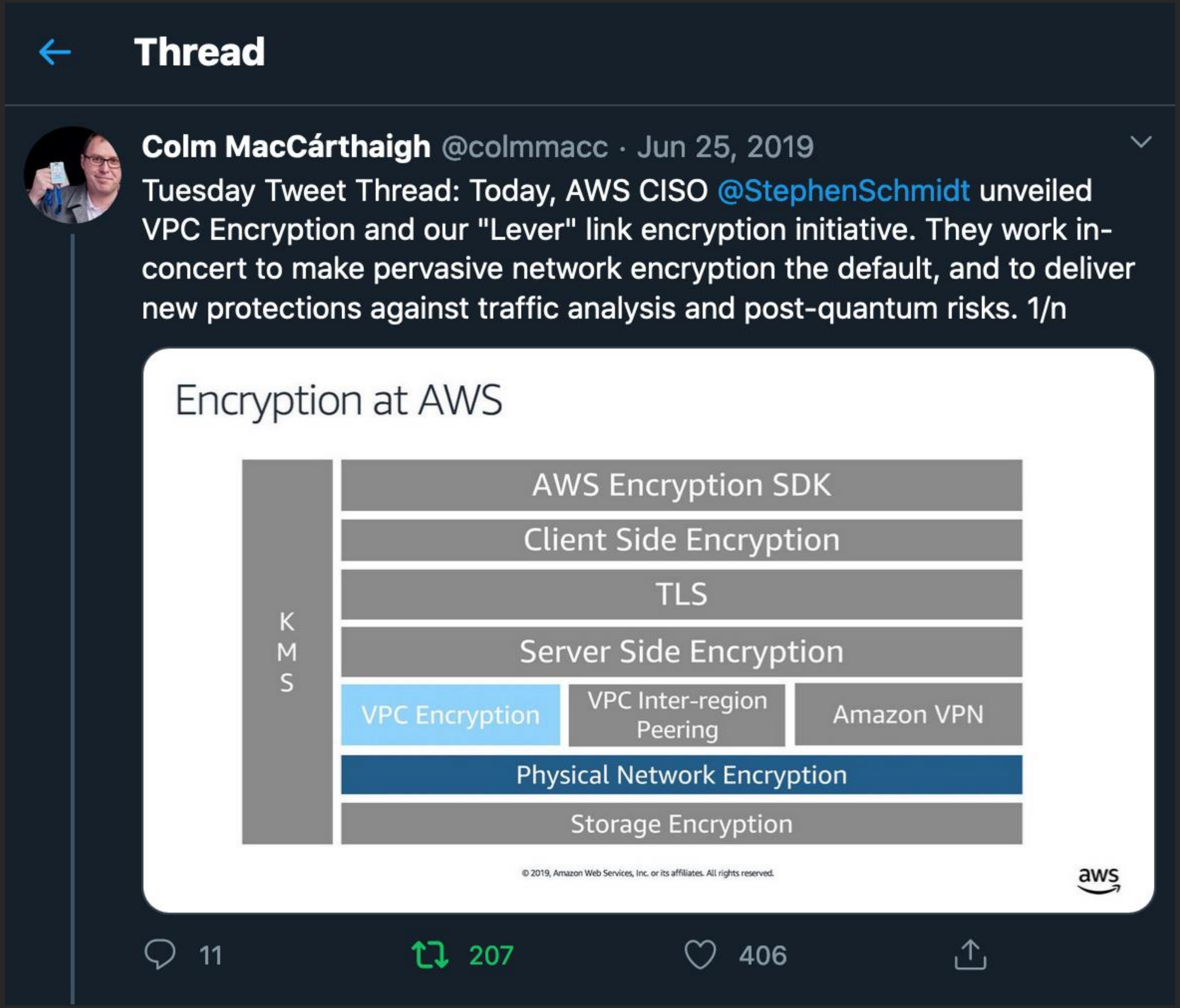


Encryption in transit

AWS provides secure and private connectivity between EC2 instances of all types. In addition, we automatically encrypt in-transit traffic between supported instances in the same VPC or in peered VPCs, using AEAD algorithms with 256-bit encryption. This encryption feature uses the offload capabilities of the underlying hardware, and there is no impact on network performance. The following instances support the additional in-transit traffic encryption: C5n, G4, I3en, M5dn, M5n, P3dn, R5dn, and R5n.

