Client de chiffrement Amazon DynamoDB

Guide du développeur

Client de chiffrement Amazon DynamoDB: Guide du développeur

Table of Contents

Qu'est-ce que le client de chiffrement Amazon DynamoDB ?

Amazon DynamoDB Encryption Client est une bibliothèque logicielle qui vous permet d'inclure le chiffrement côté client dans votreAmazon DynamoDBconception. Le client de chiffrement DynamoDB est conçu pour être mis en œuvre dans de nouvelles bases de données non renseignées. Le chiffrement de vos données sensibles en transit et au repos contribue à garantir que vos données en texte brut ne sont pas accessibles à un tiers, y comprisAWS. DynamoDB Encryption Client est fourni gratuitement dans le cadre de la licence Apache 2.0.

> **Important**
>
> Le client de chiffrement DynamoDB ne prend pas en charge le chiffrement des données de table DynamoDB non chiffrées existantes.

Ce manuel des développeurs présente les concepts relatifs au client de chiffrement DynamoDB, y comprisprésentation de son architecture (p. 5), des détails surcomment il protège les données des tables DynamoDB (p. 3)et en quoi elle diffère deChiffrement côté serveur DynamoDB (p. 7), des conseils survsélection des composants critiques pour votre application (p. 13), et des exemples dans chaqueLangage de programmation (p. 35)pour vous aider à démarrer.

DynamoDB Encryption Client présente les avantages suivants :

Spécifiquement conçu pour les applications DynamoDB

Vous n'avez pas besoin d'être un expert en chiffrement pour utiliser DynamoDB Encryption Client. Les implémentations incluent les méthodes d'annotation conçues pour travailler avec vos applications DynamoDB existantes.

Après que vous avez créé et configuré les composants requis, DynamoDB Encryption Client chiffre et signe de façon transparente vos éléments de table lorsque vous les ajoutez à une table, et les vérifie et les déchiffre lorsque vous les extrayez.

DynamoDB Encryption Client prend en chargeTypes de données DynamoDBet la plupart des fonctionnalités Amazon DynamoDB, y comprisTables globales. Toutefois, vous devrez peut-être apporter quelques modifications à la configuration si vous utilisez une ancienne version des tables globales. Pour plus d'informations, consultez Problèmes avec les tables globales des versions antérieures (p. 59).

Inclut le chiffrement et la signature sécurisées

DynamoDB Encryption Client inclut les implémentations sécurisées qui chiffrent les valeurs d'attribut de chaque élément de table à l'aide d'une clé de chiffrement unique, puis signent l'élément pour le protéger contre les modifications non autorisées, comme l'ajout ou la suppression d'attributs, ou la permutation de valeurs chiffrées.

Utilise les matériaux de chiffrement de n'importe quelle source

Vous pouvez utiliser DynamoDB Encryption Client avec les clés de chiffrement de n'importe quelle source, y compris votre implémentation personnalisée ou un service de chiffrement, tel queAWS Key Management Service(AWS KMS) ouAWS CloudHSM. Le client de chiffrement DynamoDB ne nécessite pasCompte AWSou n'importe quelAWSservice.

Les implémentations des langages de programmation pris en charge sont interopérables

Les bibliothèques client de chiffrement DynamoDB sont développées dans des projets open source sur GitHub. Elles sont actuellement disponibles en Java et en Python. Toutes les implémentations des

langages de programmation pris en charge du client de chiffrement DynamoDB sont interopérables. Par exemple, vous pouvez chiffrer les données avec le client Java et les déchiffrer avec le client Python.

Cependant, DynamoDB Encryption Client n'est pas compatible avecAWS Encryption SDKou leClient de chiffrement Amazon S3. Vous ne pouvez pas chiffrer avec une bibliothèque côté client et déchiffrer avec une autre.

Développé dans des référentiels open source

Le client de chiffrement Amazon DynamoDB est développé dans des référentiels open source sur GitHub. Vous pouvez utiliser ces référentiels pour afficher le code, lire et soumettre des problèmes, et trouver des informations spécifiques à l'implémentation de votre langage.

- Client de chiffrement DynamoDB pour Java —aws-dynamodb-encryption-java
- Client de chiffrement DynamoDB pour Python —aws-dynamodb-encryption-python

Support et maintenance

Le client de chiffrement DynamoDB utilise le mêmepolitique de maintenanceque leAWSUtilisation du SDK et des outils, y compris ses phases de gestion des versions et de cycle de vie. En tant que bonne pratique, nous vous recommandons d'utiliser la dernière version disponible du client de chiffrement DynamoDB pour votre langage de programmation, et de la mettre à niveau à mesure que de nouvelles versions sont publiées.

Chaque implémentation de langage de programmation du client de chiffrement DynamoDB est développée dans un environnement open source distinct GitHub repository. Le cycle de vie et la phase de support de chaque version sont susceptibles de varier selon les référentiels. Par exemple, une version donnée de DynamoDB Encryption Client peut être en phase de disponibilité générale (prise en charge complète) dans un langage de programmation, mais le end-of-supportphase dans un langage de programmation différent. Nous vous recommandons d'utiliser une version entièrement prise en charge dans la mesure du possible et d'éviter les versions qui ne le sont plus.

Pour connaître la phase de cycle de vie des versions de DynamoDB Encryption Client pour votre langage de programmation, consultez le`SUPPORT_POLICY.rst`dans chaque référentiel DynamoDB Encryption Client.

- Client de chiffrement DynamoDB pour Java —Support_Policy.RST
- Client de chiffrement DynamoDB pour Python —Support_Policy.RST

Pour plus d'informations, consultez le .AWSPolitique de maintenance des outilsdans leAWSRéférence des outils

Envoyer un commentaire

Nous apprécions vos commentaires. Si vous avez une question ou un commentaire, ou un problème à signaler, veuillez utiliser les ressources suivantes.

- Si vous découvrez une vulnérabilité de sécurité potentielle dans DynamoDB Encryption Client,notifierAWSsécurité. Ne pas créer de public GitHub Problème.
- Pour communiquer des commentaires sur DynamoDB Encryption Client, soumettez un problème dans leaws-dynamodb-encryption-javaouaws-dynamodb-encryption-python GitHub repository.

- Pour émettre des commentaires sur cette documentation, utilisez le lien des commentaires sur n'importe quelle page. Vous pouvez également soumettre un ticket ou contribuer àaws-dynamodb-encryption-docs, le référentiel open source pour cette documentation sur GitHub.

Quels sont les champs chiffrés et signés ?

Dans DynamoDB, untableest une collection d'éléments. Chaque élément est une collection d'attributs. Chaque attribut a un nom et une valeur.

Le client de chiffrement DynamoDB chiffre les valeurs des attributs. Puis, il calcule une signature sur les attributs. Vous pouvez spécifier les valeurs d'attribut à chiffrer et celles à inclure dans la signature. Toutefois, le client de chiffrement DynamoDB est conçu pour être implémenté dans de nouvelles bases de données non remplies. Vous devez ajouter les fonctions de chiffrement à vos applications DynamoDB avant d'envoyer des données à DynamoDB.

Le chiffrement protège la confidentialité de la valeur d'attribut. La signature assure l'intégrité de tous les attributs signés et de leurs relations entre eux, et fournit l'authentification. Elle vous permet de détecter les modifications non autorisées sur l'élément comme un tout, y compris l'ajout ou la suppression d'attributs, ou le remplacement d'une valeur chiffrée par une autre.

Dans un élément chiffré, certaines données demeurent en texte brut, y compris le nom de table, tous les noms d'attribut, les valeurs d'attribut que vous ne chiffrez pas, et les noms et valeurs des attributs de la clé primaire (clé de partition et clé de tri). Ne stockez pas les données sensibles dans ces champs.

Rubriques

Chiffrement des valeurs d'attribut

Le client de chiffrement DynamoDB chiffre les valeurs (mais pas les noms) des attributs que vous spécifiez. Pour déterminer quelles sont les valeurs d'attribut chiffrées, utilisez les actions d'attribut (p. 9).

Par exemple, cet élément inclut les attributs `example` et `test`.

```
'example': 'data',
'test': 'test-value',
...
```

Si vous chiffrez l'attribut `example`, mais pas l'attribut `test`, les résultats se présentent comme suit. La valeur d'attribut `example` chiffrée est une donnée binaire, et non une chaîne.

```
'example': Binary(b"'b\x933\x9a+s\xf1\xd6a\xc5\xd5\x1aZ\xed\xd6\xce\xe9X\xf0T\xcb\x9fY\x9f
\xf3\xc9C\x83\r\xbb\\"),
'test': 'test-value'
...
```

Les attributs de clé primaire (clé de partition et clé de tri) de chaque élément doivent rester en texte brut car DynamoDB les utilise pour trouver l'élément dans la table. Ils doivent être signés, mais pas chiffrés.

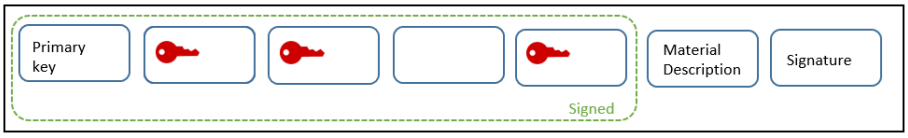
> Warning
>
> Ne chiffrez pas les attributs de la clé primaire. Ceux-ci doivent rester en texte brut pour que DynamoDB puisse trouver l'élément sans exécuter une analyse complète de la table.

Les annotations de chaque langage de programmation identifient automatiquement les attributs de clé primaire et garantissent que leurs valeurs sont signées, mais pas chiffrées. Et, si vous identifiez votre clé primaire, puis essayez de la chiffrer, le client lève une exception. Si vous devez chiffrer la clé primaire pour un scénario particulier, utilisez le bas niveauchiffreur d'élément (p. 9)mais n'oubliez pas que DynamoDB ne pourra pas trouver votre élément sans exécuter une analyse complète de la table.

Le client de chiffrement DynamoDB ne chiffre pas non plus ni ne signe leattribut de description du matériau (p. 10), qui stocke les informations dont le client de chiffrement DynamoDB a besoin pour vérifier et déchiffrer l'élément.

Signature de l'élément

Après avoir chiffré les valeurs d'attribut spécifiées, le client de chiffrement DynamoDB calcule une signature numérique sur les noms et les valeurs des attributs que vous spécifiez dans leactions d'attribut (p. 9)objet. Le client enregistre la signature dans un attribut qu'il ajoute à l'élément.



Si vous fournissez un nom de table, il est inclus dans la signature. Il est vous est ainsi possible de détecter qu'un élément signé a été déplacé vers une autre table, peut-être par malveillance, comme dans le cas du déplacement d'un enregistrement employé de la table `AllEmployees` vers la table `TrustedEmployees`. Le client de chiffrement DynamoDB obtient le nom de la table dans leClient de chiffrement DynamoDB (p. 11), où il est un champ facultatif.

Veillez à inclure la clé primaire dans la signature. Il s'agit du comportement par défaut quand vous utilisez une annotation. La signature capture la relation entre la clé primaire et les autres attributs de l'élément, et la validation de la signature vérifie que la relation n'a pas été modifiée.

L'attribut de description de matériau (p. 10) n'est pas chiffré ou signé.

Élément chiffré et signé

Quand le client de chiffrement DynamoDB chiffre et signe un élément de table, le résultat est un élément de table DynamoDB standard avec les valeurs d'attribut chiffrées.

L'illustration suivante présente une partie d'un exemple d'élément de table chiffré et signé.

```
{
    '*amzn-ddb-map-desc*': Binary(b'\x00\x00\x00\x00\x00\x00\x10amzn-ddb-env-alg\
\x00\x00\x00\xe0AQEBAHhA84wnXjEJdBbBBylRUFcZZK2j7xwh6UyLoL28nQ
+0FAAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgkqhkiG9w0BBwEwHgYJYIZIAWUDBAEuMBEEDPeFBydmoJD
izYl0R0C4M7wAK6E1/N/bgTmHI=\x00\x00\x00\x17amzn-ddb-map-signingAlg\x00\x00\x00\nHmacS
\x00\x00\x00\x11/CBC/PKCS5Padding\x00\x00\x00\x10amzn-ddb-sig-alg\x00\x00\x00\x0eHmac
\x00\x00\x00\x0faws-kms-ec-attr\x00\x00\x00\x06*keys*'),
    '*amzn-ddb-map-sig*': Binary(b"\xd3\xc6\xc7\n\xb7#\x13\xd1Y\xea\xe4.|^\xbd\xdf\xe
    'binary': Binary(b'!"\xc5\x92\xd7\x13\x1d\xe8Bs\x9b\x7f\xa8\x8e\x9c\xcf\x10\x1e\x
    'example': Binary(b"'b\x933\x9a+s\xf1\xd6a\xc5\xd5\x1aZ\xed\xd6\xce\xe9X\xf0T\xcb
    'numbers': Binary(b'\xd5\xa0\\d\xcc\x85\xf5\x1e\xb9-f!\xb9\xb8\x8a\x1aT\xbaq\xf7\
    'partition_attribute': 'value1',
    'sort_attribute': 55,
    'test': 'test-value'
}
```

La figure présente les caractéristiques suivantes des éléments de table chiffrés et signés par le client de chiffrement DynamoDB :

- Tous les noms d'attribut sont en texte brut.
- Les valeurs des attributs de clé primaire sont en texte brut. Dans cet exemple, le nom de la clé de partition est `partition_attribute` et le nom de la clé de tri est `sort_attribute`.
- Les valeurs des attributs que vous dites au client de ne pas chiffrer demeurent en texte brut. Dans cet exemple, la valeur de l'attribut `test` est en texte brut.
- Les valeurs des attributs chiffrés sont des données binaires.
- Le client ajoute un attribut de signature (`*amzn-ddb-map-sig*`) à l'élément. Sa valeur est la signature de l'élément.
- Le client ajoute un attribut de description de matériau (p. 10) (`*amzn-ddb-map-desc*`) à l'élément. Sa valeur décrit comment l'attribut a été chiffré et signé. Le client utilise ces informations pour vérifier et déchiffrer l'élément. L'attribut de description de matériau n'est pas chiffré ou signé.

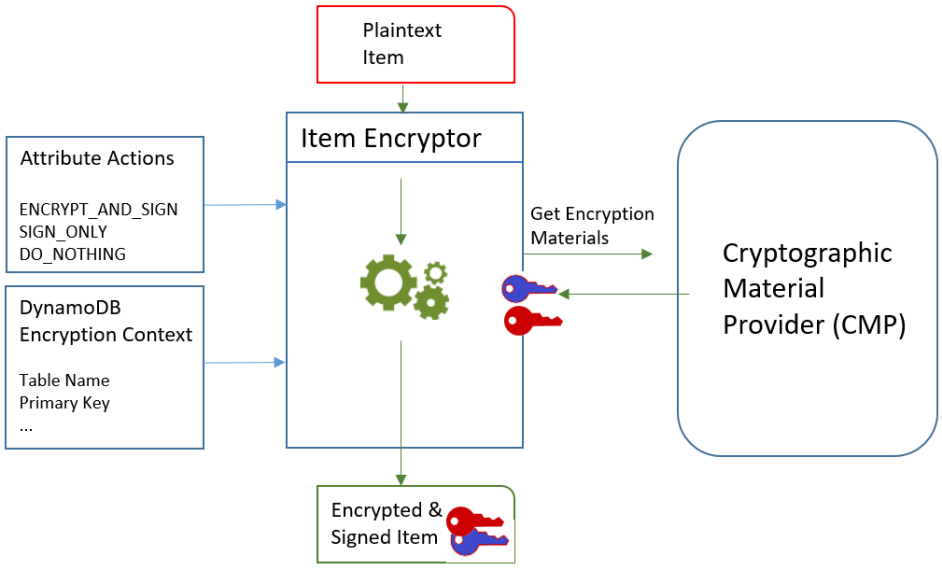
Fonctionnement du client de chiffrement DynamoDB

Le client de chiffrement DynamoDB est spécifiquement conçu pour protéger les données que vous stockez dans DynamoDB. Les bibliothèques incluent les implémentations sécurisées que vous pouvez étendre ou utiliser inchangées. Et, la plupart des éléments sont représentés par des éléments abstraits afin que vous puissiez créer et utiliser des composants personnalisés compatibles.

Chiffrement et signature des éléments de table

Au cœur du client de chiffrement DynamoDB se trouve unChiffreur d'élémentqui chiffre, signe, vérifie et déchiffre des éléments de table. Il prend les informations sur les éléments de table et les instructions sur les éléments à chiffrer et signer. Il obtient les matériaux de chiffrement et les instructions sur leur utilisation auprès d'un fournisseur CMP (p. 9) que vous sélectionnez et configurez.

