



AWS Black Belt Online Seminar

AWS Key Management Service (AWS KMS)

平賀 敬博

Security Solutions Architect

2024/03

自己紹介

平賀 敬博 Hiraga Takahiro

アマゾン ウェブ サービス ジャパン
セキュリティソリューションアーキテクト

好きな AWS サービス
AWS Key Management Service



Part. 1 基礎編



本セミナーの対象者

- AWS 環境における保管中データ暗号化に関心をお持ちの方
- AWS Key Management Service をご利用予定の方
- AWS Key Management Service について理解を深めたい方

本セミナーで得られること

- AWS Key Management Service の保管中データ暗号化の仕組み
- AWS Key Management Service の鍵管理の仕組み
- AWS Key Management Service による暗号鍵保管・管理の仕組みを社内のセキュリティ部門、コンプライアンス部門に説明できる

Part.1 アジェンダ

1. 暗号化を支える暗号鍵

2. AWS Key Management Service 概要

3. AWS Key Management Service で扱う暗号鍵

4. AWS Key Management Service の鍵管理

ライフサイクル管理
アクセス制御
モニタリング

5. AWS Key Management Service 暗号技術の詳細

【初級】
AWS KMS を利用される
全ての方向け

【中級】
AWS KMS の
暗号鍵を管理する方向け

【中級～上級】
AWS KMS の
安全性を確認する方向け

参考：Part.2 アジェンダ

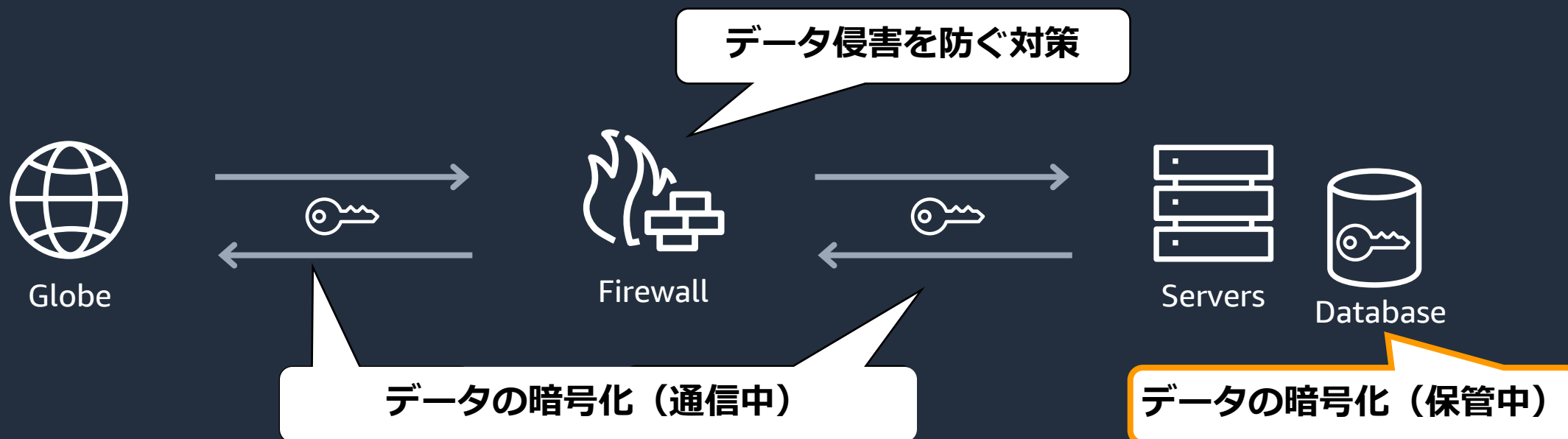
1. AWS KMS キーの種別
2. AWS Key Management Service の鍵管理 詳細
ライフサイクル管理
アクセス制御
3. AWS Key Management Service
様々な暗号化ユースケース

暗号化を支える暗号鍵



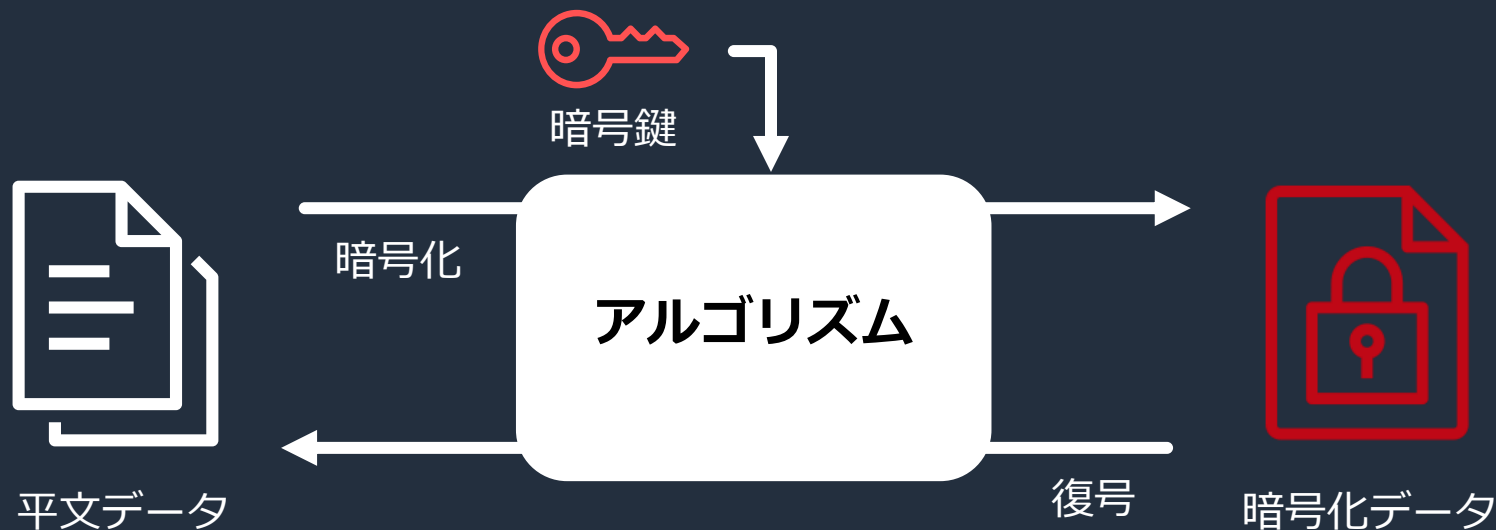
ビジネスにおけるデータ保護の必要性

- データは組織に固有のビジネス資産
 - 顧客情報、事業計画、設計文書、コードなど
- **ビジネスを保護するには、データの保護が必要**



暗号化とは

- 平文を暗号化し、暗号文にすることもしくは、暗号文を復号し、平文データに戻すこと
- 暗号化（復号）は、暗号鍵とアルゴリズムの組み合わせ
- 暗号鍵は秘密の文字列、アルゴリズムは公開されたアルゴリズム



暗号化の用途と暗号鍵管理の重要性

- 暗号化で実現できる性質
 - 機密性 : 許可された人だけが対象の情報にアクセスできる性質
 - 真正性 : 本物であることを確実に証明する性質
 - 完全性 : データが全て正確である状態を維持する性質
 - 非否認性 : 当事者による行動を後から否定をされないようにする性質
- 上記の性質を保つため、暗号鍵は決して漏洩してはならない
 - 暗号鍵は平文の秘密情報と同等の価値を持っている
 - 暗号鍵の安全な管理は重要な課題

暗号鍵の安全な管理において考慮すべきポイント

- **暗号鍵の保管**
 - どこに保管されるのか
 - 保管されているハードウェアは安全なのか
- **暗号鍵の管理**
 - 誰が鍵を使えるのか
 - いつ使えるのか
 - どこで鍵は使われるのか
 - どのような用途で鍵は使われるのか

鍵自体のセキュリティを担保する管理の仕組みがあるか？

暗号鍵管理の仕組みを構築・運用するには様々な検討が必要

- 参考：「暗号鍵管理システム設計指針 (基本編)」

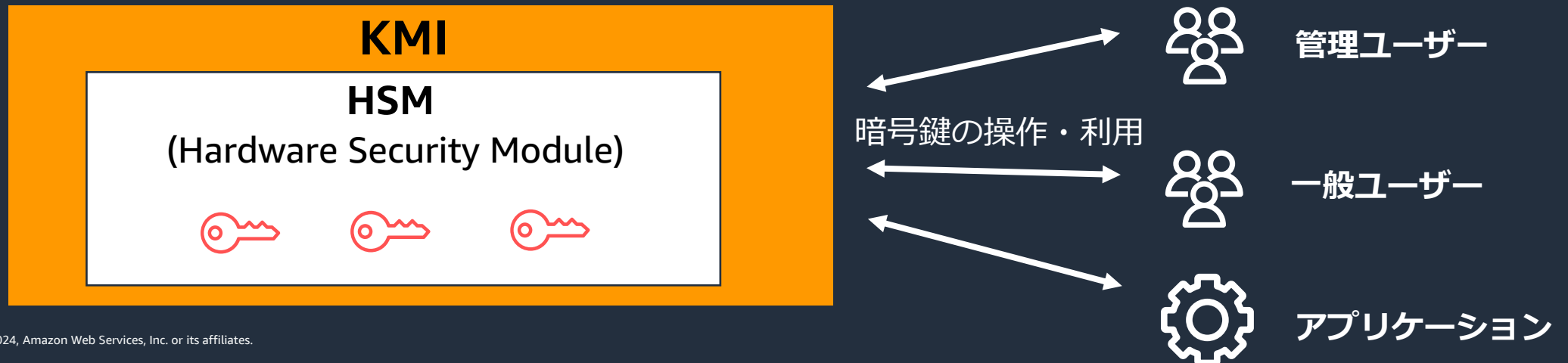
<https://www.ipa.go.jp/security/crypto/guideline/gmcbt80000005u7d-att/ipa-cryptrec-gl-3002-1.0.pdf>

- IPA、CRYPTREC による暗号鍵管理を安全に行うための考慮事項

#	タイトル	検討項目例	項目数
A	暗号鍵管理システムの設計原理と運用ポリシー	<ul style="list-style-type: none">どのようなポリシーで運用するのか参加者が誰でどういった権限を有しているのか	69
B	暗号アルゴリズム運用のための暗号鍵管理オペレーション対策	<ul style="list-style-type: none">どのような目的を持つ暗号鍵を利用するのかその暗号鍵はどこでどのように保管されるのか	81
C	暗号アルゴリズムの選択	<ul style="list-style-type: none">どのようなセキュリティ強度の暗号アルゴリズムを利用するか	2
D	暗号アルゴリズム運用に必要な鍵情報の管理	<ul style="list-style-type: none">どのように鍵情報を生成するかどのように窃取や改ざんなどから鍵情報を保護するか	10
E	暗号鍵管理デバイスのセキュリティ対策	<ul style="list-style-type: none">デバイスのセキュリティ確認のためにどのようなセキュリティ評価試験を実施するか	37
F	暗号鍵管理システムのオペレーション対策	<ul style="list-style-type: none">全体に対する包括的なセキュリティ対策をどうするか	57

暗号鍵の管理を行う、キー管理インフラストラクチャ (Key Management Infrastructure : KMI)

- 暗号鍵の保管：
鍵を安全に保管するストレージ
耐タンパ性を持つハードウェアセキュリティモジュール (HSM)
- 暗号鍵の管理：
ライフサイクル管理、アクセス制御、モニタリング



AWS Key Management Service

概要

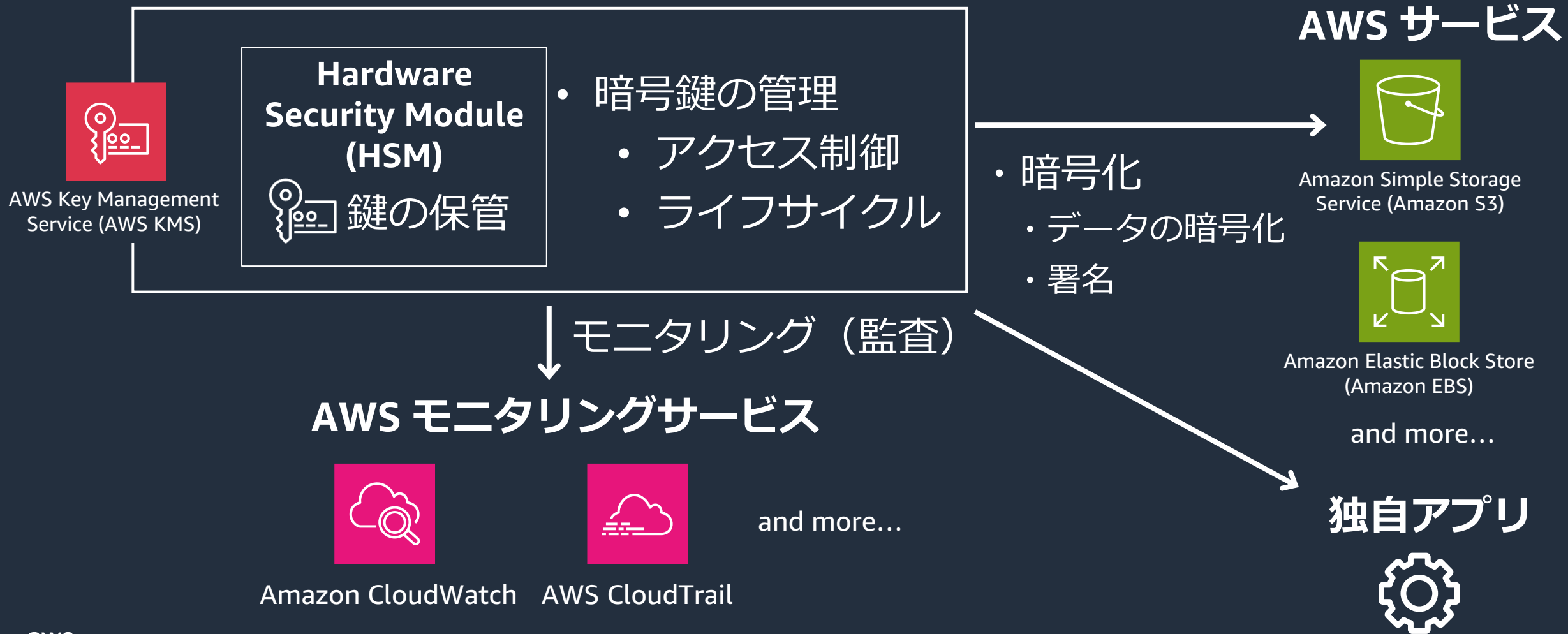
AWS Key Management Service (AWS KMS)



AWS Key Management Service (AWS KMS)

- 運用に様々な検討が必要で、暗号鍵の作成、保管、管理を実現するために必要な KMI を AWS が運用するマネージドサービス
- 100 以上の AWS サービスに統合されており、様々な暗号化のユースケースを実現可能
- AWS CloudTrail などのモニタリングサービスとの統合で暗号鍵の監査を実現

AWS KMS の概要



AWS KMS を利用した暗号鍵の保管・管理

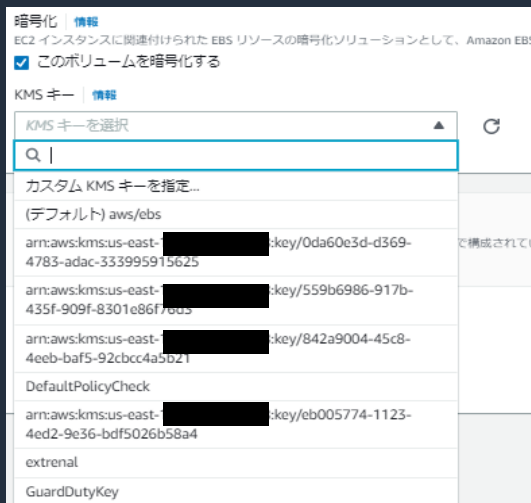
- 暗号鍵の保管場所の指定、ライフサイクル・アクセス制御などの暗号鍵の管理を、AWS KMS から一元的に操作可能
- 暗号鍵自体のセキュリティをマネージドな仕組みで提供

The screenshot displays the AWS KMS console interface. The top section, titled "一般設定" (General Settings), shows the key's details: "エイリアス" (Alias) is "SecretsEncryption", "ARN" is "arn:aws:kms:us-east-1:[redacted]:key/91b2273c-3ce1-4121-82d4-562cf575bf7e", "ステータス" (Status) is "有効" (Enabled), "作成日" (Created) is "2023年10月10日 10:05 JST", and "リージョンごと" (Per Region) is "単一リージョン" (Single Region). Below this is a navigation bar with tabs for "キーポリシー" (Key Policy), "暗号化設定" (Encryption Settings), "タグ" (Tags), "キーローテーション" (Key Rotation), and "エイリアス" (Alias). The "暗号化設定" (Encryption Settings) tab is active, showing "キーのタイプ" (Key Type) as "対称" (Symmetric), "オリジン" (Origin) as "AWS KMS", "キーの仕様" (Key Spec) as "SYMMETRIC_DEFAULT", and "キーの使用方法" (Key Usage) as "暗号化および復号化" (Encryption and Decryption).