- Implemented in AWS hardware, by Annupurna Labs, as part of Nitro
- We encrypt your data AND our network virtualization protocol
- Encryption is applied within and between availability zones
- Forward-secrecy for between hours and one day

https://twitter.com/colmmacc/status/1143569907369594881



Thread

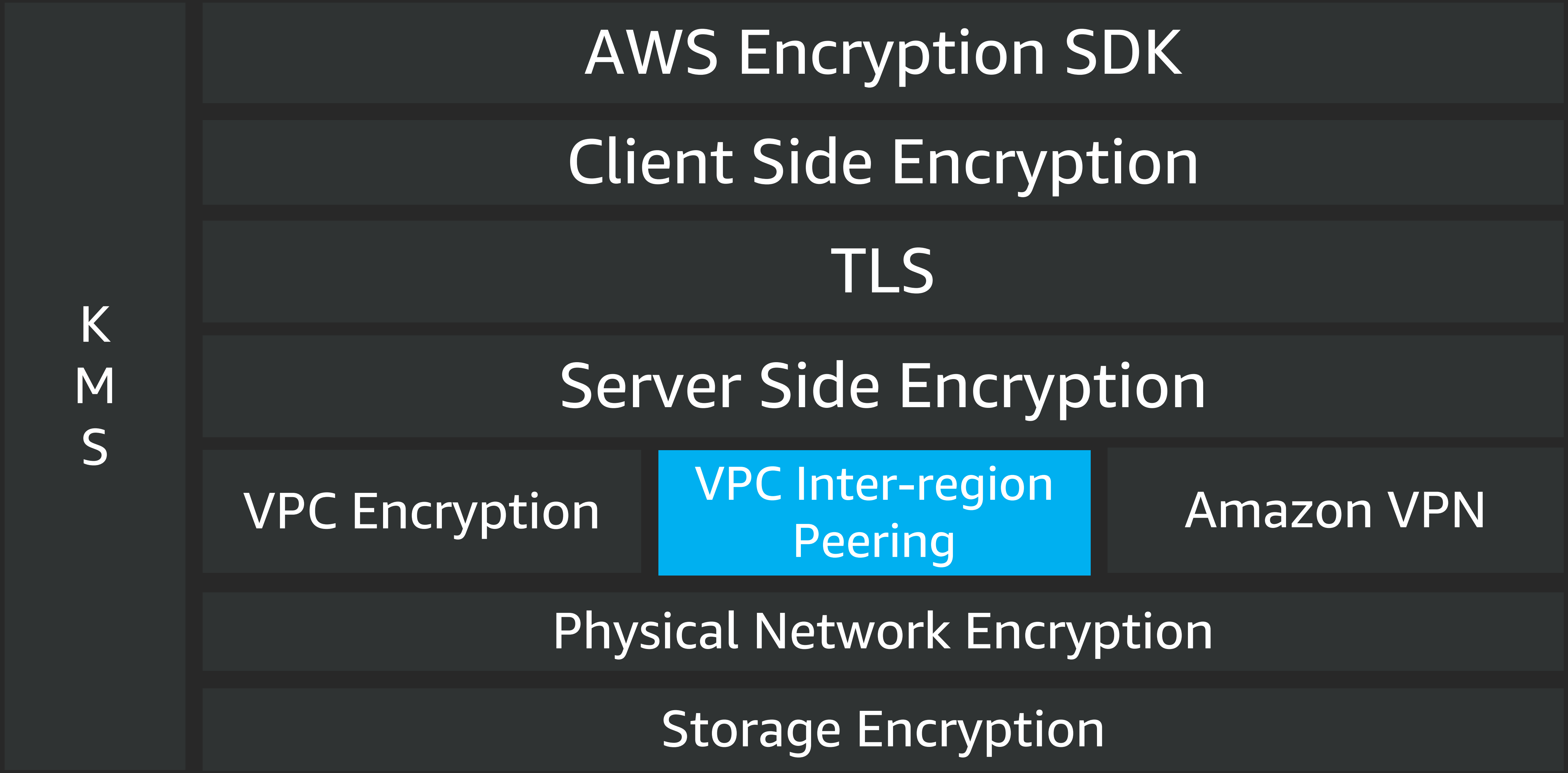
Colm MacCárthaigh @colmmacc · Jun 25, 2019