Le schéma suivant illustre un aperçu de haut niveau du processus.



Pour chiffrer et signer un élément de table, a besoin des éléments suivants :

- Informations sur la table. Il obtient les informations sur la table auprès d'unClient de chiffrement DynamoDB (p. 11)que vous fournissez. Certaines animations obtiennent les informations requises

auprès de DynamoDB auprès de DynamoDB et créent automatiquement le contexte de chiffrement DynamoDB.

Note

LeClient de chiffrement DynamoDBdans DynamoDB Encryption Client n'est pas lié aucontexte de chiffrementdansAWS Key Management Service(AWS KMS) et leAWS Encryption SDK.

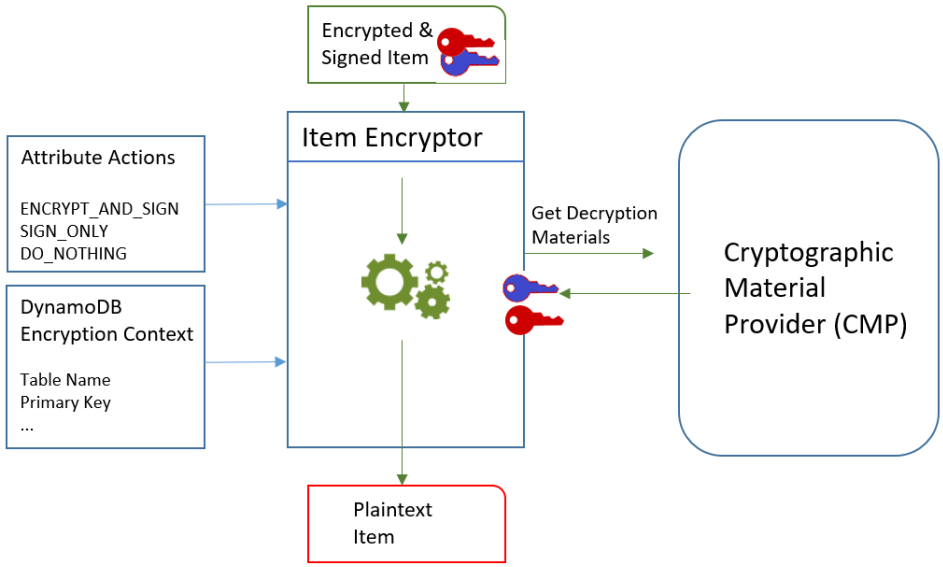
- Attributs à chiffrer et signer. Il obtient ces informations à partir des actions d'attribut (p. 9) que vous fournissez.
- Matériaux de chiffrement, clés de chiffrement et de signature incluses. Il obtient ces informations auprès d'un fournisseur CMP (p. 9) que vous sélectionnez et configurez.
- Instructions pour le chiffrement et la signature de l'élément. Le fournisseur CMP ajoute les instructions sur l'utilisation des matériaux de chiffrement, algorithmes de chiffrement et de signature inclus, à la description du matériau réel (p. 10).

Le chiffreur d'élément (p. 9) utilise tous ces éléments pour chiffrer et signer l'élément. Le chiffreur d'élément ajoute aussi deux attributs à l'élément : un attribut de description de matériau (p. 10) qui contient les instructions de chiffrement et de signature (description du matériau réel), et un attribut qui contient la signature. Vous pouvez interagir directement avec le chiffreur d'élément, ou utilisez les fonctions d'annotation qui interagissent avec le chiffreur d'élément pour que vous implémentiez le comportement par défaut sécurisé.

Il en résulte un élément DynamoDB contenant les données chiffrées et signées.

Vérification et déchiffrement des éléments de table

Ces composants fonctionnent aussi ensemble pour vérifier et déchiffrer votre élément, comme illustré dans le schéma suivant.



Pour vérifier et déchiffrer un élément, a besoin des mêmes composants, de composants avec la même configuration ou de composants spécialement conçus pour déchiffrer les éléments, comme suit :

- Informations sur la tableà partir desClient de chiffrement DynamoDB (p. 11).
- Attributs à vérifier et à déchiffrer. Il obtient ces informations à partir des actions d'attribut (p. 9).

- Matériaux de déchiffrement, clés de vérification et de déchiffrement incluses, depuis le fournisseur CMP (p. 9) que vous sélectionnez et configurez.

  L'élément chiffré n'inclut pas d'enregistrement du CMP qui a été utilisé pour le chiffrer. Vous devez fournir le même CMP, un CMP avec la même configuration ou un CMP qui a été conçu pour déchiffrer les éléments.
- Informations sur la façon dont l'élément a été chiffré et signé, algorithmes de chiffrement et de signature inclus. Le client obtient ces informations à partir de l'attribut de description du matériau (p. 10) de l'élément.

Le chiffreur d'élément (p. 9) utilise tous ces éléments pour vérifier et déchiffrer l'élément. Il supprime aussi les attribut de description de matériau et de signature. Le résultat est un élément DynamoDB en texte brut.

Chiffrement côté client et côté serveur

Le client de chiffrement DynamoDB prend en chargeChiffrement côté client, où vous chiffrez les données de vos tables avant de les envoyer à DynamoDB. Toutefois, DynamoDB fournit un côté serveurChiffrement au reposqui chiffre votre table de façon transparente quand elle est conservée sur disque et la déchiffre quand vous y accédez.

Les outils que vous choisissez dépendent de la sensibilité de vos données et des exigences de sécurité de votre application. Vous pouvez utiliser le client de chiffrement DynamoDB et le chiffrement au repos. Lorsque vous envoyez des éléments chiffrés et signés à DynamoDB, DynamoDB ne reconnaît pas les éléments comme étant protégés. Il détecte simplement les éléments de table classiques avec ses valeurs d'attribut binaires.

Chiffrement côté serveur au repos

Prise en charge de DynamoDBChiffrement au repos, unchiffrement côté serveurdans laquelle DynamoDB chiffre automatiquement de façon transparente vos tables quand la table est conservée sur disque et les déchiffre quand vous accédez aux données de la table.

Avec le chiffrement côté serveur, vos données sont chiffrées en transit via une connexion HTTPS, déchiffrées au point de terminaison DynamoDB, puis à nouveau chiffrées avant d'être stockées dans DynamoDB.

- Chiffrement par défaut. DynamoDB chiffre et déchiffre de façon transparente toutes les tables quand elles sont écrites sur le disque. Aucune option ne permet d'activer ou de désactiver le chiffrement au repos.
- DynamoDB crée et gère les clés de chiffrement.La clé unique de chaque table est protégée par uneAWS KMS keyqui ne part jamaisAWS Key Management Service(AWS KMS) non chiffré. Par défaut, DynamoDB utilise unClé détenue par AWSdans le compte de service DynamoDB, mais vous pouvez choisir unClé gérée par AWSouclé gérée par le clientdans votre compte pour protéger une partie ou la totalité de vos tables.
- Toutes les données des tables sont chiffrées sur disque.Lorsqu'une table chiffrée est enregistrée sur le disque, DynamoDB chiffre toutes les données de la table, inclus leclé primaireet local et mondialIndex secondaires. Si votre table a une clé de tri, certaines clés de tri qui marquent les limites de plage sont stockées en texte brut dans les métadonnées de la table.
- Les objets liés aux tables sont également chiffrés. Le chiffrement au repos protègeDynamoDB Streams,Tables globales, etsauvegardeschaque fois qu'ils sont écrits sur des supports durables.
- Vos éléments sont déchiffrés lorsque vous y accédez.Lorsque vous accédez à la table, DynamoDB déchiffre la partie de la table qui inclut votre élément cible et vous retourne l'élément en texte brut.

Client de chiffrement DynamoDB

Le chiffrement côté client fournitend-to-endla protection de vos données, en transit et au repos, depuis sa source jusqu'au stockage dans DynamoDB. Vos données en texte brut ne sont jamais exposées à une partie tierce, AWS inclus. Toutefois, le client de chiffrement DynamoDB est conçu pour être implémenté dans de nouvelles bases de données non remplies. Vous devez ajouter les fonctions de chiffrement à vos applications DynamoDB avant d'envoyer des données à DynamoDB.

- Vos données sont protégées en transit et au repos. Elles ne sont jamais exposées à une partie tierce, inclusAWS.
- Vous pouvez signer les éléments de vos tables. Vous pouvez demander au client de chiffrement DynamoDB de calculer une signature sur tout ou partie d'un élément de table, attributs de clé primaire et nom de table inclus. Cette signature vous permet de détecter les modifications non autorisées sur l'élément comme un tout, y compris l'ajout ou la suppression d'attributs, ou le remplacement d'une valeur d'attribut par une autre.
- Vous choisissez de quelle façon vos clés de chiffrement sont générées et protégées. Vous pouvez les créer et les gérer vous-même, ou utiliser un service de chiffrement comme AWS Key Management Service ou AWS CloudHSM pour générer et protéger vos clés.
- Vous déterminez la façon dont vos données sont protégées en sélectionnant un fournisseur CMP (p. 13) ou en en écrivant un. Le CMP détermine la stratégie de chiffrement utilisée, y compris à quel moment les clés uniques sont générées, ainsi que les algorithmes de chiffrement et de signature utilisés.
- Le client de chiffrement DynamoDB ne chiffre pas la totalité de la table. Vous pouvez chiffrer les éléments sélectionnés dans une table, ou les valeurs d'attribut sélectionnées dans tout ou partie des éléments. Cependant, le client de chiffrement DynamoDB ne chiffre pas la totalité d'un élément. Il ne chiffre pas les noms d'attribut, ou les noms ou valeurs des attributs de clé primaire (clé de partition et clé de tri). Pour plus d'informations sur ce qui est chiffré (et ce qui ne l'est pas), consultez Quels sont les champs chiffrés et signés ? (p. 3).

AWS Encryption SDK

Si vous chiffrez les données que vous stockez dans DynamoDB, nous vous recommandons d'utiliser DynamoDB Chiffrement Client.

Le kit AWS Encryption SDK est une bibliothèque de chiffrement côté serveur qui vous aide à chiffrer et déchiffrer les données génériques. Même s'il peut protéger tout type de données, il n'est pas conçu pour fonctionner avec des données structurées, comme les enregistrements de base de données. Contrairement au client de chiffrement DynamoDB, leAWS Encryption SDKne peut pas fournir de contrôle d'intégrité au niveau élément et n'a pas de logique pour reconnaître les attributs ou empêcher le chiffrement des clés primaires.

Si vous utilisez le pluginAWS Encryption SDKPour chiffrer un élément de votre table, n'oubliez pas qu'il n'est pas compatible avec le client de chiffrement DynamoDB. Vous ne pouvez pas chiffrer avec une bibliothèque et déchiffrer avec l'autre.

Concepts du client de chiffrement Amazon DynamoDB

Cette rubrique explique les concepts et la terminologie utilisés dans Amazon DynamoDB Encryption Client.

Pour apprendre comment les composants du client de chiffrement DynamoDB interagissent, consultezFonctionnement du client de chiffrement DynamoDB (p. 5).

Rubriques

Fournisseur CMP (Cryptographic Materials Provider)

Lorsque vous implémentez le client de chiffrement DynamoDB, l'une de vos premières tâches consiste à :Sélection d'un fournisseur CMP (p. 13)(CMP) (également appeléFournisseur de matériaux de chiffrement). Votre choix détermine une grande part du reste de l'implémentation.

Un fournisseur CMP recueille, assemble et retourne les matériaux de chiffrement que le chiffreur d'élément (p. 9) utilise pour chiffrer et signer vos éléments de table. Le CMP détermine les algorithmes de chiffrement à utiliser, ainsi que la façon de générer et de protéger le chiffrement et les clés de signature.

Le fournisseur CMP interagit avec le chiffreur d'élément. Celui-ci demande les matériaux de chiffrement ou de déchiffrement au CMP, et le CMP les retourne au chiffreur d'élément. Puis, celui-ci utilise les matériaux de chiffrement pour chiffrer et signer, ou vérifier et déchiffrer, l'élément.

Vous spécifiez le CMP lorsque vous configurez le client. Vous pouvez créer un CMP personnalisé compatible ou utiliser l'un des nombreux CMP de la bibliothèque. La plupart des CMP sont disponibles en plusieurs langages de programmation.

Chiffreurs d'éléments

LeChiffreur d'élémentest un composant de bas niveau qui effectue les opérations de chiffrement pour le client de chiffrement DynamoDB. Il demande les matériaux de chiffrement auprès d'un fournisseur CMP (p. 9), puis utilise les matériaux retournés par le CMP pour chiffrer et signer, ou vérifier et déchiffrer, votre élément de table.

Vous pouvez interagir directement avec le chiffreur d'élément ou utiliser les annotations fournis par votre bibliothèque. Par exemple, DynamoDB Encryption Client for Java inclut un`AttributeEncryptor`classe d'assistance que vous pouvez utiliser avec le`DynamoDBMapper`, au lieu d'interagir directement avec le`DynamoDBEncryptor`Chiffreur d'élément. La bibliothèque Python inclut les classes d'annotations `EncryptedTable`, `EncryptedClient` et `EncryptedResource` qui interagissent avec le chiffreur d'éléments pour vous.

Actions d'attribut

Les actions d'attribut informent le chiffreur d'élément des actions à exécuter sur chaque attribut de l'élément.

L'action d'attribut peut avoir l'une des valeurs suivantes :

- Chiffrer et signer— Chiffrer la valeur d'attribut. Incluez l'attribut (nom et valeur) dans la signature de l'élément.
- Signer uniquement— Inclure l'attribut dans la signature de l'élément.

- Ne rien faire— Ne pas chiffrer ou signer l'attribut.

Pour tout attribut pouvant stocker des données sensibles, utilisez Chiffrer et signer. Pour les attributs de clé primaire (clé de partition et clé de tri), utilisez Signer uniquement. L'attribut de description de matériau (p. 10) et l'attribut de signature ne sont pas signés ou chiffrés. Vous n'avez pas besoin de spécifier les actions d'attribut pour ces attributs.

Choisissez soigneusement vos actions d'attribut. En cas de doute, utilisez Chiffrer et signer. Une fois que vous avez utilisé le client de chiffrement DynamoDB pour protéger vos éléments de table, vous ne pouvez pas modifier l'action d'un attribut sans risquer une erreur de validation de signature. Pour en savoir plus, consultez Modification de votre modèle de données (p. 54).

> Warning
>
> Ne chiffrez pas les attributs de la clé primaire. Ceux-ci doivent rester en texte brut pour que DynamoDB puisse trouver l'élément sans exécuter une analyse complète de la table.

Si l'icôneClient de chiffrement DynamoDB (p. 11)Identifie vos attributs de clé primaire, le client génère une erreur si vous essayez de les chiffrer.

La technique que vous utilisez pour spécifier les actions d'attribut dépend du langage de programmation que vous utilisez. Elle peut aussi être spécifique aux classes d'annotations que vous utilisez.

Pour plus d'informations, consultez la documentation de votre langage de programmation.

- Python (p. 47)
- Java (p. 37)

Description du matériau

La description du matériau pour un élément de table chiffré se compose d'informations, telles que les algorithmes de chiffrement, sur la façon dont l'élément de table est chiffré et signé. Le fournisseur CMP (p. 9) enregistre la description du matériau tandis qu'il rassemble les matériaux de chiffrement pour le chiffrement et la signature. Par la suite, quand il doit rassembler les matériaux de chiffrement pour vérifier et chiffrer l'élément, il utilise la description du matériau comme guide.

Dans DynamoDB Encryption Client, la description du matériau fait référence à trois éléments associés :

Description du matériau demandé

> Certains fournisseurs CMP (p. 9) permettent de spécifier des options avancées, comme un algorithme de chiffrement. Pour indiquer vos choix, vous ajoutez des paires nom-valeur à la propriété de description du matériau duClient de chiffrement DynamoDB (p. 11)dans votre demande de chiffrement d'un élément de tableau. L'élément est appelé description du matériau demandé. Les valeurs autorisées pour la description du matériau demandé sont définies par le fournisseur CMP que vous choisissez.
>
> > Note
> >
> > Comme la description du matériau peut remplacer les valeurs par défaut sécurisées, il est recommandé d'omettre la description du matériau demandé à moins que vous n'ayez une excellente raison de l'utiliser.

Description du matériau réel

> La description du matériau que les fournisseurs CMP (p. 9) retournent est appelée description du matériau réel. Elle décrit les valeurs réelles que le CMP a utilisées quand il a rassemblé les matériaux

de chiffrement. Elle se compose généralement de la description du matériau demandé, le cas échéant, avec les ajouts et les modifications.