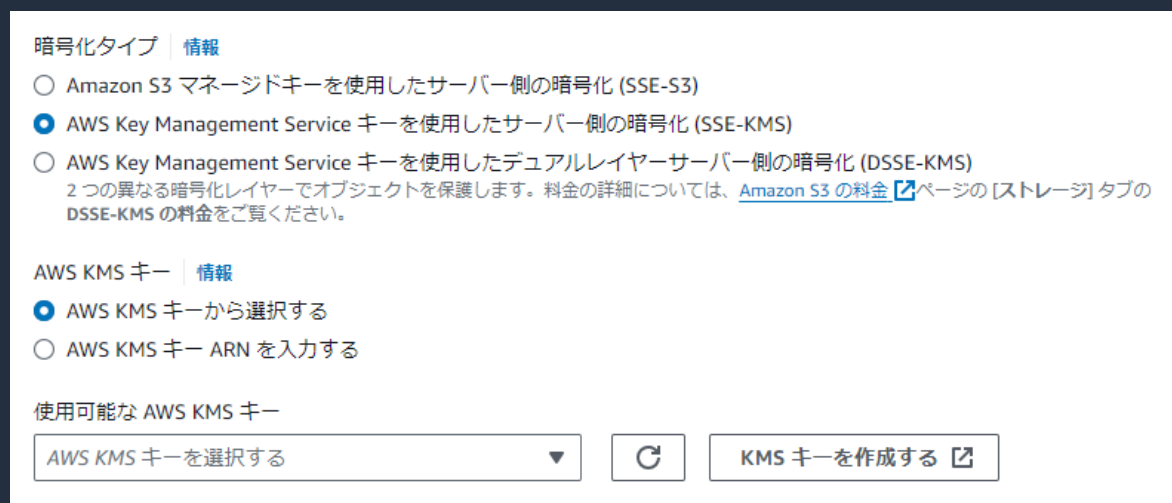
AWS KMS と AWS サービスの統合によるデータの暗号化

- データの暗号化は、サービスのバックエンドで実行されるため、ユーザーが必要な操作は利用する暗号鍵の指定のみ
- 暗号化オプションの詳細は、各サービスのユーザーガイドを参照

Amazon EBS



Amazon S3



プルダウンから利用したい暗号鍵を指定することで、保管中のデータを暗号化

参考：AWS KMS と統合されている AWS サービス

- 100 を超える AWS サービスとの統合により、データの暗号化や署名などのユースケースを実現

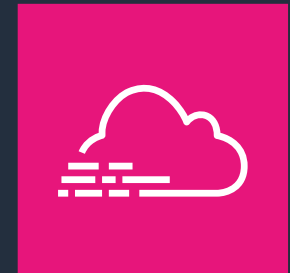
Alexa for Business ^[1]	Amazon FSx	Amazon Rekognition	AWS CodePipeline
Amazon AppFlow	Amazon GuardDuty	Amazon Relational Database Service (RDS)	AWS Control Tower
Amazon Athena	Amazon HealthLake	Amazon Route 53	AWS Data Exchange
Amazon Aurora	Amazon Inspector	Amazon Simple Storage Service (Amazon S3) ^[3]	AWS Database Migration Service
Amazon Chime SDK	Amazon Kendra	Amazon SageMaker	AWS Elastic Disaster Recovery
Amazon CloudWatch Logs	Amazon Keyspaces (Apache Cassandra 向け)	Amazon Simple Email Service (SES)	AWS Elemental MediaTailor
Amazon CloudWatch Synthetics	Amazon Kinesis Data Streams	Amazon Simple Notification Service (SNS)	AWS Entity Resolution
Amazon CodeGuru	Amazon Kinesis Firehose	Amazon Simple Queue Service (SQS)	AWS GameLift
Amazon CodeWhisperer	Amazon Kinesis Video Streams	Amazon Textract	AWS Glue
Amazon Comprehend	Amazon Lex	Amazon Timestream	AWS Glue DataBrew
Amazon Connect	Amazon Lightsail ^[1]	Amazon Transcribe	AWS Ground Station
Amazon Connect Customer Profiles	Amazon Location Service	Amazon Translate	AWS IoT SiteWise

Amazon Connect Wisdom	Amazon Lookout for Metrics	Amazon WorkSpaces	AWS Lambda
Amazon DocumentDB	Amazon Lookout for Vision	Amazon WorkSpaces シンククライアント	AWS License Manager
Amazon DynamoDB	Amazon Macie	Amazon WorkSpaces Web	AWS Mainframe Modernization
Amazon DynamoDB Accelerator (DAX) ^[1]	Amazon Managed Blockchain	AWS AppConfig	AWS Network Firewall
Amazon EBS	Amazon Managed Service for Prometheus	AWS AppFabric	AWS Proton
Amazon EC2 Image Builder	Amazon Managed Streaming for Kafka (MSK)	AWS Application Cost Profiler	AWS Secrets Manager
Amazon EFS	Amazon Managed Workflows for Apache Airflow (MWAA)	AWS Application Migration Service	AWS Snowball
Amazon Elastic Container Registry (ECR)	Amazon MemoryDB	AWS App Runner	AWS Snowball Edge
Amazon Elastic Kubernetes Service (EKS)	Amazon Monitron	AWS Audit Manager	AWS Snowcone
Amazon Elastic Transcoder	Amazon MQ	AWS Backup	AWS Snowmobile
Amazon ElastiCache	Amazon Neptune	AWS Certificate Manager ^[1]	AWS Storage Gateway



AWS KMS を利用した暗号鍵のモニタリングの実現

- 鍵管理に必要な要素を網羅する AWS CloudTrail によるログ管理
 - 誰が
 - いつ
 - どの AWS リソースから
 - どの AWS リソースに対して
 - どの API を用いたか（どのような操作を実行したか）



AWS CloudTrail

AWS Key Management Service で扱う暗号鍵

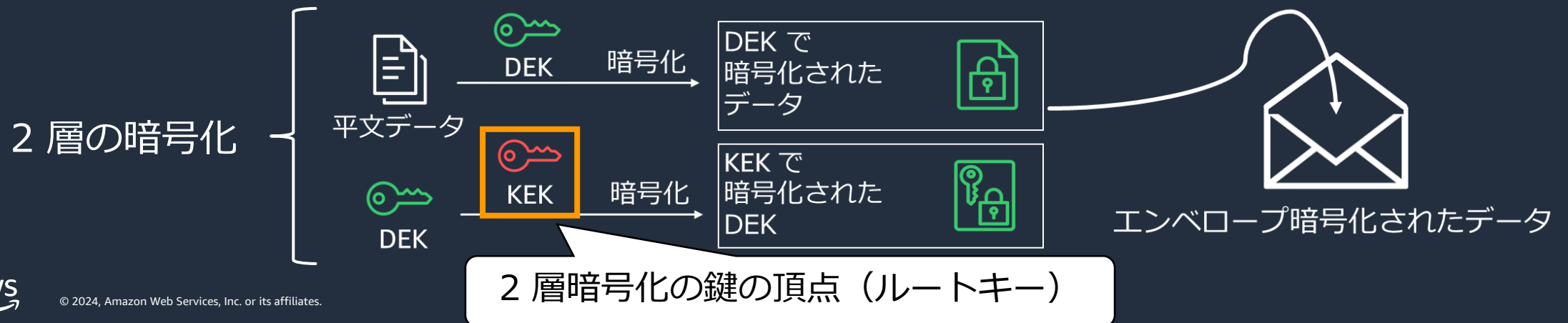
データ暗号化のプラクティス：エンベロープ暗号化

- 複数の暗号鍵を用いた暗号化方式
 - データ暗号化鍵 (Data Encryption Key: DEK) を用いてデータを暗号化
 - DEK は別の暗号鍵 (Key Encryption Key: KEK) を用いて暗号化
 - DEK で暗号化されたデータと、KEK で暗号化された DEK を一緒に保管
平文の DEK は利用が終わり次第、廃棄



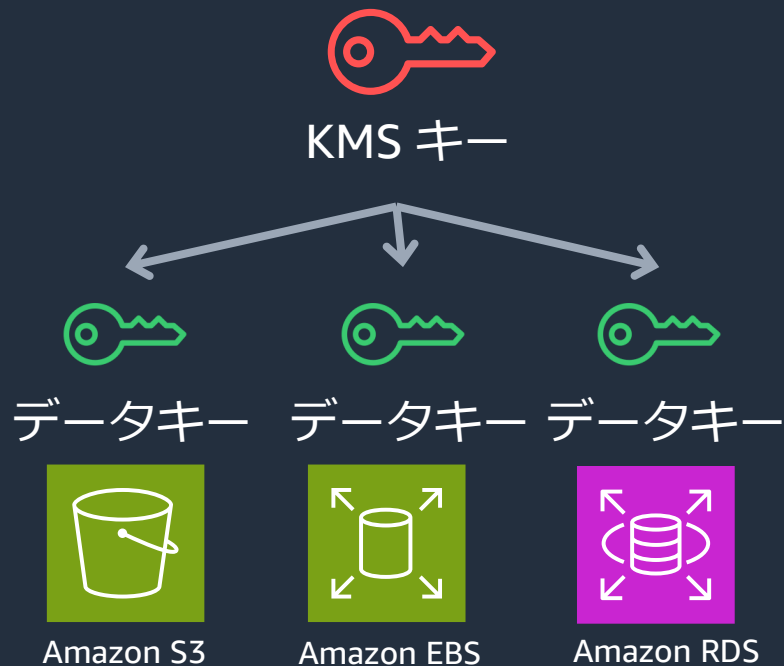
エンベロープ暗号化のメリット

- 2層レイヤーでの暗号化によるデータの保護
- 暗号鍵の中央管理の簡易化
 - KEK が安全に保管すべきルートとなるキー
 - データの暗号化に DEK を利用するため、KEK の数を最小限に抑えられる



AWS KMS で扱う暗号鍵と暗号鍵ヒエラルキー

- ルートキーである KEK : KMS キー (旧 Customer Master Key: CMK)
- DEK : データキー



- KMS キーから、データキーを生成
- 各サービスにおける暗号化は、**エンベロープ暗号化**によって実現

AWS KMS の KMS キー：マスターとなる暗号鍵 (KEK)

- 鍵ヒエラルキーの頂点に位置する “論理的な” 暗号鍵
 - キーマテリアル（暗号鍵の基となるプレーンテキスト）やメタデータなどがセットになっている単位で、関連するデータを保管する **コンテナ**
 - ローテーションなどを通じてキーマテリアルは変化するが、論理的なコンテナとしての KMS キーはそのまま利用可能
- 暗号化された状態で耐久性の高いストレージに保管
- AWS KMS 外でキーマテリアルの利用・エクスポートができないよう設計



KMS キー • キーマテリアル（暗号鍵の基）
• キー ID、キー用途 などのメタデータ

コンテキストに応じて利用可能な KMS キーの種別

- AWS KMS では、暗号鍵の所有、用途、利用できるリージョン、キーマテリアル（暗号鍵の基となるプレーンテキスト）のオリジン、など、お客様のコンテキストに基づく、暗号鍵の種別の選択が可能

 KMS キー	鍵の所有	: カスタマーマネージド / AWS マネージド / AWS 所有
	鍵の用途	: 対称 / 非対称 / HMAC
	鍵のリージョン	: 単一リージョン / マルチリージョン
	鍵のオリジン	: KMS / インポート / カスタムキーストア

AWS KMS のデータキー：データを暗号化する暗号鍵 (DEK)

- 256 ビット、128 ビット、最大 1,024 バイトまでの任意の鍵長
- エンベロープ暗号化を通じた大容量データの暗号化が可能
- KMS キーで直接暗号化する場合、最大 4 KB
- データキーは暗号化操作を行うクライアント側で管理
- 暗号化処理が終了次第、平文のデータキーは即削除



データキー

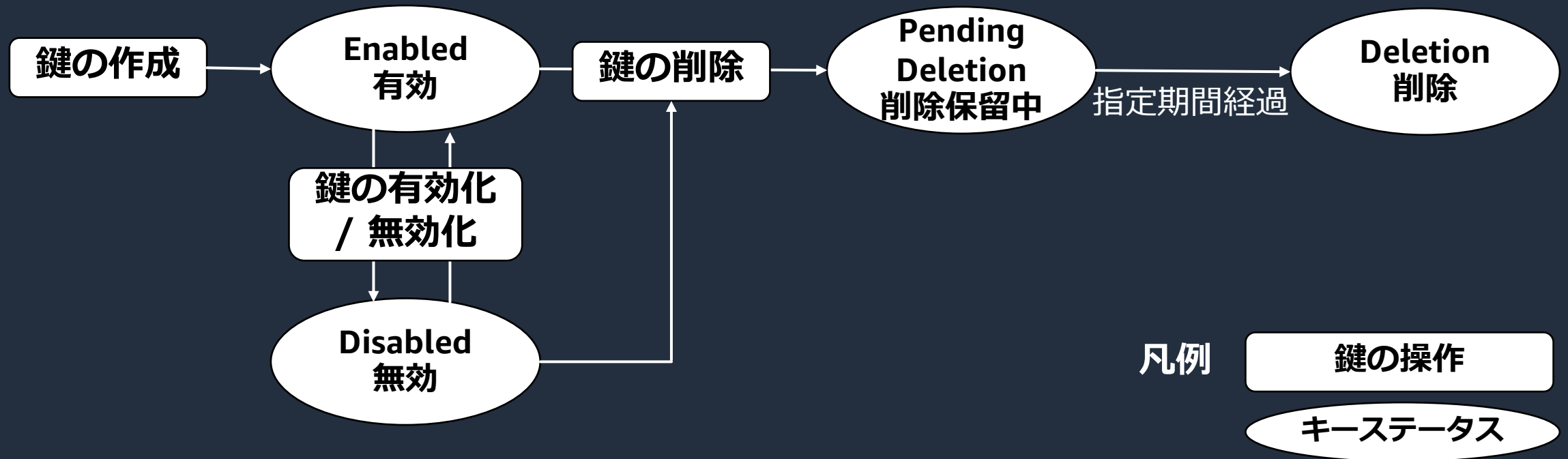
AWS Key Management Service の鍵管理

AWS KMS の鍵管理機能

- ライフサイクル
 - キーの作成
 - キーのローテーション
 - キーの削除
- アクセス制御
 - キーポリシー
 - IAM ポリシー
- モニタリング

KMS キーのライフサイクル

- KMS キーの作成から削除までのライフサイクルの状態を表すキーステータス



KMS キーの作成

- GUI、API、AWS CloudFormation (Infrastructure as Code)

The screenshot displays the AWS Key Management Service (KMS) console interface. On the left, a navigation pane shows 'Key Management Service (KMS)' with options for 'AWS マネージド型キー', 'カスタマー管理型のキー', and '▼ カスタムキーストア'. The main content area features the heading 'AWS Key Management Service' and the sub-heading 'AWS やその外部で容易にキーを作成し暗号化を管理'. Below this, a paragraph describes KMS as a managed service for key creation and management. A prominent orange circle highlights the 'キーの作成' (Create Key) button in the '今すぐ始める' (Get started now) section. Other sections visible include '料金' (Pricing) with a link to '詳細はこちら' (Learn more here).

KMS キーのローテーション

- 暗号化のベストプラクティスにおいて、暗号化キーの広範な再利用は推奨されない
- 有効な KMS キーを一定期間ごとにローテーションする必要
- 自動方式と手動方式のローテーションを選択可能
 - 自動：キーマテリアルを毎年追加、古いマテリアルも保存
 - 手動：KMS キー自体を新たに作成、キーの参照の変更が必要

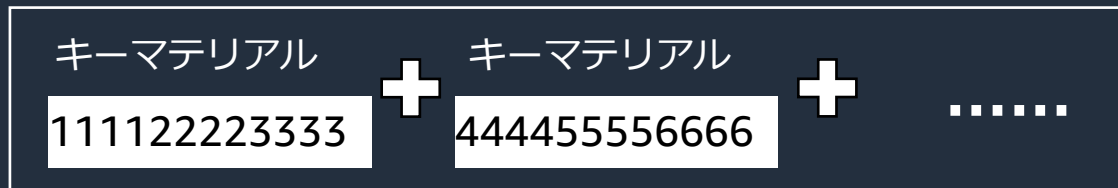


KMS キー

- キーマテリアル（暗号鍵の基）
- キー ID、キー用途 などのメタデータ

KMS キーローテーションの更新イメージ

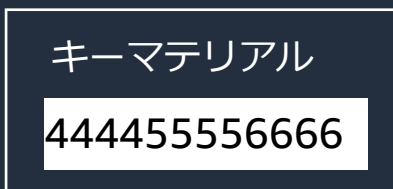
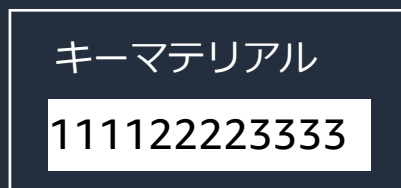
自動



コンテナは不変
キー ID 等を継続利用可能

キーマテリアルが毎年自動更新（追加）

手動



コンテナ自体を新規作成
キー ID も新規に割り当て
参照関係の見直しが必要



.....

参考：KMS キーのローテーション方式比較

方式	説明	対象	特徴
自動	新しいキーマテリアルを生成 コンテナとしての AWS KMS キーは不変	AWS KMS で 作成した 対称暗号化キー	<ul style="list-style-type: none">定期的な自動更新ローテーション後は、新しいキーマテリアルでデータを暗号化過去のキーマテリアルを保管するため、過去暗号化されたデータの復号が容易
手動	新しい KMS キーを作成 サービスやアプリケーションの参照先を変更必要	上記以外のキー	<ul style="list-style-type: none">更新頻度はお客様判断過去のキーマテリアルの保管有無はお客様判断過去のキーマテリアルの有無に応じ、ローテーション時のデータの再暗号化が必要

KMS キーの削除

KMS > カスタマー管理型のキー > キーの削除をスケジュール

キーの削除をスケジュール

⚠ キーを削除すると、そのキーで暗号化したすべてのデータが回復できなくなります。 [詳細はこちら](#)

待機期間中にキーを使用しようとした場合にアラートを送信する、Amazon CloudWatch アラームを作成できます。 [詳細はこちら](#)

待機期間

待機期間が終了する前は、いつでも削除をキャンセルできます。待機期間が終了すると、AWS KMS によってキーが削除されます。 [詳細はこちら](#)

待機期間 (日数)

7~30 日間の待機期間を入力します。

削除するキー

以下のキーは削除が予定されます。

エイリアス	キー ID	キーの使用	リージョンごと
DefaultPolicyCheck	[REDACTED]	暗号化および復号化	単一リージョン

確認

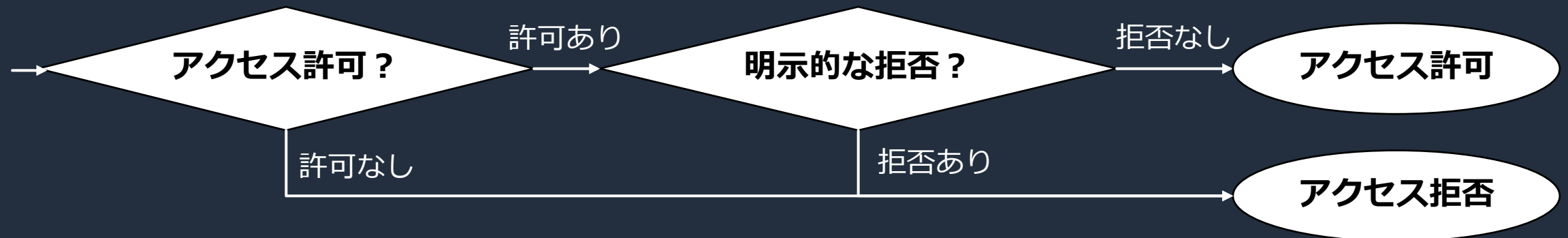
30 日の待機期間後にこれらのキーの削除をスケジュールすることを確認します。

キャンセル **削除をスケジュール**

- KMS キーの削除操作は元に戻せない
- コンテナ含む KMS キーの削除
- 暗号化されたデータの復号不可能
- 将来の利用が想定される場合、無効化を活用
- 7 – 30 日間の削除待機期間を指定
- 待機期間中に削除のキャンセルが可能
- 削除予定の鍵のモニタリング推奨
 - AWS CloudTrail
 - Amazon CloudWatch Logs

KMS キーのアクセス制御：ポリシーの利用

- KMS キーに紐づいているキーポリシーと KMS キーに適用される全ての認可（IAM ポリシーなど）に基づく
 - アクセス権限に “Allow” があった場合、アクセス許可
 - ただし、明示的な “Deny” がある場合、アクセス拒否
 - デフォルトアクセス拒否 < アクセス許可 < 明示的なアクセス拒否