Tuesday Tweet Thread: Today, AWS CISO @StephenSchmidt unveiled VPC Encryption and our "Lever" link encryption initiative. They work in-concert to make pervasive network encryption the default, and to deliver new protections against traffic analysis and post-quantum risks. 1/n

Encryption at AWS

11 207 406

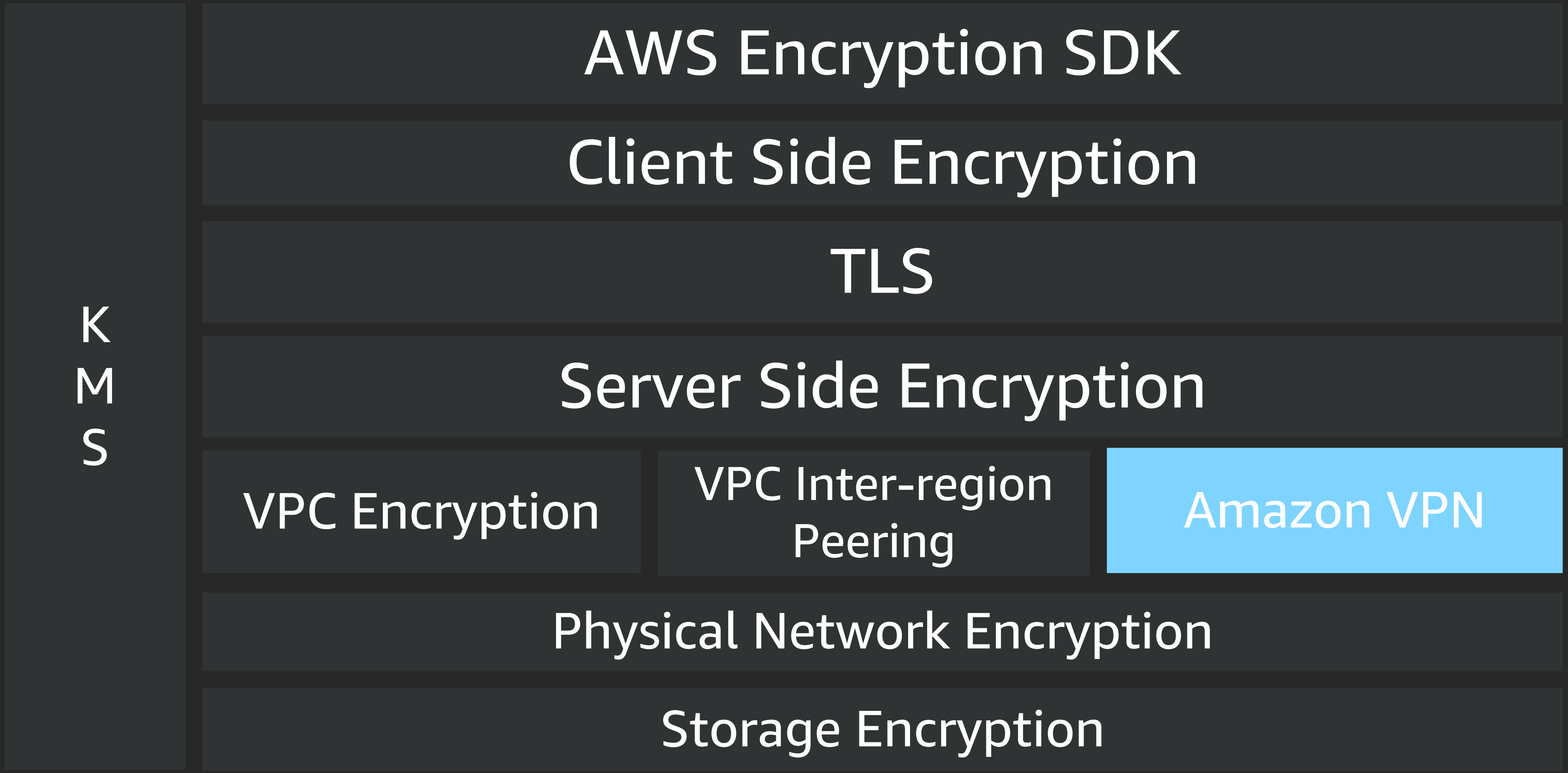
VPC inter-region peering



VPC inter-region peering

- Allows peering and exchange of traffic between VPCs in different regions
- Implemented on our ”Blackfoot” layer of edge devices
- We encrypt your data AND our network virtualization protocol
- Forward-secrecy for between hours and one day