Attribut de description du matériau

Le client enregistre la description du matériau réel dans l'attribut de description du matériau de l'élément chiffré. Le nom de l'attribut de description du matériau est `amzn-ddb-map-desc` et sa valeur est la description du matériau réel. Le client utilise les valeurs de l'attribut de description du matériau pour vérifier et déchiffrer l'élément.

Client de chiffrement DynamoDB

LeClient de chiffrement DynamoDBfournit les informations sur la table et l'élément auFournisseur de matériaux de chiffrement (p. 9)(CMP). Dans les implémentations avancées, le contexte de chiffrement DynamoDB peut inclure unDescription du matériau demandé (p. 10).

Lorsque vous chiffrez des éléments de table, le contexte de chiffrement DynamoDB est lié cryptographiquement aux valeurs d'attribut chiffrées. Lorsque vous déchiffrez, si le contexte de chiffrement DynamoDB n'est pas une correspondance exacte et sensible à la casse pour le contexte de chiffrement DynamoDB utilisé pour chiffrer, l'opération de déchiffrement échoue. Si vous interagissez avecChiffreur d'élément (p. 9)directement, vous devez fournir un contexte de chiffrement DynamoDB lorsque vous appelez une méthode de chiffrement ou de déchiffrement. La plupart des assistants créent automatiquement le contexte de chiffrement DynamoDB.

Note

LeClient de chiffrement DynamoDBdans DynamoDB Encryption Client n'est pas lié aucontexte de chiffrementdansAWS Key Management Service(AWS KMS) et leAWS Encryption SDK.

Le contexte de chiffrement DynamoDB peut inclure les champs suivants. Tous les champs et valeurs sont facultatifs.

- Nom de la table
- Nom de la clé de partition
- Nom de la clé de tri
- Paires nom-valeur des attributs
- Description du matériau demandé (p. 10)

Magasin de fournisseur

Un magasin de fournisseur est un composant qui retourne les fournisseurs CMP (p. 9). Le magasin de fournisseur peut créer les CMP ou les obtenir d'une autre source, telle qu'un autre magasin de fournisseur. Le fournisseur de magasin enregistre les versions des CMP qu'il crée dans un stockage permanent où chaque fournisseur CMP stocké est identifié par le nom du matériau du demandeur et le numéro de version.

LeÀ propos du fournisseur le plus récent (p. 23)Dans DynamoDB Encryption Client, obtient ses fournisseurs CMP d'un magasin de fournisseur, mais vous pouvez utiliser le magasin de fournisseur pour fournir les CMP à quelque composant que ce soit. Chaque fournisseur le plus récent est associé à un magasin de fournisseur, mais un magasin de fournisseur peut fournir les CMP à la plupart des demandeurs sur plusieurs hôtes.

Le magasin de fournisseur crée de nouvelles versions des fournisseurs CMP à la demande, et retourne les versions nouvelles et existantes. Il retourne aussi le dernier numéro de version d'un nom de matériau donné. Le demandeur peut ainsi savoir lorsque le magasin de fournisseur a une nouvelle version de son fournisseur, puis la demander.

Le client de chiffrement DynamoDB inclut unMetaStore (p. 26), c'est-à-dire un magasin de fournisseur qui crée des fournisseurs CMP encapsulés ; ceux-ci sont stockés dans DynamoDB et chiffrés à l'aide d'un client de chiffrement DynamoDB interne.

En savoir plus :

- Magasin de fournisseur : Java,Python
- MetaStore : Java,Python

Comment choisir un fournisseur CMP

L'une des plus importantes décisions que vous prenez lors de l'utilisation du client de chiffrement DynamoDB consiste à sélectionner unfournisseur de matériaux de chiffrement (p. 9)(CMP). Le fournisseur CMP assemble et retourne les matériaux de déchiffrement au chiffreur d'élément. Il détermine aussi la façon dont les clés de chiffrement et les clés de signature sont générées, que les nouveaux matériaux de clés soient générés pour chaque élément ou soient réutilisés, et les algorithmes de chiffrement et de signature qui sont utilisés.

Vous pouvez choisir un CMP parmi les implémentations fournies dans les bibliothèques DynamoDB Enchiffrement Client ou générer un CMP personnalisé compatible. Le choix du fournisseur CMP peut aussi dépendre du langage de programmation (p. 35) que vous utilisez.

Cette rubrique décrit les CMP les plus courants et propose quelques conseils pour vous aider à choisir le meilleur fournisseur pour votre application.

Fournisseur de matériaux KMS direct

Le fournisseur Direct KMS Materials Provider protège vos éléments de table sous uneAWS KMS keyqui ne part jamaisAWS Key Management Service(AWS KMS) non chiffré. Votre application n'a pas à générer ou gérer des matériaux de chiffrement. Comme il utilise la AWS KMS key pour générer des clés de chiffrement et de signature uniques pour chaque élément, ce fournisseur appelle AWS KMS chaque fois qu'il chiffre ou déchiffre un élément.

Si vous utilisez AWS KMS et qu'un appel AWS KMS par transaction convient à votre application, ce fournisseur constitue un bon choix.

Pour en savoir plus, consultez Fournisseur de matériaux KMS direct (p. 14).

Fournisseur CMP encapsulé

Le fournisseur CMP encapsulé vous permet de générer et de gérer vos clés d'encapsulation et de signature en dehors du client de chiffrement DynamoDB.

Le fournisseur CMP encapsulé génère une clé de chiffrement unique pour chaque élément. Puis, il utilise les clés d'encapsulation (ou de désencapsulation) et de signature que vous fournissez. En tant que tel, vous déterminez de quelle façon les clés d'encapsulation et de signature sont générées, et si elles sont propres à chaque élément ou sont réutilisées. Le fournisseur CMP encapsulé constitue une alternative sécurisée au fournisseur KMS direct (p. 14) pour les applications qui n'utilisent pas AWS KMS et peuvent gérer de façon sécurisée les matériaux de chiffrement.

Pour en savoir plus, consultez Fournisseur de matériaux encapsulé (p. 19).

À propos du fournisseur le plus récent

Le fournisseur le plus récent est un fournisseur CMP (p. 9) conçu pour travailler avec un magasin de fournisseur (p. 11). Il obtient les fournisseurs CMP auprès d'un magasin de fournisseur et les matériaux de chiffrement qu'il retourne auprès des CMP. Le fournisseur le plus récent utilise généralement chaque fournisseur CMP pour satisfaire plusieurs demandes de matériaux de chiffrement, mais vous pouvez utiliser les fonctions du magasin de fournisseur pour contrôler l'étendue à laquelle les matériaux sont réutilisés, déterminer à quelle fréquence a lieu la rotation des fournisseurs CMP et, même, modifier le type de fournisseur CMP utilisé sans modifier le fournisseur le plus récent.

Vous pouvez utiliser le fournisseur le plus récent avec n'importe quel magasin de fournisseur compatible. Le DynamoDB de chiffrement inclut un métastore, lequel est un magasin de fournisseur qui retourne les CMP encapsulés.

Le fournisseur le plus récent constitue un bon choix pour les applications qui doivent minimiser les appels à leur source de chiffrement, et pour les applications qui peuvent réutiliser certains matériaux de chiffrement sans enfreindre leurs exigences de sécurité. Par exemple, il vous permet de protéger vos matériaux de chiffrement sous uneAWS KMS keydansAWS Key Management Service(AWS KMS) sans appelerAWS KMSchaque fois que vous chiffrez ou déchiffrez un élément.

Pour en savoir plus, consultez À propos du fournisseur le plus récent (p. 23).

Fournisseur de matériaux statique

Le fournisseur CMP statique est destiné aux tests, aux démonstrations preuve de concept et à la compatibilité de l'existant. Il ne génère pas de matériaux de chiffrement uniques pour chaque élément. Il retourne les mêmes clés de chiffrement et de signature que vous fournissez, et ces clés sont utilisées directement pour chiffrer, déchiffrer et signer vos éléments de table.

Note

Le fournisseur statique asymétrique de la bibliothèque Java n'est pas un fournisseur statique. Il fournit juste d'autres constructeurs au fournisseur CMP encapsulé (p. 19). Il est sûr pour une utilisation en production, mais vous devez utiliser directement le CMP encapsulé chaque fois que possible.

Rubriques

Fournisseur de matériaux KMS direct

LeFournisseur de matériaux KMS direct(fournisseur KMS direct) protège vos éléments de table sous unAWS KMS keyqui ne part jamaisAWS Key Management Service(AWS KMS) non chiffré. Le fournisseur CMP (p. 9) retourne une clé de chiffrement et une clé de signature uniques pour chaque élément de table. Pour ce faire, il appelle AWS KMS chaque fois que vous chiffrez ou déchiffrez un élément.

Si vous traitez les éléments DynamoDB à une fréquence élevée et à une grande échelle, il se peut que vous dépassiez leAWS KMS limites de demandes par seconde, ce qui entraîne des retards de traitement. Si vous devez dépasser une limite, créez une requête dans laAWS SupportCenter. Vous pouvez également envisager d'utiliser un fournisseur de matériel cryptographique avec une réutilisation limitée des clés, comme leÀ propos du fournisseur le plus récent (p. 23).

Pour utiliser le fournisseur KMS direct, l'appelant doit disposerunCompte AWS, au moins uneAWS KMS key, et l'autorisation d'appeler leGenerateDataKeyetDecryptsur leAWS KMS key. LeAWS KMS keydoit être une clé de chiffrement symétrique ; le client de chiffrement DynamoDB ne prend pas en charge le chiffrement asymétrique. Si vous utilisez uneTable globale DynamoDB, vous pouvez spécifier uneAWS KMSClé multi-région. Pour plus d'informations, consultez Comment l'utiliser (p. 15).

Note

Lorsque vous utilisez le fournisseur KMS direct, les noms et les valeurs de vos attributs de clé principale apparaissent en texte brut dans leAWS KMScontexte de chiffrementetAWS CloudTrailjournaux des associésAWS KMS. Toutefois, DynamoDB Encryption Client n'expose jamais le texte brut des valeurs d'attributs chiffrées.

Le fournisseur KMS direct est l'un des nombreuxfournisseurs de matériaux de chiffrement (p. 9)(CPM) pris en charge par le client de chiffrement DynamoDB. Pour plus d'informations sur les autres CMP, consultez Comment choisir un fournisseur CMP (p. 13).

Pour obtenir un exemple de code, consultez :

- Java: AwsKmsEncryptedItem
- Python : aws-kms-encrypted-table, aws-kms-encrypted-item

Rubriques

Comment l'utiliser

Pour créer un fournisseur KMS direct, utilisez le paramètre d'ID de clé pour spécifier un chiffrement symétrique.Clé KMSdans votre compte. La valeur du paramètre d'ID de clé peut être l'ID de clé, l'ARN de clé, le nom d'alias ou l'ARN d'alias duAWS KMS key. Pour plus d'informations sur les identificateurs de clés, consultez.Identificateurs clésdans leAWS Key Management ServiceManuel du développeur.

Le fournisseur Direct KMS nécessite une clé KMS de chiffrement symétrique. Vous ne pouvez pas utiliser de clé KMS asymétrique. Toutefois, vous pouvez utiliser une clé KMS multi-région, une clé KMS avec du matériau de clé importé ou une clé KMS dans un magasin de clés personnalisé. Vous devez avoirkms:GenerateDataKeyetkms:Decryptautorisation sur la clé KMS. En tant que tel, vous devez utiliser une clé gérée par le client, et non uneAWSgérée ouAWSpropriétaire de la clé KMS.

Le client de chiffrement DynamoDB pour Python détermine la région à appeler.AWS KMSà partir de la région dans la valeur du paramètre ID clé, s'il en inclut un. Sinon, la région est utilisée dans leAWS KMSclient, si vous en spécifiez un, ou la région que vous configurez dans leAWS SDK for Python (Boto3). Pour plus d'informations sur la sélection de régions dans Python, consultez.Configurationdans leAWSRéférence de l'API SDK for Python (Boto3).

Le client de chiffrement DynamoDB pour Java détermine la région à appelerAWS KMSde la région dans leAWS KMSclient, si le client que vous spécifiez inclut une région. Sinon, la région que vous configurez dans leAWS SDK for Java. Pour plus d'informations sur la sélection de régions dans leAWS SDK for Java, voirRégion AWSsélectiondans leAWS SDK for JavaManuel du développeur

Java

```
// Replace the example key ARN and Region with valid values for your application
final String keyArn = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'
final String region = 'us-west-2'

final AWSKMS kms = AWSKMSClientBuilder.standard().withRegion(region).build();
final DirectKmsMaterialProvider cmp = new DirectKmsMaterialProvider(kms, keyArn);
```

Python

L'exemple suivant utilise l'ARN de clé pour spécifier laAWS KMS key. Si votre identifiant de clé n'inclut pas deRégion AWS, le client de chiffrement DynamoDB obtient la région à partir de la session Botocore configurée, s'il y en a une, ou des valeurs par défaut Boto.

```
# Replace the example key ID with a valid value
kms_key = 'arn:aws:kms:us-west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'
kms_cmp = AwsKmsCryptographicMaterialsProvider(key_id=kms_key)
```

Si vous utilisezTables globales Amazon DynamoDB, nous vous recommandons de chiffrer vos données sous unAWS KMSClé multi-région. Les clés multi-région sontAWS KMS keysdans différentsRégions

AWSqui peuvent être utilisés de manière interchangeable car ils ont un ID de clé et un matériau de clé identiques. Pour plus d'informations, consultez .Utilisation de clés multi-régionsdans leAWS Key Management ServiceManuel du développeur.

> Note
>
> Si vous utilisez les tables globalesversion 2017.11.29, vous devez définir des actions attributaires afin que les champs de réplication réservés ne soient ni chiffrés ni signés. Pour plus d'informations, consultez Problèmes avec les tables globales des versions antérieures (p. 59).

Pour utiliser une clé multi-régions avec DynamoDB Encryption Client, créez une clé multi-régions et répliquez-la dans les régions dans lesquelles votre application s'exécute. Configurez ensuite le fournisseur Direct KMS pour qu'il utilise la clé multi-régions dans la région dans laquelle le client DynamoDB Encryption appelleAWS KMS.

L'exemple suivant configure le client de chiffrement DynamoDB pour chiffrer les données dans la région USA Est (Virginie du Nord) (us-east-1) et les déchiffrer dans la région USA Ouest (Oregon) (us-west-2) à l'aide d'une clé multi-région.

Java

Dans cet exemple, le client de chiffrement DynamoDB obtient la région à appelerAWS KMSde la région dans leAWS KMSclient. Le`keyArn`value identifie une clé multi-région dans la même région.

```
// Encrypt in us-east-1

// Replace the example key ARN and Region with valid values for your application
final String usEastKey = 'arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab'
final String region = 'us-east-1'

final AWSKMS kms = AWSKMSClientBuilder.standard().withRegion(region).build();
final DirectKmsMaterialProvider cmp = new DirectKmsMaterialProvider(kms, usEastKey);
```

```
// Decrypt in us-west-2

// Replace the example key ARN and Region with valid values for your application
final String usWestKey = 'arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab'
final String region = 'us-west-2'

final AWSKMS kms = AWSKMSClientBuilder.standard().withRegion(region).build();
final DirectKmsMaterialProvider cmp = new DirectKmsMaterialProvider(kms, usWestKey);
```

Python

Dans cet exemple, le client de chiffrement DynamoDB obtient la région à appelerAWS KMSdepuis la région dans l'ARN clé.