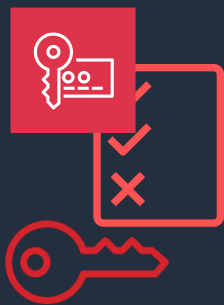


KMS キーの主要なアクセスコントロール

- AWS KMS における主要な 3 つのアクセスコントロール

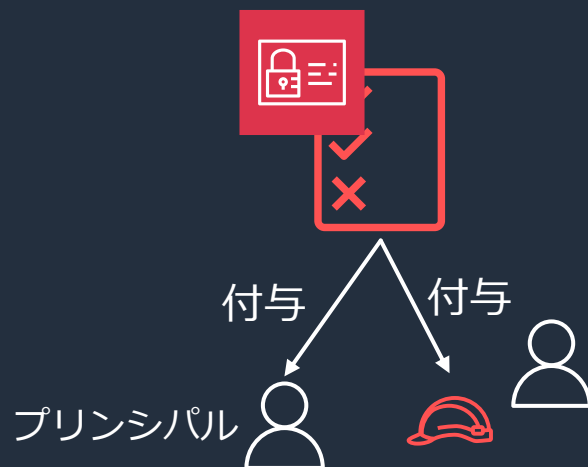
キーポリシー

キーと 1 : 1 に紐づく、
リソースベースの
パーミッション



IAM ポリシー

プリンシパルベース
のパーミッション



グラント

クライアントによる
他プリンシパルへの
パーミッション委任

Part.2 スコープ



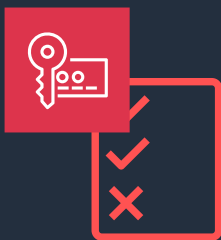
KMS キーのアクセスコントロールのイメージ

デフォルトキーポリシーで、IAM ポリシーを利用した認可を許可

プリンシパル

KMS キー

IAM ユーザー IAM ロール



キーポリシー

IAM ポリシー

KMS キーの
操作

KMS キー操作への
認可有無を確認

アクセス許可があればアクセス許可
どちらかで明示的な拒否があればアクセス拒否

KMS キーのキーポリシー

- キーごとに1つ設定可能なリソースベースのポリシー
- JSON 形式
- キーごとのアクセス制御が行える
キーポリシーの利用をアクセス制御運用の上で優先することが
AWS KMS のベストプラクティス
- IAM ポリシーでのアクセス許可を行うには、
キーポリシー内での IAM ポリシー利用の有効化が必要
(デフォルトのキーポリシーでは IAM ポリシー有効)

KMS キーのデフォルトキーポリシー

- AWS アカウントに対してアクセス許可を与え、IAM ポリシーを利用したアクセス制御を有効化
- アカウントに制御を与えることで、鍵の管理不能リスクを軽減

```
{  
  "Sid": "Enable IAM policies",  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": "arn:aws:iam::111122223333:root"  
  },  
  "Action": "kms:*",  
  "Resource": "*" }  
}
```

1. IAM ポリシーを利用した認可を可能にする
2. アカウントに権限を付与

KMS キーのキーポリシーを利用した認可のイメージ



鍵の管理者

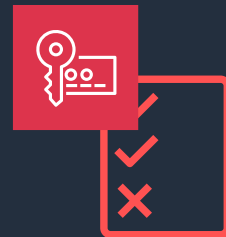
鍵の削除、鍵のタグ付け、鍵のローテーション設定の変更 など、鍵の管理操作



KMS キー



KMS キー



キーポリシー

1つのポリシー内に
各プリンシパルに必要な認可を記述



鍵の利用者（アプリケーション開発者など）

暗号化操作、鍵の表示 など、鍵の利用操作



暗号操作の対象リソース

KMS キーへのアクセス許可を付与するキーポリシー例

- 1つのキーポリシー内に管理者と利用者に対するアクセス許可

```
{
  "Sid": "Allow access for Key Administrators",
  "Effect": "Allow",
  "Principal": {"AWS": [
    "arn:aws:iam::111122223333:user/KMSAdminUser",
    "arn:aws:iam::111122223333:role/KMSAdminRole"
  ]},
  "Action": [
    "kms:Create*", "kms:Describe*", "kms:Enable*",
    "kms:List*", "kms:Put*", "kms:Update*",
    "kms:Revoke*", "kms:Disable*", "kms:Get*",
    "kms>Delete*", "kms:TagResource",
    "kms:UntagResource", "kms:ScheduleKeyDeletion",
    "kms:CancelKeyDeletion"
  ],
  "Resource": "*"
}
```

管理用ユーザー・ロールに
鍵の削除などの管理操作を許可

```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": [
    "arn:aws:iam::111122223333:user/ExampleUser",
    "arn:aws:iam::111122223333:role/ExampleRole",
    "arn:aws:iam::444455556666:root"
  ]},
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

鍵の利用ユーザーに
暗号化、データキーの作成などの
データ暗号化に必要な操作を許可

KMS キーの IAM ポリシー

- プリンシパルベースの JSON 形式のポリシー
- (前提) IAM ポリシーを使用して KMS キーへのアクセスを制御するには、KMS キーのキーポリシーが IAM ポリシーを使用するアクセス許可をアカウントに付与する必要がある
- アクセス制御ではキーポリシーの利用を運用上で優先するが、キーポリシーでは制御できない KMS キーの作成操作の制御に有効

KMS キーの IAM ポリシーを利用した認可のイメージ

デフォルトキーポリシーで、IAM ポリシーを利用した認可を許可



プリンシパルに対して、必要な鍵操作の認可を付与

プリンシパル

**KMS キーの
操作**

KMS キーへのアクセス許可を付与する IAM ポリシー例

- KMS キーの暗号操作への利用に対するアクセス許可

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "kms:DescribeKey",
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": [ "arn:aws:kms:*:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890a" ]
  }
}
```

特定の KMS キーを利用した
暗号操作を許可するポリシー

KMS キーの IAM ポリシーにおけるベストプラクティス

- キーの作成 (CreateKey) 権限を最少範囲に制限するために利用
- 権限を付与する KMS キー (Resource 句) をポリシーで指定する
- Resource 句での、ワイルドカード "*" 指定を避ける
 - 他の AWS アカウントでの KMS キー操作に対する許可となるリスク
 - キーの表示 : DescribeKey
 - ローテーション設定の確認 : GetKeyRotationStatus
 - 暗号化と復号に関わる操作 : GenerateDataKey, Decrypt など

参考：KMS キーの認可パターン詳解

	キーポリシー		IAM ポリシー	認可の結果
	IAM ポリシーを利用した認可の許可	キーポリシーによる鍵操作の許可	鍵操作の許可	
パターン 1	あり	あり	あり	アクセス許可
パターン 2	あり	あり	なし	アクセス許可
パターン 3	あり	なし	あり	アクセス許可
パターン 4	あり	なし	なし	アクセス拒否
パターン 5	なし	あり	あり	アクセス許可
パターン 6	なし	あり	なし	アクセス許可
パターン 7	なし	なし	あり	アクセス拒否
パターン 8	なし	なし	なし	アクセス拒否

詳細は Part.2

KMS キーのモニタリング

- AWS CloudTrail
 - コンソールや CLI による KMS キーの操作を全て記録
 - 暗号鍵について、誰が、いつ、何の操作をしたか、追跡可能
- Amazon CloudWatch
 - KMS キーの使用や変更に関するメトリクスの監視、アラートも可能

AWS KMS の AWS CloudTrail を利用したログ管理

- ユーザー、ロール、その他の AWS のサービスからの全ての呼び出し（オペレーション成功・失敗）を記録
- キーマテリアルの自動キーローテーション等、AWS KMS 内部のオペレーションも全て記録
- デフォルトで 90 日間の履歴を保管
長期の保管が必要な際には、証跡を作成
- 平文情報などの機微情報はログから省略される

AWS KMS の AWS CloudTrail ログ例

```
{
  "eventVersion": "1.02",
  "userIdentity": {
    "type": "IAMUser", "principalId": "EX_PRINCIPAL_ID",
    "arn": "arn:aws:iam:111122223333:user/Alice",
    "accountId": "111122223333", "accessKeyId": "EXAMPLE_KEY_ID", "userName": "Alice"
  },
  "eventTime": "2022-07-14T20:17:42Z",
  "eventSource": "kms.amazonaws.com", "eventName": "Encrypt",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "AWS Internal",
  "requestParameters": { "encryptionContext": { "Department": "Engineering" },
    "keyId": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
    "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
  },
  "responseElements": null,
  "requestID": "f3423043-63bc-11e4-bc2b-4198b6150d5c",
  "eventID": "91235988-eb87-476a-ac2c-0cdc244e6dca",
  "readOnly": true,
  "resources": [{
    "ARN": "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab", "accountId": "111122223333"
  }],
  "eventType": "AwsApiCall", "recipientAccountId": "111122223333"
}
```

誰が

いつ

何をしたか

どの鍵を用いて

AWS KMS の Amazon CloudWatch を利用したログ管理

- KMS キーの重要なメトリクス・イベントについて監視やメールによるアラートが可能
 - AWS KMS に対するリクエスト数
 - 削除保留中のキーの利用
 - キーの自動ローテーション
 - キーの削除

など

AWS Key Management Service

暗号技術の詳細

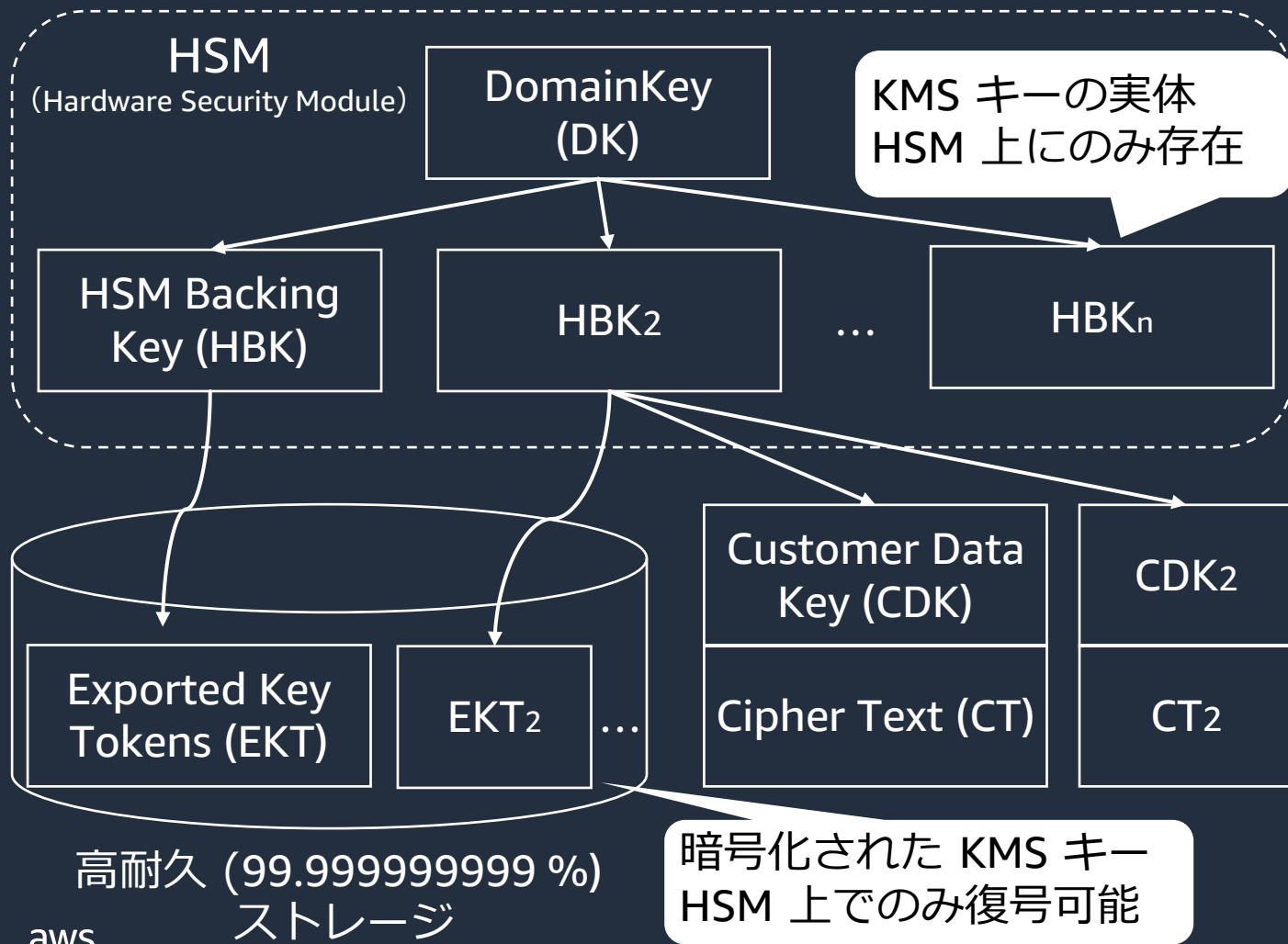


AWS KMS の内部アーキテクチャ

- FIPS 140-2 セキュリティレベル 3 認証済の HSM
- 暗号鍵は、暗号化処理の時のみ HSM 上で利用可能
- KMS アーキテクチャ外への KMS キーのエクスポートはできない



AWS KMS の内部アーキテクチャ詳細



種類	説明
Domain Key	<ul style="list-style-type: none"> • HSM メモリ上でのみ存在 • AES-GCM 256 ビットキー • 日次ローテーション
HSM Backing Key	<ul style="list-style-type: none"> • KMS キーの実体 • 256 ビット対称キー、もしくは、RSA または 楕円曲線キー • エクスポート不可 • 利用時に HSM の揮発性メモリ上で復号
Exported Key Tokens	<ul style="list-style-type: none"> • DK で暗号化された HBK • KMS アーキテクチャ内の高耐久ストレージに保管
Customer Data Key	<ul style="list-style-type: none"> • データキーの実体 • HBK (KMS キー) で暗号化



AWS KMS の Hardware Security Module (HSM) 詳細



Figure 1 – Cryptographic Module Boundary (Front)



Figure 2 - Cryptographic Module Boundary (Back)

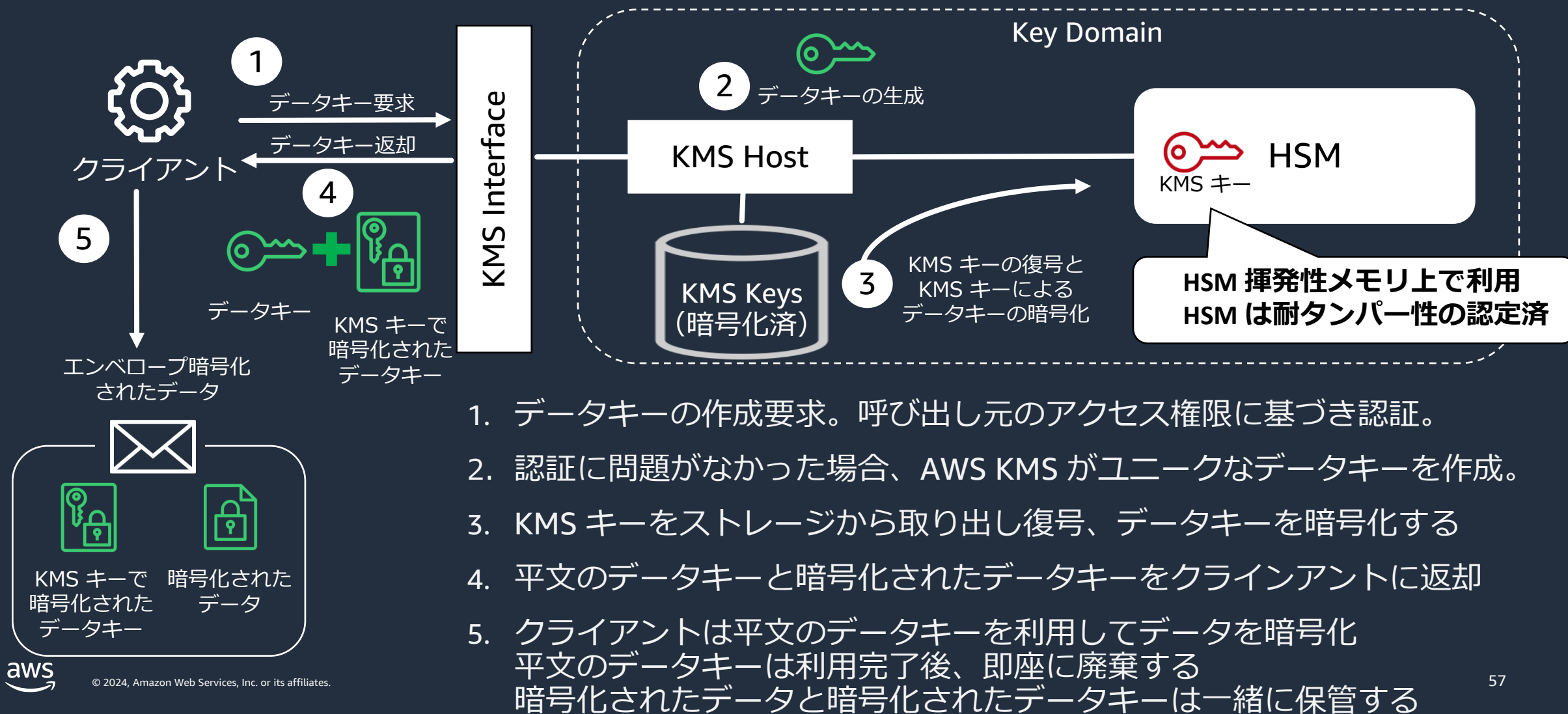
FIPS 140-2 Section Title	Validation Level
Cryptographic Module Specification	3
Cryptographic Module Ports and Interfaces	3
Roles, Services, and Authentication	3
Finite State Model	3
Physical Security	3
Operational Environment	N/A
Cryptographic Key Management	3
Electromagnetic Interference / Electromagnetic Compatibility	3
Self-Tests	3
Design Assurance	3
Mitigation of Other Attacks	N/A
Overall Level	3

AWS KMS で利用可能な暗号アルゴリズム

- FIPS 140-2 では、弱い暗号アルゴリズムの利用に制限

Approved Algorithms	
AES	Certs. #A1791 and #A1908
CKG	vendor affirmed
CVL	Cert. #A1908
DRBG	Certs. #A1791 and #A1908
ECDSA	Cert. #A1908
ENT	P
HMAC	Cert. #A1908
KAS	Cert. #A1908 ; key establishment methodology provides 192 bits of encryption strength
KBKDF	Cert. #A1910
KDA	Cert. #A1908
KTS	AES Cert. #A1908
KTS-RSA	Cert. #A1908 ; key establishment methodology provides between 112 and 150 bits of encryption strength
RSA	Cert. #A1908
SHS	Cert. #A1908
Allowed Algorithms	RSA (key wrapping; key establishment methodology provides between 112 and 150 bits of encryption strength)