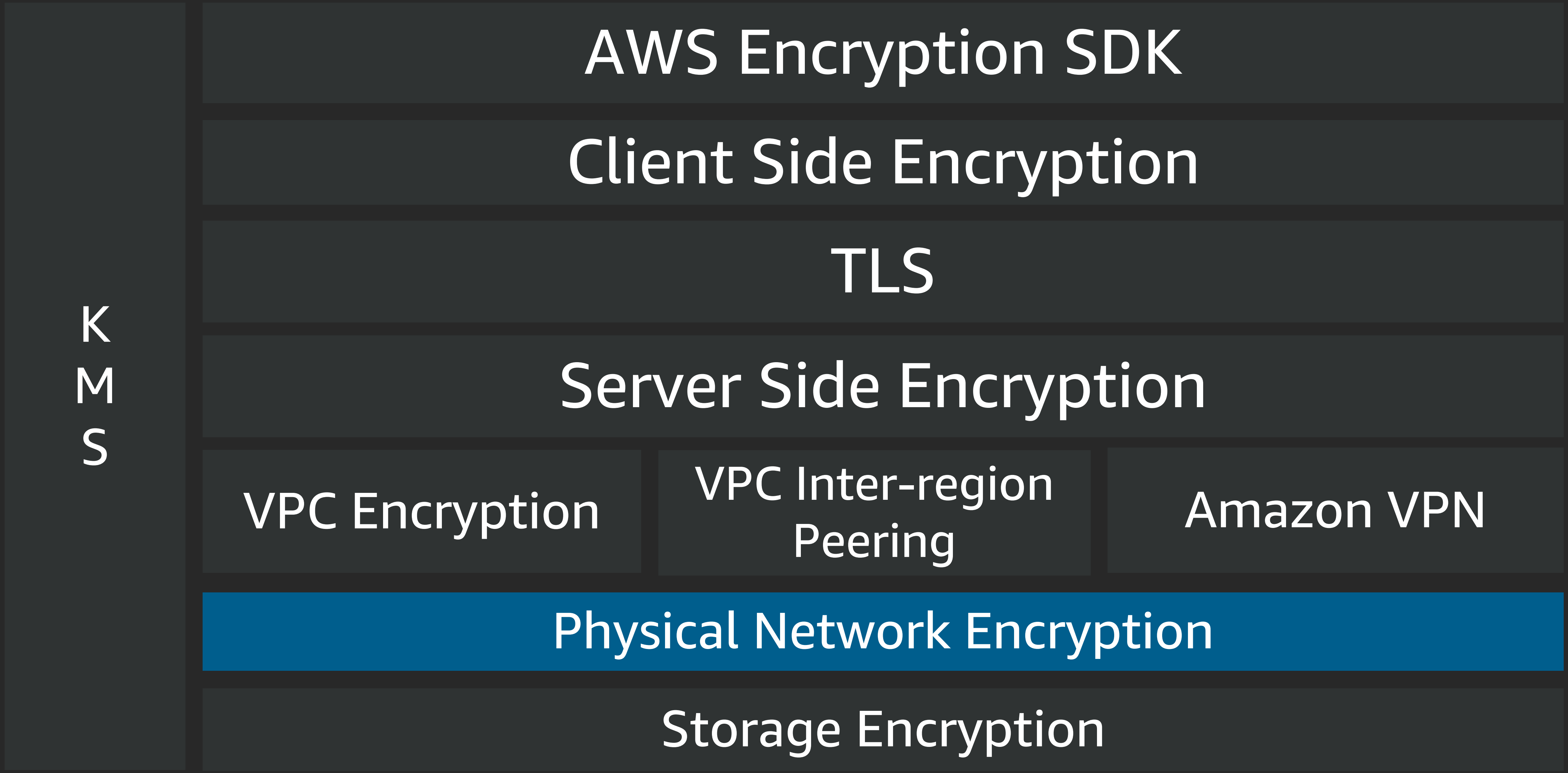
Amazon VPN


K M S
AWS Encryption SDK
Client Side Encryption
TLS
Server Side Encryption
VPC Encryption
VPC Inter-region Peering
Amazon VPN
Physical Network Encryption
Storage Encryption

Amazon VPN

- Site-to-site and client VPN options
- Based on the IPSec and OpenVPN protocols
- Both include per-session forward secrecy that lasts ~hours
- If you’re using site-to-site VPN: watch out for “Group 2”