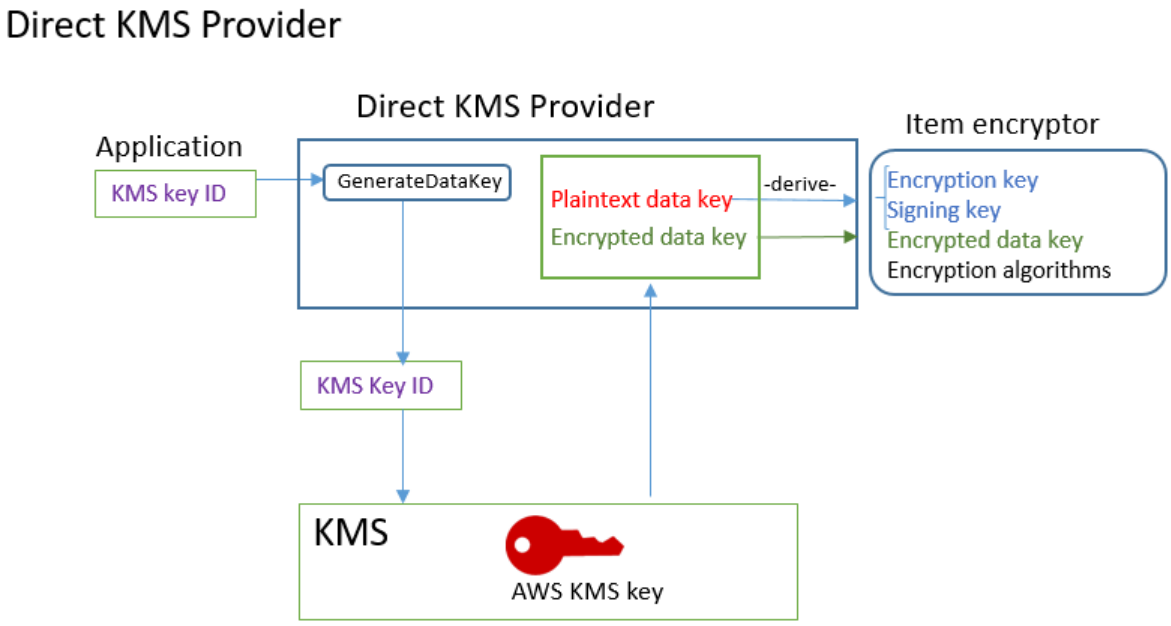
```
# Encrypt in us-east-1

# Replace the example key ID with a valid value
us_east_key = 'arn:aws:kms:us-east-1:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab'
kms_cmp = AwsKmsCryptographicMaterialsProvider(key_id=us_east_key)
```

```
# Decrypt in us-west-2

# Replace the example key ID with a valid value
```

```
us_west_key = 'arn:aws:kms:us-west-2:111122223333:key/
mrk-1234abcd12ab34cd56ef1234567890ab'
kms_cmp = AwsKmsCryptographicMaterialsProvider(key_id=us_west_key)
```

Fonctionnement

Le fournisseur KMS direct renvoie les clés de chiffrement et de signature qui sont protégées par unAWS KMS keyque vous spécifiez, comme indiqué dans le diagramme suivant.



- Pour générer les matériaux de chiffrement, le fournisseur KMS direct demandeAWS KMSpourgénérer une clé de données uniquepour chaque article utilisant unAWS KMS keyque vous spécifiez. Il dérive les clés de chiffrement et de signature de l'élément depuis la copie en texte brut de la clé de données, puis retourne les clés de chiffrement et de signature, ainsi que la clé des données chiffrées, qui est stockée dans l'attribut de description de matériau (p. 10) de l'élément.

  Le chiffreur d'élément utilise les clés de chiffrement et de signature, et les supprime de la mémoire dès que possible. Seule la copie chiffrée de la clé de données à partir de laquelle elles ont été dérivées est enregistrée dans l'élément chiffré.
- Pour générer les matériaux de déchiffrement, le fournisseur KMS direct demande à AWS KMS de déchiffrer la clé de données chiffrée. Puis, il dérive les clés de vérification et de signature de la clé de données en texte brut, et les retourne au chiffreur d'élément.

  Le chiffreur d'élément vérifie l'élément et, si la vérification aboutit, déchiffre les valeurs chiffrées. Puis, il supprime les clés de la mémoire dès que possible.

Obtention des matériaux de chiffrement

Cette section décrit en détail les entrées, les sorties et le traitement du fournisseur KMS direct quand il reçoit une demande de matériaux de chiffrement de la part du chiffreur d'élément (p. 9).

Entrée (depuis l'application)

- ID de clé d'uneAWS KMS key.

Entrée (depuis le chiffreur d'élément)

- Client de chiffrement DynamoDB (p. 11)

Sortie (vers le chiffreur d'élément)

- Clé de chiffrement (texte brut)
- Clé de signature
- Dansdescription du matériau réel (p. 10) : Ces valeurs sont enregistrées dans l'attribut de description du matériau que le client ajoute à l'élément.
  - amzn-ddb-env-key : clé de données codée en base 64 chiffrée par leAWS KMS key
  - amzn-ddb-env-alg : Algorithme de chiffrement, par défautAES/256
  - amzn-ddb-sig-alg : Algorithme de signature, par défaut,HmacSHA256/256
  - amzn-ddb-wrap-alg: kms

En cours

1. Le fournisseur KMS direct envoieAWS KMSune demande d'utilisation de l'option spécifiéeAWS KMS keypourgénérer une clé de données uniquepour l'élément. L'opération retourne une clé en texte brut et une copie chiffrée sous la AWS KMS key. Ce matériau est appelé matériau de clé initial.

   La demande inclut les valeurs suivantes en texte brut dans le contexte de chiffrement AWS KMS. Ces valeurs non secrètes sont liées en termes de chiffrement à l'objet chiffré, si bien que le même contexte de chiffrement est requis au déchiffrement. Vous pouvez utiliser ces valeurs pour identifier l'appel de AWS KMS dans les journaux AWS CloudTrail.
   - amzn-ddb-env-alg — Algorithme de chiffrement, par défaut AES/256
   - amzn-ddb-sig-alg — Algorithme de signature, par défaut HmacSHA256/256
   - (Facultatif) aws-kms-table –*nom de table*
   - (Facultatif)*nom de la clé de partition*–*valeur de clé de partition*(valeurs binaires codées en base 64)
   - (Facultatif)*nom de la clé de tri*–*valeur de clé de tri*(valeurs binaires codées en base 64)

   Le fournisseur Direct KMS obtient les valeurs duAWS KMSConsignation du contexte de chiffrementClient de chiffrement DynamoDB (p. 11)pour l'élément. Si le contexte de chiffrement DynamoDB n'inclut pas de valeur, comme le nom de table, la paire nom-valeur est omise duAWS KMScontexte de chiffrement.
2. Le fournisseur KMS direct dérive une clé de chiffrement symétrique et une clé de signature à partir de la clé de données. Par défaut, il utilise SHA (Secure Hash Algorithm) 256 et la fonction de dérivation de clé basée sur HMAC RFC5869 pour dériver une clé de chiffrement symétrique AES 256 bits et une clé de signature HMAC-SHA-256 256 bits.
3. Le fournisseur KMS direct retourne la sortie du chiffreur d'élément.
4. Le chiffreur d'élément utilise la clé de chiffrement pour chiffrer les attributs spécifiés et la clé de signature pour les signer, à l'aide des algorithmes spécifiés dans la description du matériau réel. Il supprime les clés en texte brut de la mémoire dès que possible.

Obtention des matériaux de déchiffrement

Cette section décrit en détail les entrées, les sorties et le traitement du fournisseur KMS direct quand il reçoit une demande de déchiffrement de matériaux de la part du chiffreur d'élément (p. 9).

Entrée (depuis l'application)

- ID de clé d'uneAWS KMS key.

  La valeur de l'ID de clé peut être l'ID de clé, l'ARN de clé, le nom d'alias ou l'ARN d'alias duAWS KMS key. Toutes les valeurs qui ne sont pas incluses dans l'ID de clé, comme la région, doivent être disponibles dans leAWSprofil nommé. L'ARN de clé fournit toutes les valeurs quiAWS KMSbesoins.

Entrée (depuis le chiffreur d'élément)

- Une copie duClient de chiffrement DynamoDB (p. 11)qui contient le contenu de l'attribut de description du matériau.

Sortie (vers le chiffreur d'élément)

- Clé de chiffrement (texte brut)
- Clé de signature

En cours

1. Le fournisseur KMS direct obtient la clé de données chiffrée depuis l'attribut de description du matériau dans l'élément chiffré.
2. Il demande à AWS KMS d'utiliser la AWS KMS key spécifiée pour déchiffrer la clé de données chiffrée. L'opération retourne une clé en texte brut.

   Cette demande doit utiliser le même contexte de chiffrement AWS KMS que celui utilisé pour générer et chiffrer la clé de données.

   - aws-kms-table —*`nom de table`*
   - *`nom de la clé de partition`*–*`valeur de clé de partition`*(valeurs binaires codées en base 64)
   - (Facultatif)*`nom de la clé de tri`*–*`valeur de clé de tri`*(valeurs binaires codées en base 64)
   - amzn-ddb-env-alg — Algorithme de chiffrement, par défaut AES/256
   - amzn-ddb-sig-alg — Algorithme de signature, par défaut HmacSHA256/256
3. Le fournisseur KMS direct utilise SHA (Secure Hash Algorithm) 256 et la fonction de dérivation de clé basée sur HMAC RFC5869 pour dériver une clé de chiffrement symétrique AES 256 bits et une clé de signature HMAC-SHA-256 256 bits depuis la clé de données.
4. Le fournisseur KMS direct retourne la sortie du chiffreur d'élément.
5. Le chiffreur d'élément utilise la clé de signature pour vérifier l'élément. S'il réussit, il utilise la clé de chiffrement symétrique pour déchiffrer les valeurs d'attribut chiffrées. Ces opérations utilisent les algorithmes de chiffrement et de signature spécifiés dans la description du matériau réel. Le chiffreur d'élément supprime les clés en texte brut de la mémoire dès que possible.

Fournisseur de matériaux encapsulé

LeFournisseur de matériaux encapsulé(CMP encapsulé) vous permet d'utiliser des clés d'habillage et de signature de n'importe quelle source avec le client de chiffrement DynamoDB. Le CMP encapsulé ne dépend d'aucunAWSservice. Cependant, vous devez générer et gérer vos clés d'encapsulation et de signature en dehors du client, y compris la fourniture des clés appropriées pour vérifier et déchiffrer l'élément.

Le fournisseur CMP encapsulé génère une clé de chiffrement d'élément unique pour chaque élément. Il encapsule la clé de chiffrement d'élément avec la clé d'encapsulation que vous fournissez et enregistre

la clé de chiffrement d'élément encapsulée dans l'attribut de description de matériau (p. 10) de l'élément. Comme vous fournissez les clés d'encapsulation et de signature, vous déterminez de quelle façon les clés d'encapsulation et de signature sont générées, et si elles sont propres à chaque élément ou sont réutilisées.

Le fournisseur CMP encapsulé constitue une implémentation sécurisée et un bon choix pour les applications qui peuvent gérer les matériaux de chiffrement.

Le CMP enveloppé est l'un des nombreuxfournisseurs de matériaux de chiffrement (p. 9)(CPM) pris en charge par le client de chiffrement DynamoDB. Pour plus d'informations sur les autres CMP, consultez Comment choisir un fournisseur CMP (p. 13).

Pour obtenir un exemple de code, consultez :

- Java: AsymmetricEncryptedItem
- Python : wrapped-rsa-encrypted-table, wrapped-symmetric-encrypted-table

Rubriques

Comment l'utiliser

Pour créer un fournisseur CMP encapsulé, spécifiez une clé d'encapsulation (requise au chiffrement), une clé de désencapsulation (requise au déchiffrement) et une clé de signature. Vous devez fournir les clés lorsque vous chiffrez et déchiffrez les éléments.

Les clés d'encapsulation, de désencapsulation et de signature peuvent être des clés symétriques ou des paires de clés asymétriques.

Java

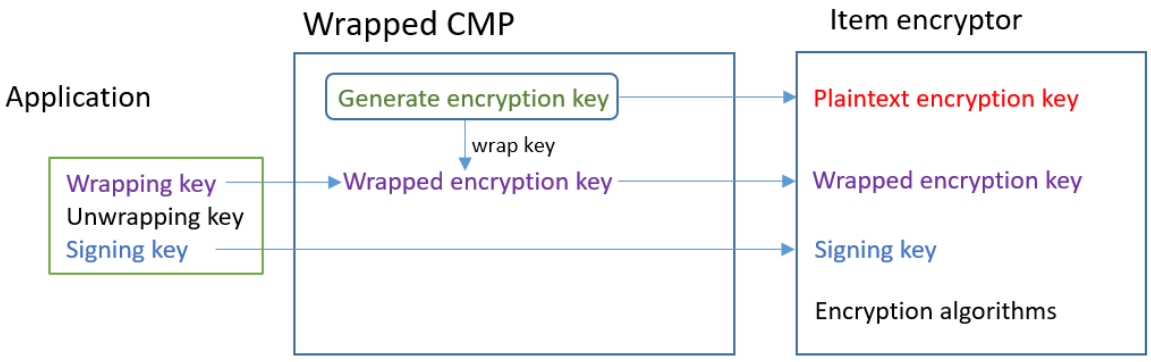
```
// This example uses asymmetric wrapping and signing key pairs
final KeyPair wrappingKeys = ...
final KeyPair signingKeys = ...

final WrappedMaterialsProvider cmp =
    new WrappedMaterialsProvider(wrappingKeys.getPublic(),
                                 wrappingKeys.getPrivate(),
                                 signingKeys);
```

Python

```
# This example uses symmetric wrapping and signing keys
wrapping_key = ...
signing_key  = ...

wrapped_cmp = WrappedCryptographicMaterialsProvider(
    wrapping_key=wrapping_key,
    unwrapping_key=wrapping_key,
    signing_key=signing_key
)
```

Fonctionnement

Le fournisseur CMP encapsulé génère une nouvelle clé de chiffrement d'élément pour chaque élément. Il utilise les clés d'encapsulation, de désencapsulation et de signature que vous fournissez, comme illustré dans le schéma suivant.



Obtention des matériaux de chiffrement

Cette section décrit en détail les entrées, les sorties et le traitement du fournisseur CMP encapsulé quand il reçoit une demande de matériaux de chiffrement.

Entrée (depuis l'application)

- Clé d'encapsulation : UnNorme de chiffrement avancée(AES), ou une clé symétriqueRSAClé publique. Obligatoire si les valeurs d'attribut sont chiffrées. Sinon, elle est facultative et ignorée.
- Clé de déballage : Facultatif et ignoré.
- Clé de signature

Entrée (depuis le chiffreur d'élément)

- Client de chiffrement DynamoDB (p. 11)

Sortie (vers le chiffreur d'élément) :

- Clé de chiffrement d'élément en texte brut
- Clé de signature (inchangée)
- Description du matériau réel (p. 10) : Ces valeurs sont enregistrées dans leAttribut de description du matériau (p. 10)que le client ajoute à l'article.
  - `amzn-ddb-env-key` : clé de chiffrement d'élément codée en base64
  - `amzn-ddb-env-alg` : algorithme de chiffrement utilisé pour chiffrer l'élément. La valeur par défaut est AES-256-CBC.
  - `amzn-ddb-wrap-alg` : algorithme d'encapsulation utilisé par le fournisseur CMP encapsulé pour encapsuler la clé de chiffrement d'élément. Si la clé d'encapsulation est une clé AES, la clé est encapsulée à l'aide de `AES-Keywrap` non complétée, comme défini dans RFC 3394. Si la clé d'encapsulation est une clé RSA, la clé est chiffrée à l'aide de RSA OAEP avec remplissage MGF1.

En cours

Lorsque vous chiffrez un élément, vous transmettez une clé d'encapsulation et une clé de signature. Une clé de désencapsulation est facultative et ignorée.

1. Le fournisseur CMP encapsulé génère une clé de chiffrement d'élément symétrique unique pour l'élément de table.
2. Il utilise la clé d'encapsulation que vous spécifiez pour encapsuler la clé de chiffrement d'élément. Puis, il la supprime de la mémoire dès que possible.
3. Il retourne la clé de chiffrement d'élément en texte brut, la clé de signature que vous avez fournie et une description de matériau réel (p. 10) qui inclut la clé de chiffrement d'élément encapsulé, ainsi que les algorithmes de chiffrement et d'encapsulation.
4. Le chiffreur d'élément utilise la clé de chiffrement en texte brut pour chiffrer l'élément. Il utilise la clé de signature que vous avez fournie pour signer l'élément. Puis, il supprime les clés en texte brut de la mémoire dès que possible. Il copie les champs de la description du matériel réel, y compris la clé de chiffrement encapsulée (`amzn-ddb-env-key`), dans l'attribut de la description de matériau de l'élément.

Obtention des matériaux de déchiffrement

Cette section décrit en détail les entrées, les sorties et le traitement du fournisseur CMP encapsulé quand il reçoit une demande de matériaux de déchiffrement.

Entrée (depuis l'application)

- Clé d'encapsulation : Facultatif et ignoré.
- Clé de déballage : Identique àNorme de chiffrement avancée(AES) à clé symétrique ouRSAClé privée qui correspond à la clé publique RSA utilisée pour chiffrer. Obligatoire si les valeurs d'attribut sont chiffrées. Sinon, elle est facultative et ignorée.
- Clé de signature

Entrée (depuis le chiffreur d'élément)

- Une copie duClient de chiffrement DynamoDB (p. 11)qui contient le contenu de l'attribut de la description du matériau.