エンベロープ暗号化のフロー詳細：KMS キーは HSM 上のみ



AWS KMS の KMS キーのセキュリティ

- KMS キー（マスターとなる Key Encryption Key）は AWS KMS アーキテクチャ外にエクスポートできないよう設計
 - 平文のKMS キーは、ディスクに書き込まず、保護された HSM の揮発性メモリでのみ使用
 - FIPS140-2 セキュリティレベル 3 検証済の耐タンパ性を持つ HSM を利用
 - AWS KMS サービス内のソフトウェアアップデートは、Amazon 内の独立したグループおよびFIPS 140-2 を順守した NIST 認証のラボによる監査を受けた複数のアクセスコントロールのもとで実施

AWS KMS のコンプライアンス

- FIPS 140-2 セキュリティレベル 3
- AWS System and Organization Controls (SOC 1、SOC 2、および SOC 3)
- PCI DSS レベル 1
- ISO および CSA STAR 認証
- FedRAMP
- HIPAA

など、その他の認証も取得



Part.1 まとめ



まとめ

- AWS KMS は、データ保護を実現するための基盤で、かつ、暗号鍵の作成、保管、管理を行うための KMI を運用する AWS マネージドサービス
- 100 以上の AWS サービスと統合されており、データ保護のプラクティスであるエンベロープ暗号化によるデータ暗号化を含めた、様々な暗号化のユースケースを実現可能
- エンベロープ暗号化のルートキーである KMS キーについて ライフサイクル管理・アクセス制御・モニタリングを一元管理
- 暗号鍵のセキュリティについて、第三者認証を取得済

Part. 2 発展編

本セミナーの対象者

- AWS 環境における暗号化のユースケースに関心をお持ちの方
- AWS Key Management Service をご利用予定の方
- AWS Key Management Service について理解を深めたい方

本セミナーで得られること

- AWS Key Management Service の暗号化の仕組み
- AWS Key Management Service の鍵管理の仕組み
- AWS Key Management Service による暗号鍵保管・管理の仕組みを社内のセキュリティ部門、コンプライアンス部門に説明できる

Part.2 アジェンダ

1. AWS KMS キーの種別
2. AWS Key Management Service の鍵管理 詳細
(ライフサイクル管理・アクセス制御)
3. AWS Key Management Service 様々な暗号化ユースケース

参考 : Part.1 アジェンダ

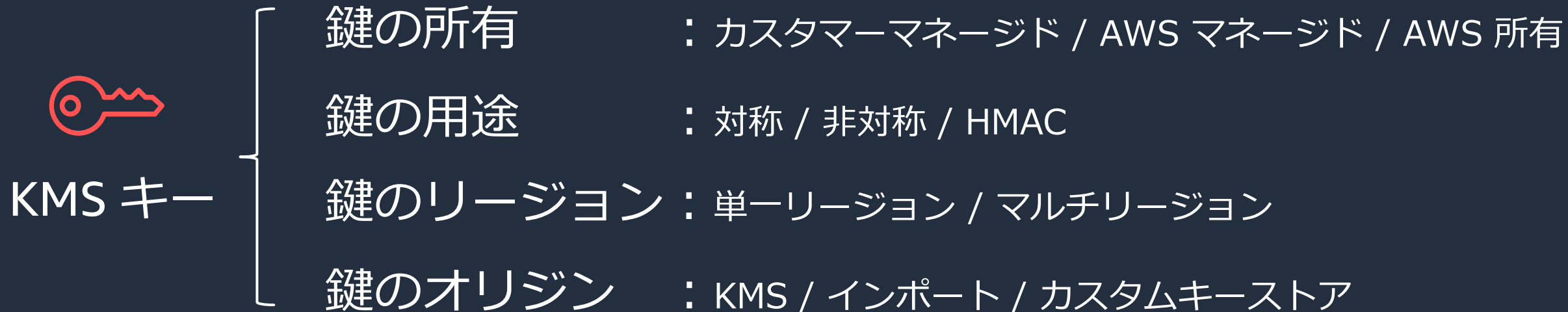
1. 暗号化を支える暗号鍵
2. AWS Key Management Service 概要
3. AWS Key Management Service で扱う暗号鍵
4. AWS Key Management Service の鍵管理
(ライフサイクル管理・アクセス制御・モニタリング)
5. AWS Key Management Service 暗号技術の詳細

AWS KMS キーの種類別



コンテキストに応じて利用可能な KMS キーの種別

- AWS KMS では、暗号鍵のオーナーシップ、用途
利用できるリージョン、キーマテリアル（暗号鍵の基となるプ
レーンテキスト）のオリジン、など、お客様のコンテキストに基
づく、暗号鍵の種別の選択が可能



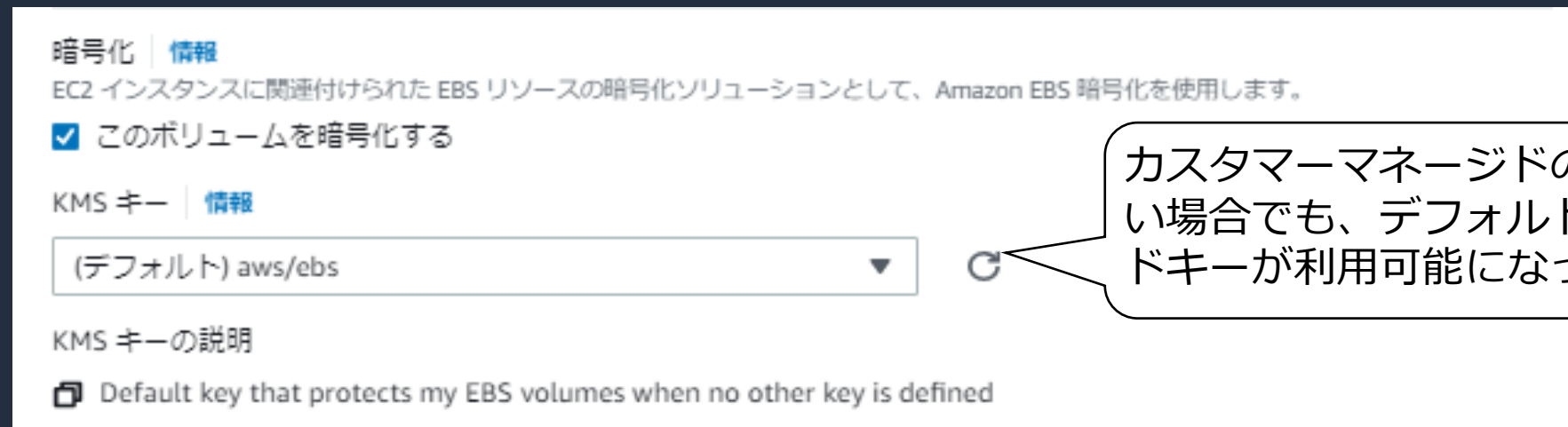
鍵の所有に基づく KMS キー種別

種類	作成	管理			お客様アカウント以外での鍵利用
		アクセス制御	ライフサイクル	モニタリング	
カスタマーマネージドキー	お客様	お客様	お客様 自動方式を選択した場合、毎年更新	お客様	なし
AWS マネージドキー	AWS	AWS お客様も閲覧は可能	AWS 自動更新毎年	AWS お客様も閲覧は可能	なし
AWS 所有のキー	AWS	AWS	AWS 自動更新頻度は可変	AWS お客様の閲覧不可	あり

AWS マネージドキーの利用

- 一部の AWS サービスでは、AWS 管理のキーを使用した暗号化が可能
- 暗号化オプションの詳細は、各サービスのドキュメント参照

Amazon EBS の例



暗号化 | 情報

EC2 インスタンスに関連付けられた EBS リソースの暗号化ソリューションとして、Amazon EBS 暗号化を使用します。

このボリュームを暗号化する

KMS キー | 情報

(デフォルト) aws/ebs

KMS キーの説明

Default key that protects my EBS volumes when no other key is defined

カスタマーマネージドの鍵を作成していない場合でも、デフォルトで、AWS マネージドキーが利用可能になっている

鍵の用途に基づく KMS キー種別

種類	概要	ユースケース
対称暗号化	デフォルト 共通鍵形式 256 ビット長	AWS サービス内のデータ暗号化 ※最もよく利用される鍵種類
非対称	公開鍵形式 公開鍵はエクスポート可能	メッセージの署名 AWS 外データの暗号化 (ECC 除く)
HMAC	共通鍵形式 様々な長さの鍵を作成可能	メッセージ認証コード

カスタマーマネージドキーのみ対象

利用可能な AWS KMS キー仕様とアルゴリズム

種類	キータイプ	アルゴリズム
対称	256 ビットキー	AES-GCM
非対称	RSA 2048 RSA 3072 RSA 4096 ECC NIST P-256 ECC NIST P-384 ECC NIST-521 ECC SECG P-256k1	非対称暗号 RSAES_OAEP_SHA_1 RSAES_OAEP_SHA_256 署名 RSASSA_PSS_SHA_256 RSASSA_PSS_SHA_384 RSASSA_PSS_SHA_512 RSASSA_PKCS1_V1_5_SHA_256 RSASSA_PKCS1_V1_5_SHA_384 RSASSA_PKCS1_V1_5_SHA_512

利用可能な AWS KMS キー仕様とアルゴリズム (続き)

種類	キータイプ	アルゴリズム
HMAC	HMAC_224	HMAC_SHA_224
	HMAC_256	HMAC_SHA_256
	HMAC_384	HMAC_SHA_384
	HMAC_512	HMAC_SHA_512

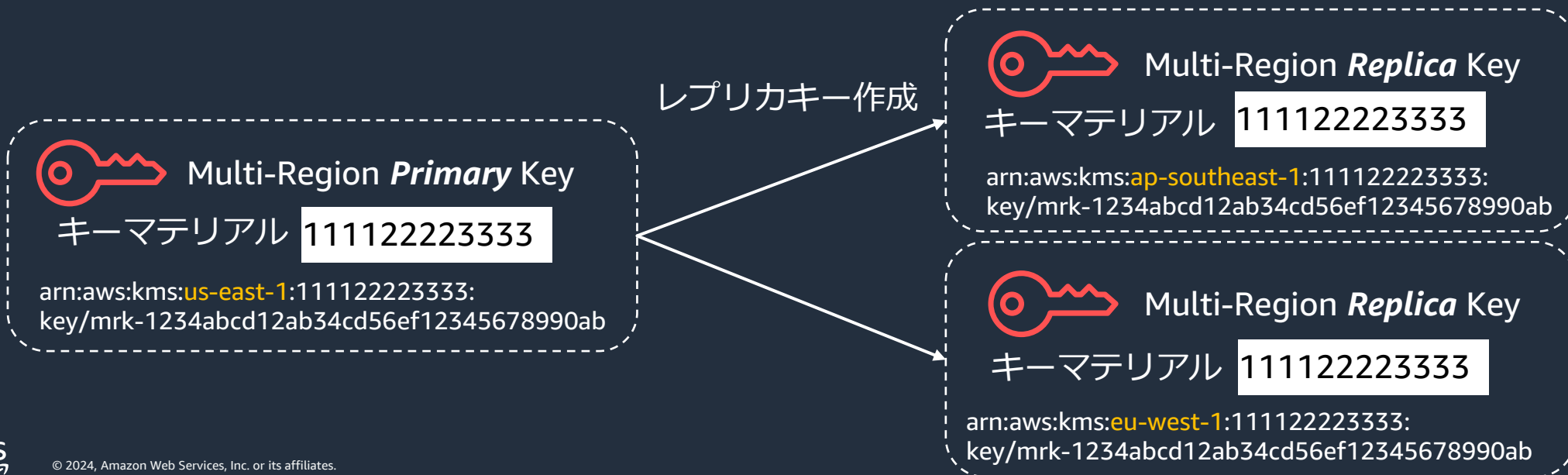
鍵のリージョンに基づく KMS キー種別

- マルチリージョンキーは、アクセス制御、データの暗号化ポリシー、キーのモニタリングや監査など複雑化
- マルチリージョンキーは、本当に必要な時だけ利用

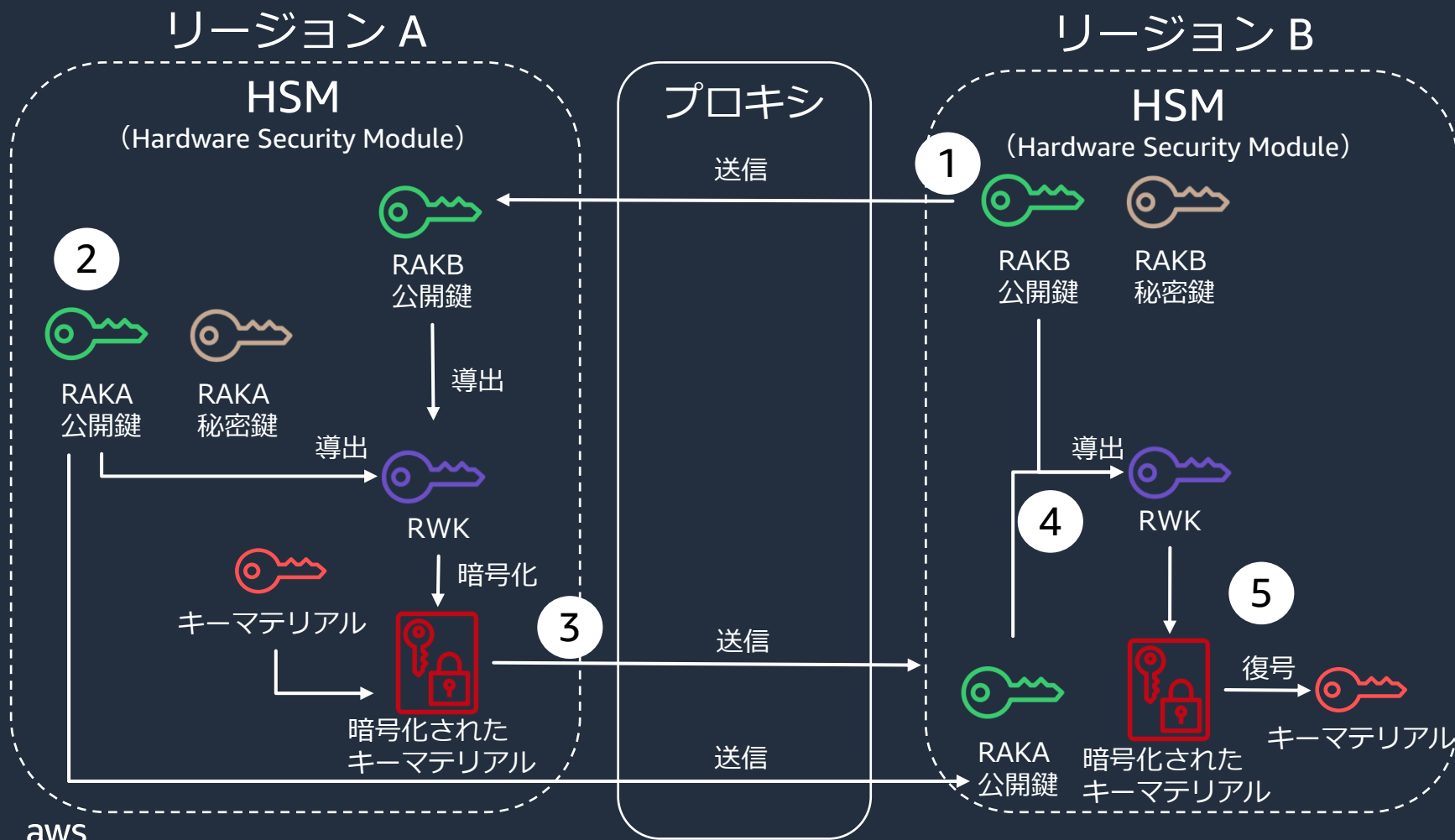
種類	概要	ユースケース
単一リージョン	デフォルト方式 1 リージョンでキーを作成	• デフォルト
マルチリージョン	複数のリージョンに複製可能なキーを作成	• リージョン停止などに備えた DR • グローバルに展開しているアプリケーション

マルチリージョンキーの仕組み

- 複数リージョンで同じキーのように利用可能
- 災害時のバックアップ / グローバルなデータ管理 など
- 統合された AWS サービスによる暗号化は透過的に行われるため、主にクライアントサイドの暗号化向け



参考：マルチリージョンキーのレプリケーション詳細



1. リージョン B の HSM は、NIST P-384 曲線上に EDCH キー (RAKB) を作成 RAKB 公開鍵をリージョン A に送付
2. リージョン A の HSM は、NIST P-384 曲線上に EDCH キー (RAKA) を作成 RAKB と RAKB から対称キー (RWK) を導出
3. RAKA 公開鍵と RWK で暗号化されたキーマテリアルをリージョン B に送付
4. リージョン B の HSM は、RAKB と RAKB から対称キー (RWK) を導出
5. リージョン B の HSM は、RWK を使用して、キーマテリアルを復号

キーマテリアルのオリジンに基づく KMS キー種別

種類	鍵の生成	鍵の保管	鍵の管理
AWS KMS	KMS	KMS	KMS
キーのインポート	お客様	KMS	KMS
カスタムキーストア	AWS CloudHSM or お客様	AWS CloudHSM or お客様	KMS

カスタマーマネージドキーのみ対象

キーマテリアルのオリジンに基づく KMS キー種別イメージ

モデル 1

AWS KMS によるキー生成

モデル 2

キーのインポート

モデル 3

カスタムキーストア



KMS キーのインポート : Bring Your Own Key (BYOK)

- お客様独自のキーマテリアルを AWS KMS に持ち込みたい場合に利用可能
- 主要な利用ユースケース
 - 要件を満たすエントロピーのソースを使用したことを証明したい
 - 自社のキーマテリアルを利用したいが、AWS KMS のインフラを活用したい
 - 暗号鍵の有効期限を設定したい
 - オリジナルのキーマテリアルは AWS 環境外に保有しておきたい