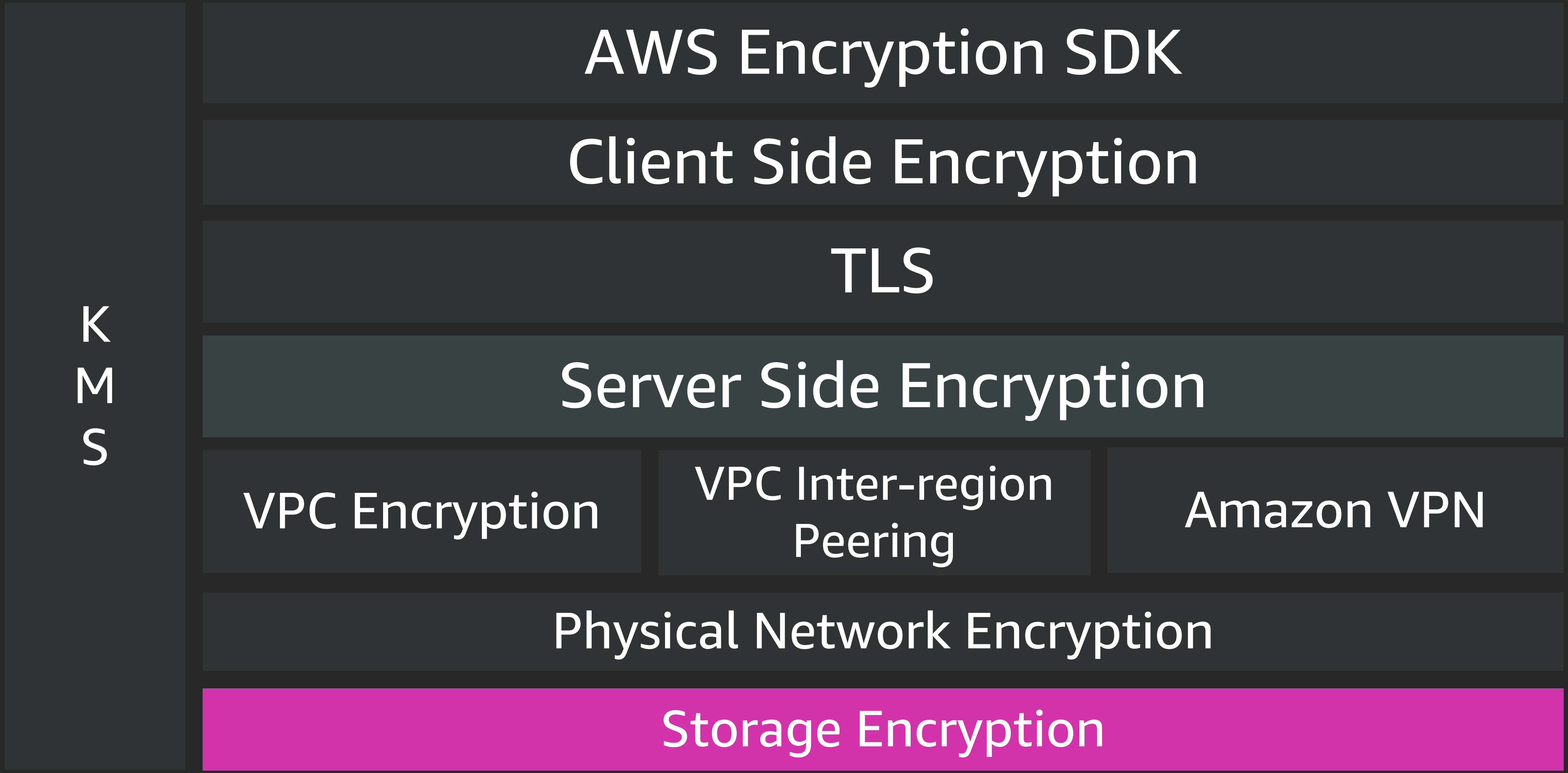
Physical network encryption



Physical network encryption

- Any link outside of AWS physical control, including between AWS datacenters, and the AWS backbone is protected
- Reminder: all traffic between AWS regions (except China) is carried on the AWS backbone
- Most links are protected with MACSEC and Optical encryption using AES-256
- Small number of short-distance links use laser monitoring

Storage Encryption



Storage Encryption

- Server-side encryption for all storage type: Block, File, Object
  - Amazon EBS
  - Amazon EFS
  - Amazon FSX for Lustre
  - Amazon FSX for Microsoft
  - Amazon S3
- EBS Encryption at hardware level with Nitro
- Memory encryption for Amazon Ec2 instances powered by AWS Graviton2 ARM processors

Let’s encrypt everything, really everything.

aws SUMMIT ONLINE

APN Security Competency Partners

accenture
AllCloud
Check Point SOFTWARE TECHNOLOGIES LTD
CloudHealth by vmware
CLOUDZONE by matrix
DATADOG
Deloitte.
New Relic
paloalto NETWORKS
REPLY STORM
SENTIA LEAD THE WAY
TREND MICRO

Visit the Partner Discovery Zone to meet these partners and view the full list of APN Competency Partners

Let’s Encrypt Everything

Physical Layer	Secured infrastructure and AES-256 encryption
Data Link	MACsec AES-256 (IEEE 802.1AE)
Network Layer	VPC Encryption | Cross-Region Peering | Amazon VPN
Transport Layer	Amazon s2n | NLB-TLS | ALB | CloudFront | ACM
Application Layer	AWS Crypto SDK | KMS | CloudHSM | Secret Manager

Thank you !

Sebastien Stormacq

@sebsto