AWS re:Invent

NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

SEC403

Protecting secrets, keys, and data: Cryptography for the long term

Peter M. O’Donnell (he/him)

Principal Security Solutions Architect
Strategic Accounts
AWS

Matthew Campagna (he/him)

Senior Principal Security Engineer
AWS Cryptography
AWS

What we are talking about today

- TLS versions, cipher suites
- Secrets Manager
- CloudHSM
- AWS KMS and the new XKS
- Encryption SDK
- What’s happening with post-quantum cryptography
- Cryptographic computing

TLS versions and cipher suites

- TLS is understood with two primary vectors: versions and cipher suites
- Versions are the protocol itself: how sessions are established and maintained
- Cipher suites are the various cryptographic building blocks that are used for the TLS session
- Cipher suites in TLS 1.2 have four components: key exchange, signature, bulk encryption, message authentication

Change is coming to AWS TLS endpoints

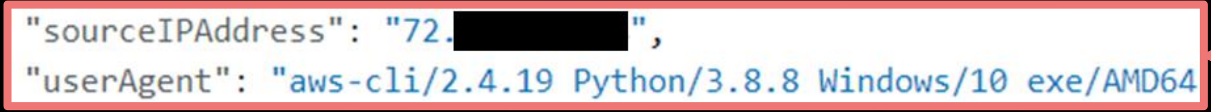
- FIPS endpoints in US East & West *already* require TLS 1.2+ as of March 2022
- All AWS endpoints are being *uplifted* to require 1.2+ next year – **June 2023**
- AWS SDKs & AWS CLI v2 support TLS 1.2+
- Most services show TLS version in CloudTrail

Analyze your API calls in CloudTrail

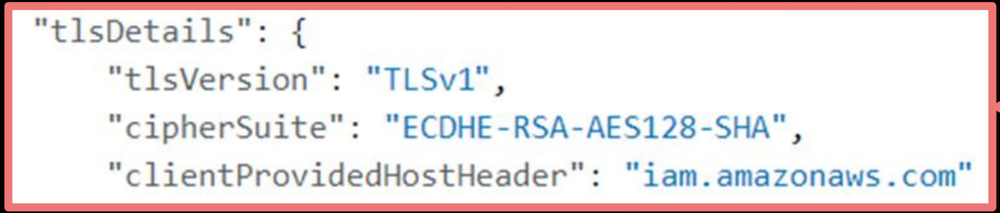
DETERMINE IMPACT

Directly examine event records in CloudTrail for supported services

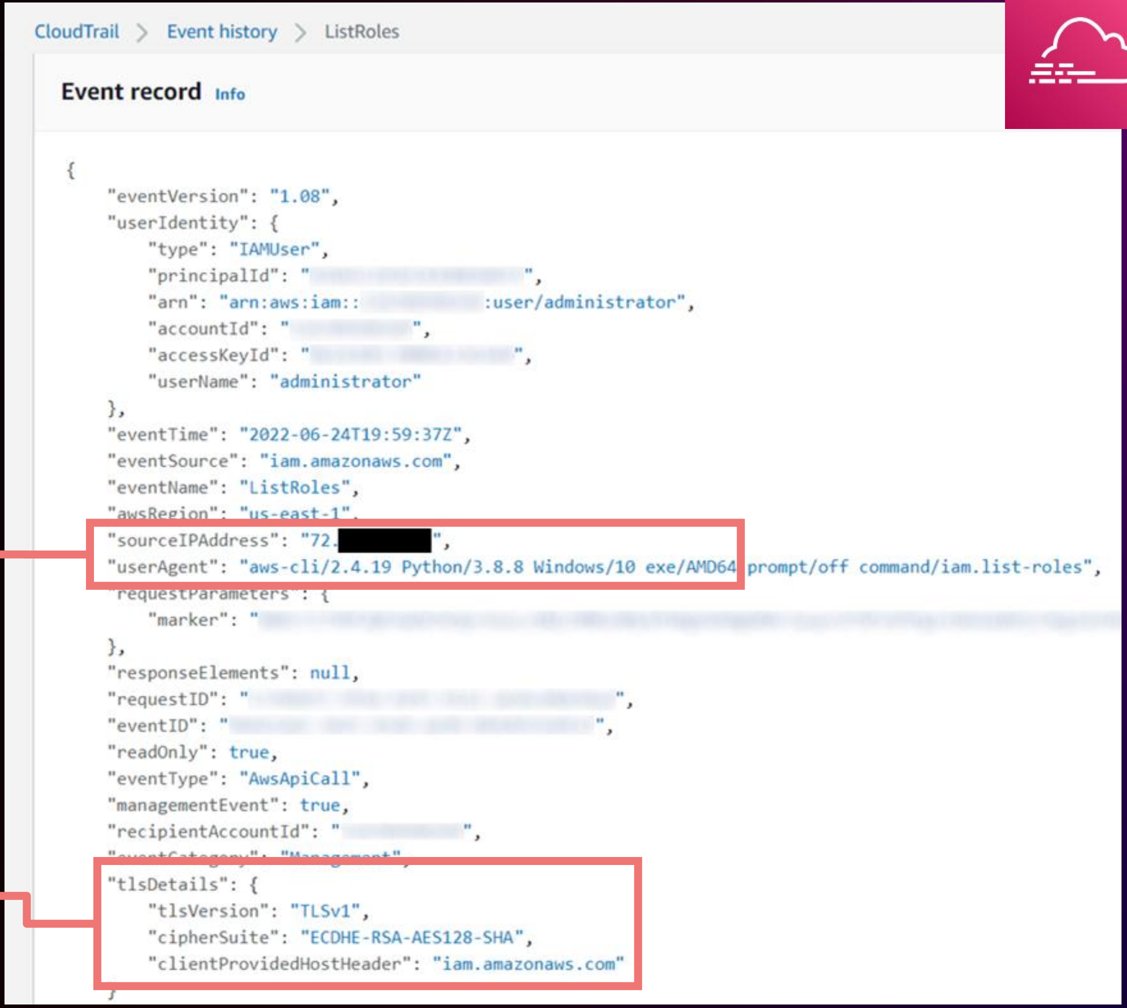
List of supported services: https://go.aws/3zrVTos



```
"sourceIPAddress": "72.[redacted]",
"userAgent": "aws-cli/2.4.19 Python/3.8.8 Windows/10 exe/AMD64
```



```
"tlsDetails": {
    "tlsVersion": "TLSv1",
    "cipherSuite": "ECDHE-RSA-AES128-SHA",
    "clientProvidedHostHeader": "iam.amazonaws.com"
```