KMS キーのインポートプロセス

ステップ 1 : KMS キーを空のコンテナとして作成

- キーマテリアルオリジン “外部 (キーマテリアルのインポート) ” を選択

ステップ 2 : ラップキーとインポートトークンのダウンロード

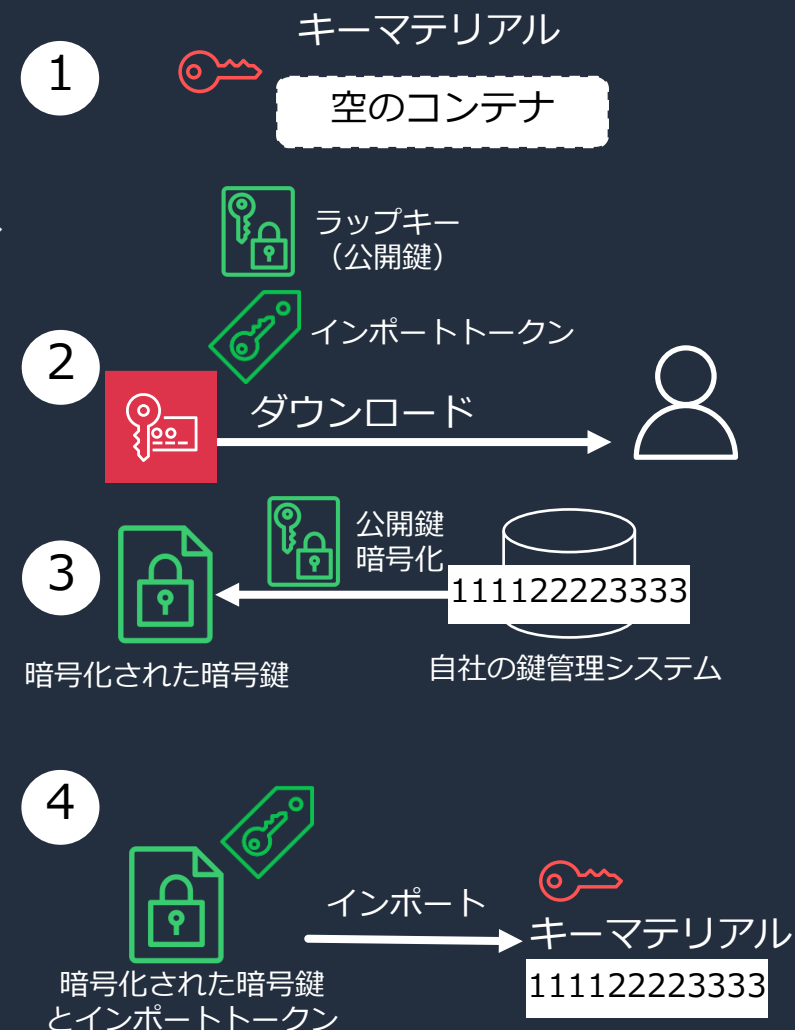
- インポートするキーを暗号化するラップキー (RSA 2048 bit)
- アップロード時に必要なインポートトークン
- キーマテリアルの暗号化に利用する暗号アルゴリズムの選択

ステップ 3 : 自社で管理している暗号鍵の暗号化

- 選択した暗号アルゴリズムとラップキーを利用した暗号化

ステップ 4 : キーマテリアルのインポート

- 必要に応じて、キーマテリアルの有効期限を設定



KMS キーのインポートに関する前提と制約

- KMS キーの耐久性が AWS KMS 生成のキーと異なる
 - AWS KMS で生成した場合と同様のアーキテクチャに保管
 - ただし、リージョン障害などの場合、自動復旧不可
- インポートした暗号鍵の表示・エクスポートは不可能
 - 耐久性も踏まえ、お客様の責任でコピーを保持しておくことが運用の前提
- キーマテリアルは即時削除が可能
 - 有効期限切れの場合も、即時削除される
 - これらの場合、空のコンテナとしての KMS キーは削除されない

KMS キーのインポートに関する前提と制約（続き）

- 同じ KMS キー ではキーマテリアル変更不可
 - 同じキーマテリアルの再インポートのみ可能
- キーマテリアルオリジンの変更不可
 - 後から AWS KMS 生成や、カスタムキーストアの利用はできない
- 鍵の手動ローテーションが必要
 - 手動ローテーションは、新規の KMS キーの作成
- （対象キーの場合） AWS KMS 外部で保管している同一のキーマテリアルで暗号化したデータでも、インポートした KMS キーによる復号は不可

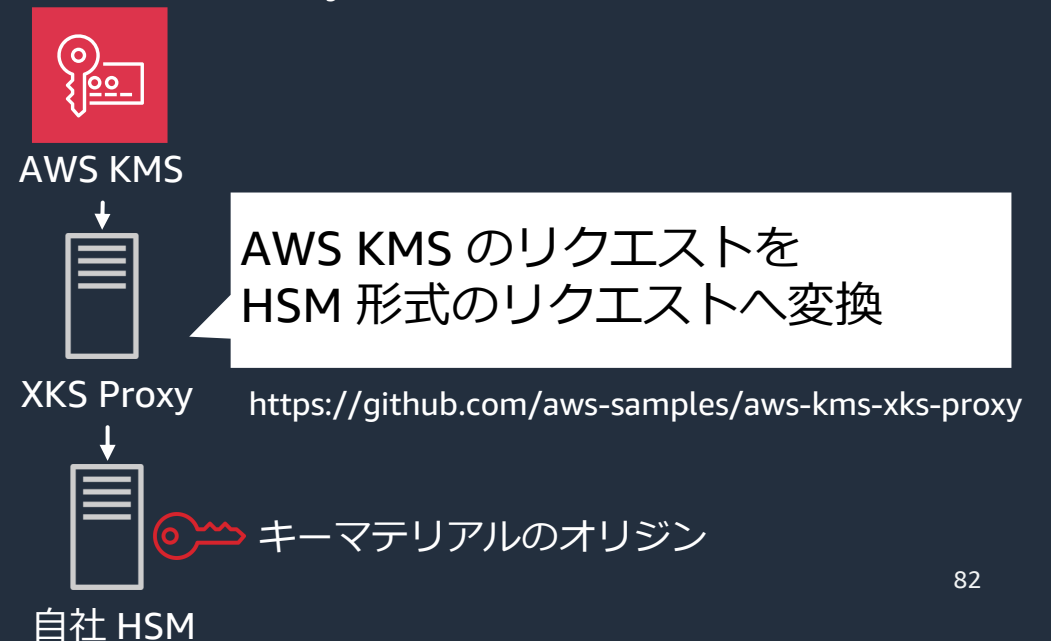
カスタムキーストア：キーマテリアル保管の外部化

- キーマテリアルの保存を AWS KMS から外部化
- コンプライアンス要件やお客様独自のポリシーなど、HSM の独自管理が必要な場合に利用

カスタムキーストア (AWS CloudHSM)



カスタムキーストア (外部キーストア)
AWS KMS External Key Store (XKS)



カスタムキーストアと AWS KMS オリジンの比較

	AWS KMS	AWS CloudHSM	XKS
暗号鍵の生成	AWS	AWS	お客様
ハードウェアの管理	AWS	AWS	お客様
ハードウェアの共有	マルチテナント	シングルテナント	お客様の設定次第
パフォーマンス、 可用性などの設定	AWS	お客様	お客様
ユースケース	デフォルト	シングルテナント、 暗号処理のオフ ロード	AWS からアクセス させない要件など、 お客様固有のニーズ

カスタムキーストアの前提と制約

- 設計に応じて、大きく責任の所在が変化
- カスタムキーストア側の仕様に応じて、レイテンシー、パフォーマンス、可用性に変化が出る可能性
- 対称暗号化キーのみ対応
- 鍵の手動ローテーションが必要
- マルチリージョンキーのサポート無し

参考 : AWS CloudHSM



AWS CloudHSM

- クラウドベースのハードウェアセキュリティモジュール (HSM)
- プラットフォームやアプリケーションの暗号処理をオフロード
- ハードウェア専有型シングルテナント
- 初期費用不要で時間単位の従量課金

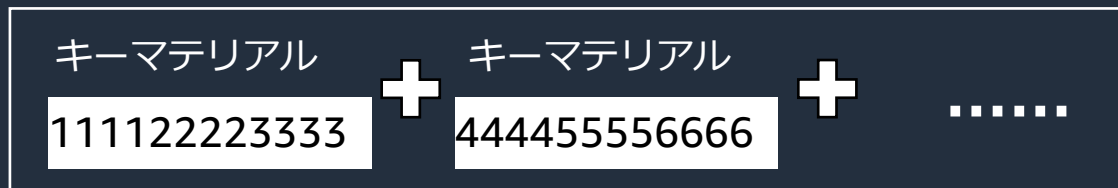
AWS Key Management Service の鍵管理 詳細

AWS KMS の鍵管理機能

- ライフサイクル
 - 自動ローテーション
 - 手動ローテーション
- アクセス制御
 - KMS キーの認可パターン詳解
 - グラント
 - クロスアカウント キーアクセス

(Part.1 から再掲) KMS キーローテーションの更新イメージ

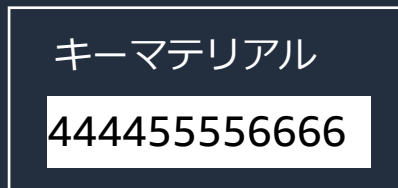
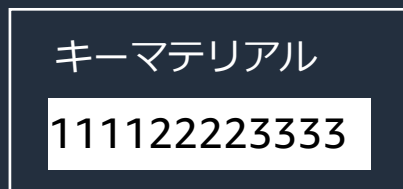
自動



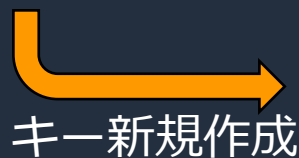
コンテナは不変
キー ID 等を継続利用可能

キーマテリアルが毎年自動更新 (追加)

手動



コンテナ自体を新規作成
キー ID も新規に割り当て
参照関係の見直しが必要

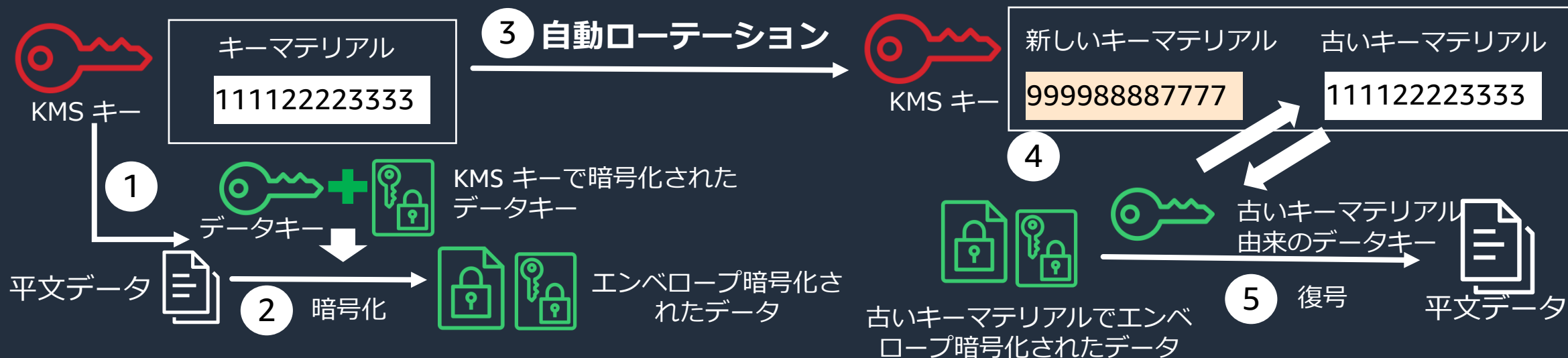


.....

(Part.1 から再掲) KMS キーのローテーション方式比較

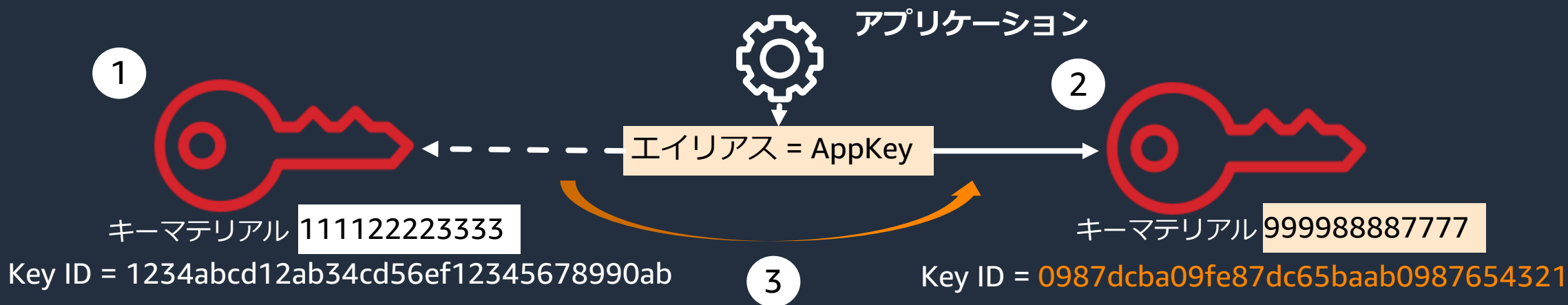
方式	説明	対象	特徴
自動	新しいキーマテリアルを生成 コンテナとしての AWS KMS キーは不変	AWS KMS で 作成した 対称暗号化キー	<ul style="list-style-type: none">定期的な自動更新ローテーション後は、新しいキーマテリアルでデータを暗号化過去のキーマテリアルを保管するため、過去暗号化されたデータの復号が容易
手動	新しい KMS キーを作成 サービスやアプリケーションの参照先を変更必要	上記以外のキー	<ul style="list-style-type: none">更新頻度はお客様判断過去のキーマテリアルの保管有無はお客様判断過去のキーマテリアルの有無に応じ、ローテーション時のデータの再暗号化が必要

自動ローテーションでは過去のデータの復号が容易



1. データキーを作成、KMS キーでデータキーを暗号化
2. データキーを用いてデータを暗号化、暗号化されたデータキーと一緒に保管
3. KMS キーの自動ローテーションにより、キーマテリアルが新たに追加
4. 自動ローテーション後に、KMS キーで暗号化されたデータキーの復号をリクエスト。KMS キー内に過去のキーマテリアルが保管されているため、暗号化されたデータキーの復号が可能
5. 暗号化されたデータを復号、次の暗号化では新しいキーマテリアルを利用

手動ローテーションにおける エイリアスを活用した参照関係の維持



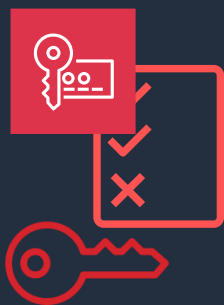
1. KMS キーを生成
アプリケーションの参照先をエイリアス “AppKey” に指定
エイリアスの参照先は、Key ID = 1234abcd12ab34cd56ef12345678990ab
2. 手動ローテーションして、新たな KMS キーを生成
3. UpdateAlias API を用いて、エイリアスの参照先を、
Key ID = 0987dcba09fe87dc65baab0987654321 に変更
アプリケーションは、コードの変更なく新しい KMS キーを利用することが可能

KMS キーの主要なアクセスコントロール

- AWS KMS における主要な 3 つのアクセスコントロール

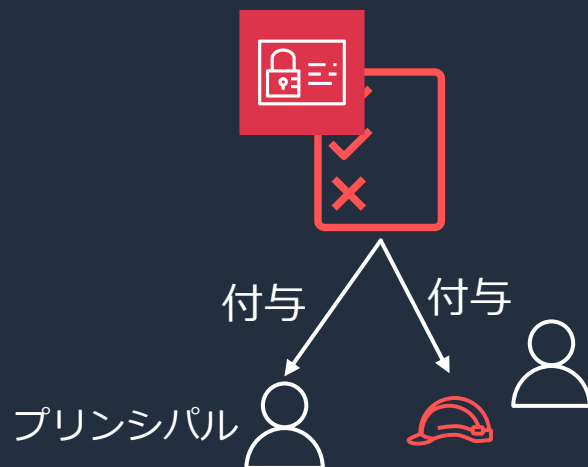
キーポリシー

キーと 1 : 1 に紐づく、
リソースベースの
パーミッション



IAM ポリシー

プリンシパルベース
のパーミッション



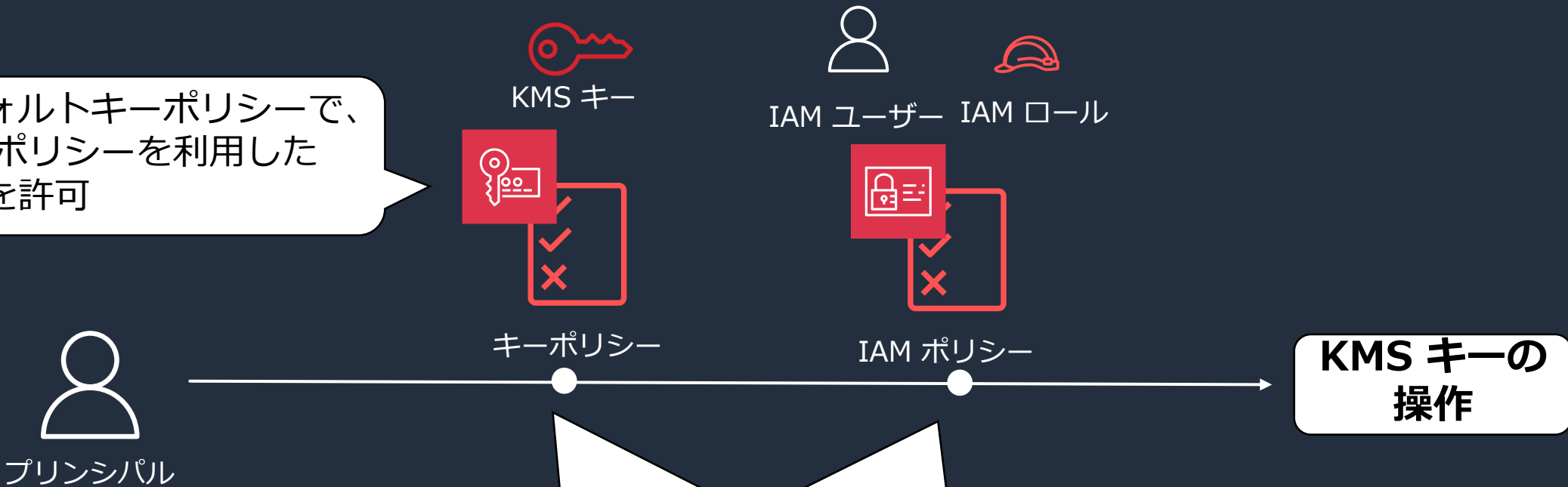
グラント

クライアントによる
他プリンシパルへの
パーミッション委任



(Part.1 から再掲) KMS キーのアクセスコントロールのイメージ

デフォルトキーポリシーで、IAM ポリシーを利用した認可を許可



KMS キー操作への認可有無を確認

アクセス許可があればアクセス許可
どちらかで明示的な拒否があればアクセス拒否

KMS キーの認可パターン详解

	キーポリシー		IAM ポリシー	認可の結果
	IAM ポリシーを利用した認可の許可	キーポリシーによる鍵操作の許可	鍵操作の許可	
パターン 1	あり	あり	あり	アクセス許可
パターン 2	あり	あり	なし	アクセス許可
パターン 3	あり	なし	あり	アクセス許可
パターン 4	あり	なし	なし	アクセス拒否
パターン 5	なし	あり	あり	アクセス許可
パターン 6	なし	あり	なし	アクセス許可
パターン 7	なし	なし	あり	アクセス拒否
パターン 8	なし	なし	なし	アクセス拒否