Sortie (vers le chiffreur d'élément)

- Clé de chiffrement d'élément en texte brut
- Clé de signature (inchangée)

En cours

Lorsque vous déchiffrez un élément, vous transmettez une clé de désencapsulation et une clé de signature. Une clé d'encapsulation est facultative et ignorée.

1. Le fournisseur CMP encapsulé obtient la clé de chiffrement d'élément encapsulé depuis l'attribut de description du matériau de l'élément.
2. Il utilise la clé et l'algorithme de désencapsulation pour désencapsuler la clé de chiffrement d'élément.
3. Il retourne la clé de chiffrement de l'élément en texte brut, la clé de signature, et les algorithmes de chiffrement et de signature au chiffreur d'élément.
4. Le chiffreur d'élément utilise la clé de signature pour vérifier l'élément. S'il réussit, il utilise la clé de chiffrement d'élément pour déchiffrer l'élément. Puis, il supprime les clés en texte brut de la mémoire dès que possible.

À propos du fournisseur le plus récent

Le fournisseur le plus récent est un fournisseur CMP (p. 9) conçu pour travailler avec un magasin de fournisseur (p. 11). Il obtient les fournisseurs CMP auprès d'un magasin de fournisseur et les matériaux de chiffrement qu'il retourne auprès des CMP. Il utilise généralement chaque CMP pour répondre à plusieurs demandes de matériaux de chiffrement. Cependant, vous pouvez utiliser les fonctions de son magasin de fournisseur pour contrôler jusqu'à quelle mesure les matériaux sont réutilisés, déterminer la fréquence de rotation de son fournisseur CMP et, même, modifier le type de fournisseur CMP utilisé sans modifier le fournisseur le plus récent.

> Note
>
> Code associé au`MostRecentProvider`pour le fournisseur le plus récent peut stocker des documents cryptographiques en mémoire pendant toute la durée de vie du processus. Il peut permettre à un appelant d'utiliser des clés qu'il n'est plus autorisé à utiliser. Le`MostRecentProvider`est obsolète dans les anciennes versions prises en charge du client DynamoDB Encryption Client et supprimé de la version 2.0.0. Il est remplacé par le`CachingMostRecentProvider`Symbole. Pour en savoir plus, consultez Mises à jour du fournisseur le plus récent (p. 31).

Le fournisseur le plus récent constitue un bon choix pour les applications qui doivent minimiser les appels au magasin de fournisseur et à sa source de chiffrement, et pour les applications qui peuvent réutiliser certains matériaux de chiffrement sans enfreindre leurs exigences de sécurité. Par exemple, il vous permet de protéger vos matériaux de chiffrement sous unAWS KMS keydansAWS Key Management Service(AWS KMS) sans appelerAWS KMSchaque fois que vous chiffrez ou déchiffrez un élément.

Le magasin de fournisseur que vous choisissez détermine le type de CMP que le fournisseur le plus récent utilise et à quelle fréquence il obtient un nouveau fournisseur CMP. Vous pouvez utiliser tout magasin de fournisseur compatible avec le fournisseur le plus récent, y compris les magasins de fournisseur personnalisés que vous concevez.

Le client de chiffrement DynamoDB inclut unMetaStorequi crée et retourneFournisseurs de matériaux encapsulés (p. 19)(CMP enveloppés). Le MetaStore enregistre plusieurs versions des fournisseurs CMP encapsulés qu'il génère dans une table DynamoDB interne et les protège avec le chiffrement côté client par une instance interne du client de chiffrement DynamoDB.

Vous pouvez configurer le MetaStore pour utiliser n'importe quel type de CMP interne pour protéger les matériaux de la table, y compris unFournisseur KMS direct (p. 14)qui génère des matériaux cryptographiques protégés par votreAWS KMS key, CMP encapsulé qui utilise des clés d'emballage et de signature que vous fournissez, ou un CMP personnalisé compatible que vous concevez.

Pour obtenir un exemple de code, consultez :

- Java: MostRecentEncryptedItem
- Python : most_recent_provider_encrypted_table

Rubriques

Comment l'utiliser

Pour créer un fournisseur le plus récent, vous devez créer et configurer un magasin de fournisseur, puis créer un fournisseur le plus récent qui utilise le magasin de fournisseur.

Les exemples suivants montrent comment créer un fournisseur le plus récent qui utilise un MetaStore et protège les versions de sa table DynamoDB interne avec des matériaux de chiffrement à partir d'unFournisseur KMS direct (p. 14). Ces exemples utilisent l'`CachingMostRecentProvider` (p. 31)Symbole.

Chaque fournisseur le plus récent porte un nom qui identifie ses CMP dans la table MetaStore, undurée de vie (p. 27)(TTL) et un paramètre de taille de cache qui détermine le nombre d'entrées que le cache peut contenir. Ces exemples définissent la taille du cache à 1000 entrées et un TTL de 60 secondes.

Java

```
// Set the name for MetaStore's internal table
final String keyTableName = 'metaStoreTable'

// Set the Region and AWS KMS key
final String region = 'us-west-2'
final String keyArn = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'

// Set the TTL and cache size
final long ttlInMillis = 60000;
final long cacheSize = 1000;

// Name that identifies the MetaStore's CMPs in the provider store
final String materialName = 'testMRP'

// Create an internal DynamoDB client for the MetaStore
final AmazonDynamoDB ddb =
 AmazonDynamoDBClientBuilder.standard().withRegion(region).build();

// Create an internal Direct KMS Provider for the MetaStore
final AWSKMS kms = AWSKMSClientBuilder.standard().withRegion(region).build();
final DirectKmsMaterialProvider kmsProv = new DirectKmsMaterialProvider(kms, keyArn);

// Create an item encryptor for the MetaStore,
// including the Direct KMS Provider
final DynamoDBEncryptor keyEncryptor = DynamoDBEncryptor.getInstance(kmsProv);

// Create the MetaStore
final MetaStore metaStore = new MetaStore(ddb, keyTableName, keyEncryptor);

//Create the Most Recent Provider
final CachingMostRecentProvider cmp = new CachingMostRecentProvider(metaStore,
 materialName, ttlInMillis, cacheSize);
```

Python

```
# Designate an AWS KMS key
kms_key_id = 'arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'

# Set the name for MetaStore's internal table
meta_table_name = 'metaStoreTable'

# Name that identifies the MetaStore's CMPs in the provider store
material_name = 'testMRP'

# Create an internal DynamoDB table resource for the MetaStore
meta_table = boto3.resource('dynamodb').Table(meta_table_name)

# Create an internal Direct KMS Provider for the MetaStore
kms_cmp = AwsKmsCryptographicMaterialsProvider(key_id=kms_key_id)
```

```
# Create the MetaStore with the Direct KMS Provider
meta_store = MetaStore(
    table=meta_table,
    materials_provider=kms_cmp
)

# Create a Most Recent Provider using the MetaStore
#    Sets the TTL (in seconds) and cache size (# entries)
most_recent_cmp = MostRecentProvider(
    provider_store=meta_store,
    material_name=material_name,
    version_ttl=60.0,
    cache_size=1000
)
```

Fonctionnement

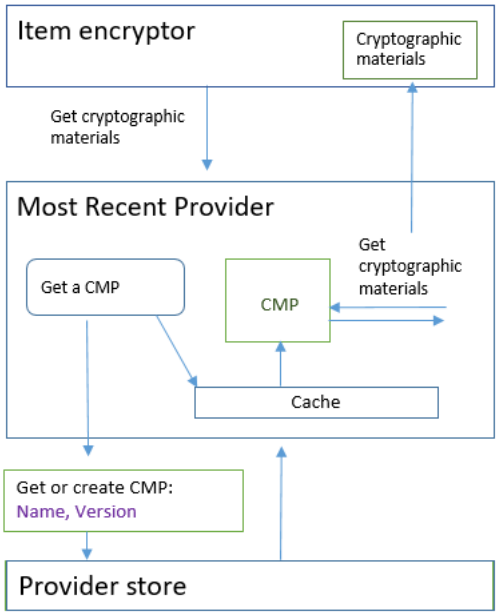
Le fournisseur le plus récent obtient les fournisseurs CMP auprès d'un magasin de fournisseur. Puis, il utilise le fournisseur CMP pour générer les matériaux de chiffrement qu'il retourne au chiffreur d'élément.

À propos du fournisseur le plus récent

Le fournisseur le plus récent obtient un fournisseur CMP (p. 9) à partir d'un magasin de fournisseur (p. 11). Puis, il utilise le fournisseur CMP pour générer les matériaux de chiffrement qu'il retourne. Chaque fournisseur le plus récent est associé à un magasin de fournisseur, mais un magasin de fournisseur peut fournir les CMP à plusieurs fournisseurs répartis sur plusieurs hôtes.

Le fournisseur le plus récent peut travailler avec n'importe quel fournisseur CMP compatible d'un magasin de fournisseur. Il demande les matériaux de chiffrement ou de déchiffrement auprès du fournisseur CMP et retourne la sortie du chiffreur d'élément. Il n'effectue pas d'opération de chiffrement.

Pour demander un fournisseur CMP auprès de son magasin de fournisseur, le fournisseur le plus récent fournit son nom de matériau et la version d'un fournisseur CMP existant qu'il veut utiliser. Pour les matériaux de chiffrement, le fournisseur le plus récent demande toujours la version maximale (la « plus récente »). Pour les matériaux de chiffrement, il demande la version du fournisseur CMP qui a été utilisée pour créer les matériaux de chiffrement, comme illustré dans le diagramme suivant.



Le fournisseur le plus récent enregistre les versions des fournisseurs CMP que le magasin de fournisseur retourne dans un cache LRU (Least Recently Used) local en mémoire. Le cache permet au fournisseur le plus récent d'obtenir les fournisseurs CMP dont il a besoin sans appeler le magasin de fournisseur pour chaque élément. Vous pouvez effacer le cache à la demande.

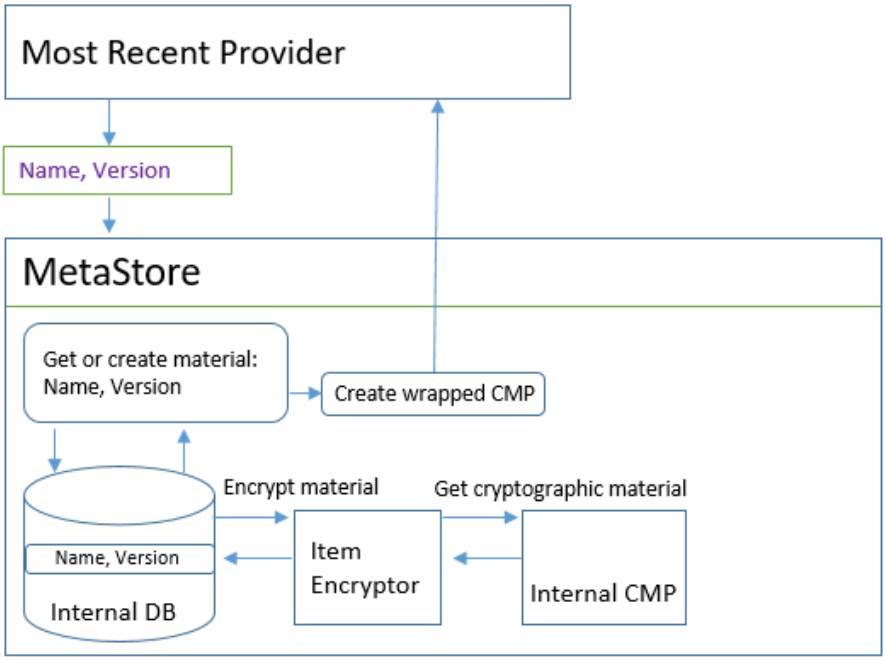
Le fournisseur le plus récent utilise un fournisseur configurableValeur de durée de vie (p. 27)que vous pouvez ajuster en fonction des caractéristiques de votre application.

À propos du métastore

Vous pouvez utiliser un fournisseur le plus récent avec n'importe quel magasin de fournisseur, y compris un magasin de fournisseur personnalisé compatible. Le client de chiffrement DynamoDB inclut un MetaStore, implémentation sécurisée que vous pouvez configurer et personnaliser.

Un métastore est un magasin de fournisseur (p. 11) qui crée et retourne les fournisseurs CMP encapsulés (p. 19) configurés avec la clé d'encapsulation, la clé de désencapsulation et la clé de signature que les fournisseurs CMP encapsulés requièrent. Un MetaStore est une option sécurisée du fournisseur le plus récent, car les fournisseurs CMP encapsulés génèrent toujours des clés de chiffrement d'élément uniques pour chaque élément. Seule la clé d'encapsulation qui protège la clé de chiffrement d'élément et la clé de signature sont réutilisées.

Le diagramme suivant montre les composants du métastore et la façon dont il interagit avec le fournisseur le plus récent.



Le MetaStore génère les fournisseurs CMP encapsulés, puis les stocke sous une forme chiffrée dans une table DynamoDB interne. La clé de partition est le nom du matériel fournisseur le plus récent ; la clé de tri est son numéro de version. Les matériaux de la table sont protégés par un client de chiffrement DynamoDB interne, y compris un chiffreur d'élément et interne.Fournisseur de matériaux de chiffrement (p. 9)(CMP).

Vous pouvez utiliser n'importe quel type de fournisseur CMP interne dans votre métastore, y compris un fournisseur KMS direct (p. 19), un fournisseur encapsulé avec les matériaux de chiffrement que vous fournissez ou un fournisseur CMP personnalisé compatible. Si le fournisseur CMP interne dans votre MetaStore est un fournisseur KMS direct, vos clés d'emballage et de signature réutilisables sont protégées sous unAWS KMS keydansAWS Key Management Service(AWS KMS). Le métastore appelle AWS KMS chaque fois qu'il ajoute une nouvelle version CMP à sa table interne ou qu'il obtient une version CMP de sa table interne.

Définition d'une valeur de durée de vie

Vous pouvez définir une valeur de durée de vie (TTL) pour chaque fournisseur le plus récent que vous créez. En général, utilisez la valeur TTL la plus faible qui est pratique pour votre application.

L'utilisation de la valeur TTL est modifiée dans le`CachingMostRecentProvider`Symbole du fournisseur le plus récent.

> **Note**
>
> Le`MostRecentProvider`pour le fournisseur le plus récent est obsolète dans les anciennes versions prises en charge du client DynamoDB Encryption et supprimé de la version 2.0.0. Il est remplacé par le`CachingMostRecentProvider`Symbole. Nous vous recommandons de mettre votre code à jour dès que possible. Pour en savoir plus, consultez Mises à jour du fournisseur le plus récent (p. 31).

`CachingMostRecentProvider`

Le`CachingMostRecentProvider`utilise la valeur TTL de deux manières différentes.

- Le TTL détermine à quelle fréquence le fournisseur le plus récent vérifie dans le magasin du fournisseur une nouvelle version du CMP. Si une nouvelle version est disponible, le fournisseur

le plus récent remplace son CMP et actualise ses matériaux cryptographiques. Sinon, il continue d'utiliser son CMP actuel et son matériel cryptographique.

- Le TTL détermine la durée d'utilisation des CMP dans le cache. Avant d'utiliser un CMP mis en cache pour le chiffrement, le fournisseur le plus récent évalue son temps dans le cache. Si le temps de cache CMP dépasse le TTL, le CMP est expulsé du cache et le fournisseur le plus récent obtient un nouveau CMP de la dernière version de son magasin de fournisseurs.

`MostRecentProvider`

Dans`MostRecentProvider`, le TTL détermine à quelle fréquence le fournisseur le plus récent vérifie dans le magasin du fournisseur une nouvelle version du CMP. Si une nouvelle version est disponible, le fournisseur le plus récent remplace son CMP et actualise ses matériaux cryptographiques. Sinon, il continue d'utiliser son CMP actuel et son matériel cryptographique.

Le TTL ne détermine pas la fréquence de création d'une nouvelle version CMP. Vous créez de nouvelles versions CMP parrotation des matériaux de chiffrement (p. 28).

La valeur TTL idéale varie en fonction de l'application et de ses objectifs de latence et de disponibilité. Un TTL inférieur améliore votre profil de sécurité en réduisant le temps pendant lequel les matériaux cryptographiques sont stockés en mémoire. De plus, un TTL inférieur actualise plus fréquemment les informations critiques. Par exemple, si votre CMP interne est unFournisseur KMS direct (p. 14), il vérifie plus fréquemment que l'appelant est toujours autorisé à utiliser unAWS KMS key.