Event record

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "[redacted]",
        "arn": "arn:aws:iam::[redacted]:user/administrator",
        "accountId": "[redacted]",
        "accessKeyId": "[redacted]",
        "userName": "administrator"
    },
    "eventTime": "2022-06-24T19:59:37Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "ListRoles",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.[redacted]",
    "userAgent": "aws-cli/2.4.19 Python/3.8.8 Windows/10 exe/AMD64 prompt/off command/iam.list-roles",
    "requestParameters": {
        "marker": "[redacted]
    },
    "responseElements": null,
    "requestID": "[redacted]",
    "eventID": "[redacted]",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "[redacted]",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1",
        "cipherSuite": "ECDHE-RSA-AES128-SHA",
        "clientProvidedHostHeader": "iam.amazonaws.com"
    }
```

Analyzing impact at scale

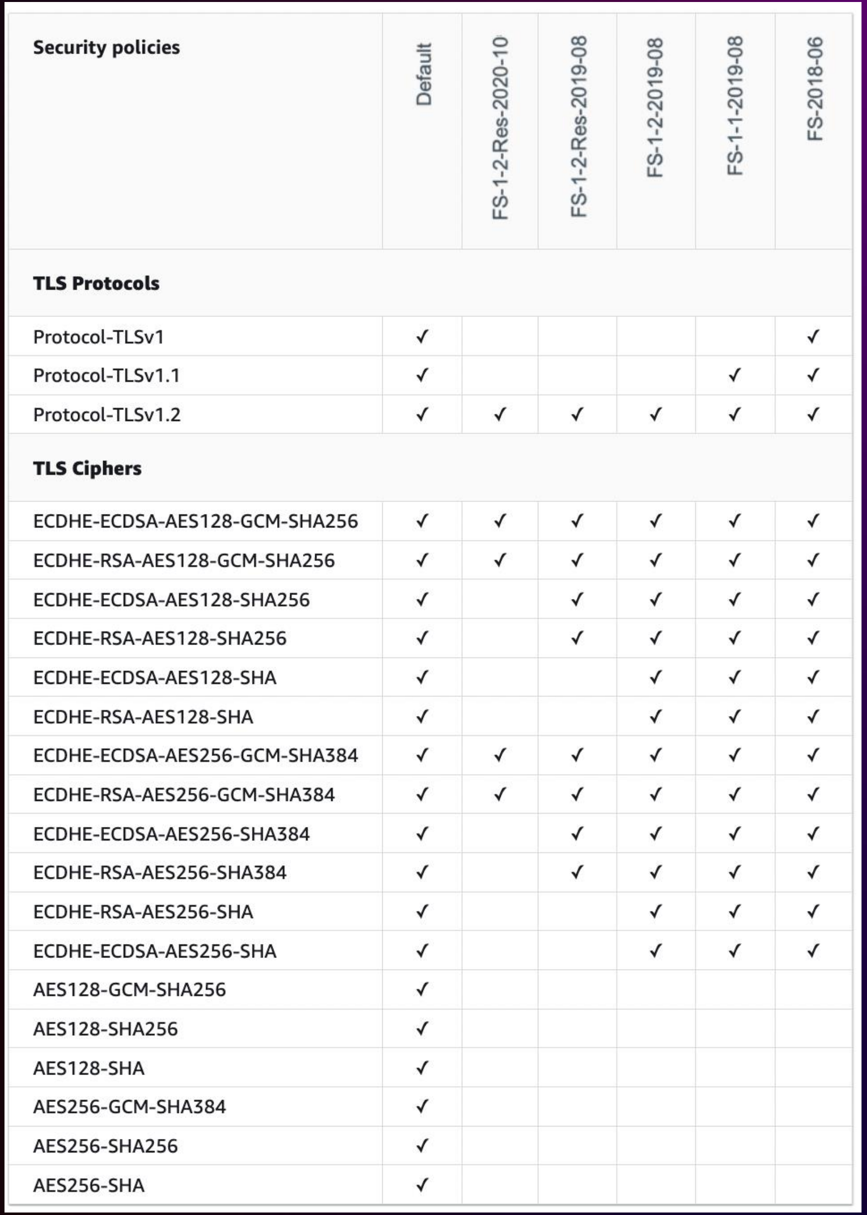
DETERMINE IMPACT

At large scale, there are multiple options to analyze your API calls, including:

AWS service	Method
AWS CloudTrail Lake	SQL queries https://go.aws/3O7a6ep
Amazon CloudWatch Logs Insights	Purpose-built queries https://go.aws/3PvDOLs
Aggregate AWS Health events	Use organizational view https://go.aws/3IXh1WV
AWS Health API automation	SDK/application code https://go.aws/3zceAfQ

What about your own endpoints on AWS?

- Clients, not the server, choose TLS preferences
- Server resources present what is possible/allowed to clients
- ELB & CloudFront support *security policies* for fine-grained preferences
- Balance compatibility with stringency
- Consider the threat model and network topology
- Narrowest policy is narrowest compatibility



Security policies	Default	FS-1-2-Res-2020-10	FS-1-2-Res-2019-08	FS-1-2-2019-08	FS-1-1-2019-08	FS-2018-06
TLS Protocols						
Protocol-TLSv1	✓					✓
Protocol-TLSv1.1	✓				✓	✓
Protocol-TLSv1.2	✓	✓	✓	✓	✓	✓
TLS Ciphers						
ECDHE-ECDSA-AES128-GCM-SHA256	✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES128-GCM-SHA256	✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA256	✓		✓	✓	✓	✓
ECDHE-RSA-AES128-SHA256	✓		✓	✓	✓	✓
ECDHE-ECDSA-AES128-SHA	✓			✓	✓	✓
ECDHE-RSA-AES128-SHA	✓			✓	✓	✓
ECDHE-ECDSA-AES256-GCM-SHA384	✓	✓	✓	✓	✓	✓
ECDHE-RSA-AES256-GCM-SHA384	✓	✓	✓	✓	✓	✓
ECDHE-ECDSA-AES256-SHA384	✓		✓	✓	✓	✓
ECDHE-RSA-AES256-SHA384	✓		✓	✓	✓	✓
ECDHE-RSA-AES256-SHA	✓			✓	✓	✓
ECDHE-ECDSA-AES256-SHA	✓			✓	✓	✓
AES128-GCM-SHA256	✓					
AES128-SHA256	✓					
AES128-SHA	✓					
AES256-GCM-SHA384	✓					
AES256-SHA256	✓					
AES256-SHA	✓					

What is s2n?

Family of AWS open-source libraries focused on encryption in transit

s2n-tls

s2n-quic

s2n-bignum

Secure

Backwards-compatible

Simple and focused

Well-tested

s2n-quic

NEW, WITH ENHANCED SECURITY AND PERFORMANCE

Security

Written in Rust, an efficient thread- and memory-safe language

PQ-hybrid key exchange

Verified correctness

Extensive testing – fuzzing, integration, interop, Monte Carlo

RFC compliance

Performance

Congestion controller

Packet pacing

Generic Segmentation Offload (GSO)

Path MTU discovery

AWS Secrets Manager everywhere

Secure centrally

Fine-grained access control