KMS キーの認可パターン详解

	キーポリシー		IAM 鍵	キーポリシーでの鍵操作の 許可があればアクセス許可
	IAM ポリシーを 利用した認可の許可	キーポリシーによる 鍵操作の許可		
パターン 1	あり	あり	あり	アクセス許可
パターン 2	あり	あり	なし	アクセス許可
パターン 3	あり	なし	あり	アクセス許可
パターン 4	あり	なし	なし	アクセス拒否
パターン 5	なし	あり	あり	アクセス許可
パターン 6	なし	あり	なし	アクセス許可
パターン 7	なし	なし	あり	アクセス拒否
パターン 8	なし	なし	なし	アクセス拒否

KMS キーの認可パターン详解

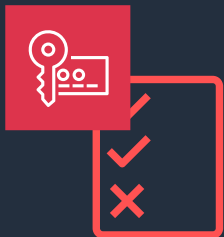
	キーポリシー		IAM ポリシー	認可の結果
	IAM ポリシーを利用した認可の許可	キーポリシーによる鍵操作の許可	鍵操作の許可	
パターン 1	あり	あり	あり	アクセス許可
パターン 2	あり	あり	なし	アクセス許可
パターン 3	あり	なし	あり	アクセス許可
パターン 4	あり	なし	なし	アクセス拒否
パターン 5	なし	あり	あり	アクセス許可
パターン 6	なし	なし	なし	アクセス拒否
パターン 7	なし	なし	あり	アクセス拒否
パターン 8	なし	なし	なし	アクセス拒否

IAM ポリシーを利用した認可制御には、キーポリシーでの “IAMポリシーを利用した認可の許可” が必要

パターン 6 アクセス許可：キーポリシーを利用した認可



パターン 6 アクセス許可：キーポリシーを利用した認可



キーポリシー

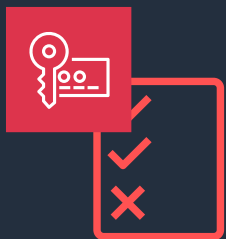
```
{
  "Sid": "Allow use of the key",
  "Effect": "Allow",
  "Principal": {"AWS": [
    "arn:aws:iam::111122223333:user/ExampleUser",
    "arn:aws:iam::111122223333:role/ExampleRole",
    "arn:aws:iam::444455556666:root"
  ]},
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

鍵の利用ユーザーに
暗号化、データキーの作成などの
データ暗号化に必要な操作を許可

パターン 3 アクセス許可 : IAM ポリシーを利用した認可



パターン 3 アクセス許可 : IAM ポリシーを利用した認可



キーポリシー

1. IAM ポリシーを利用した認可を可能にする
2. アカウントに権限を付与

```
{  
  "Sid": "Enable IAM policies",  
  "Effect": "Allow",  
  "Principal": {  
    "AWS": "arn:aws:iam::111122223333:root"  
  },  
  "Action": "kms:*",  
  "Resource": "*" }  
}
```



IAM ポリシー

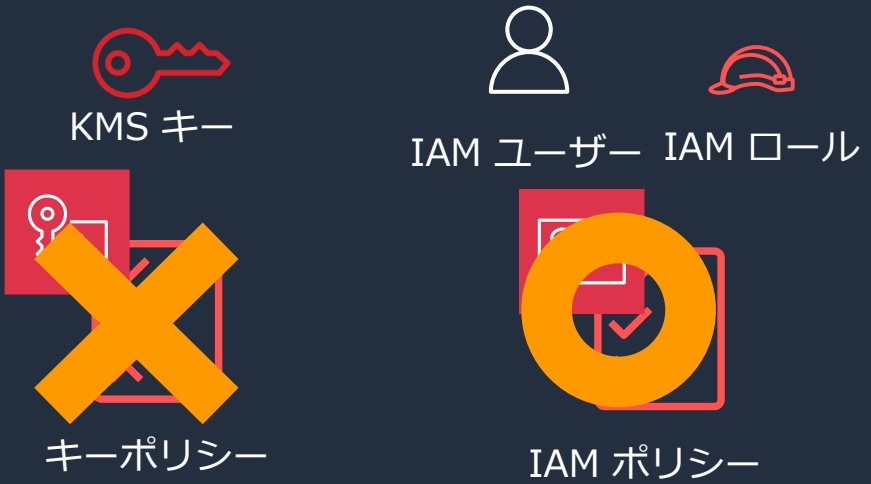
特定の KMS キーを利用した暗号操作を許可するポリシー

```
{  
  "Version": "2012-10-17",  
  "Statement": {  
    "Effect": "Allow",  
    "Action": [ "kms:DescribeKey", "kms:GenerateDataKey",  
      "kms:Decrypt" ],  
    "Resource":  
    [ "arn:aws:kms:*:111122223333:key/1234abcd-12ab-  
      34cd-56ef-1234567890a" ]  
  }  
}
```

パターン7 アクセス拒否： IAM ポリシーの利用がキーポリシーで未許可

デフォルトのキーポリシーで、IAM ポリシーを利用した認可を許可

プリンシパル



KMS キーへのアクセス

KMS キー操作への認可有無を確認

アクセス許可があればアクセス許可
どちらかで明示的な拒否があればアクセス拒否

パターン7 アクセス拒否： IAM ポリシーの利用がキーポリシーで未許可

IAM ポリシーを利用した認可を可能にするキーポリシーが付与されていないため、右記の IAM ポリシーは評価されず、アクセス拒否となる

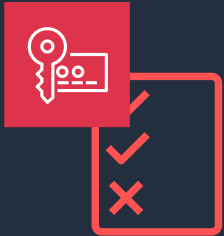


特定の KMS キーを利用した暗号操作を許可するポリシー

IAM ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [ "kms:DescribeKey", "kms:GenerateDataKey",
"kms:Decrypt" ],
    "Resource":
[ "arn:aws:kms:*:111122223333:key/1234abcd-12ab-
34cd-56ef-1234567890a" ]
  }
}
```

明示的な Deny はどのパターンでもアクセス拒否



キーポリシー

```
{
  "Sid": "Deny use of the key",
  "Effect": "Deny",
  "Principal": {"AWS": [
    "arn:aws:iam::111122223333:user/ExampleUser",
    "arn:aws:iam::111122223333:role/ExampleRole" ]},
  "Action": [ "kms:Encrypt", "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*", "kms:DescribeKey" ],
  "Resource": "*"
}
```



IAM ポリシー

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": [ "kms:DescribeKey", "kms:GenerateDataKey",
      "kms:Decrypt" ],
    "Resource":
      [ "arn:aws:kms:*:111122223333:key/1234abcd-12ab-
        34cd-56ef-1234567890a" ]
  }
}
```

KMS キーのアクセス許可をテストする : DryRun

- API リクエストの際、“DryRun” パラメータを指定することで、KMS キーのアクセス許可状況を確認可能

```
aws kms create-grant ¥  
  --key-id 1234abcd-12ab-34cd-56ef-1234567890ab ¥  
  --grantee-principal arn:aws:iam::111122223333:role/keyUserRole ¥  
  --operations Decrypt ¥  
  --dry-run
```


グラント

- 他の AWS プリンシパルに KMS キーの利用を委任
- 暗号化オペレーションなど、一部オペレーション (*) のみ委任可能
- 付与された権限は、明示的に削除しない限り永続的に有効
 - 暗号化コンテキストの利用によるグラントの制約の活用を推奨
- ユースケース：統合された AWS サービスによるデータの暗号化
 - Amazon EC2 がユーザーに代わってボリュームを暗号化

暗号化コンテキスト Encryption Context

- 対称暗号化 KMS キーで利用可能
- 暗号化の際に追加で指定できるキー / バリューのペア
- 暗号化の際に指定した値と復号の際に指定する値の一致が必要
- 追加認証データ (Additional Authenticated Data: AAD)
- AWS CloudTrail に平文で出力されるため、機微情報は非推奨



グラントアクセス許可コマンドの例

```
{
  "Grants": [
    {
      "Name": "",
      "IssuingAccount": "arn:aws:iam::111122223333:root",
      "GrantId": "abcde1237f76e4ba7987489ac329fbfa6ad343d6f7075dbd1ef191f0120514a",
      "Operations": [
        "Decrypt"
      ],
      "GranteePrincipal": "arn:aws:iam::111122223333:role/keyUserRole",
      "CreationDate": 1568565290.0,
      "Constraints": { "EncryptionContextEquals": { "Department": "IT" } },
      "KeyId":
        "arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
      "RetiringPrincipal": "arn:aws:iam::111122223333:role/adminRole"
    }
  ]
}
```

委任される権限

権限を委任する先のプリンシパル

暗号化コンテキスト

委任した権限を削除できるプリンシパル

権限を委任する KMS キー

キーポリシーにおけるグラントの許可

- 統合された AWS サービスで暗号化オペレーションを行うのに必要なパーミッションを委任（グラント）する際の条件の指定

```
{
  "Sid": "Allow attachment of persistent resources",
  "Effect": "Allow",
  "Principal": {"AWS": "arn:aws:iam::111122223333:user/ExampleUser"},
  "Action": [
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
  ],
  "Resource": "*",
  "Condition": {"Bool": {"kms:GrantIsForAWSResource": true}}
}
```

対象のプリンシパルが
KMS と統合された AWS サービスで
KMS キーを使用する許可

クロスアカウントのキーアクセス

- 異なる AWS アカウントのプリンシパルに KMS キーの利用を許可する場合、2 つのアクセス許可の付与が必要
 - KMS キーを保有するアカウントのキーポリシー
 - KMS キーを利用するアカウントの IAM ポリシー
- 予期しないアクセスを防ぐため、クロスアカウントアクセスの許可は最小限にとどめる
 - 付与する操作権限の限定
 - 利用させるプリンシパルの限定
 - 利用するリソース（KMS キー）の限定

クロスアカウントのキーアクセスのためのキーポリシー

```
{
  "Sid": "Allow an external account to use this KMS key",
  "Effect": "Allow",
  "Principal": {
    "AWS":
      "arn:aws:iam::444455556666:role/ExampleRole"
  },
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "*"
}
```

異なる AWS アカウントの IAM ロールを指定

- プリンシパルには、アカウントユーザーロールの指定が可能。
- プリンシパルへの、ワイルドカード "*" 設定は非推奨

参考：AWS コンソールから別の AWS アカウントの指定



- アカウントレベルの認可には、AWS コンソールからの GUI 操作が利用可能

クロスアカウントのキーアクセスのための IAM ポリシー