Toutefois, si le TTL est trop court, les appels fréquents vers le magasin fournisseur peuvent augmenter vos coûts et amener votre magasin fournisseur à limiter les demandes de votre application et d'autres applications partageant votre compte de service. Vous pourriez également bénéficier de la coordination du TTL avec la vitesse à laquelle vous faites pivoter les matériaux cryptographiques.

Pendant les tests, variez le TTL et la taille du cache sous différentes charges de travail jusqu'à ce que vous trouviez une configuration qui convient à votre application et à vos normes de sécurité et de performances.

Rotation des matériaux de chiffrement

Lorsqu'un fournisseur le plus récent a besoin de matériel de chiffrement, il utilise toujours la version la plus récente de son CMP qu'il connaît. La fréquence qu'il vérifie pour une version plus récente est déterminée par ledurée de vie (p. 27)(TTL) que vous définissez lorsque vous configurez le fournisseur le plus récent.

Lorsque le TTL expire, le fournisseur le plus récent vérifie dans le magasin des fournisseurs la plus récente version du CMP. Si un fournisseur est disponible, le fournisseur le plus récent l'obtient et remplace le CMP dans son cache. Il utilise ce CMP et ses matériaux cryptographiques jusqu'à ce qu'il découvre que le magasin de fournisseurs possède une version plus récente.

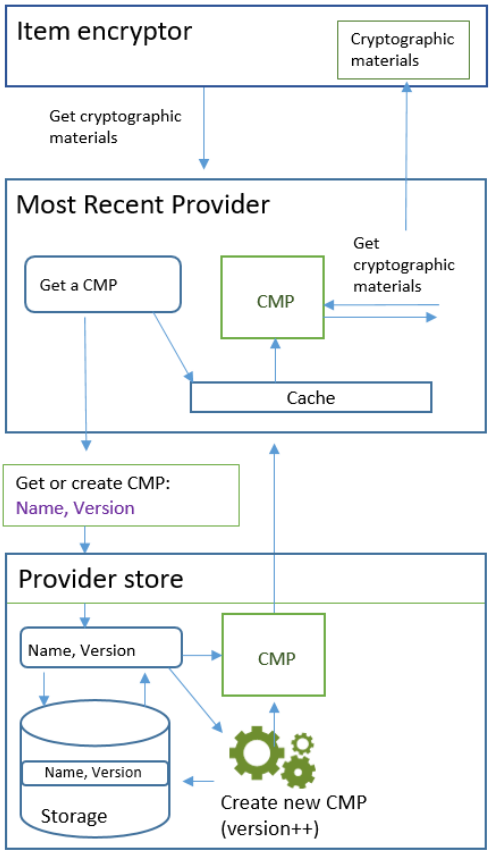
Pour demander au magasin de fournisseur de créer une version d'un CMP pour un fournisseur le plus récent, appelez l'opération Create New Provider du magasin de fournisseur avec le nom de matériau du fournisseur le plus récent. Le magasin de fournisseur crée un CMP et enregistre une copie chiffrée dans son stockage interne avec un numéro de version supérieur. (Il retourne aussi un CMP, mais vous pouvez l'ignorer.) En conséquence, la prochaine fois où le fournisseur le plus récent demande au magasin de fournisseur le numéro de version maximal de ses CMP, il obtient le nouveau numéro de version supérieur, et l'utilise dans les demandes suivantes au magasin pour voir si une nouvelle version du fournisseur CMP a été créée.

Vous pouvez planifier vos appels Create New Provider en fonction de l'heure, du nombre d'éléments ou d'attributs traités, ou de toute autre métrique qui revêt un sens pour votre application.

Obtention des matériaux de chiffrement

Le fournisseur le plus récent utilise le processus suivant, illustré dans le diagramme, pour obtenir les matériaux de chiffrement qu'il retourne au chiffreur d'élément. La sortie dépend du type de fournisseur CMP

que le magasin de fournisseur retourne. Le fournisseur le plus récent peut utiliser n'importe quel magasin de fournisseur compatible, y compris le MetaStore inclus dans le client de chiffrement DynamoDB.



Lorsque vous créez un fournisseur le plus récent en utilisant le`CachingMostRecentProvider`symbole (p. 31), vous spécifiez un magasin de fournisseurs, un nom pour le fournisseur le plus récent et undurée de vie (p. 27)(TTL). Vous pouvez également spécifier une taille de cache, qui détermine le nombre maximal de matériaux cryptographiques pouvant exister dans le cache.

Quand le chiffreur d'élément demande au fournisseur le plus récent les matériaux de chiffrement, le fournisseur le plus récent commence par chercher dans le cache le dernier numéro de version de ses fournisseurs CMP.

- S'il trouve la dernière version dans le cache et que le fournisseur CMP n'a pas dépassé la valeur TTL, le fournisseur le plus récent utilise le fournisseur CMP pour générer les matériaux de chiffrement. Puis, il retourne les matériaux de chiffrement au chiffreur d'élément. Cette opération ne requiert pas d'appel au magasin de fournisseur.
- Si la dernière version du fournisseur CMP n'est pas dans le cache, ou si elle est dans le cache mais a dépassé sa valeur TTL, le fournisseur le plus récent demande un fournisseur CMP auprès de son magasin de fournisseur. La demande inclut le nom de matériau du fournisseur le plus récent et le numéro de version maximal qu'il connaît.

  1. Le magasin de fournisseur retourne un fournisseur CMP à partir de son stockage permanent. Si le magasin de fournisseur est un MetaStore, il obtient un fournisseur CMP chiffré encapsulé depuis sa table DynamoDB interne en utilisant le nom de matériau du fournisseur le plus récent comme clé de partition et le numéro de version comme clé de tri. Le métastore utilise son chiffreur d'élément interne et son fournisseur CMP interne pour déchiffrer le fournisseur CMP encapsulé. Ensuite, il retourne le

fournisseur CMP en texte brut au fournisseur le plus récent. Si le CMP interne est un fournisseur KMS direct (p. 14), cette étape inclut un appel de AWS Key Management Service (AWS KMS).

2. Le fournisseur CMP ajoute le champ `amzn-ddb-meta-id` à la description du matériau réel (p. 10). Sa valeur est le nom de matériau et la version du CMP dans sa table interne. Le magasin de fournisseur retourne le fournisseur CMP au fournisseur le plus récent.
3. Le fournisseur le plus récent met en cache mémoire le fournisseur CMP.
4. Le fournisseur le plus récent utilise le fournisseur CMP pour générer les matériaux de chiffrement. Puis, il retourne les matériaux de chiffrement au chiffreur d'élément.

Obtention des matériaux de déchiffrement

Quand le chiffreur d'élément demande au fournisseur le plus récent les matériaux de chiffrement, le fournisseur le plus récent utilise le processus suivant pour les obtenir et les retourner.

1. Le fournisseur le plus récent demande au magasin de fournisseur le numéro de version des matériaux de chiffrement utilisés pour chiffrer l'élément. Il transmet la description du matériau réel depuis l'attribut de description du matériau (p. 10) de l'élément.
2. Le magasin de fournisseur obtient le numéro de version du CMP en charge du chiffrement auprès du champ `amzn-ddb-meta-id` de la description du matériau réel et le retourne au fournisseur le plus récent.
3. Le fournisseur le plus récent recherche dans son cache le numéro de version du CMP utilisé pour chiffrer et signer l'élément.

- S'il trouve la version correspondante du fournisseur CMP dans le cache et que le CMP n'a pas dépassé leValeur de durée de vie (TTL) (p. 27), le fournisseur le plus récent utilise le fournisseur CMP pour générer les matériaux de déchiffrement. Puis, il retourne les matériaux de déchiffrement au chiffreur d'élément. Cette opération ne requiert pas d'appel au magasin de fournisseur ou à un autre fournisseur CMP.
- Si la version correspondante du fournisseur CMP n'est pas dans le cache, ou si le cacheAWS KMS keya dépassé sa valeur TTL, le fournisseur le plus récent demande un fournisseur CMP auprès de son magasin de fournisseur. Il envoie le nom de matériau et le numéro de version du CMP de chiffrement dans la demande.

  1. Le magasin de fournisseur recherche dans le stockage permanent le fournisseur CMP en utilisant le nom du fournisseur le plus récent comme clé de partition et le numéro de version comme clé de tri.
     - Si le nom et le numéro de version ne sont pas dans le stockage permanent, le magasin de fournisseur lève une exception. Si le magasin de fournisseur a été utilisé pour générer le CMP, celui-ci doit être stocké dans son stockage permanent, à moins qu'il ne soit intentionnellement supprimé.
     - Si le CMP avec le nom et le numéro de version correspondants se trouve dans le stockage permanent du magasin de fournisseur, celui-ci retourne le CMP spécifié au fournisseur le plus récent.

       Si le magasin de fournisseur est un MetaStore, il obtient le fournisseur CMP chiffré auprès de sa table DynamoDB. Puis, il utilise les matériaux de chiffrement de son fournisseur CMP interne pour déchiffrer le CMP chiffré avant de retourner le fournisseur CMP au fournisseur le plus récent. Si le CMP interne est un fournisseur KMS direct (p. 14), cette étape inclut un appel de AWS Key Management Service (AWS KMS).
  2. Le fournisseur le plus récent met en cache mémoire le fournisseur CMP.
  3. Le fournisseur le plus récent utilise le fournisseur CMP pour générer les matériaux de déchiffrement. Puis, il retourne les matériaux de déchiffrement au chiffreur d'élément.

Mises à jour du fournisseur le plus récent

Le symbole du fournisseur le plus récent est remplacé par`MostRecentProvider`pour`CachingMostRecentProvider`.

> **Note**
>
> Le`MostRecentProvider`, qui représente le fournisseur le plus récent, est obsolète dans la version 1.15 du client de chiffrement DynamoDB pour Java et la version 1.3 de DynamoDB Encryption Client for Python et supprimé des versions 2.0.0 du client DynamoDB Encryption Client dans les deux implémentations linguistiques. À la place, utilisez le`CachingMostRecentProvider`.

Le`CachingMostRecentProvider`implémente les modifications suivantes :

- Le`CachingMostRecentProvider`supprime périodiquement les matériaux cryptographiques de la mémoire lorsque leur temps de mémoire dépasse la valeur configuréeValeur de durée de vie (TTL) (p. 27).

  Le`MostRecentProvider`peut stocker du matériel cryptographique en mémoire pendant toute la durée de vie du processus. Par conséquent, le fournisseur le plus récent peut ne pas être au courant des changements d'autorisation. Il peut utiliser des clés de chiffrement une fois que les autorisations de l'appelant pour les utiliser ont été révoquées.

  Si vous ne pouvez pas mettre à jour cette nouvelle version, vous pouvez obtenir un effet similaire en appelant périodiquement le`clear()`sur le cache. Cette méthode vide manuellement le contenu du cache et exige que le fournisseur le plus récent demande un nouveau CMP et de nouveaux matériaux cryptographiques.
- Le`CachingMostRecentProvider`inclut également un paramètre de taille de cache qui vous donne plus de contrôle sur le cache.

Pour mettre à jour le`CachingMostRecentProvider`, vous devez modifier le nom du symbole dans votre code. À tous les autres égards, le`CachingMostRecentProvider`est entièrement rétrocompatible avec le`MostRecentProvider`. Vous n'avez pas besoin de chiffrer les éléments de table.

Cependant, le`CachingMostRecentProvider`génère davantage d'appels vers l'infrastructure clé sous-jacente. Il appelle le magasin de fournisseur au moins une fois par intervalle de durée de vie (TTL). Les applications avec de nombreux CMP actifs (en raison de la rotation fréquente) ou les applications avec de grandes flottes sont les plus susceptibles d'être sensibles à ce changement.

Avant de publier votre code mis à jour, testez-le minutieusement pour vous assurer que les appels les plus fréquents n'affectent pas votre application ou ne provoquent pas de limitation des services dont dépend votre fournisseur, tels queAWS Key Management Service(AWS KMS) ou Amazon DynamoDB. Pour atténuer les problèmes de performances, ajustez la taille du cache et la durée de vie du`CachingMostRecentProvider`en fonction des caractéristiques de performance que vous observez. Pour de plus amples informations, consultez Définition d'une valeur de durée de vie (p. 27).

Fournisseur de matériaux statique

LeFournisseur de matériaux statique(CMP statique) est unfournisseur de matériaux de chiffrement (p. 9)(CMP) destiné aux tests, aux démonstrations preuve de concept et à la compatibilité de l'existant.

Pour utiliser le fournisseur CMP statique afin de chiffrer un élément de table, vous fournissez une clé de chiffrement symétrique AES (Advanced Encryption Standard) et une clé ou paire de clés de signature.

Vous devez fournir les mêmes clés pour déchiffrer l'élément chiffré. Le fournisseur CMP statique n'assure aucune opération de chiffrement. Au lieu de cela, il transmet inchangées les clés de chiffrement que vous fournissez au chiffreur d'élément. Le chiffreur d'élément chiffre les éléments directement sous la clé de chiffrement. Puis, il utilise directement la clé de signature pour les signer.

Comme le fournisseur CMP statique ne génère pas de matériau de chiffrement unique, tous les éléments de table que vous traitez sont chiffrés avec la même clé de chiffrement et signés par la même clé de signature. Lorsque vous utilisez la même clé pour chiffrer les valeurs d'attribut de nombreux éléments, ou que vous utilisez la même clé ou paire de clés pour signer tous les éléments, vous risquez de dépasser les limites de chiffrement des clés.

> Note
>
> Le fournisseur statique asymétrique de la bibliothèque Java n'est pas un fournisseur statique. Il fournit juste d'autres constructeurs au fournisseur CMP encapsulé (p. 19). Il est sûr pour une utilisation en production, mais vous devez utiliser directement le CMP encapsulé chaque fois que possible.

Le CMP statique est l'un des nombreuxfournisseurs de matériaux de chiffrement (p. 9)(CPM) pris en charge par le client de chiffrement DynamoDB. Pour plus d'informations sur les autres CMP, consultez Comment choisir un fournisseur CMP (p. 13).

Pour obtenir un exemple de code, consultez :

- Java: SymmetricEncryptedItem

Rubriques

- Comment l'utiliser (p. 32)
- Fonctionnement (p. 33)

Comment l'utiliser

Pour créer un fournisseur statique, fournissez une clé ou paire de clés de chiffrement, et une clé ou paire de clés de signature. Vous devez fournir le matériau de clé pour chiffrer et déchiffrer les éléments de table.

Java

```
// To encrypt
SecretKey cek = ...;        // Encryption key
SecretKey macKey =  ...;    // Signing key
EncryptionMaterialsProvider provider = new SymmetricStaticProvider(cek, macKey);

// To decrypt
SecretKey cek = ...;        // Encryption key
SecretKey macKey =  ...;    // Verification key
EncryptionMaterialsProvider provider = new SymmetricStaticProvider(cek, macKey);
```

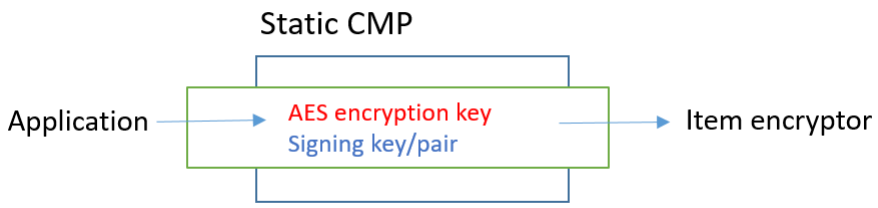
Python

```
# You can provide encryption materials, decryption materials, or both
encrypt_keys = EncryptionMaterials(
    encryption_key = ...,
    signing_key = ...
)

decrypt_keys = DecryptionMaterials(
    decryption_key = ...,
    verification_key = ...
```

```
)

static_cmp = StaticCryptographicMaterialsProvider(
    encryption_materials=encrypt_keys
    decryption_materials=decrypt_keys
)
```

Fonctionnement

Le fournisseur statique transmet les clés de chiffrement et de signature que vous fournissez au chiffreur d'élément, où elles sont utilisées directement pour chiffrer et signer vos éléments de table. À moins que vous ne fournissiez différentes clés pour chaque élément, les mêmes clés sont utilisées pour chaque élément.



Obtention des matériaux de chiffrement

Cette section décrit en détail les entrées, les sorties et le traitement du fournisseur CMP statique quand il reçoit une demande de matériaux de chiffrement.

Entrée (depuis l'application)

- Clé de chiffrement : il doit s'agir d'une clé symétrique, telle qu'une clé de chiffrementNorme de chiffrement avancée(AES).
- Clé de signature : il peut s'agir d'une clé symétrique ou d'une key pair asymétrique.