Rotate secrets safely

Auditing and monitoring

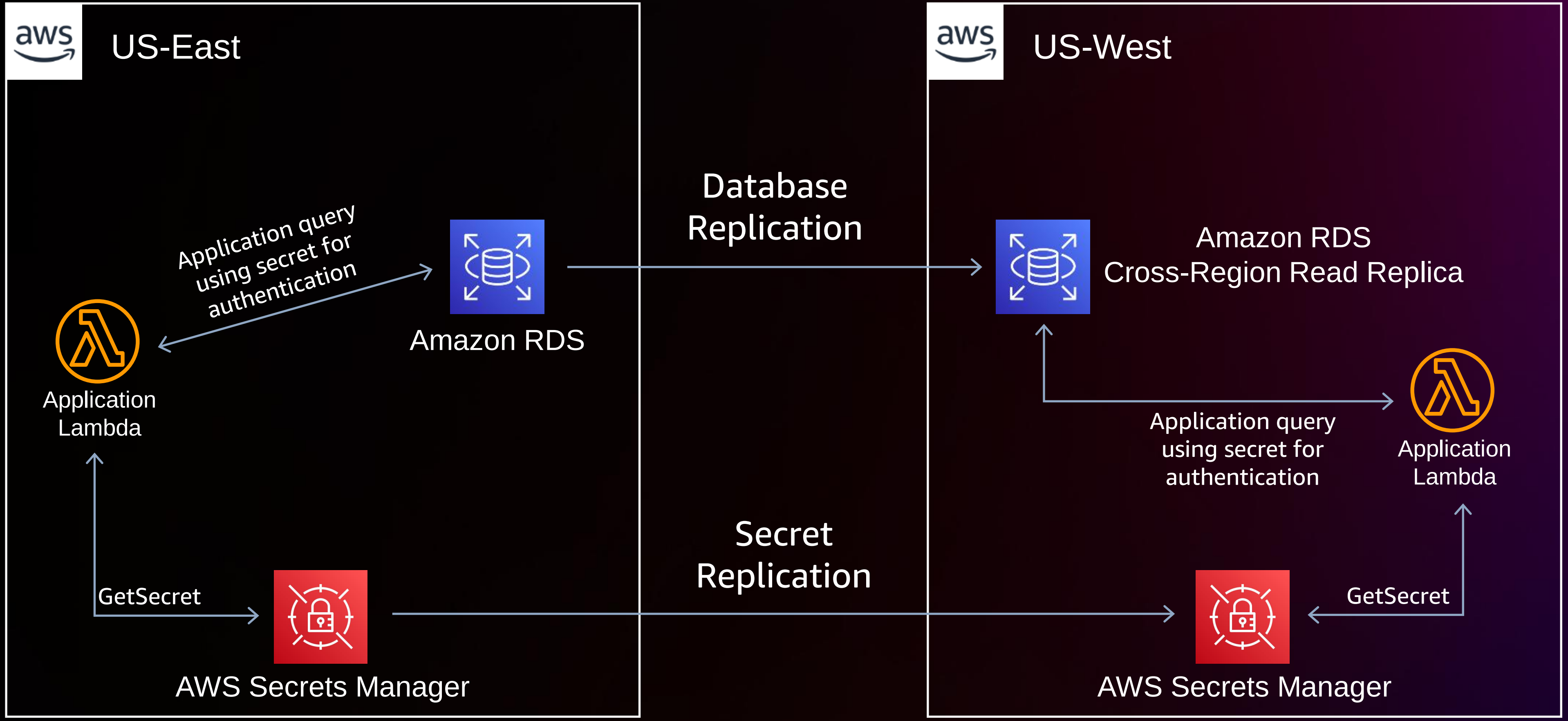
Secrets lifecycle management

What is a secret, exactly?

“A generic term for any secret value that an attacker could use to impersonate the subscriber in an authentication protocol […] “ ~ NIST SP 800-63-3

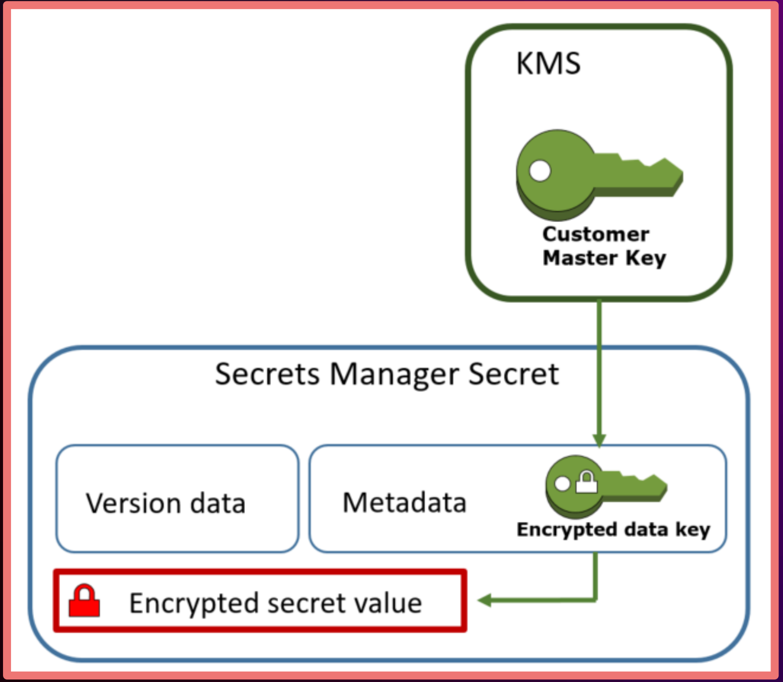
Multi-Region replication example: Disaster recovery

Creating *replica secrets* in multiple AWS Regions allows customers to plan for potential disaster recovery scenarios where failover to a new Region is necessary. In the below example, a replica secret (replicated from US-East) is used by a serverless application in US-West to access an RDS Cross-Region Read Replica.



How encrypting secrets works

- Encryption of secrets is enabled by default and cannot be disabled
- Each secret can be associated with a single CMK
- A single CMK can be used to encrypt many secrets
- Secrets are encrypted with a data key, and that data key is encrypted with a AWS KMS CMK
- Encrypted data key is stored along with secret (in metadata)
- Secrets are scrubbed from memory after encryption and are not saved, unencrypted, in durable storage



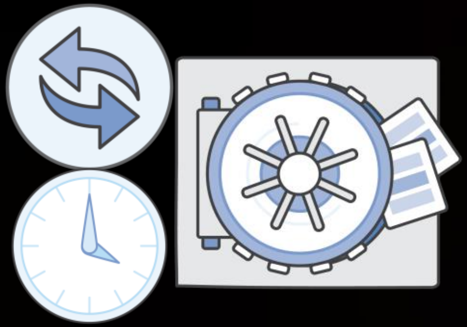
API actions that request access to CMK for encryption

- CreateSecretValue
- PutSecretValue
- UpdateSecret

API actions that request access to CMK for decryption

- GetSecretValue
- PutSecretValue
- UpdateSecret

Rotate secrets safely



- Built-in integrations for rotating all Amazon Relational Database Service (Amazon RDS) database types
- Extensible with AWS Lambda
- Use versioning so that applications don't break when secrets are rotated
- Configuring rotation causes the secret to rotate once as soon as you store the secret
- Configure secrets to rotate with alignment to organizational requirements

Transform a long-term secret into a short-term secret that is rotated automatically

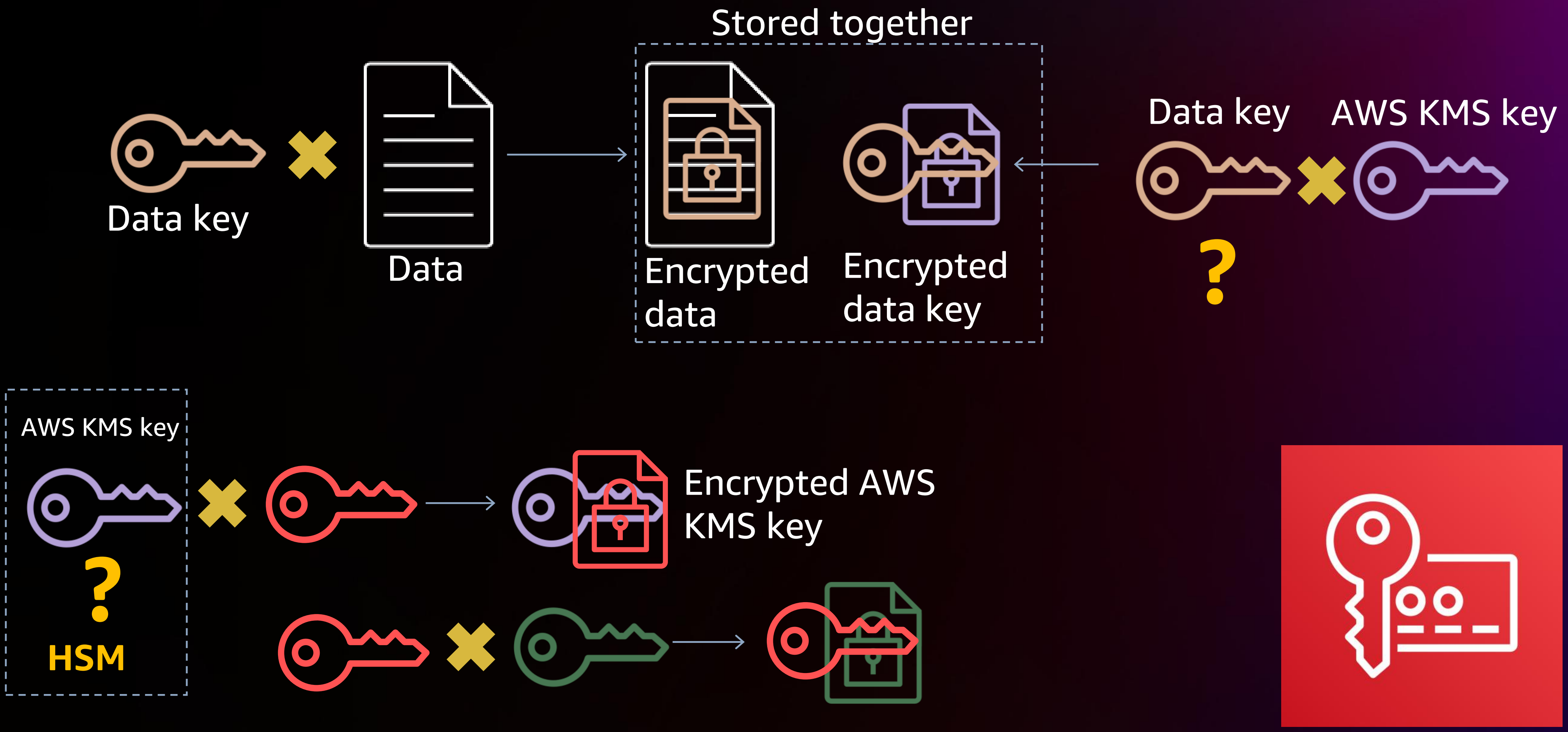
Audit and monitor using AWS Config

- Review changes in configurations and relationships between AWS resources
- Track detailed resource configuration histories
- Determine your overall compliance with configurations specified in your internal guidelines

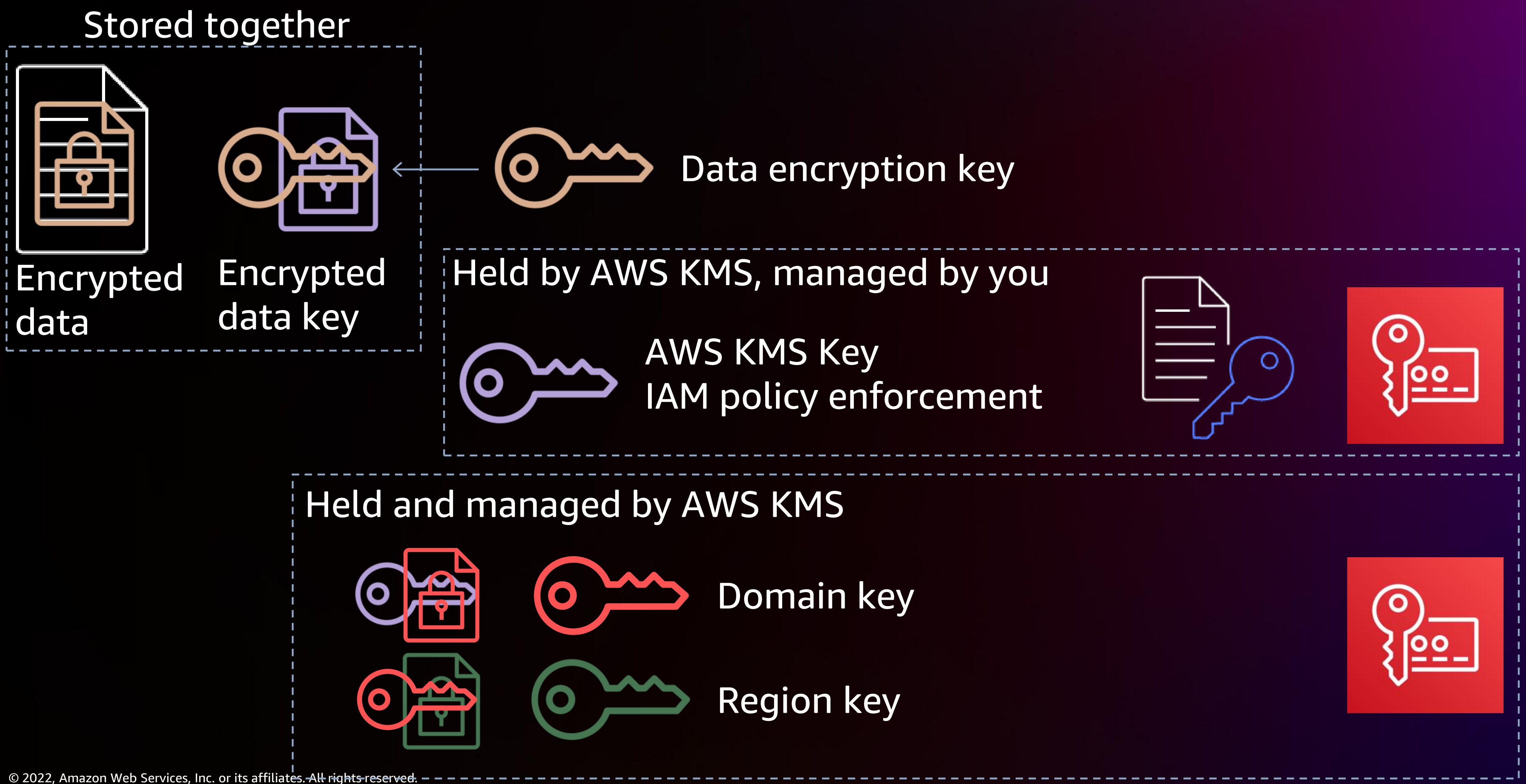
AWS Config

- Current Managed Rules
- Secretsmanager-rotation-enabled-check
- Secretsmanager-scheduled-rotation-success-check
- Secretsmanager-using-cmk
- Secretsmanager-secret-unused
- Secretsmanager-secret-periodic-rotation

Envelope encryption primer



AWS KMS key hierarchy



Stored together

Encrypted data

Encrypted data key

Data encryption key

Held by AWS KMS, managed by you

AWS KMS Key
IAM policy enforcement

Held and managed by AWS KMS

Domain key

Region key

Using keys securely in AWS

AWS Key Management Service (AWS KMS)

Multi-tenant, managed service for keys

AWS CloudHSM

Single-tenant, hardware security module (HSM) instance you control

Use cases for CloudHSM

FIPS 140-2 Level 3 requirement eligibility (HIPAA, FedRAMP, PCI)

Offload TLS/SSL processing

Signing and verification

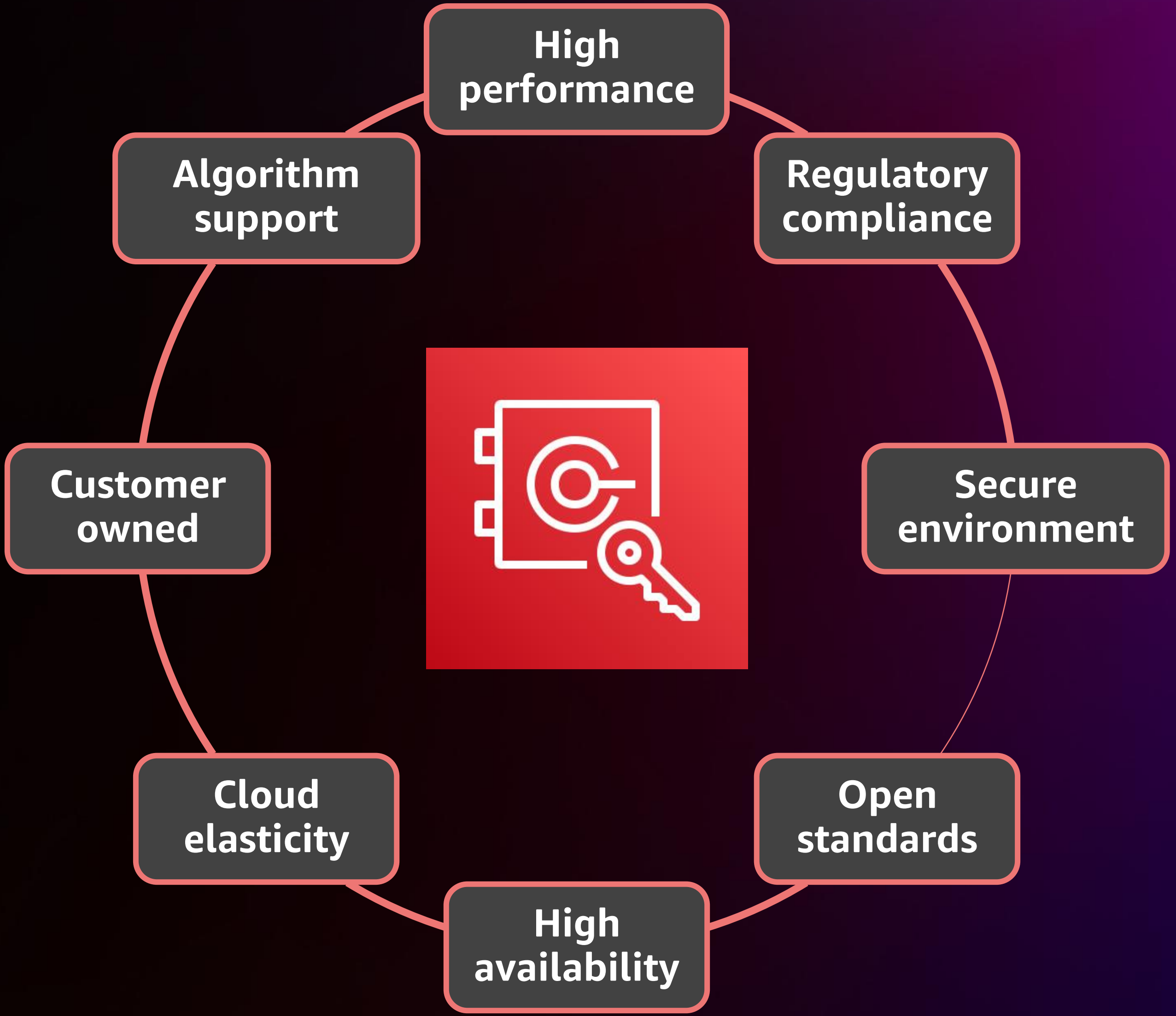
Certificate authority (CA)

Transparent Data Encryption (TDE) for Oracle databases