```
{
  "Sid": "AllowUseOfKeyInAccount111122223333",
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey"
  ],
  "Resource": "arn:aws:kms:us-west-
2:111122223333:key/1234abcd-12ab-34cd-56ef-
1234567890ab"
}
```

利用したい KMS キーを保有
する AWS アカウントの
KMS キーを指定

- リソースには複数のキーの指定が可能だが、必要最小限にとどめることを推奨
- キーポリシーで許可されていない操作の認可はできない

AWS Key Management Service

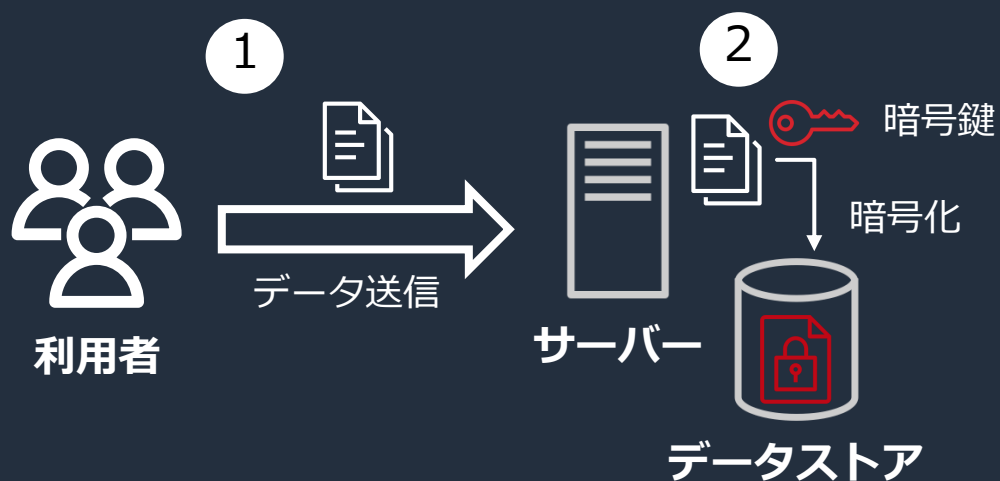
様々な暗号化ユースケース

様々な暗号化のユースケース

- データ暗号化
 - サーバーサイド暗号化
 - クライアントサイド暗号化
- 署名
- メッセージ認証コード

データ暗号化の方式

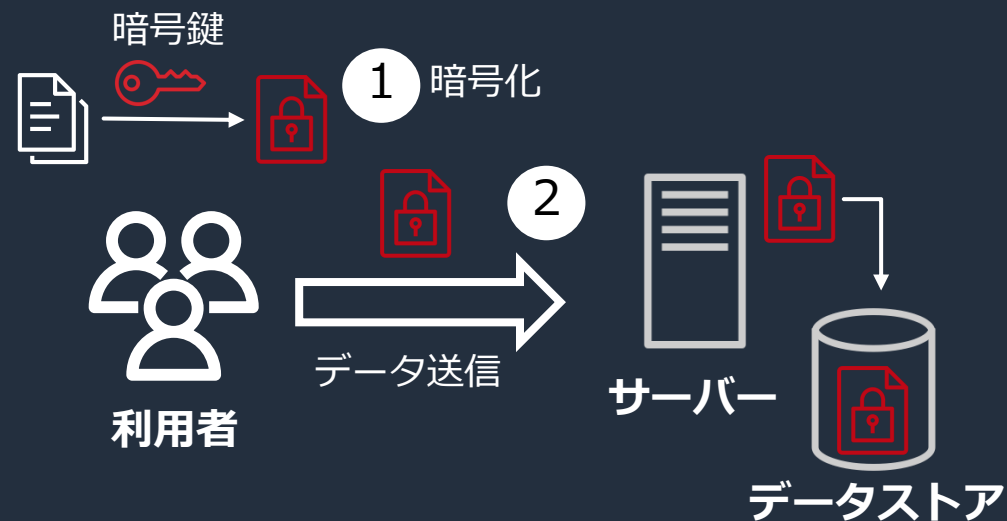
・サーバーサイド暗号化



1. クライアントからサーバーへデータ送信
2. サーバ側で暗号鍵を利用した暗号化を実施
暗号化されたデータをデータストアに保管

統合された AWS サービスによる
エンベロープ暗号化

・クライアントサイド暗号化



1. クライアントにて、
暗号鍵を利用した暗号化を実施
2. クライアントからサーバーへデータ送信

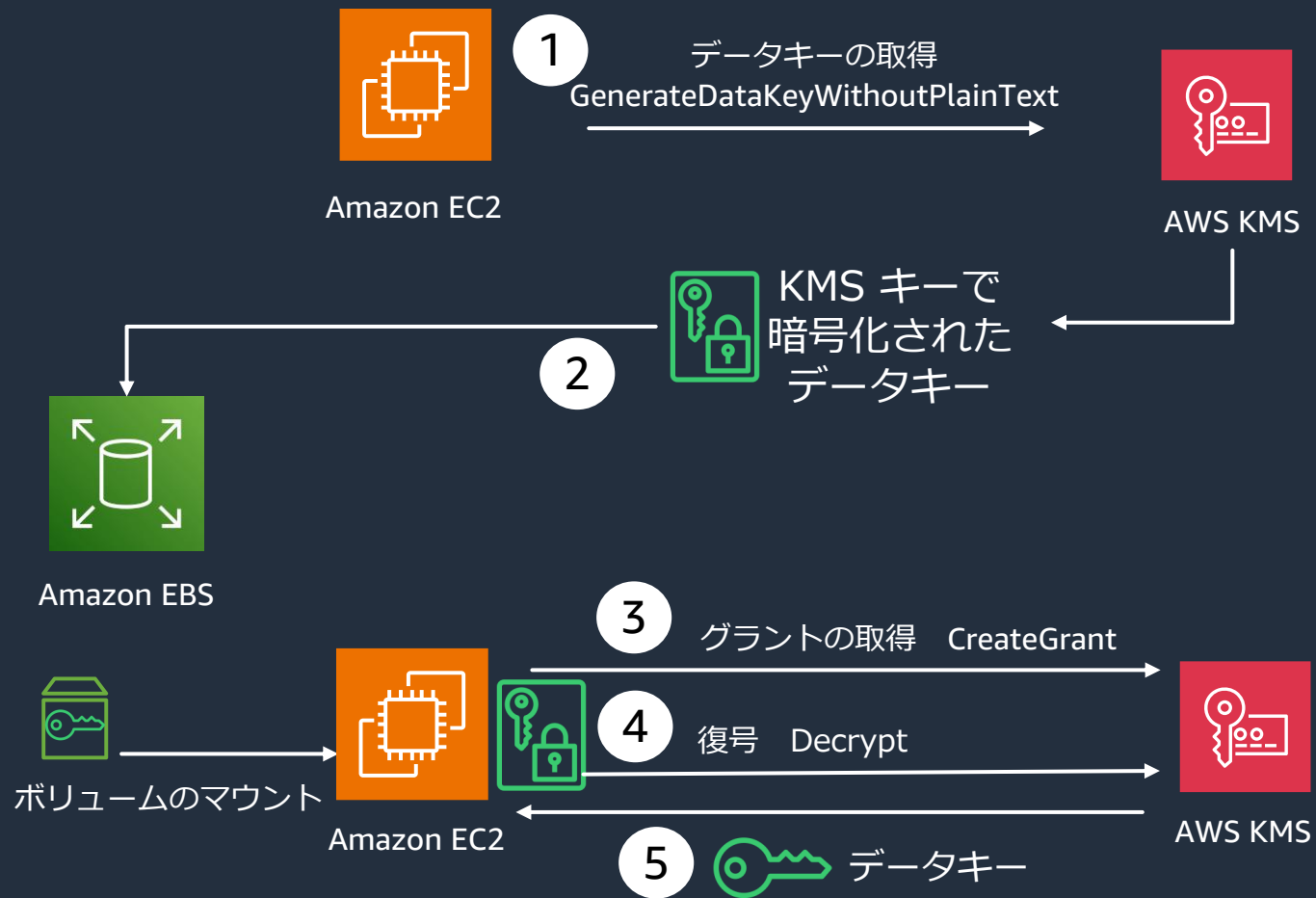
お客様アプリや SDK を使った暗号化

AWS KMS でのサーバーサイド暗号化

- データキーの作成やエンベロープ暗号化などの処理は、AWS サービス側で実行

API	概要
Encrypt	KMS キーを用いた暗号化
Decrypt	KMS キーを用いた復号
GenerateDataKey	平文のデータキーと暗号化されたデータキーの生成
GenerateDataKeyWithoutPlaintext	暗号化されたデータキーの生成
CreateGrant	ユーザーやロールに KMS キー使用を許可するグラントを作成

Amazon EBS (暗号化されたスナップショット) 暗号化フロー

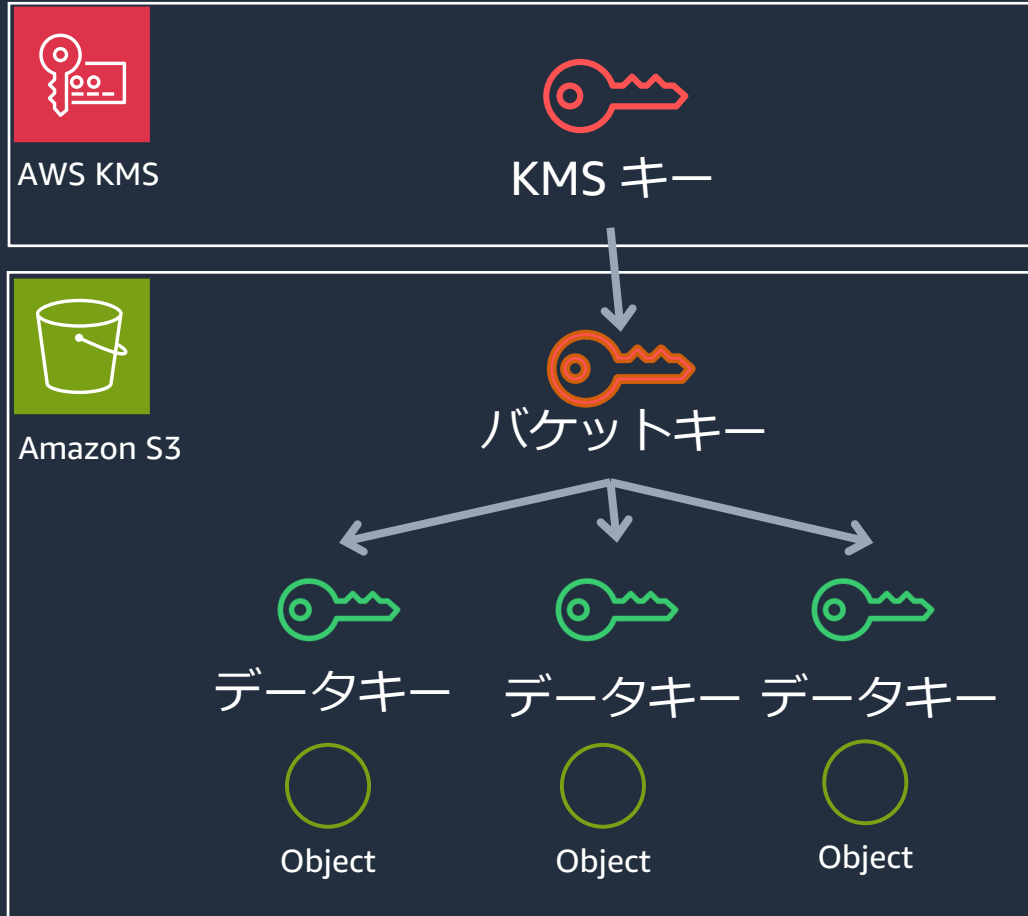


1. KMS キーを指定して、データキーを取得
2. AWS KMS は、スナップショットの暗号化に利用されたデータキーと同じデータキーを、暗号化したうえで返却
暗号化されたデータキーは、ボリュームのメタデータに保存
3. ボリュームがインスタンスにマウントされると、Amazon EC2 は、AWS KMS にデータキーの復号操作をするためグラントを要求
4. 暗号化されたデータキーの復号
5. 復号されたデータキーを用いて、ボリュームの I/O 処理を行う
データキーはメモリに保持

統合された AWS サービスによるデータキーの利用

- リソースのデータ保護のため、統合された各 AWS サービスでは、データキーの作成が行われる
- データキーの利用数は保護するリソースごとに異なる
 - 1 データキー / EBS ボリューム
 - 1 データキー / RDS インスタンス
 - 1 データキー / Amazon S3 に保管されているオブジェクト
- Amazon S3 では、大量のデータキー作成が行われる可能性

Amazon S3 バケットキー によるコストの削減

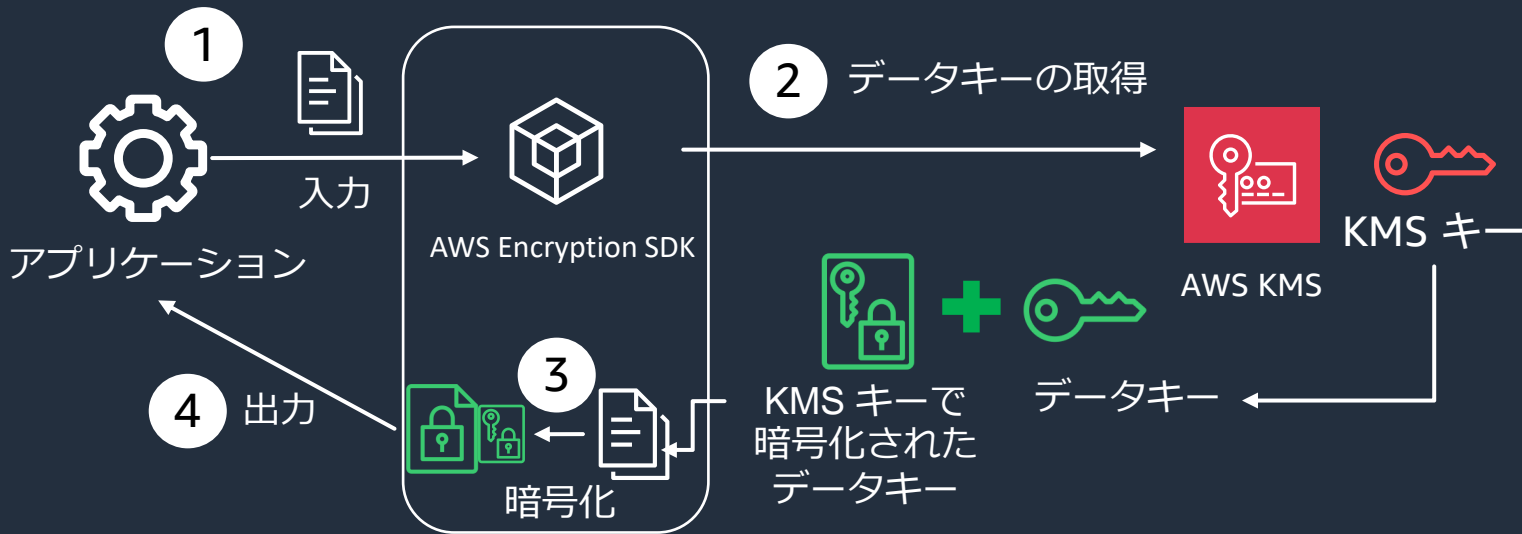


- バケットキーは、“中間”のキー
- Amazon S3 が AWS KMS にバケットキーの作成をリクエスト
- Amazon S3 はバケットキーから、オブジェクトを暗号化するためのデータキーを導出して利用
- AWS KMS へのリクエスト回数を削減し、最大 99% のコスト削減

AWS KMS でのクライアントサイド暗号化

- データをローカルで暗号化したうえでサーバに送付する方式
- AWS Encryption SDK, Amazon S3 Encryption Client, AWS Database Encryption SDK などの AWS が提供する暗号化ライブラリを通じた実装が可能
- データキーを暗号化するラッピングキーとして、KMS キー を指定すれば、残りの暗号化処理はライブラリ側で実施
- 様々なプログラミング言語をサポート
 - AWS Encryption SDK : C, .NET, Java, JavaScript, Python, CLI
 - Amazon S3 Encryption Client : Java, Go, C++, .NET, Ruby, PHP
 - AWS Database Encryption SDK : Java, Python

AWS Encryption SDK による暗号化フロー



1. 使用する KMS キー、暗号化したい平文データ、暗号化コンテキストを指定して SDK に入力
2. SDK が AWS KMS にデータキー取得をリクエスト
3. SDK が暗号化を実行
暗号化されたデータキーも保存
4. 暗号化されたデータの出力

AWS Encryption SDK CLI Bash での暗号化

- 言語実装は全て相互運用可能
- 各プログラミング言語による実装の詳細はドキュメント参照

¥¥ この例を実際のコマンドラインで実行する場合、暗号鍵 ARN などの置き換えが必要

```
$ keyArn=arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab
```

```
$ aws-encryption-cli --encrypt ¥  
  --input hello.txt ¥  
  --wrapping-keys key=$keyArn ¥  
  --metadata-output ~/metadata ¥  
  --encryption-context purpose=test ¥  
  --commitment-policy require-encrypt-require-decrypt ¥  
  --output .
```