Entrée (depuis le chiffreur d'élément)

- Client de chiffrement DynamoDB (p. 11)

Sortie (vers le chiffreur d'élément)

- Clé de chiffrement transmise comme entrée.
- Clé de signature transmise comme entrée.
- Description du matériau réel : LeDescription du matériau demandé (p. 10), le cas échéant, inchangé.

Obtention des matériaux de déchiffrement

Cette section décrit en détail les entrées, les sorties et le traitement du fournisseur CMP statique quand il reçoit une demande de matériaux de déchiffrement.

Même s'il comporte des méthodes distinctes pour l'obtention des matériaux de chiffrement et celle des matériaux de déchiffrement, le comportement est le même.

Entrée (depuis l'application)

- Clé de chiffrement : il doit s'agir d'une clé symétrique, telle qu'une clé de chiffrementNorme de chiffrement avancée(AES).
- Clé de signature : il peut s'agir d'une clé symétrique ou d'une key pair asymétrique.

Entrée (depuis le chiffreur d'élément)

- Client de chiffrement DynamoDB (p. 11)(non utilisé)

Sortie (vers le chiffreur d'élément)

- Clé de chiffrement transmise comme entrée.
- Clé de signature transmise comme entrée.

Langages de programmation disponibles pour Amazon DynamoDB Encryption Client

Le kit Amazon DynamoDB Encryption Client est disponible pour les langages de programmation suivants. Les bibliothèques spécifiques aux langues varient, mais les implémentations qui en résultent sont interopérables. Par exemple, vous pouvez chiffrer (et signer) un élément avec le client Java et le déchiffrer avec le client Python.

Pour plus d'informations, consultez la rubrique correspondante.

Rubriques

- Client de chiffrement Amazon DynamoDB pour Java (p. 35)
- Client de chiffrement DynamoDB pour Python (p. 45)

Client de chiffrement Amazon DynamoDB pour Java

Cette rubrique explique comment installer et utiliser le kit Amazon DynamoDB Encryption Client pour Java. Pour plus d'informations sur la programmation avec DynamoDB Encryption Client, consultez leExemples Java (p. 40), leexemplesdans leaws-dynamodb-encryptionDépôt -java surGitHub, et leJavadocpour le client de chiffrement DynamoDB.

> Note
>
> Client de chiffrement DynamoDB pour Java versions 1.h/24, j/7.h/24, j/7sont enend-of-supportphase (p. 2)à compter de juillet 2022. Passez à une version plus récente dès que possible.

Rubriques

- Prérequis (p. 35)
- Installation (p. 36)
- Utilisation du client de chiffrement DynamoDB pour Java (p. 36)
- Exemple de code pour le client de chiffrement DynamoDB pour Java (p. 40)

Prérequis

Avant d'installer le kit Amazon DynamoDB Encryption Client pour Java, veillez à remplir les conditions prérequises suivantes.

Environnement de développement Java

Vous aurez besoin de Java 8 ou version ultérieure. Sur le site web d'Oracle, consultez la page Téléchargements Java SE, puis téléchargez et installez le kit Java SE Development (JDK).

Si vous utilisez le kit JDK Oracle, vous devez également télécharger et installer les fichiers Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy.

AWS SDK for Java

Le client DynamoDB Encryption Client nécessite le module DynamoDB duAWS SDK for Javamême si votre application n'interagit pas avec DynamoDB. Vous pouvez installer la totalité du kit SDK ou le seul module. Si vous utilisez Maven, ajoutez `aws-java-sdk-dynamodb` à votre fichier `pom.xml`.

Pour plus d'informations sur l'installation et la configuration du kit AWS SDK for Java, veuillez consulter AWS SDK for Java.

Installation

Vous pouvez installer le kit Amazon DynamoDB Encryption Client pour Java comme suit.

Manuellement

Pour installer le kit Amazon DynamoDB Encryption Client pour Java, clonez ou téléchargez le kitaws-dynamodb-encryptionjava GitHubrepository.

Utilisation d'Apache Maven

Le kit Amazon DynamoDB Encryption Client pour Java est disponible viaApache Mavenavec la définition de dépendance suivante.

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-dynamodb-encryption-java</artifactId>
  <version>version-number</version>
</dependency>
```

Après avoir installé le kit SDK, commencez en regardant l'exemple de code du présent guide et la documentationJavadoc Client de chiffrement DynamoDBsurGitHub.

Utilisation du client de chiffrement DynamoDB pour Java

Cette rubrique explique certaines fonctionnalités du client de chiffrement DynamoDB en Java qui peuvent ne pas être disponibles dans d'autres implémentations de langages de programmation.

Pour plus de détails sur la programmation avec DynamoDB Encryption Client, consultez leExemples Java (p. 40), leexemplesdans le`aws-dynamodb-encryption-java repository`sur GitHub, et leJavadocpour le client de chiffrement DynamoDB.

Rubriques

- Chiffreurs d'éléments : AttributeEncryptor et DynamoDBEncryptor (p. 36)
- Configuration du comportement d'enregistrement (p. 37)
- Actions d'attribut en Java (p. 37)
- Remplacer les noms des tables (p. 39)

Chiffreurs d'éléments : AttributeEncryptor et DynamoDBEncryptor

Le client de chiffrement DynamoDB en Java possède deuxéléments (p. 9): le niveau inférieurDynamoDBEncryptoret l'AttributeEncryptor (p. 36).

Le`AttributeEncryptor`est une classe d'annotations qui vous permet d'utiliser leDynamoDBMapper DynamoDdans leAWS SDK for Javaavec le`DynamoDB Encryptor`dans DynamoDB Encryption Client. Lorsque vous utilisez le `AttributeEncryptor` avec le `DynamoDBMapper`, il chiffre et signe vos éléments de manière transparente lorsque vous les enregistrez. Il vérifie et déchiffre également vos éléments de manière transparente lorsque vous les chargez.

Configuration du comportement d'enregistrement

Vous pouvez utiliser le plugin`AttributeEncryptor`et`DynamoDBMapper`pour ajouter ou remplacer des éléments de table avec des attributs signés uniquement ou chiffrés et signés. Pour ces tâches, nous vous recommandons de le configurer pour utiliser le comportement d'enregistrement `PUT`, comme illustré dans l'exemple suivant. Sinon, il est possible que vous ne puissiez pas déchiffrer vos données.

```
DynamoDBMapperConfig mapperConfig =
 DynamoDBMapperConfig.builder().withSaveBehavior(SaveBehavior.PUT).build();
DynamoDBMapper mapper = new DynamoDBMapper(ddb, mapperConfig, new
 AttributeEncryptor(encryptor));
```

Si vous utilisez le comportement d'enregistrement par défaut, qui met à jour uniquement les attributs modélisés dans l'élément de table, les attributs qui ne sont pas modélisés ne sont pas inclus dans la signature et ne sont pas modifiés par les écritures de table. Par conséquent, lors des lectures ultérieures de tous les attributs, la signature ne sera pas validée, car elle n'inclut pas d'attributs non modélisés.

Vous pouvez également utiliser le comportement de sauvegarde `CLOBBER`. Comportement d'enregistrement est identique au comportement d'enregistrement `PUT`, si ce n'est qu'il désactive le verrouillage optimiste et remplace l'élément dans la table.

Pour éviter les erreurs de signature, le client de chiffrement DynamoDB génère une exception d'exécution si un`AttributeEncryptor`est utilisé avec un`DynamoDBMapper`qui n'est pas configuré avec un comportement d'enregistrement de`CLOBBER`ou`PUT`.

Pour voir l'utilisation de ce code dans un exemple, veuillez consulterUtilisation du DynamoDBMapper (p. 42)et l'AwsKmsEncryptedObject.javaexemple dans le`aws-dynamodb-encryption-java`repository dans GitHub.

Actions d'attribut en Java

Les actions d'attribut (p. 9) déterminent les valeurs d'attribut chiffrées et signées, qui sont uniquement signées et qui sont ignorées. La méthode que vous utilisez pour spécifier les actions d'attribut varie selon que vous utilisez le chiffreur `DynamoDBMapper` et `AttributeEncryptor`, ou le chiffreur DynamoDBEncryptor de bas niveau.

> **Important**
>
> Après avoir utilisé vos actions d'attribut pour chiffrer vos éléments de table, l'ajouter ou la suppression d'attributs de votre modèle de données peut provoquer une erreur de validation de signature qui vous empêche de déchiffrer vos données. Pour obtenir une explication détaillée, consultez Modification de votre modèle de données (p. 54).

Actions d'attribut pour le DynamoDBMapper

Quand vous utilisez les `DynamoDBMapper` et `AttributeEncryptor`, vous utilisez les annotations pour spécifier les actions d'attribut. Le client de chiffrement DynamoDB utilise leannotations d'attributs DynamoDB standardqui définissent le type d'attribut pour déterminer comment protéger un attribut. Par défaut, tous les attributs sont chiffrés et signés à l'exception des clés primaires, qui sont signées, mais pas chiffrées.

> **Note**
>
> Ne chiffrez pas la valeur des attributs avec le@DynamoDBVersionAttributeannotation, même si vous pouvez (et devez) les signer. Sinon, les conditions qui utilisent sa valeur auront des effets inattendus.

```
// Attributes are encrypted and signed
@DynamoDBAttribute(attributeName="Description")

// Partition keys are signed but not encrypted
@DynamoDBHashKey(attributeName="Title")

// Sort keys are signed but not encrypted
@DynamoDBRangeKey(attributeName="Author")
```

Pour spécifier des exceptions, vous utilisez les annotations de déchiffrement définies dans DynamoDB Encryption Client pour Java. Si vous les spécifiez au niveau classe, elles deviennent la valeur par défaut pour la classe.

```
// Sign only
@DoNotEncrypt

// Do nothing; not encrypted or signed
@DoNotTouch
```

Par exemple, ces annotations signent mais ne chiffrent pas l'attribut `PublicationYear`, et ni ne chiffrent ou ne signent la valeur d'attribut `ISBN`.

```
// Sign only (override the default)
@DoNotEncrypt
@DynamoDBAttribute(attributeName="PublicationYear")

// Do nothing (override the default)
@DoNotTouch
@DynamoDBAttribute(attributeName="ISBN")
```

Actions d'attribut pour DynamoDBEncryptor

Pour spécifier les actions d'attribut lorsque vous utilisez directement le chiffreur DynamoDBEncryptor, créez un objet `HashMap` dans lequel les paires nom-valeur représentent les noms d'attribut et les actions spécifiées.

Les valeurs valides des actions d'attribut sont définies dans le type énuméré `EncryptionFlags`. Vous pouvez utiliser `ENCRYPT` et `SIGN` conjointement, utiliser `SIGN` seul, ou omettre les deux. Toutefois, si vous utilisez`ENCRYPT`seul, le client de chiffrement DynamoDB génère une erreur. Vous ne pouvez pas chiffrer un attribut que vous ne signez pas.

```
ENCRYPT
SIGN
```

> **Warning**
>
> Ne chiffrez pas les attributs de la clé primaire. Ceux-ci doivent rester en texte brut pour que DynamoDB puisse trouver l'élément sans exécuter une analyse complète de la table.

Si vous spécifiez une clé primaire du contexte de chiffrement, puis spécifiez`ENCRYPT`dans l'action d'attribut pour l'attribut de clé primaire, le client de chiffrement DynamoDB lève une exception

Par exemple, le code Java suivant crée un`actions` HashMap qui chiffre et signe tous les attributs dans le`record`élément. Les exceptions sont les attributs de clé de partition et de clé de tri, qui sont signés mais non chiffrés, et l'attribut `test`, qui n'est ni signé ni chiffré.

```
final EnumSet<EncryptionFlags> signOnly = EnumSet.of(EncryptionFlags.SIGN);
final EnumSet<EncryptionFlags> encryptAndSign = EnumSet.of(EncryptionFlags.ENCRYPT,
 EncryptionFlags.SIGN);
final Map<String, Set<EncryptionFlags>> actions = new HashMap<>();

for (final String attributeName : record.keySet()) {
  switch (attributeName) {
    case partitionKeyName: // no break; falls through to next case
    case sortKeyName:
      // Partition and sort keys must not be encrypted, but should be signed
      actions.put(attributeName, signOnly);
      break;
    case "test":
      // Don't encrypt or sign
      break;
    default:
      // Encrypt and sign everything else
      actions.put(attributeName, encryptAndSign);
      break;
  }
}
```

Puis, quand vous appelez la méthode encryptRecord de `DynamoDBEncryptor`, spécifiez la map comme valeur du paramètre `attributeFlags`. Par exemple, cet appel d'`encryptRecord` utilise la map `actions`.

```
// Encrypt the plaintext record
final Map<String, AttributeValue> encrypted_record = encryptor.encryptRecord(record,
 actions, encryptionContext);
```

Remplacer les noms des tables

Dans le client de chiffrement DynamoDB, le nom de la table DynamoDB est un élément duClient de chiffrement DynamoDB (p. 11)qui est transmis aux méthodes de chiffrement et de déchiffrement. Lorsque vous chiffrez ou signez des éléments de table, le contexte de chiffrement DynamoDB, y compris le nom de la table, est lié cryptographiquement au texte chiffré. Si le contexte de chiffrement DynamoDB transmis à la méthode de déchiffrement ne correspond pas au contexte de chiffrement DynamoDB transmis à la méthode de chiffrement, l'opération de déchiffrement échoue.

Parfois, le nom d'une table change, par exemple lorsque vous sauvegardez une table ou effectuez unepoint-in-time récupération. Lorsque vous déchiffrez ou vérifiez la signature de ces éléments, vous devez passer dans le même contexte de chiffrement DynamoDB que celui utilisé pour chiffrer et signer les éléments, y compris le nom de la table d'origine. Le nom de la table actuelle n'est pas nécessaire.

Lorsque vous utilisez le`DynamoDBEncryptor`, vous assemblez le contexte de chiffrement DynamoDB manuellement. Toutefois, si vous utilisez le`DynamoDBMapper`, le`AttributeEncryptor`crée le contexte de chiffrement DynamoDB pour vous, y compris le nom de la table actuelle. Pour indiquer à `AttributeEncryptor` de créer un contexte de chiffrement avec un nom de table différent, utilisez le `EncryptionContextOverrideOperator`.

Par exemple, le code suivant crée des instances du fournisseur de matériaux cryptographiques (CMP) et du `DynamoDBEncryptor`. Ensuite, il appelle la méthode `setEncryptionContextOverrideOperator` de `DynamoDBEncryptor`. Il utilise l'opérateur `overrideEncryptionContextTableName`, qui remplace un nom de table. Lorsqu'il est configuré de cette façon, le`AttributeEncryptor`crée un contexte de chiffrement DynamoDB qui inclut`newTableName`à la place de`oldTableName`. Pour obtenir un exemple complet, veuillez consulterEncryptionContextOverridesWithDynamoDBMapper.java.

```
final DirectKmsMaterialProvider cmp = new DirectKmsMaterialProvider(kms, keyArn);
final DynamoDBEncryptor encryptor = DynamoDBEncryptor.getInstance(cmp);

encryptor.setEncryptionContextOverrideOperator(EncryptionContextOperators.overrideEncryptionContextTabl
                oldTableName, newTableName));
```

Lorsque vous appelez la méthode de chargement de `DynamoDBMapper`, qui déchiffre et vérifie l'élément, vous spécifiez le nom de la table d'origine.