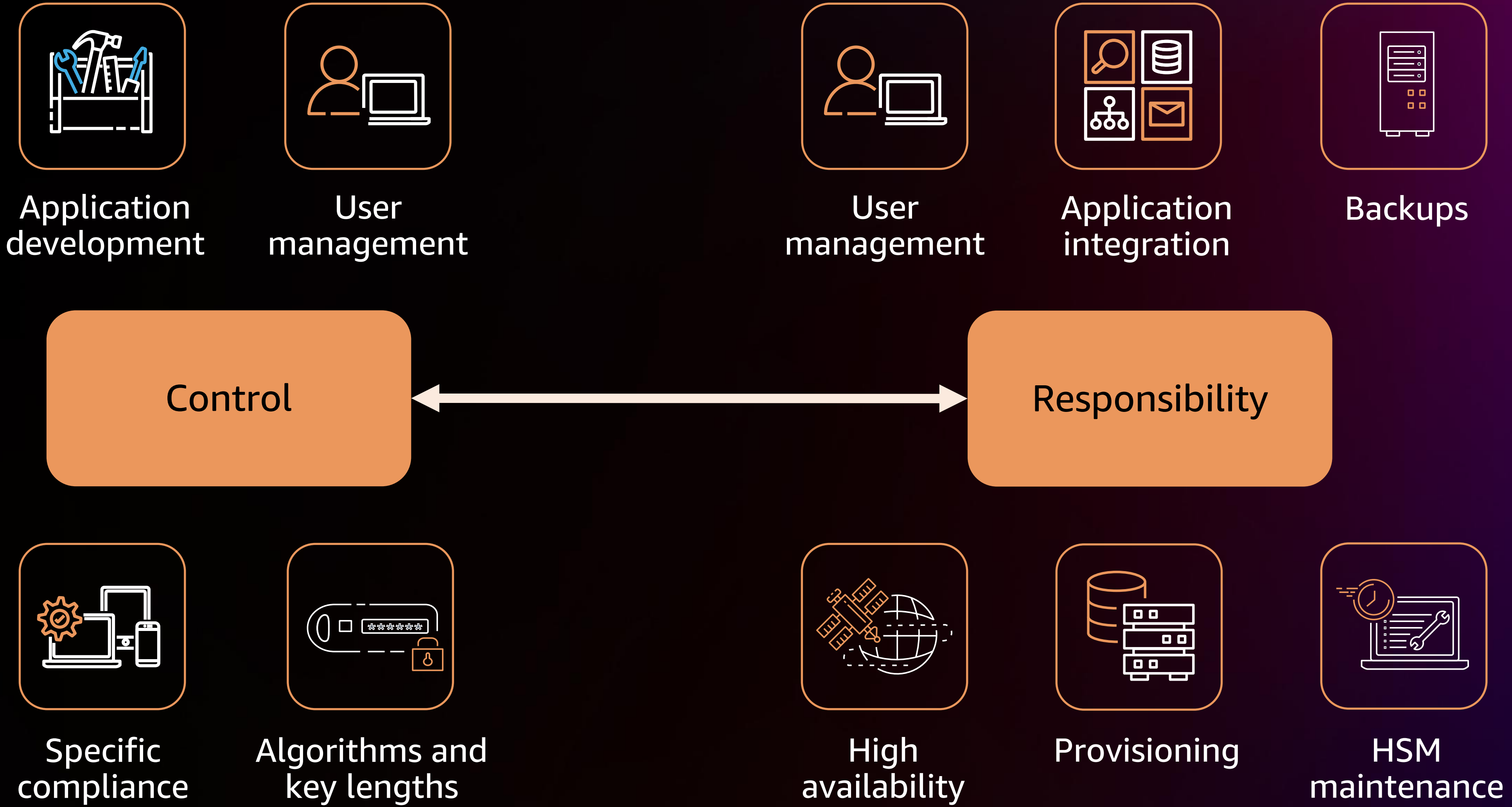
Digital rights management

Why CloudHSM?

Customers use HSMs on AWS because they need **low-latency access** to a **secure root of trust** that is under **their control**



Control implies responsibility



Application development

User management

Specific compliance

Algorithms and key lengths

Control

Responsibility

User management

Application integration

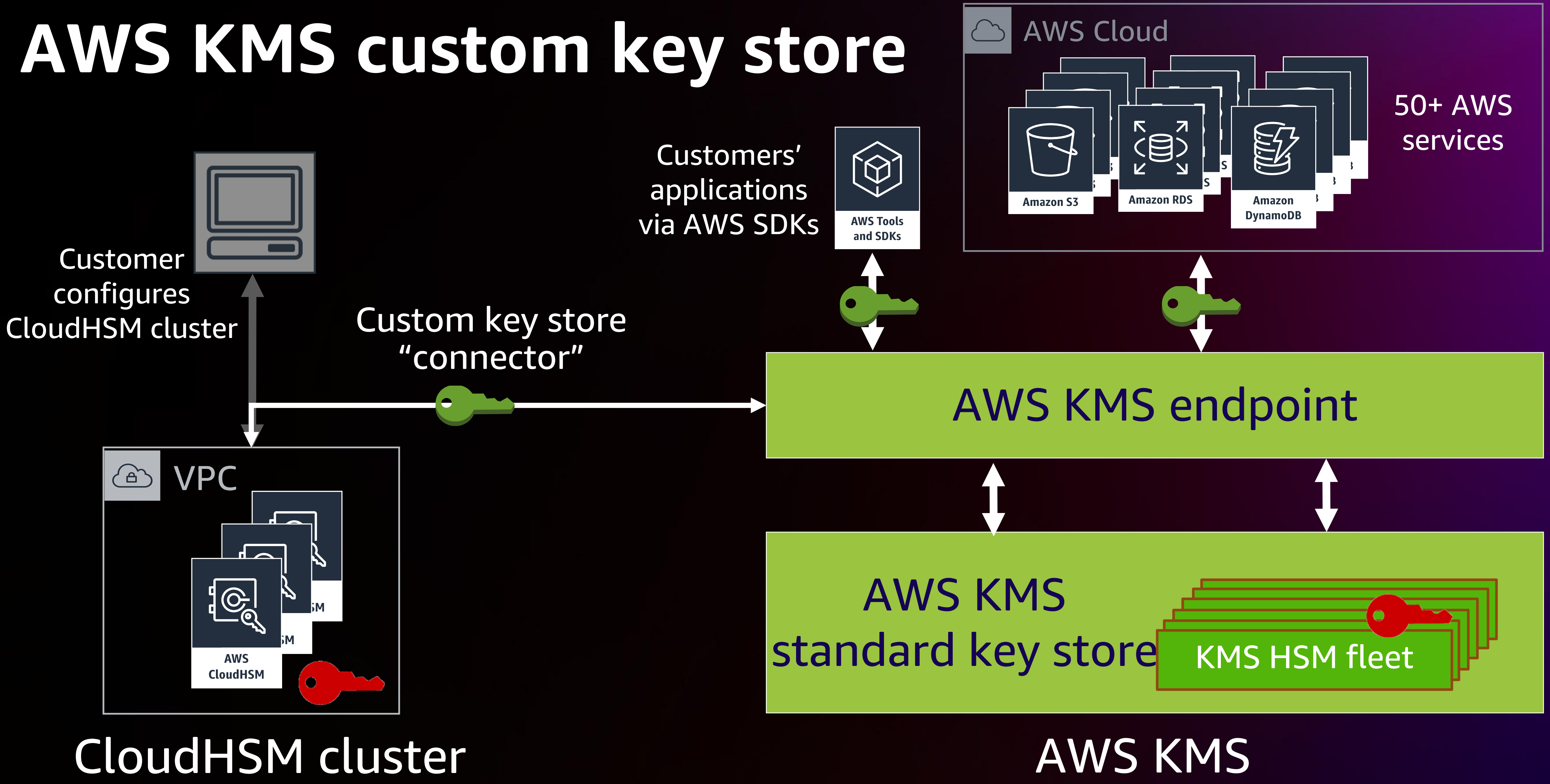
Backups

High availability

Provisioning

HSM maintenance

AWS KMS custom key store



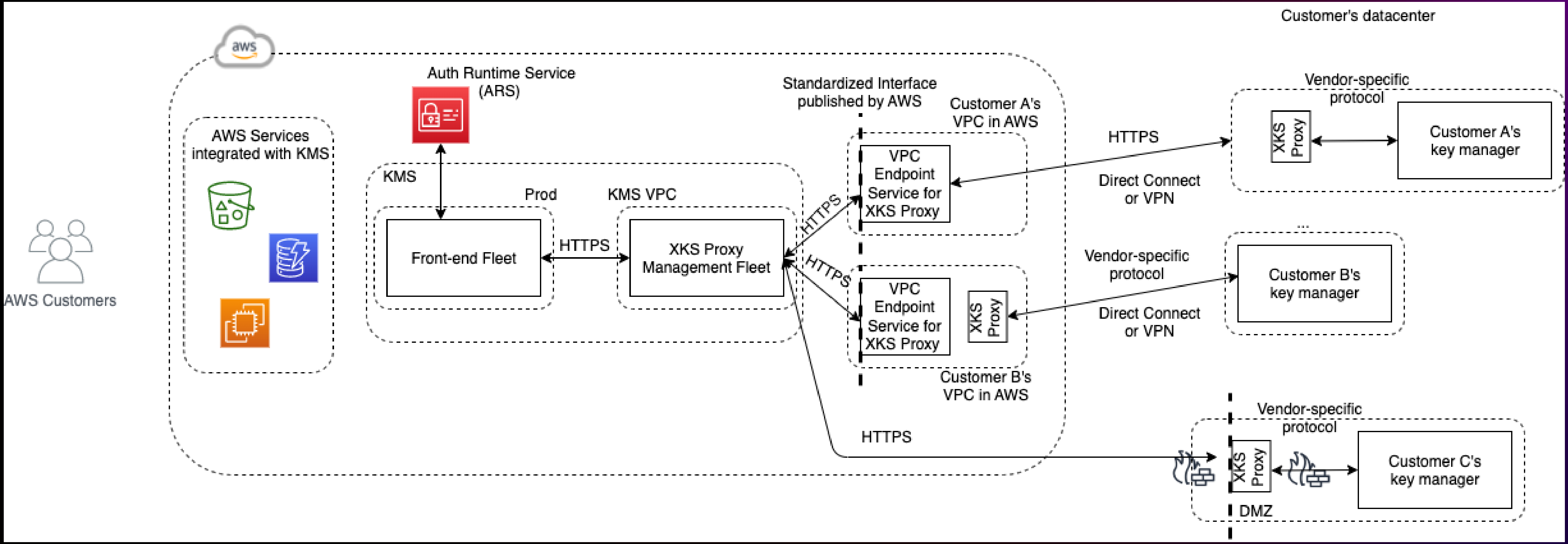
KMS HSMs are great!

- Custom hardware design minimizes surface area and maximizes security properties
- FIPS 140-2 validated modules since March, 2018
- Validated at level two overall and at level three for:
  - Cryptographic Module Specification
  - Roles, Services, and Authentication
  - Physical Security
  - Design Assurance
- Submitted for 140-2 L3 October 2021
- Submitted for 140-3 L3 September 2022

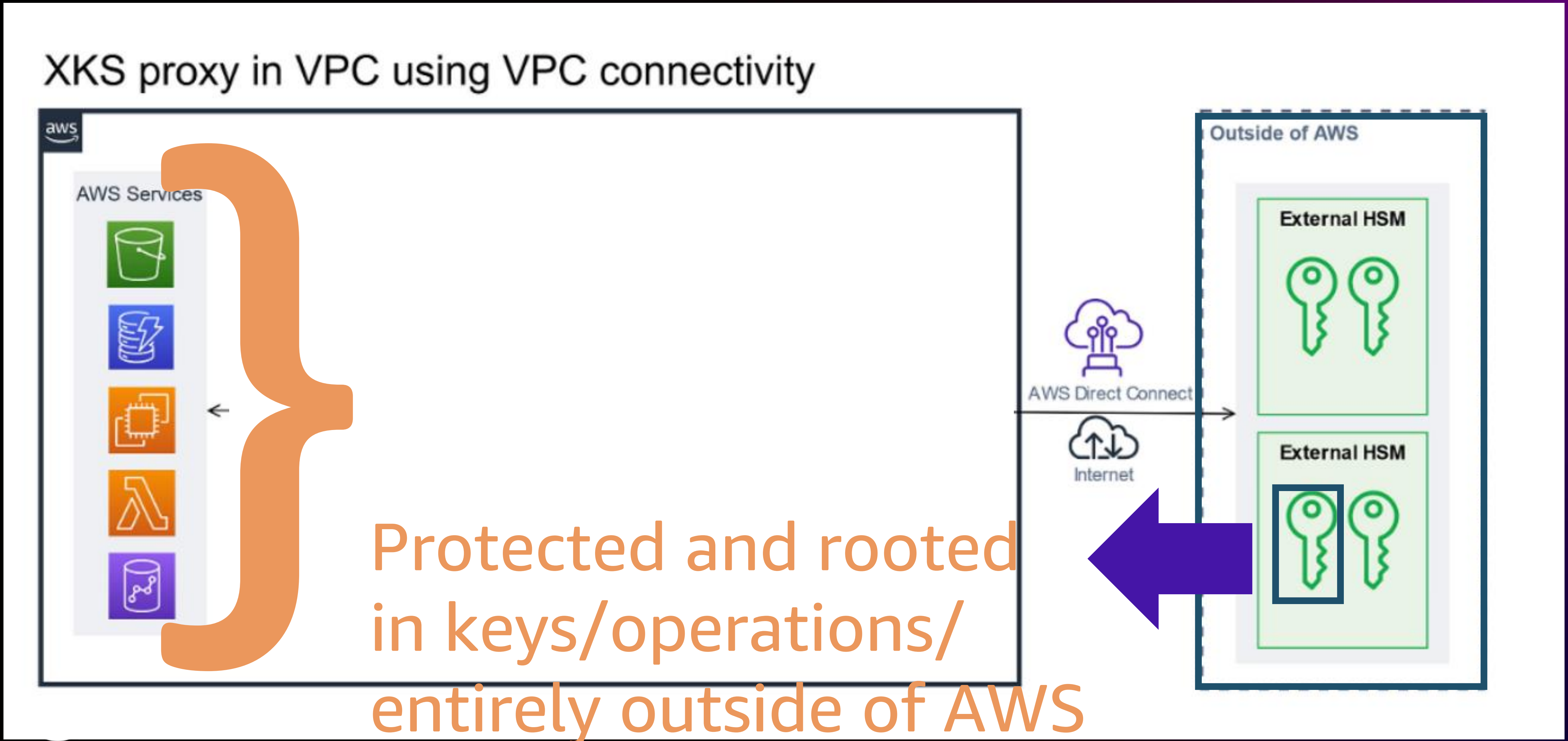
KMS HSMs

FIPS 140-3
Level 3

AWS KMS External Key Store New!



The need for double envelopes



XKS proxy in VPC using VPC connectivity

Protected and rooted in keys/operations/ entirely outside of AWS

XKS double encryption of envelope key

- Create an inner envelope first. This way, all AWS services have the identical protection offered by AWS KMS key envelopes (and we never send secret info out of the building in plaintext)

- The outer envelope ensures that only a Customer HSM, located external to AWS, can operate all decryption operations

- Quite literally the best of both worlds. No less AWS KMS security, highest XKS control

1. AWS KMS generates
a new data key

2. AWS KMS creates key envelope
(Using AWS KMS owned keys)

3. Customer creates outside envelope
(Using XKS keys encrypted in Customer HSM)

AWS Cryptography obsesses over customers

- Wide range of services to protect secrets, keys, and data
- Powerful open-source tools and SDKs for customers to use on their own
- Unrelenting innovation and responsiveness to customer demands
- Let’s go deeper into some of the most interesting innovations for our customers

Data at rest

(disk, file, DB, …)

Data in transit

(TLS, QUIC, SSH, IPsec,…)

Data in use

(cryptographic computing, enclaves)

AWS Encryption SDK

Data at rest

(disk, file, DB, …)

AWS Encryption SDK

INPUT: plaintext, CMK

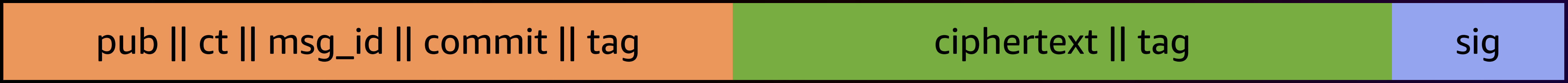
1) Generate signing key pair (prv, pub)

2) dk, ct = KMS.GenerateDataKey(CMK, pub)

3) K, commit = KDF(dk, msg_id), where msg_id random value

4) ciphertext, tag = AE(K, plaintext)

5) sig = Sign(prv, header || ciphertext || tag)



pub || ct || msg_id || commit || tag

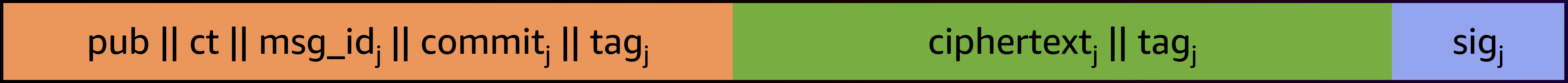
ciphertext || tag

sig

AWS Encryption SDK

INPUT: plaintext[m], CMK

1) Generate signing key pair (prv, pub)
2) dk, ct = KMS.GenerateDataKey(CMK, pub)
3) K_j, commit_j = KDF(dk, msg_id_j), random msg_id_j
4) ciphertext_j, tag_j = AE(K_j, plaintext [j])
5) sig_j = Sign(prv, header_j || ciphertext_j || tag_j)



pub || ct || msg_id_j || commit_j || tag_j

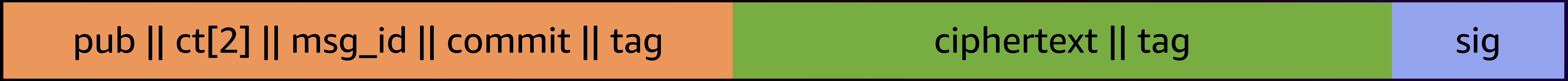
ciphertext_j || tag_j

sig_j

AWS Encryption SDK

INPUT: plaintext, CMK[2]