```
$ ls  
hello.txt hello.txt.encrypted
```

hello.txt ファイルの暗号化

利用する KMS キーの指定

暗号化されたデータが出力

クライアントサイド暗号化におけるキャッシュの利用

- データキーを暗号化操作の都度リクエストする方式では、パフォーマンスとリクエスト数に伴うコストが課題
- データキーをクライアントにキャッシュすることで、リクエストを減らし、クライアントサイドでの高速な暗号化が可能



SDK と AWS KMS の連携
データの暗号化や復号
平文データキーのキャッシュ管理

キャッシュの設定可能な閾値

- 使用期限 (時間単位)
- メッセージの最大数
- バイトの最大数

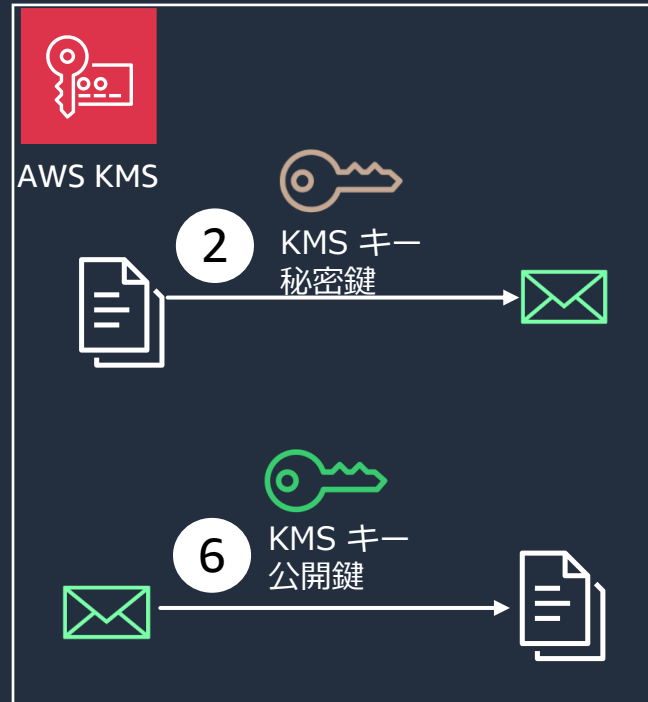
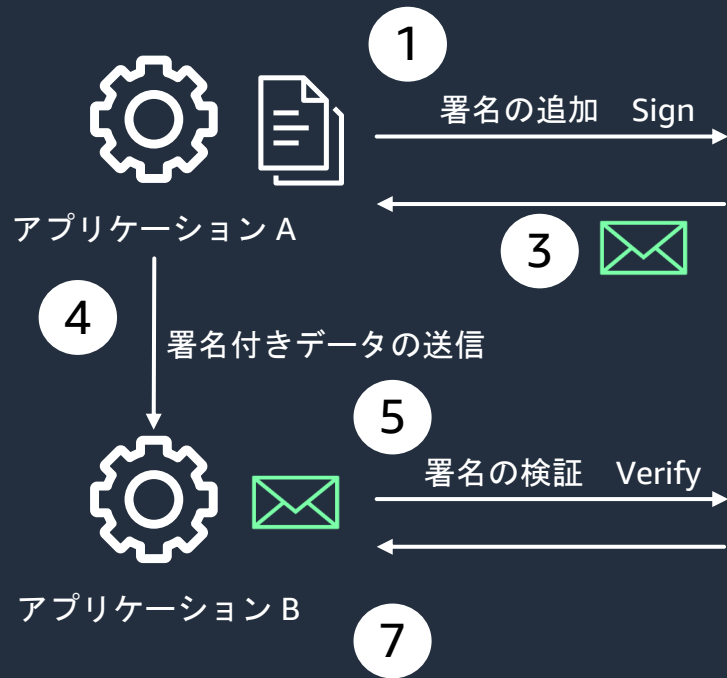
非対称キーと HMAC キーの暗号ユースケース

- 非対称キー : 送信するデータに対する署名の作成と検証
- HMAC キー : JWT * へのメッセージ認証コードの追加と署名

* JWT (JSON Web Token) : JSON 形式を利用した、
認証情報などのデータを安全に通信するための規格

API	概要
Sign	非対称 KMS キーの秘密鍵を用いた暗号化を行うことで、署名を作成する
Verify	非対称 KMS キーの公開鍵を用いた復号を行うことで、署名の検証を行う
GenerateMac	HMAC KMS キーを用いた暗号化を行うことで、メッセージ認証コードを作成する
VerifyMac	HMAC KMS キーを用いた復号を行うことで、メッセージ認証コードの検証を行う

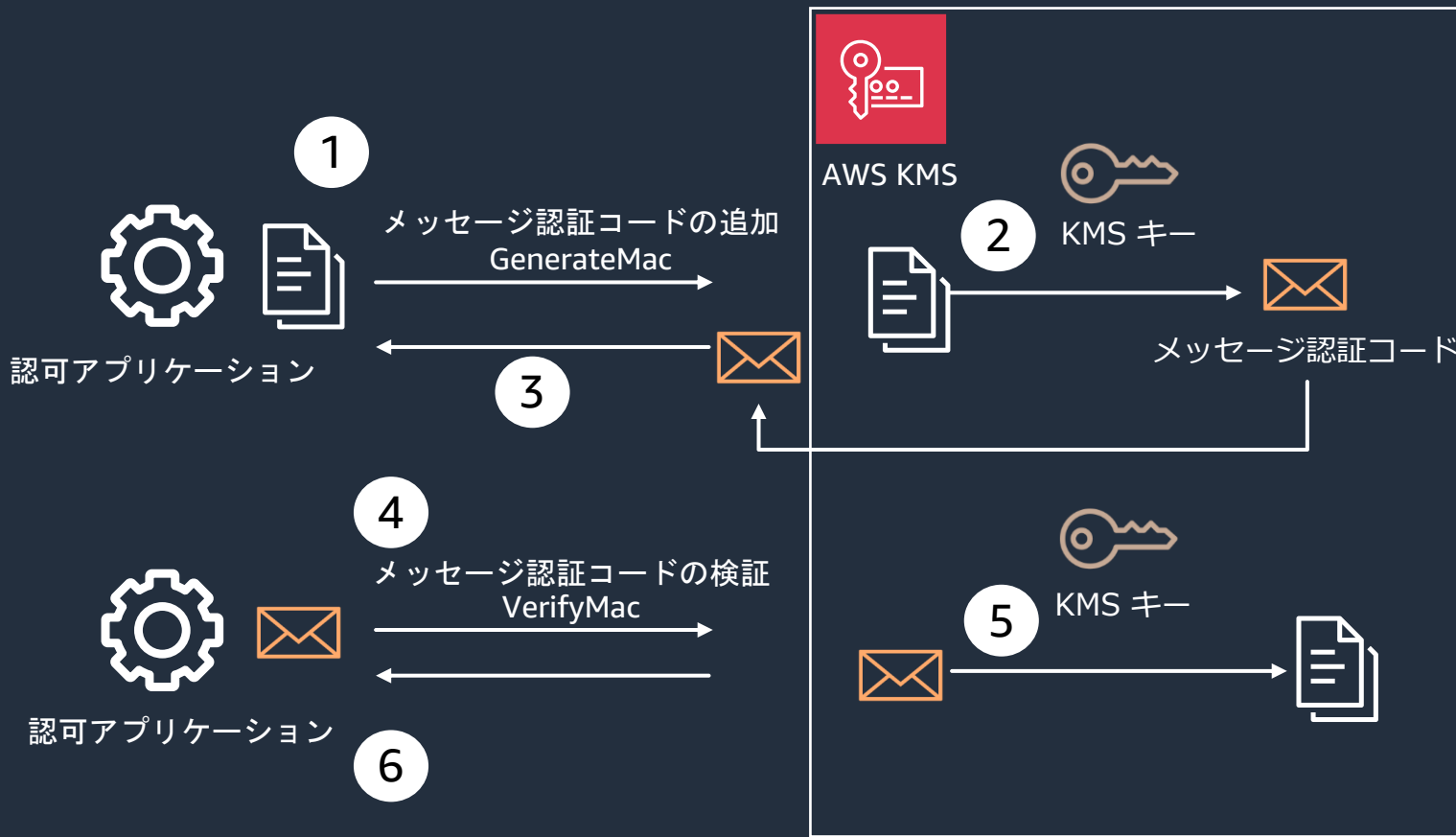
非対称 KMS キーを利用した署名の作成と検証



1. アプリケーション A がメッセージダイジェストを入力として AWS KMS に署名をリクエスト
2. AWS KMS は、指定された非対称 KMS キーの秘密鍵で署名を作成
3. AWS KMS は、署名を返却
4. アプリケーション A が元の平文データと署名をあわせてアプリケーション B に送付
5. アプリケーション B が署名の検証を AWS KMS にリクエスト
6. AWS KMS は、指定された非対称 KMS キーの公開鍵で署名を検証
7. 署名の検証結果が返却される

※ 5 以降の署名の検証は、AWS KMS の非対称 KMS キーの公開鍵をアプリケーション B にダウンロードすることで、アプリケーション B のローカルで実施する方式も可能

HMAC KMS キーを利用した JWT の作成



1. 認可アプリケーションが JSON Web トークンの claim 部と header 部を結合し Base64URL エンコードした値を入力とし MAC の追加を AWS KMS にリクエスト
2. AWS KMS は、指定された HMAC KMS キーで MAC を作成
3. AWS KMS は、MAC を返却
4. 認可アプリケーションが受信した JWT の検証を AWS KMS にリクエスト
5. AWS KMS は、指定された HMAC KMS キーで MAC を検証
6. MAC の検証結果が返却される

Part.2 まとめ



まとめ

- AWS KMS は、データ保護を実現するための基盤で、かつ、暗号鍵の作成、保管、管理を行うための KMI を運用する AWS マネージドサービス
- KMS キーについて、コンテキストに応じたキー種別が設定可能でサーバーサイドやクライアントサイドのデータ暗号化、署名検証、メッセージ認証コードなど、様々な暗号化のユースケースを実現可能
- 暗号鍵のライフサイクル管理・アクセス制御・モニタリングを一元管理と細やかな制御を両立
- 暗号鍵の安全性について、第三者認証を取得済

参考：AWS KMS の料金

- **1 USD** / カスタマーマネージド KMS キー / 月
 - 自動ローテーションを有効にしている場合、バージョンにつき同額課金
 - 削除の待機期間中の鍵は計算対象外
- 対称キー 0.03 USD / 10,000 リクエスト
- 非対称キー 鍵種別に応じた課金* / 10,000 リクエスト
 - カスタマーマネージドキー、AWS マネージドキーが計算対象
 - 月間 20,000 リクエストまでは無償

* RSA 2048 keys: 0.03 USD , ECC GenerateDataKeyPair: 0.10 USD,

* Asymmetric except RSA 2048: 0.15 USD, 0.15, RSA GenerateDataKeyPair: 12.00 USD

AWS Black Belt Online Seminar とは

- 「サービス別」「ソリューション別」「業種別」などのテーマに分け、アマゾン ウェブ サービス ジャパン 合同会社が提供するオンラインセミナーシリーズです
- AWS の技術担当者が、AWS の各サービスやソリューションについてテーマごとに動画を公開します
- 以下の URL より、過去のセミナー含めた資料などをダウンロードすることができます
 - <https://aws.amazon.com/jp/aws-jp-introduction/aws-jp-webinar-service-cut/>
 - <https://www.youtube.com/playlist?list=PLzWGOASvSx6FlwIC2X1nObr1KcMCBBBlqY>



ご感想は X (Twitter) へ！ハッシュタグは以下をご利用ください
#awsblackbelt

内容についての注意点

- 本資料では資料作成時点のサービス内容および価格についてご説明しています。AWS のサービスは常にアップデートを続けているため、最新の情報は AWS 公式ウェブサイト (<https://aws.amazon.com/>) にてご確認ください
- 資料作成には十分注意しておりますが、資料内の価格と AWS 公式ウェブサイト記載の価格に相違があった場合、AWS 公式ウェブサイトの価格を優先とさせていただきます
- 価格は税抜表記となっております。日本居住者のお客様には別途消費税をご請求させていただきます
- 技術的な内容に関しましては、有料の [AWS サポート窓口](#)へお問い合わせください
- 料金面でのお問い合わせに関しましては、[カスタマーサポート窓口](#)へお問い合わせください (マネジメントコンソールへのログインが必要です)

Appendix



暗号技術概観

- 対称暗号（共通鍵暗号）
- 公開鍵暗号
- ハイブリッド暗号（公開鍵を利用した共通鍵の交換）
- ハッシュ
- メッセージ認証コード
- デジタル署名
- デジタル証明書

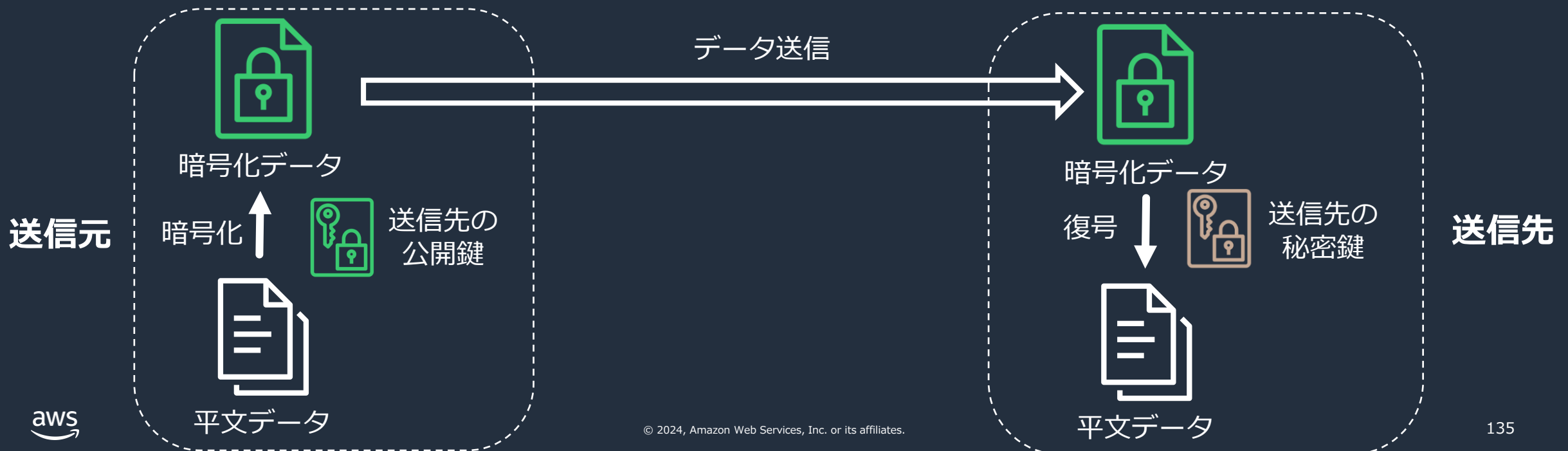
対称暗号（共通鍵暗号）

- データの暗号化と復号に**共通の鍵**を用いる暗号方式
- 公開鍵暗号方式に比べて**処理が高速**
- **安全な方法で送信元と送信先が共通の鍵を交換する必要がある**
 - 所有する鍵の数が増えるほど、鍵の保管負荷が高まる
 - 通信相手ごとに個別の鍵を用意する場合、相手の数だけ鍵の数が増える



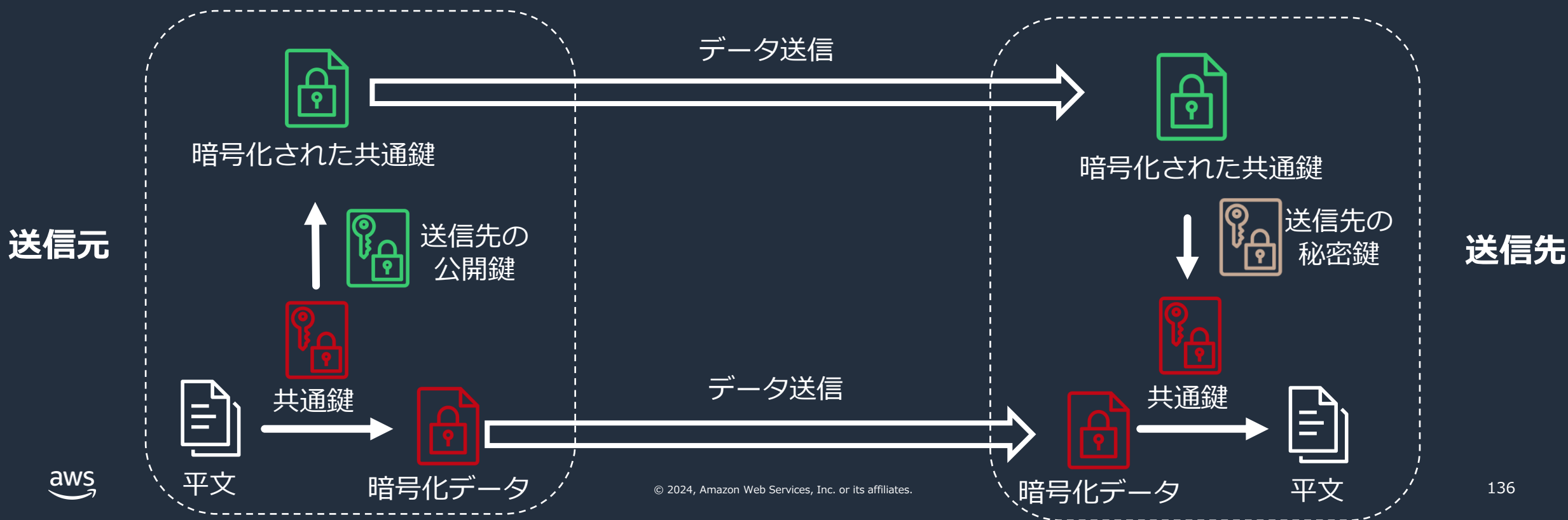
公開鍵暗号

- データの暗号化と復号に**公開鍵と秘密鍵**というペアの異なる鍵を用いる暗号方式
- 公開鍵は送信元の暗号鍵として公開**（ = 安全な鍵交換が可能）
- 共通鍵暗号と比べて、処理が複雑で遅いため、使い分けが必要



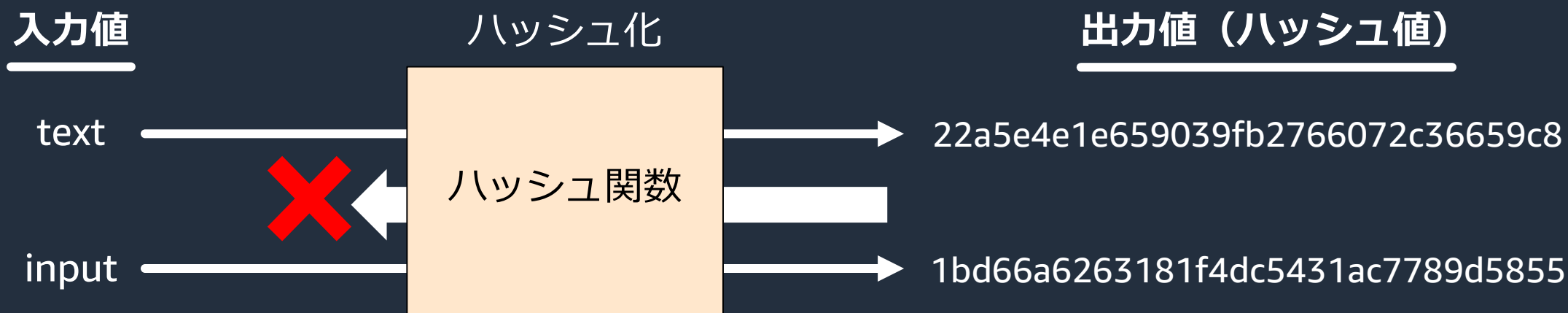
ハイブリッド暗号（公開鍵を利用した共通鍵の交換）

- 共通鍵暗号方式と公開鍵暗号方式を組み合わせた暗号方式
- 2つの暗号方式の利点を同時に利用
 - 高速処理 + 安全な鍵交換



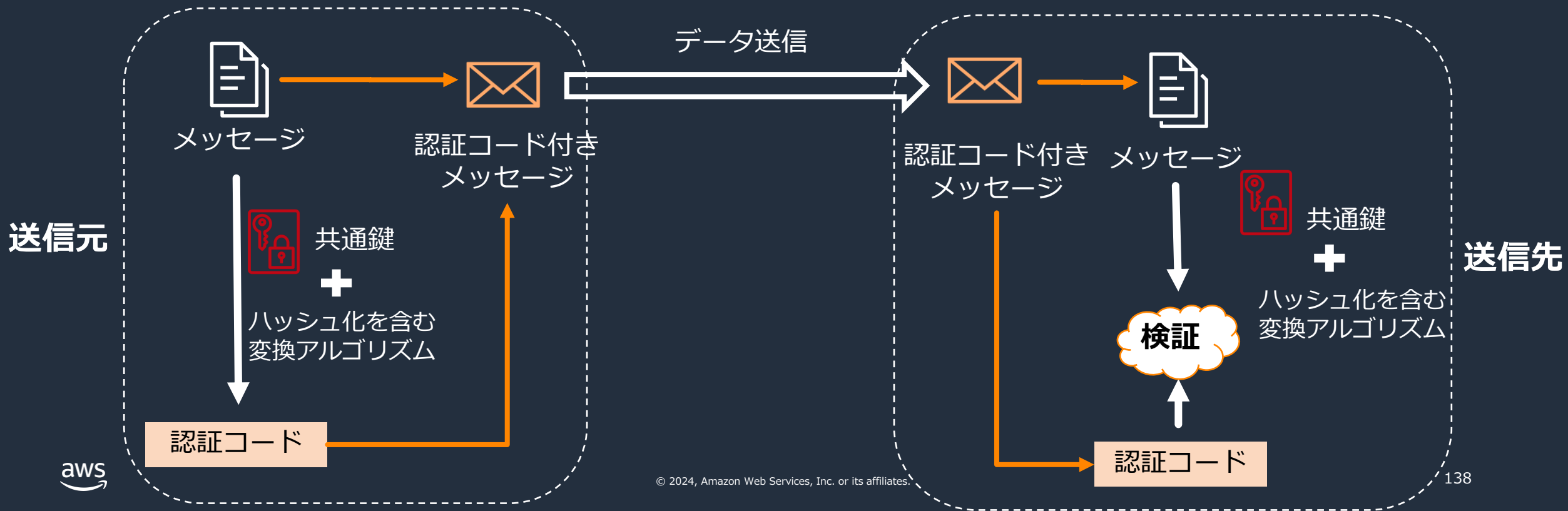
ハッシュ

- ハッシュ関数を利用して、任意のデータから、別の値を得る操作
- 計算された出力値（ハッシュ値）の性質
 - ハッシュ値から入力値を計算することが不可能
 - 同じハッシュ値となる、異なる入力値の計算が不可能



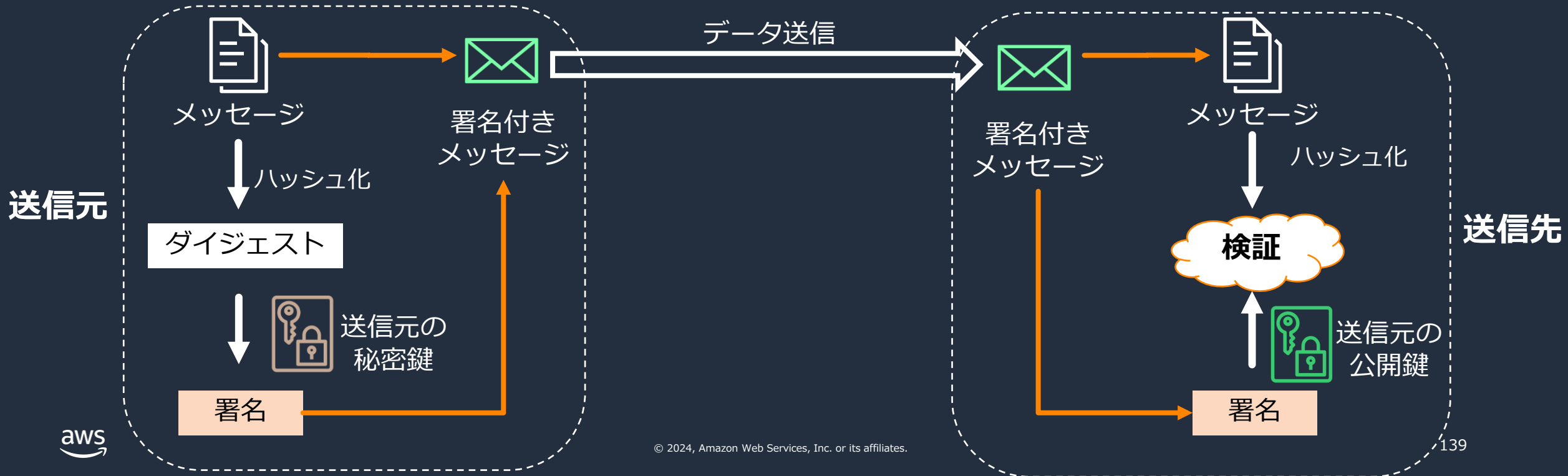
メッセージ認証コード

- 共通鍵暗号方式を利用した、送信元の真正性とメッセージの改ざん有無（完全性）を確かめる技術
- Hash-based Message Authentication Code (HMAC)



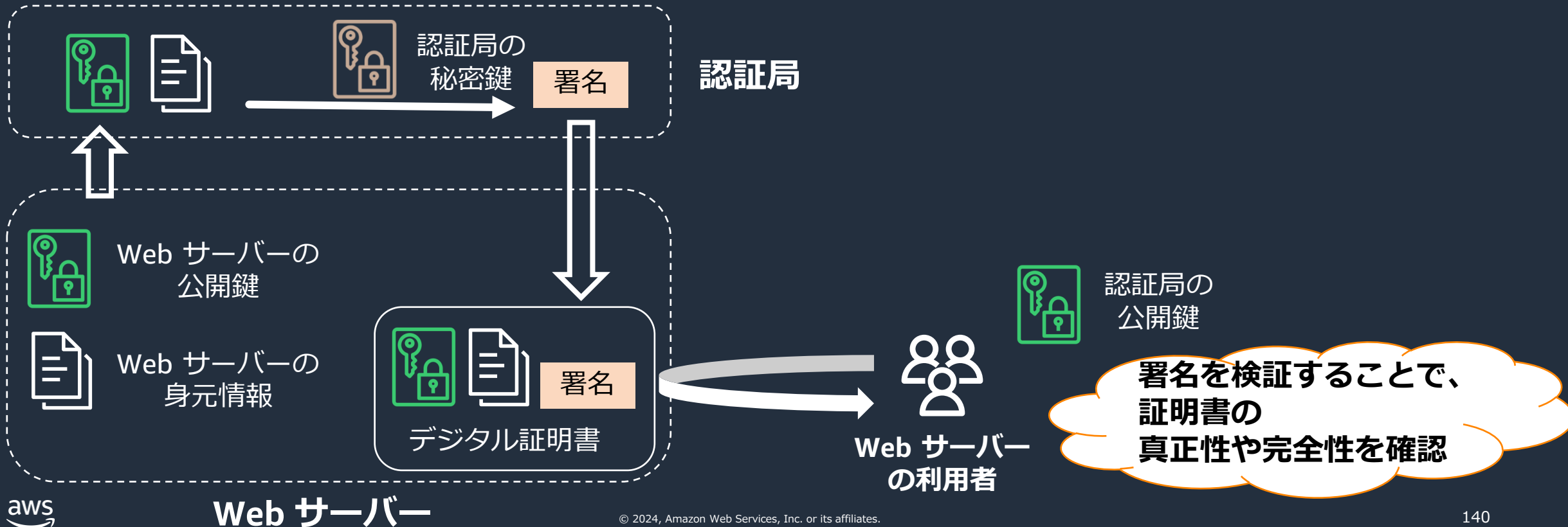
デジタル署名

- 公開鍵暗号方式を利用した、送信元の真正性、メッセージの改ざん有無（完全性）を確かめる技術
- 加えて、メッセージ送信の否認防止（非否認性）が可能
- 現実世界の署名や押印をデジタル化したもの



デジタル証明書

- 認証局（CA : Certificate Authority）が、**公開鍵の所有者の真正性や完全性を証明**
- 対象の公開鍵自体も証明書内に含まれる（公開鍵の配布）





Thank you!