```
1) Generate signing key pair (prv, pub)
2) dk, ct[0] = KMS.GenerateDataKey(CMK[0], pub)
3) ct[1] = KMS.Encrypt(dk, CMK[1], pub)
4)  K, commit = KDF(dk, msg_id), random msg_id
5) ciphertext, tag = AE(K, plaintext)
6) sig = Sign( prv, header || ciphertext || tag )
```

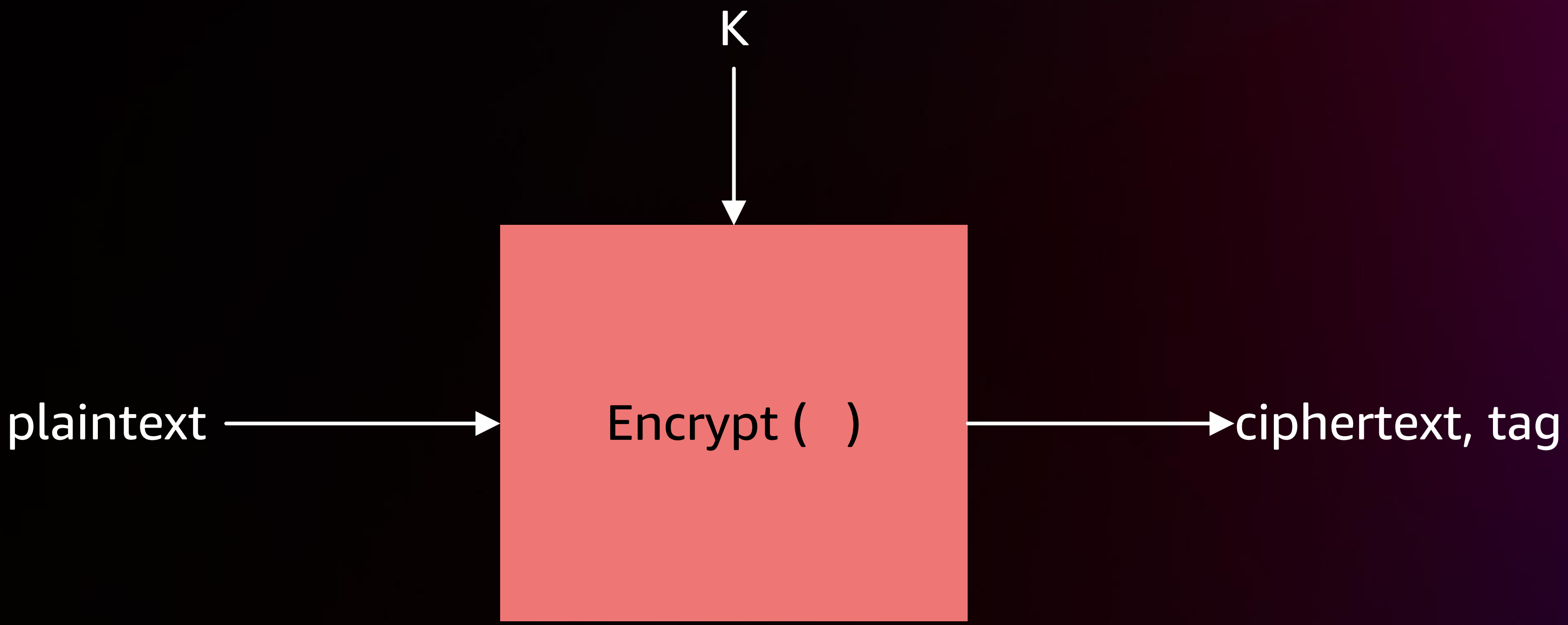


pub || ct[2] || msg_id || commit || tag

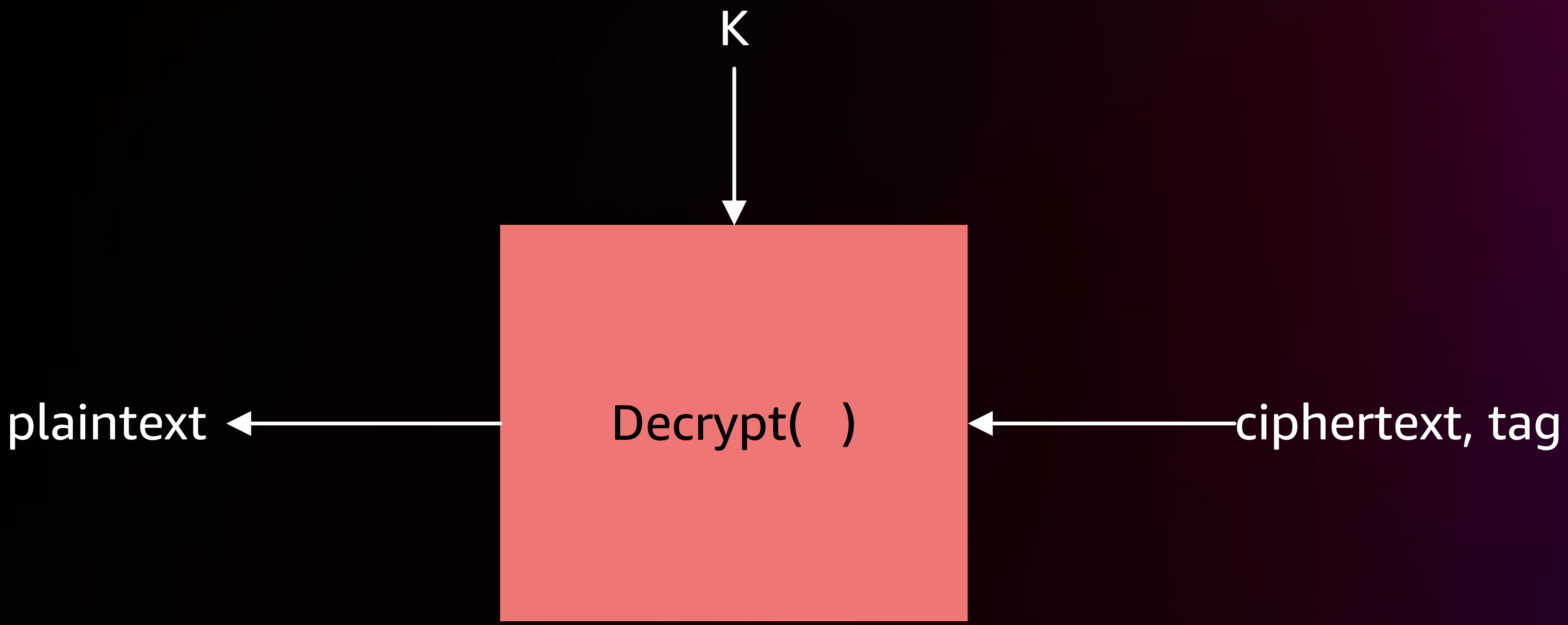
ciphertext || tag

sig

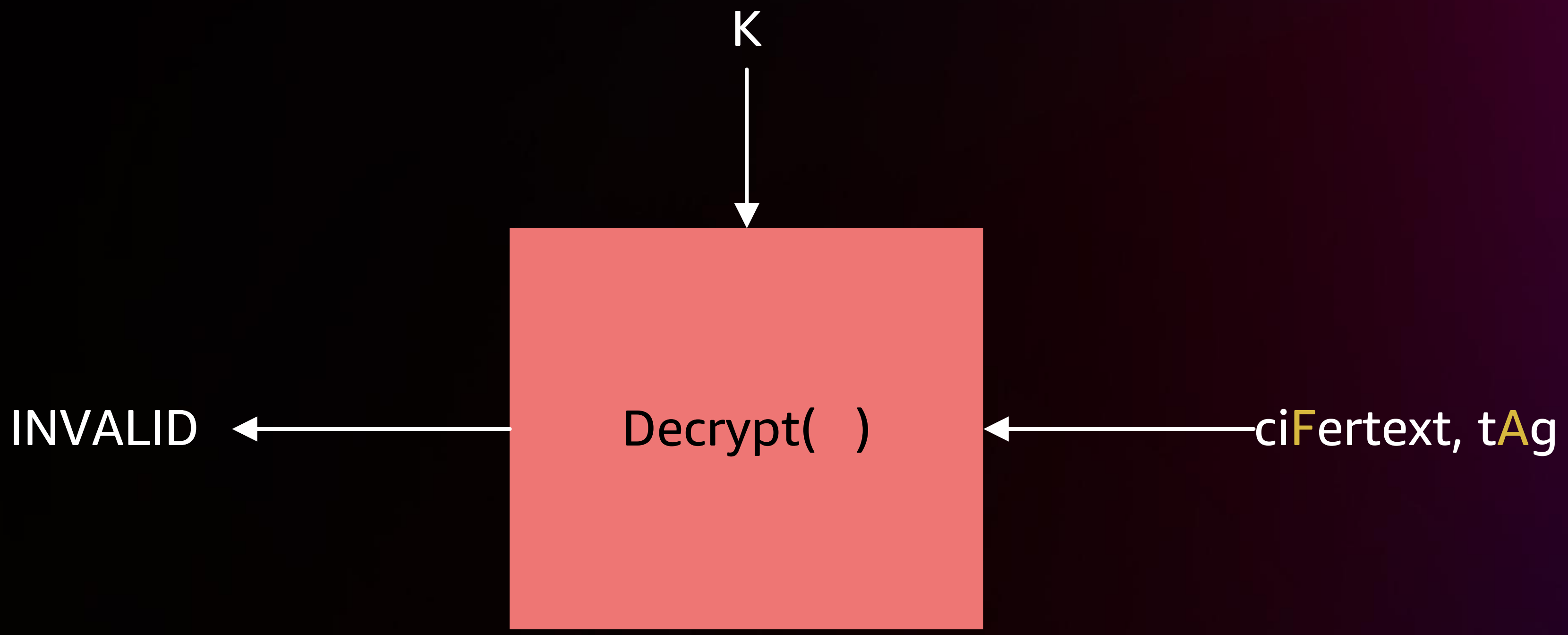
Authenticated Encryption



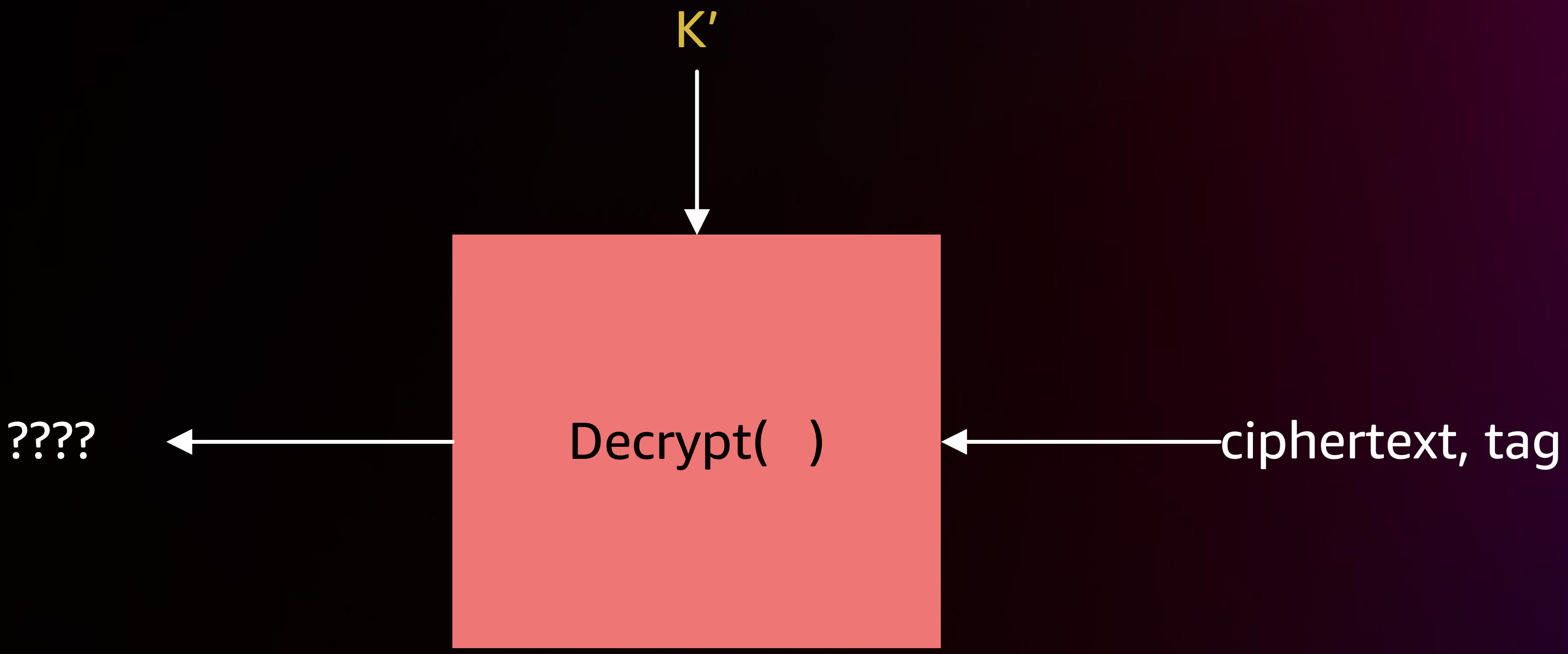
Authenticated Encryption



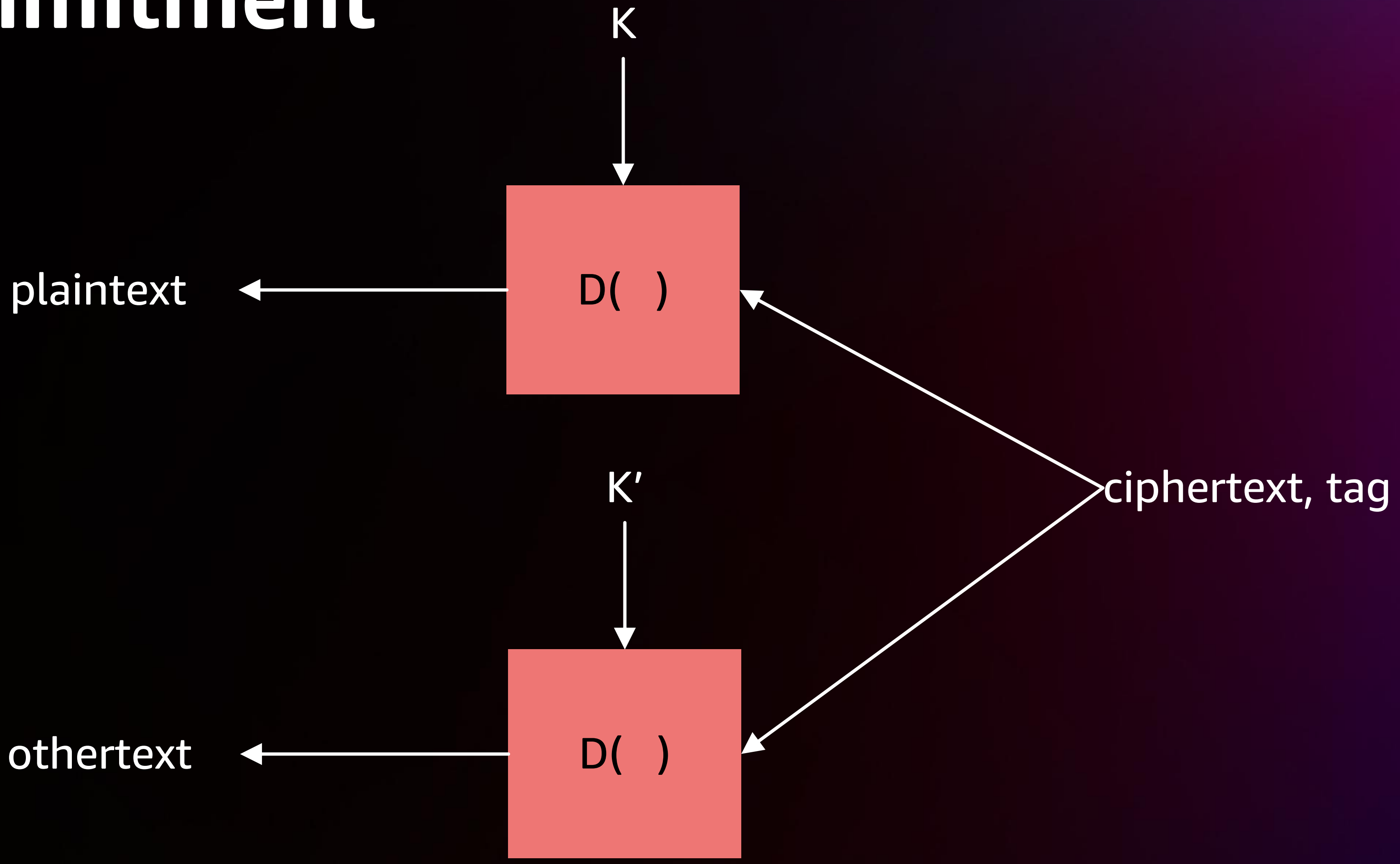
Authenticated Encryption



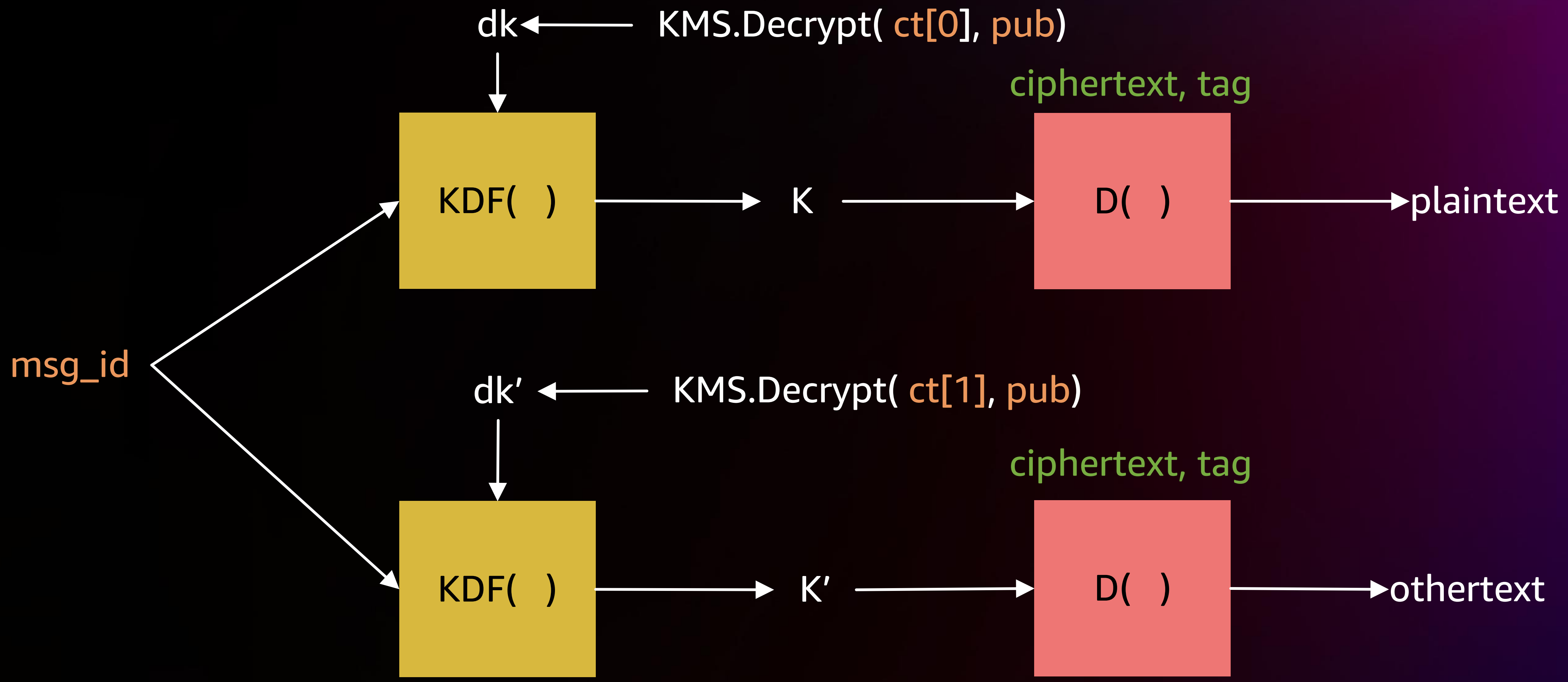
Key Commitment



Key Commitment



Key Commitment



dk ← KMS.Decrypt(ct[0], pub)

ciphertext, tag

KDF( ) → K → D( ) → plaintext

msg_id

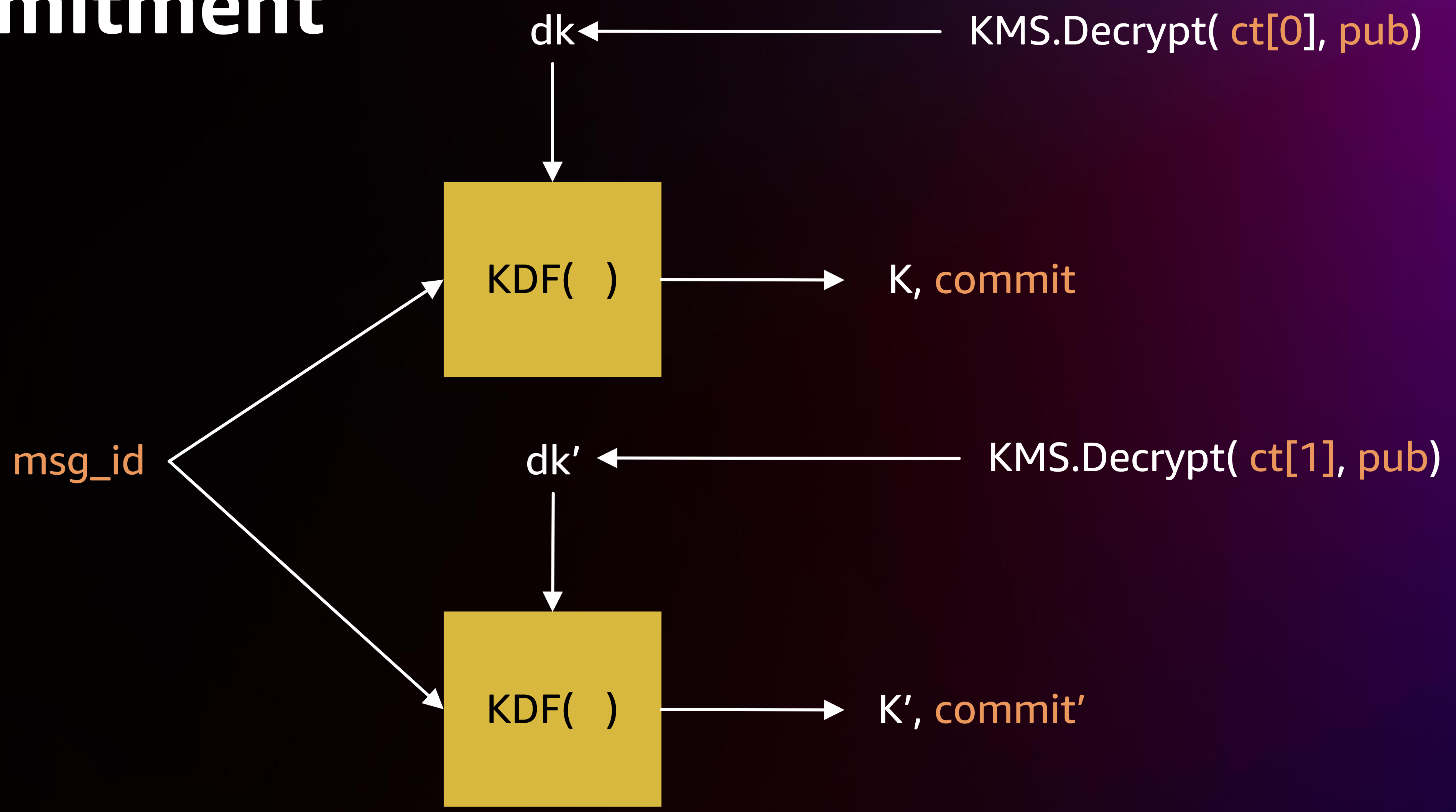
dk’ ← KMS.Decrypt(ct[1], pub)

ciphertext, tag

KDF( ) → K’ → D( ) → othertext

pub || ct[2] || msg_id || tag	ciphertext || tag	sig

Key Commitment



pub || ct[2] || msg_id || commit || tag	ciphertext || tag	sig

More on this topic

If you are encrypting data on AWS, we recommend you use the Encryption SDK

https://docs.aws.amazon.com/aws-crypto-tools/index.html

AWS completed a release of Key Commitment in all languages of the AWS Encryption SDK

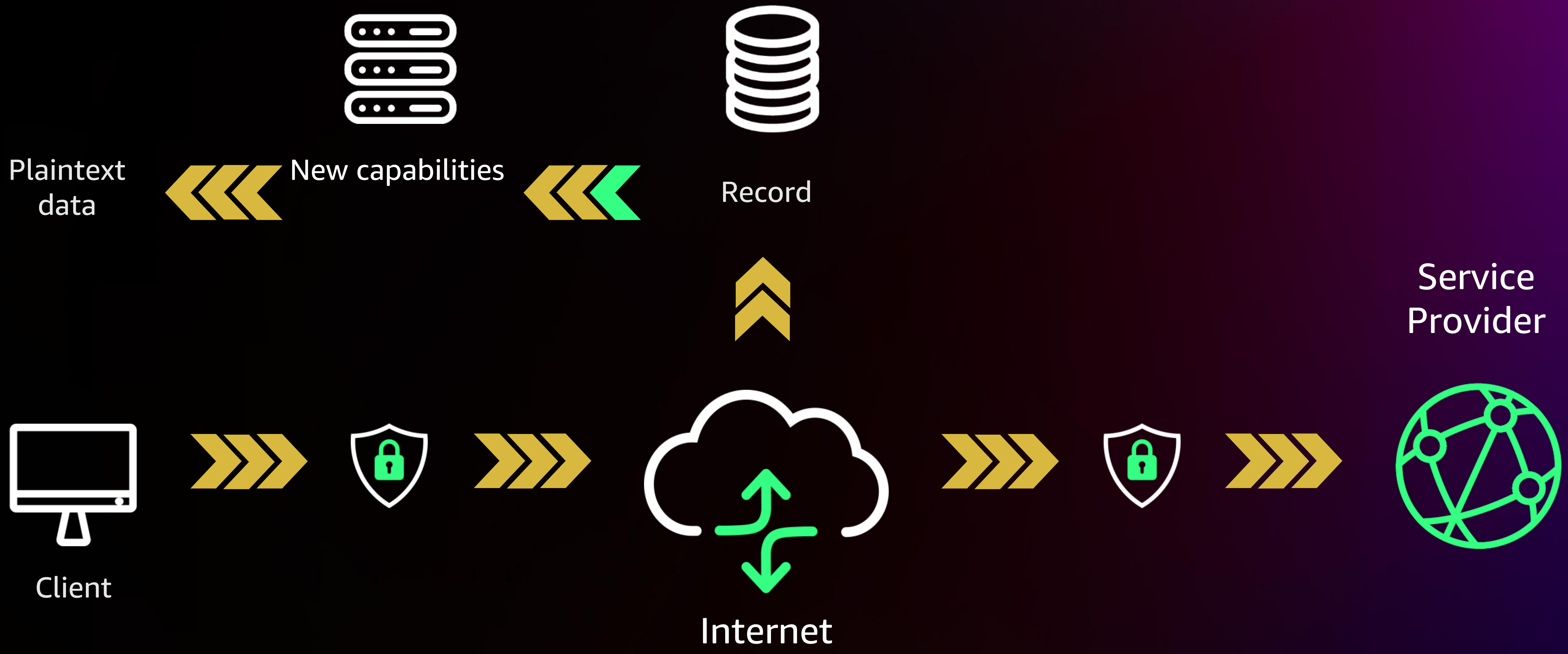
Blog post: amzn.to/2XiAP2V

Post-quantum cryptography

Data in transit

(TLS, QUIC, SSH, IPsec,…)

Long term confidentiality



Quantum computing

What’s a bit?

= 000

= 001

= 010

= 011

= 100

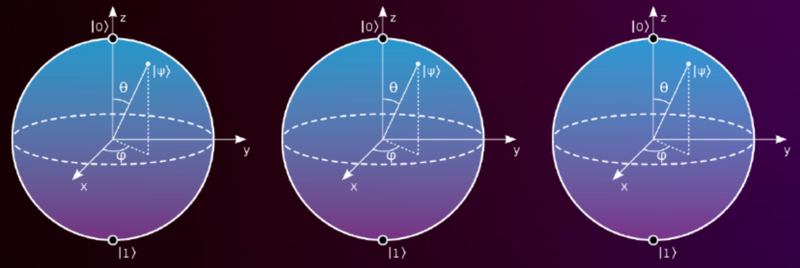
= 101

= 110

= 111

2^30 = **1,073,741,824 states**

What’s a qubit?



$|0\rangle$ $|1\rangle$ $|0\rangle$

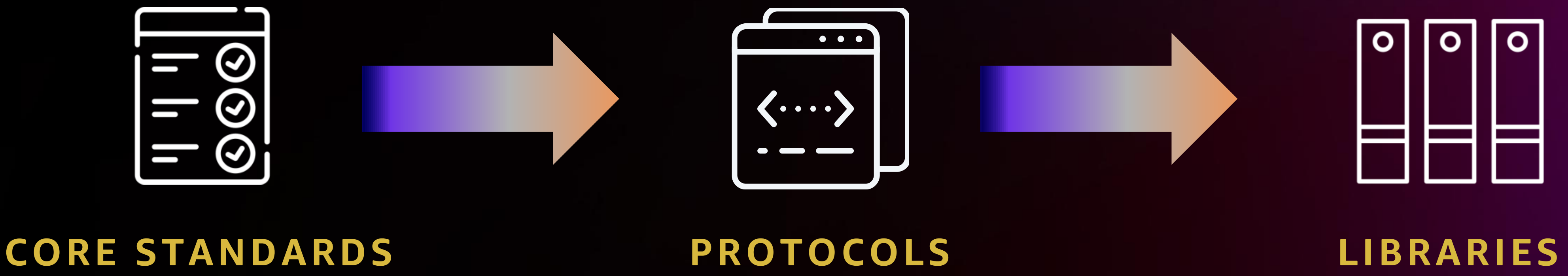
Takes 30 “pure” qubits

What can a quantum computer do?

Rapidly solve some types of “brute force” mathematical problems

Efficiently break all of our currently deployed public-key crypto systems (RSA, DH, DSA, ECC)

How is AWS preparing?



CORE STANDARDS

NIST

- Adopting the NIST standardized techniques as soon as possible

ETSI – Quantum-Safe Cryptography

- Editor of ETSI Quantum-Safe Hybrid Key Exchange TS

NCCoE

PROTOCOLS

IETF

PQ-TLS 1.3
(draft-ietf-tls-hybrid-design-04)

PQ SSH
(draft-kampanakis-curdle-ssh-pq-ke-00)

PQ Signatures in X.509
(draft-ietf-massimo-lamps-pq-pkix-00)

PQ KEMs in X.509
(draft-turner-lamps-nist-pqc-kem-certificates-01)

Other

- PQ-QUIC
- PQ-HPKE

LIBRARIES

s2n-TLS

- Deploying Post-Quantum ciphersuites for TLS 1.3 in s2n

AWS-LC

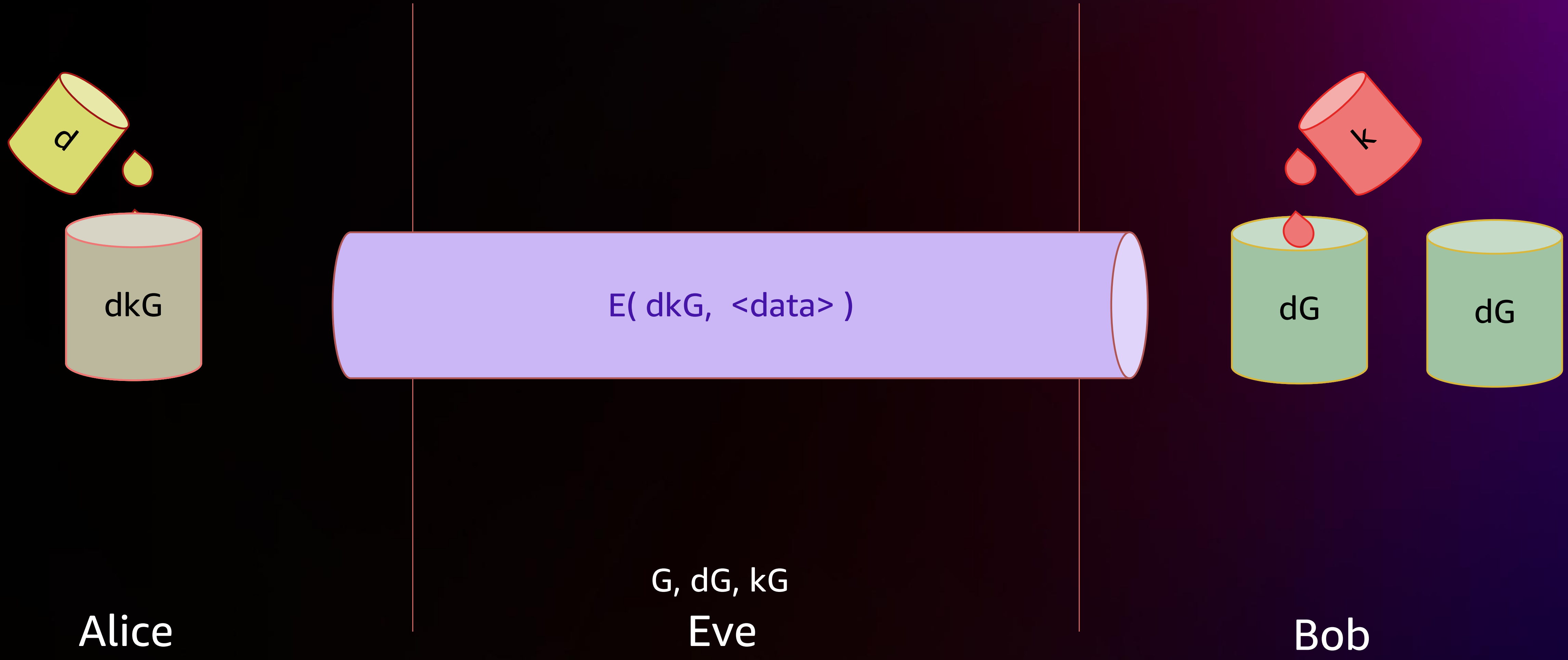
- Adopting PQ key exchange and Signature schemes

AWS Java SDK w/ CRT

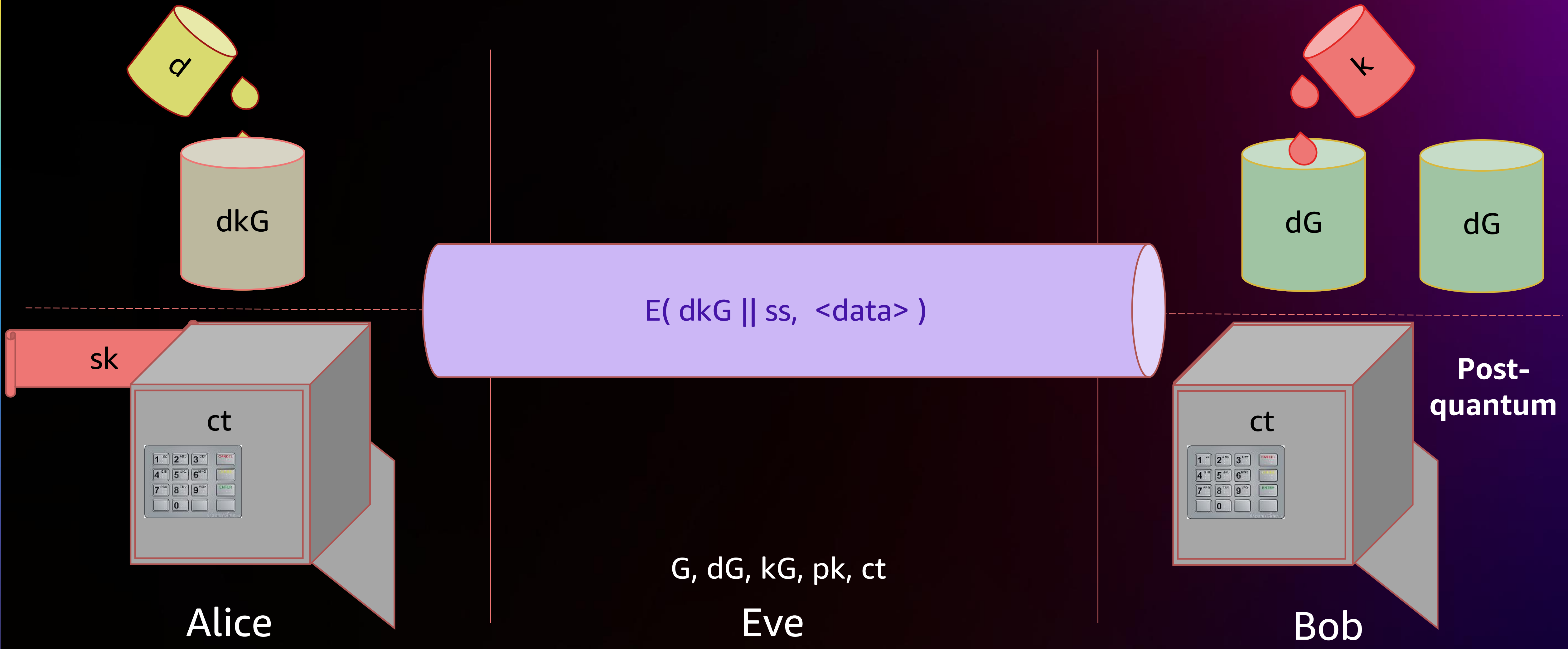
- Pq-TLS 1.3

Elliptic curve Diffie-Hellman

Plaintext data
New capabilities
Record
Service Provider
Client
Internet



Post-quantum hybrid key exchange



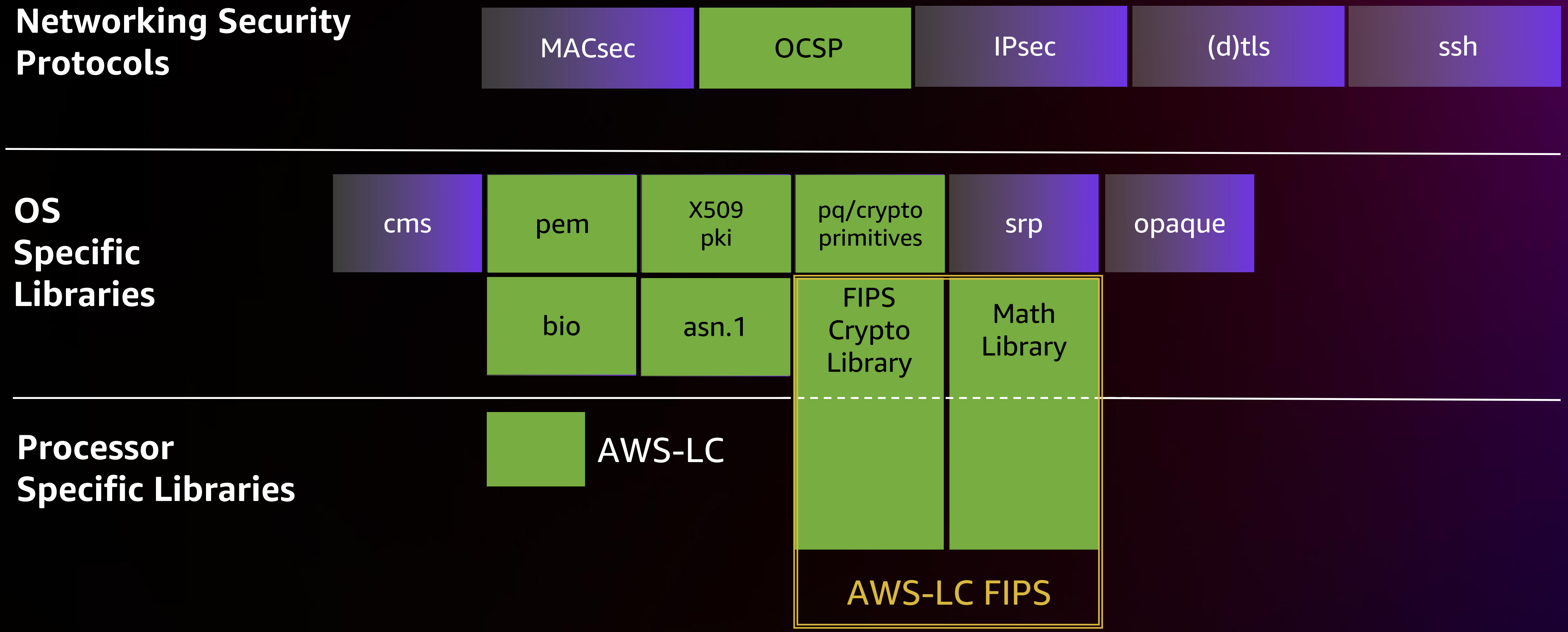
Try our PQ cryptography

PQ hybrid key exchanges

- In s2n-tls mainline, deployed everywhere s2n is, and
- used in AWS KMS, Secrets Manager and ACM

https://aws.amazon.com/security/post-quantum-cryptography/

AWS-LibCrypto



Networking Security Protocols

MACsec

OCSP

IPsec

(d)tls

ssh

OS Specific Libraries

cms

pem

X509 pki

pq/crypto primitives

srp

opaque

bio

asn.1

FIPS Crypto Library

Math Library

Processor Specific Libraries

🟩 AWS-LC

AWS-LC FIPS

AWS-LC FIPS

Submitted to NIST for FIPS 140-3 on 2021-12-21 for Graviton 2 and Intel platforms

	Prior FIPS Solution	AWS-LC 2021 FIPS	AWS-LC 2022
ECDH P-256	365/sec	17,584/sec	17,625/sec
ECDH P-384	197/sec	1,078/sec	4,156/sec
ECDH P-521	101/sec	427/sec	3,051/sec
RSA 2048 Sign 32 bytes	1,035/sec	1,719/sec	1,741/sec
RSA 2048 Verify 32 bytes	18,306/sec	53,814/sec	53,960/sec
AES-128-GCM Encrypt 16kb	131,510/sec	330,943/sec	331,031/sec

OpenSSL 1.0.2 versus AWS-LC on c5.2xlarge (Intel)

Try out AWS-LC

AWS-LC is on GitHub:

https://github.com/awslabs/aws-lc

Our PQ Branch:

https://github.com/awslabs/aws-lc/tree/integrate-pq

Cryptographic computing

Data in use

(cryptographic computing, enclaves)

What problems are customers facing?

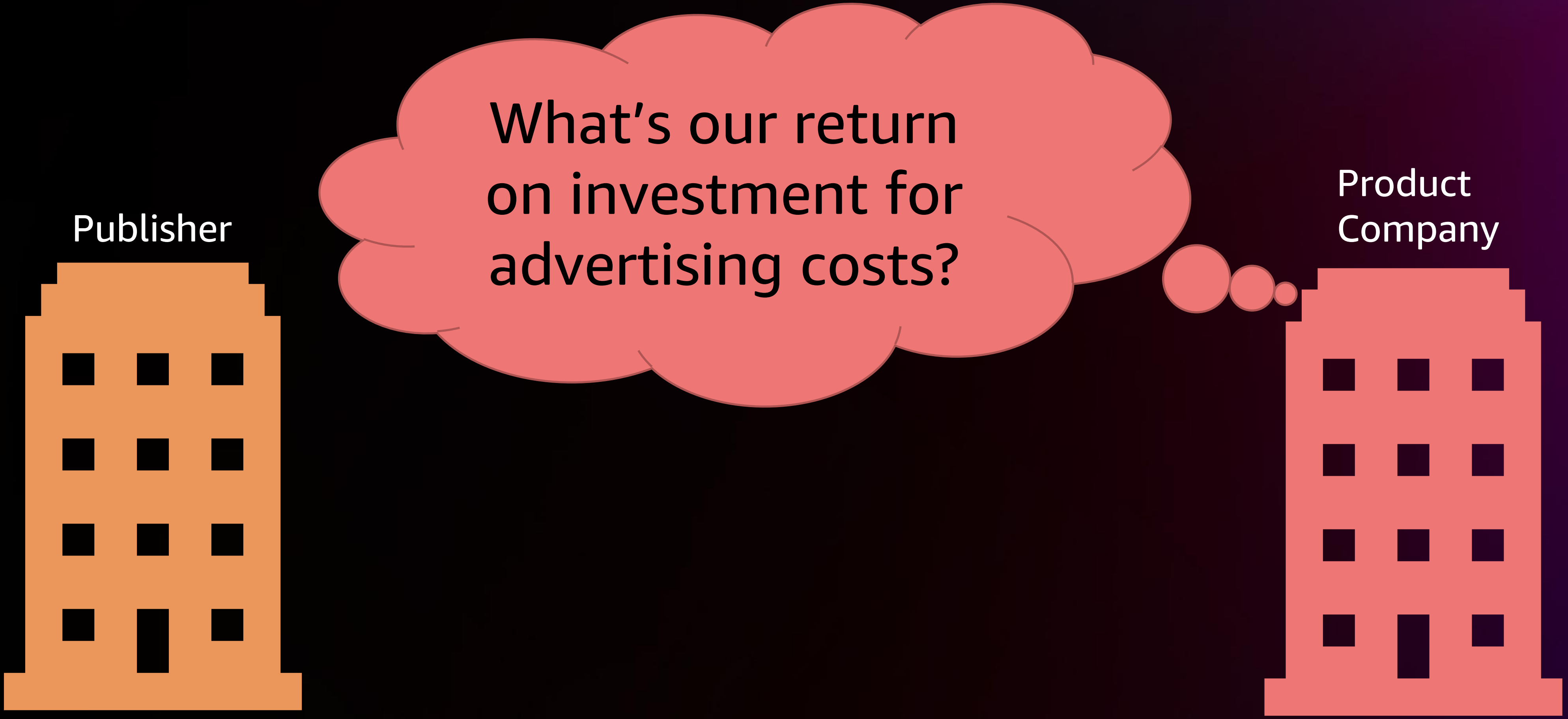
Customers

Customers want to

1. Perform collaborative data analytics in the cloud.

2. On-site encrypt sensitive/regulated data for analytics.

Example: Advertising/sales collaboration

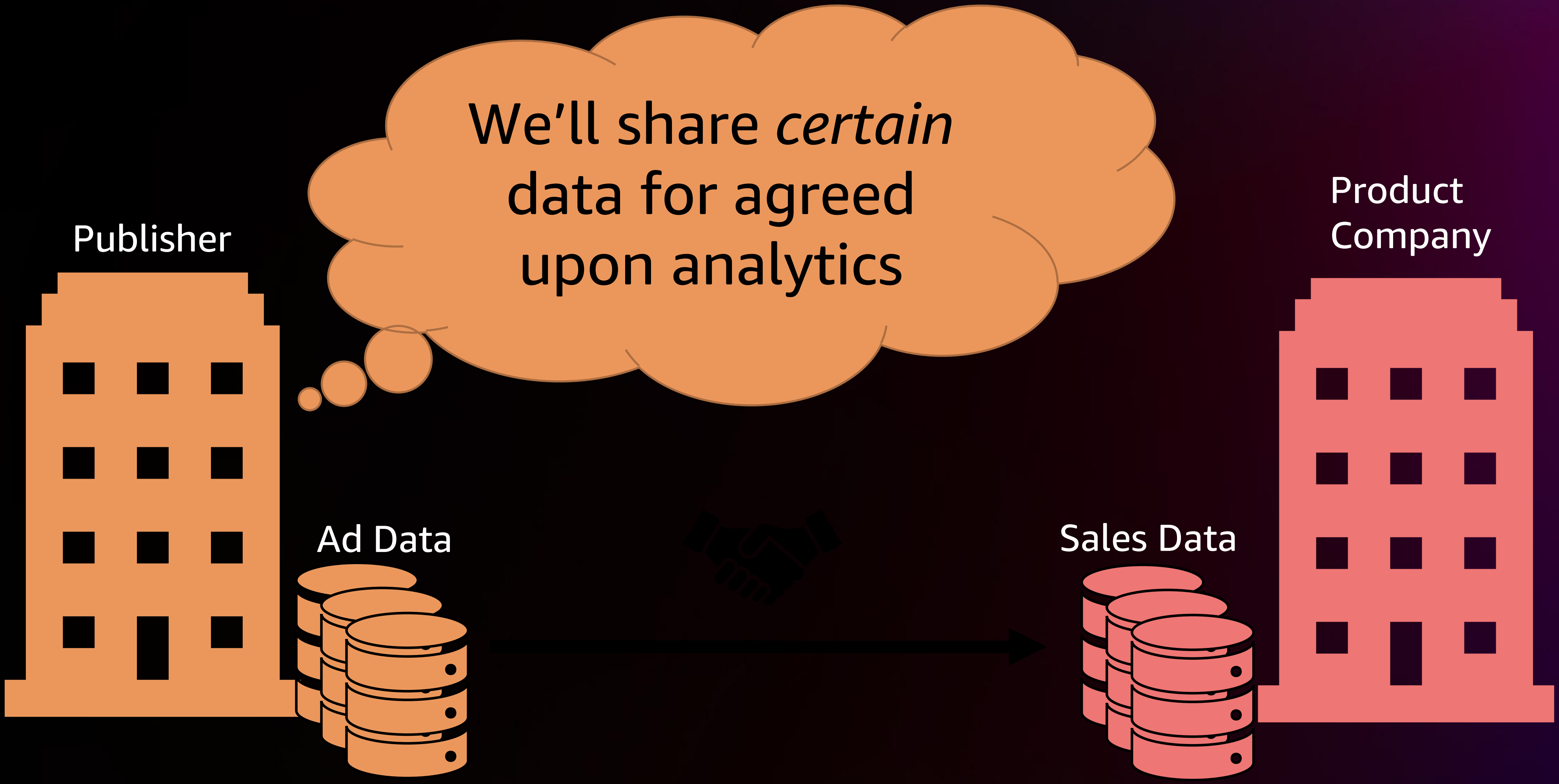


Publisher

What’s our return on investment for advertising costs?

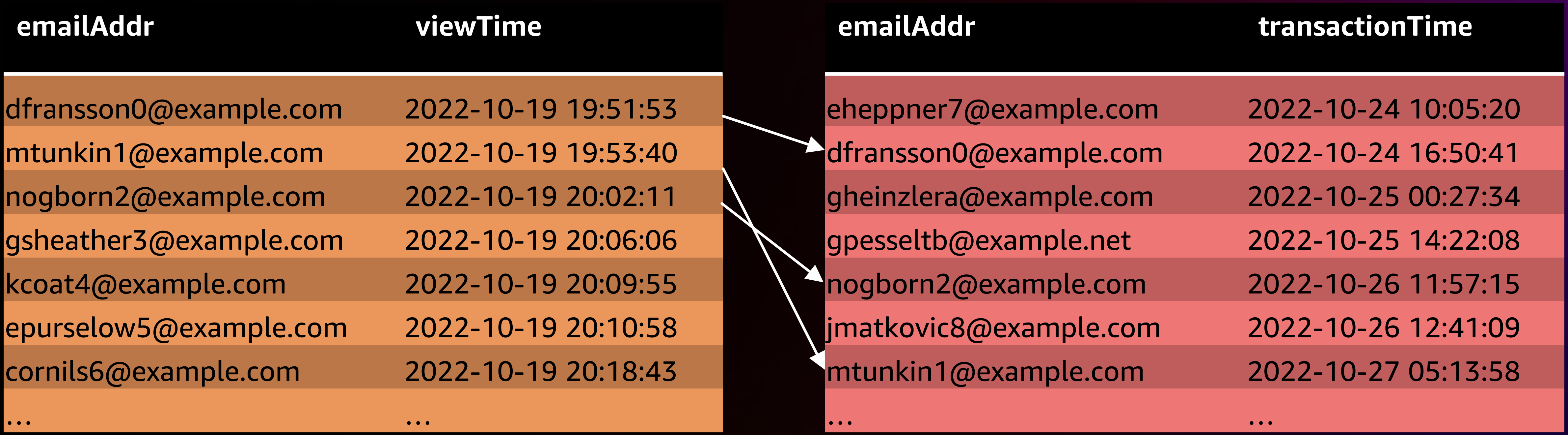
Product Company

Example: Advertising/sales collaboration



Example

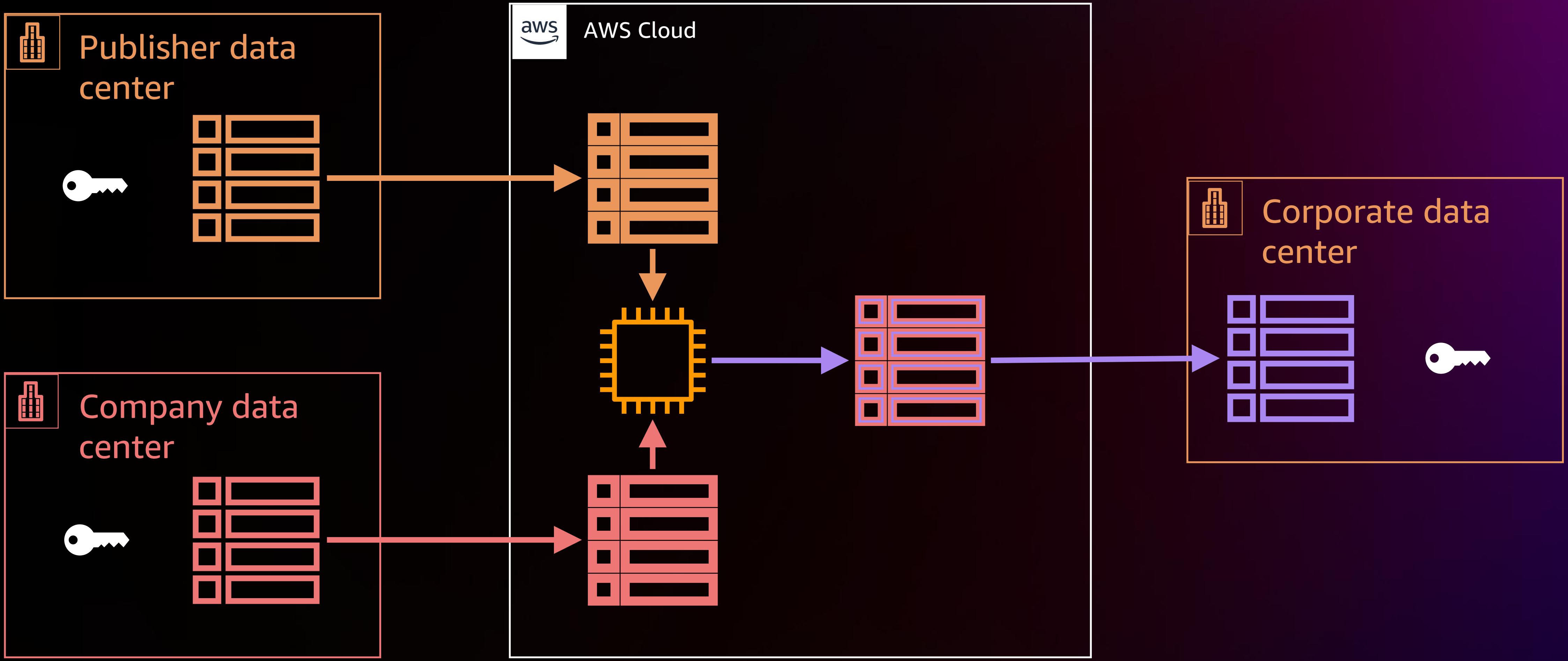
```
SELECT
    COUNT(transactions.transactionTime) AS cnt FROM transactions INNER JOIN views
    ON transactions.emailAddr = views.emailAddr
    WHERE transactions.transactionTime > views.viewTime;
```



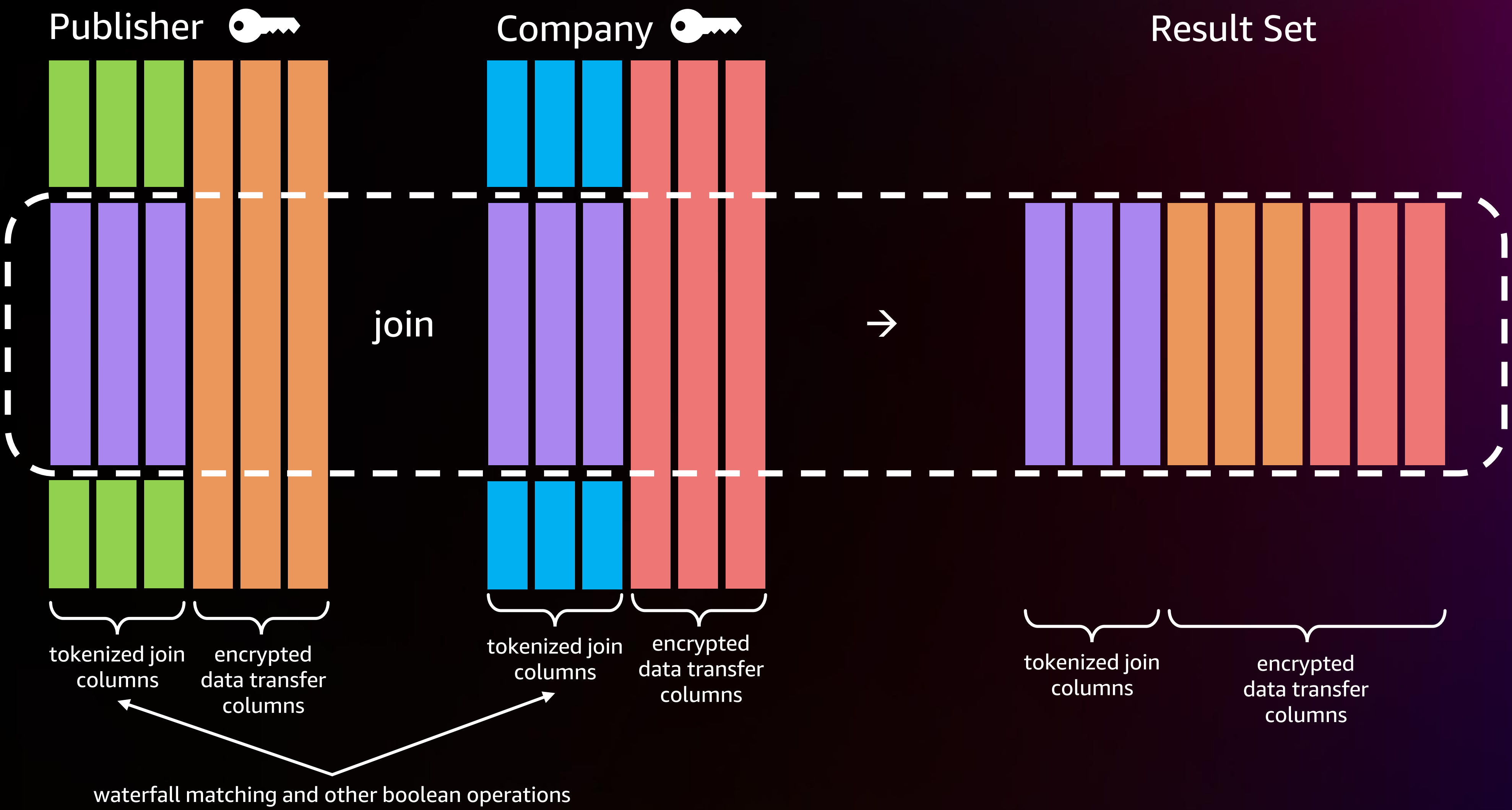
emailAddr	viewTime
dfransson0@example.com	2022-10-19 19:51:53
mtunkin1@example.com	2022-10-19 19:53:40
nogborn2@example.com	2022-10-19 20:02:11
gsheather3@example.com	2022-10-19 20:06:06
kcoat4@example.com	2022-10-19 20:09:55
epurselow5@example.com	2022-10-19 20:10:58
cornils6@example.com	2022-10-19 20:18:43
…	…

emailAddr	transactionTime
eheppner7@example.com	2022-10-24 10:05:20
dfransson0@example.com	2022-10-24 16:50:41
gheinzlera@example.com	2022-10-25 00:27:34
gpesseltb@example.net	2022-10-25 14:22:08
nogborn2@example.com	2022-10-26 11:57:15
jmatkovic8@example.com	2022-10-26 12:41:09
mtunkin1@example.com	2022-10-27 05:13:58
…	…

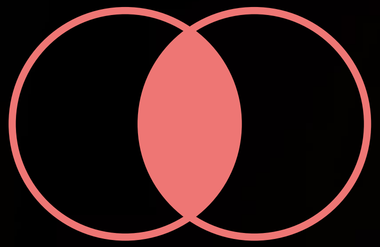
What some users want



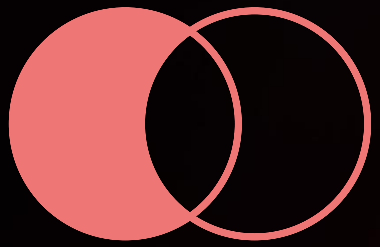
Private set intersection + data enrichment


Publisher
Company
Result Set
join
→
tokenized join columns
encrypted data transfer columns
tokenized join columns
encrypted data transfer columns
tokenized join columns
encrypted data transfer columns
waterfall matching and other boolean operations

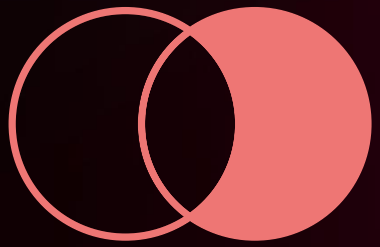
PSI as a database join



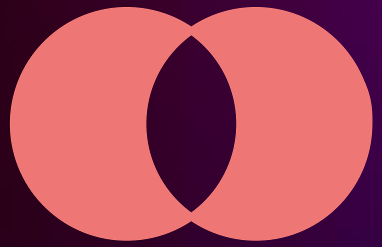
inner join



left outer join



right outer join



full outer join

Can we build an interesting subset of SQL using these kinds of techniques?

Examples

Two hospitals want to identify common patients…

- But can’t share patient lists due to privacy regulations

Insurance company wants to query third-party database…

- But queries contain trade secrets/‘secret sauce’

LE community wants cross-agency collaboration…

- But prevented from pooling data by legal ‘firewalls’

Learn more

Learn more about cryptographic computing
https://aws.amazon.com/security/cryptographic-computing/

Thank you!

Please complete the session survey in the **mobile app**