```
mapper.load(itemClass, DynamoDBMapperConfig.builder()

 .withTableNameOverride(DynamoDBMapperConfig.TableNameOverride.withTableNameReplacement(oldTableName))
                .build());
```

Vous pouvez également utiliser l'opérateur `overrideEncryptionContextTableNameUsingMap`, qui remplace plusieurs noms de table.

Les opérateurs de remplacement de nom de table sont généralement utilisés lors du déchiffrement des données et de la vérification des signatures. Toutefois, vous pouvez les utiliser pour définir le nom de la table dans le contexte de chiffrement DynamoDB sur une valeur différente lors du chiffrement et de la signature.

N'utilisez pas les opérateurs de remplacement de nom de table si vous utilisez le `DynamoDBEncryptor`. Créez plutôt un contexte de chiffrement avec le nom de la table d'origine et soumettez-le à la méthode de déchiffrement.

Exemple de code pour le client de chiffrement DynamoDB pour Java

Les exemples suivants vous montrent comment utiliser le client de chiffrement DynamoDB pour Java afin de protéger les éléments de table DynamoDB dans votre application. Vous pouvez trouver d'autres exemples (et contribuer les vôtres) dans le répertoire exemples du référentiel aws-dynamodb-encryption-java sur GitHub.

Rubriques

- Utilisation de DynamoDBEncryptor (p. 40)
- Utilisation du DynamoDBMapper (p. 42)

Utilisation de DynamoDBEncryptor

Cet exemple montre comment utiliser le chiffreur de bas niveau DynamoDBEncryptor avec le fournisseur KMS direct (p. 14). Le fournisseur KMS direct génère et protège ses matériaux cryptographiques sous unAWS KMS keydansAWS Key Management Service(AWS KMS) que vous spécifiez.

Vous pouvez utiliser tout fournisseur CMP (p. 9) compatible avec le chiffreur `DynamoDBEncryptor`, et vous pouvez utiliser le fournisseur KMS direct avec `DynamoDBMapper` et AttributeEncryptor (p. 36).

Voir l'exemple de code complet : AwsKmsEncryptedItem.java

Étape 1 : Créer le fournisseur KMS direct

Créez une instance du client AWS KMS avec la région spécifiée. Puis, utilisez l'instance cliente pour créer une instance du fournisseur KMS direct avec votre préférée.AWS KMS key.

Cet exemple utilise Amazon Resource Name (ARN) pour identifier leAWS KMS key, mais vous pouvez utiliserIdentifiant de clé valide.

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

final AWSKMS kms = AWSKMSClientBuilder.standard().withRegion(region).build();
final DirectKmsMaterialProvider cmp = new DirectKmsMaterialProvider(kms, keyArn);
```

Étape 2 : Créer un élément

Cet exemple définit un `record` HashMap qui représente un exemple d'élément de table.

```
final String partitionKeyName = "partition_attribute";
final String sortKeyName = "sort_attribute";

final Map<String, AttributeValue> record = new HashMap<>();
record.put(partitionKeyName, new AttributeValue().withS("value1"));
record.put(sortKeyName, new AttributeValue().withN("55"));
record.put("example", new AttributeValue().withS("data"));
record.put("numbers", new AttributeValue().withN("99"));
record.put("binary", new AttributeValue().withB(ByteBuffer.wrap(new byte[]{0x00, 0x01,
 0x02})));
record.put("test", new AttributeValue().withS("test-value"));
```

Étape 3 : Créer un DynamoDBEncryptor

Créez une instance de `DynamoDBEncryptor` avec le fournisseur KMS direct.

```
final DynamoDBEncryptor encryptor = DynamoDBEncryptor.getInstance(cmp);
```

Étape 4 : Créer un contexte de chiffrement DynamoDB

LeClient de chiffrement DynamoDB (p. 11)contient des informations sur la structure de la table et sur la façon dont elle est chiffrée et signée. Si vous utilisez `DynamoDBMapper`, `AttributeEncryptor` crée automatiquement le contexte de chiffrement.

```
final String tableName = "testTable";

final EncryptionContext encryptionContext = new EncryptionContext.Builder()
    .withTableName(tableName)
    .withHashKeyName(partitionKeyName)
    .withRangeKeyName(sortKeyName)
    .build();
```

Étape 5 : Créer l'objet actions d'attribut

Les actions d'attribut (p. 9) déterminent quels attributs de l'élément sont chiffrés et signés, lesquels sont uniquement signés et lesquels ne sont ni chiffrés ni signés.

En Java, pour spécifier les actions d'attribut, vous créez un HashMap du nom d'attribut et les paires de valeurs `EncryptionFlags`.

Par exemple, le code Java suivant crée un HashMap `actions` qui chiffre et signe tous les attributs de l'élément `record`, à l'exception des attributs de clé de partition et de clé de tri, qui sont signés mais pas chiffrés, et de l'attribut `test`, qui n'est ni signé ni chiffré.

```
final EnumSet<EncryptionFlags> signOnly = EnumSet.of(EncryptionFlags.SIGN);
final EnumSet<EncryptionFlags> encryptAndSign = EnumSet.of(EncryptionFlags.ENCRYPT,
 EncryptionFlags.SIGN);
```

```
final Map<String, Set<EncryptionFlags>> actions = new HashMap<>();

for (final String attributeName : record.keySet()) {
  switch (attributeName) {
    case partitionKeyName: // fall through to the next case
    case sortKeyName:
      // Partition and sort keys must not be encrypted, but should be signed
      actions.put(attributeName, signOnly);
      break;
    case "test":
      // Neither encrypted nor signed
      break;
    default:
      // Encrypt and sign all other attributes
      actions.put(attributeName, encryptAndSign);
      break;
  }
}
```

Étape 6 : Chiffrer et signer l'élément

Pour chiffrer et signer l'élément de table, appelez la méthode `encryptRecord` sur l'instance de `DynamoDBEncryptor`. Spécifiez l'élément de table (`record`), les actions d'attribut (`actions`) et le contexte de chiffrement (`encryptionContext`).

```
final Map<String, AttributeValue> encrypted_record = encryptor.encryptRecord(record,
 actions, encryptionContext);
```

Étape 7 : Placer l'élément dans la table DynamoDB

Enfin, placez l'élément chiffré et signé dans la table DynamoDB.

```
final AmazonDynamoDB ddb = AmazonDynamoDBClientBuilder.defaultClient();
ddb.putItem(tableName, encrypted_record);
```

Utilisation du DynamoDBMapper

L'exemple suivant montre comment utiliser la classe d'annotations de l'outil de mappage DynamoDB avec leFournisseur KMS direct (p. 14). Le fournisseur KMS direct génère et protège ses matériaux cryptographiques sous unAWS KMS keydansAWS Key Management Service(AWS KMS) que vous spécifiez.

Vous pouvez utiliser tout fournisseur CMP (p. 9) compatible avec `DynamoDBMapper`, et vous pouvez utiliser le fournisseur KMS direct avec le `DynamoDBEncryptor` de bas niveau.

Voir l'exemple de code complet : AwsKmsEncryptedObject.java

Étape 1 : Créer le fournisseur KMS direct

Créez une instance du client AWS KMS avec la région spécifiée. Puis, utilisez l'instance cliente pour créer une instance du fournisseur KMS direct avec votre préférée.AWS KMS key.

Cet exemple utilise Amazon Resource Name (ARN) pour identifier leAWS KMS key, mais vous pouvez utiliserIdentifiant de clé valide.

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

```
final AWSKMS kms = AWSKMSClientBuilder.standard().withRegion(region).build();
final DirectKmsMaterialProvider cmp = new DirectKmsMaterialProvider(kms, keyArn);
```

Étape 2 : Créer le chiffreur DynamoDB et l'outil de mappage et DynamoDBMapper

Utilisez le fournisseur KMS direct que vous avez créé à l'étape précédente pour créer une instance duChiffreur DynamoDB (p. 36). Vous devez instancier le chiffreur DynamoDB de bas niveau pour utiliser le mappage DynamoDB.

Créez ensuite une instance de votre base de données DynamoDB et une configuration de mappeur. Utilisez-les pour créer une instance du mappage DynamoDB.

> Important
>
> Lorsque vous utilisez `DynamoDBMapper` pour ajouter ou modifier des éléments signés (ou chiffrés et signés), configurez-le pour qu'il utilise un comportement d'enregistrement (p. 37), par exemple `PUT`, qui inclut tous les attributs, comme illustré dans l'exemple suivant. Sinon, il est possible que vous ne puissiez pas déchiffrer vos données.

```
final DynamoDBEncryptor encryptor = DynamoDBEncryptor.getInstance(cmp)
final AmazonDynamoDB ddb =
 AmazonDynamoDBClientBuilder.standard().withRegion(region).build();

DynamoDBMapperConfig mapperConfig =
 DynamoDBMapperConfig.builder().withSaveBehavior(SaveBehavior.PUT).build();
DynamoDBMapper mapper = new DynamoDBMapper(ddb, mapperConfig, new
 AttributeEncryptor(encryptor));
```

Étape 3 : Définissez votre table DynamoDB

Définissez ensuite votre table DynamoDB. Utilisez des annotations pour spécifier les actions d'attribut (p. 37). Cet exemple crée une table DynamoDB.`ExampleTable`, et un`DataPoJo`classe qui représente des éléments de table.

Dans cet exemple de table, les attributs de la clé primaire seront signés, mais pas chiffrés. Cela s'applique à l'attribut `partition_attribute`, qui est annoté avec `@DynamoDBHashKey`, et l'attribut `sort_attribute`, qui est annoté avec `@DynamoDBRangeKey`.

Les attributs qui sont annotés avec `@DynamoDBAttribute`, par exemple `some numbers`, seront chiffrés et signés. Les exceptions sont des attributs qui utilisent le`@DoNotEncrypt`(signe uniquement) ou`@DoNotTouch`(Ne pas chiffrer ou signer) annotation de chiffrement définies par le client de chiffrement DynamoDB. Par exemple, étant donné que l'attribut `leave me` comporte une annotation `@DoNotTouch`, il ne sera ni chiffré ni signé.

```
@DynamoDBTable(tableName = "ExampleTable")
public static final class DataPoJo {
  private String partitionAttribute;
  private int sortAttribute;
  private String example;
  private long someNumbers;
  private byte[] someBinary;
  private String leaveMe;

  @DynamoDBHashKey(attributeName = "partition_attribute")
  public String getPartitionAttribute() {
    return partitionAttribute;
  }

  public void setPartitionAttribute(String partitionAttribute) {
```

```
    this.partitionAttribute = partitionAttribute;
  }

  @DynamoDBRangeKey(attributeName = "sort_attribute")
  public int getSortAttribute() {
    return sortAttribute;
  }

  public void setSortAttribute(int sortAttribute) {
    this.sortAttribute = sortAttribute;
  }

  @DynamoDBAttribute(attributeName = "example")
  public String getExample() {
    return example;
  }

  public void setExample(String example) {
    this.example = example;
  }

  @DynamoDBAttribute(attributeName = "some numbers")
  public long getSomeNumbers() {
    return someNumbers;
  }

  public void setSomeNumbers(long someNumbers) {
    this.someNumbers = someNumbers;
  }

  @DynamoDBAttribute(attributeName = "and some binary")
  public byte[] getSomeBinary() {
    return someBinary;
  }

  public void setSomeBinary(byte[] someBinary) {
    this.someBinary = someBinary;
  }

  @DynamoDBAttribute(attributeName = "leave me")
  @DoNotTouch
  public String getLeaveMe() {
    return leaveMe;
  }

  public void setLeaveMe(String leaveMe) {
    this.leaveMe = leaveMe;
  }

  @Override
  public String toString() {
    return "DataPoJo [partitionAttribute=" + partitionAttribute + ", sortAttribute="
        + sortAttribute + ", example=" + example + ", someNumbers=" + someNumbers
        + ", someBinary=" + Arrays.toString(someBinary) + ", leaveMe=" + leaveMe + "]";
  }
}
```

Étape 4 : Chiffrer et enregistrer un élément de table

Désormais, lorsque vous créez un élément de table et que vous utilisez l'outil de mappage DynamoDB pour l'enregistrer, l'élément est automatiquement chiffré et signé avant d'être ajouté à la table.

Cet exemple définit un élément de table appelé `record`. Avant qu'il soit enregistré dans la table, ses attributs sont chiffrées et signés selon les annotations de la classe `DataPoJo`. Dans le cas présent, tous les attributs à l'exception de `PartitionAttribute`, `SortAttribute` et `LeaveMe` sont chiffrés

et signés. `PartitionAttribute` et `SortAttributes` sont seulement signés. L'attribut `LeaveMe` n'est ni chiffré ni signé.

Pour chiffrer et signer l'élément `record`, puis l'ajouter à `ExampleTable`, appelez la méthode `save` de la classe `DynamoDBMapper`. Parce que votre mappeur DynamoDB est configuré pour utiliser le`PUT`Enregistrer le comportement, l'élément remplace n'importe quel élément par les mêmes clés principales, au lieu de le mettre à jour. Cela garantit que les signatures correspondent et que vous pouvez déchiffrer l'élément lorsque vous le récupérez de la table.

```
DataPoJo record = new DataPoJo();
record.setPartitionAttribute("is this");
record.setSortAttribute(55);
record.setExample("data");
record.setSomeNumbers(99);
record.setSomeBinary(new byte[]{0x00, 0x01, 0x02});
record.setLeaveMe("alone");

mapper.save(record);
```

Client de chiffrement DynamoDB pour Python

Cette rubrique explique comment installer et utiliser le kit DynamoDB Encryption Client pour Python. Vous pouvez trouver le code dans le kitaws-dynamodb-encryption-Pythonrepository surGitHub, y compris complet et testéexemple de codepour vous aider à démarrer.

> Note
>
> Client de chiffrement DynamoDB pour Python versions 1.h/24, j/7.h/24, j/7et 2.h/24, j/7.h/24, j/7sont enend-of-supportphase (p. 2)à compter de juillet 2022. Mettez à niveau vers une version plus récente dès que possible.

Rubriques

Prérequis

Avant d'installer le kit Amazon DynamoDB Encryption Client pour Python, veillez à remplir les conditions prérequises suivantes.

Version prise en charge de Python

Python 3.6 ou version ultérieure est requis par Amazon DynamoDB Encryption Client pour Python versions 3.1.0 et ultérieures. Pour télécharger Python, consultez Téléchargements Python.

Les versions antérieures d'Amazon DynamoDB Encryption Client for Python prennent en charge Python 2.7 et Python 3.4 et versions ultérieures, mais nous vous recommandons d'utiliser la dernière version de DynamoDB Encryption Client.

Outil d'installation pip pour Python

Python 3.6 et versions ultérieures incluentpip, bien que vous souhaitiez peut-être le mettre à niveau. Pour plus d'informations sur la mise à niveau ou l'installation de pip, consultez Installation dans la documentation pip.

Installation

Utiliserpippour installer le kit Amazon DynamoDB Encryption Client pour Python, comme illustré dans les exemples suivants.

Pour installer la dernière version

```
pip install dynamodb-encryption-sdk
```

Pour plus d'informations sur l'utilisation de pip pour installer et mettre à niveau les packages, consultez Installing Packages.

Le client de chiffrement DynamoDB nécessite le kitbibliothèque de chiffrementsur toutes les plateformes. Toutes les versions de pip installent et créent la bibliothèque de chiffrement sous Windows. pip 8.1 et les versions ultérieures installent et créent la bibliothèque de chiffrement sous Linux. Si vous utilisez une version antérieure de pip et que votre environnement Linux ne possède pas les outils nécessaires pour générer la bibliothèque de chiffrement, vous devez les installer. Pour plus d'informations, consultez Création du chiffrement sous Linux.

Vous pouvez obtenir la dernière version de développement du client de chiffrement DynamoDB depuis le kitaws-dynamodb-encryption-Pythonrepository surGitHub.

Après avoir installé le kit DynamoDB Encryption Client, commencez en regardant l'exemple de code Python du présent guide.

Utilisation du client de chiffrement DynamoDB pour Python

Cette rubrique explique certaines fonctionnalités du Client de chiffrement DynamoDB pour Python qui peuvent ne pas être disponibles dans d'autres implémentations de langages de programmation. Ces fonctions sont conçues pour faciliter l'utilisation du client de chiffrement DynamoDB de la manière la plus sécurisée. À moins que vous n'ayez un scénario inhabituel, nous vous recommandons de les utiliser.

Pour plus d'informations sur la programmation avec DynamoDB Encryption Client, consultez leExemples Python (p. 48)dans ce guide, leexemplesdans le référentiel aws-dynamodb-encryption-python sur GitHub, et leDocumentation Pythonpour DynamoDB Encryption Client.

Rubriques

- Classes d'annotations clientes (p. 46)
- Classe TableInfo (p. 47)
- Actions d'attribut en Python (p. 47)

Classes d'annotations clientes

DynamoDB Encryption Client for Python inclut plusieurs classes d'antécédents qui reflètent les classes Boto 3 pour DynamoDB. Ces classes d'antécédents sont conçues pour faciliter l'ajout du chiffrement et de la signature à votre application DynamoDB existante et éviter les problèmes les plus courants, tels que :

- Vous empêcher de chiffrer la clé primaire de votre élément, en ajoutant une action de remplacement pour la clé primaire à l'objet AttributeActions (p. 47) ou en levant une exception si votre objet `AttributeActions` demande explicitement au client de chiffrer la clé primaire. Si l'action par défaut de votre objet `AttributeActions` est `DO_NOTHING`, les classes d'annotations clientes utilisent cette action pour la clé primaire. Sinon, elles utilisent `SIGN_ONLY`.

- Création d'unObjet TableInfo (p. 46)et remplissez-laClient de chiffrement DynamoDB (p. 11)basé sur un appel à DynamoDB. Cela permet de s'assurer que votre contexte de chiffrement DynamoDB est exact et que le client peut identifier la clé primaire.
- Méthodes de support, telles que`put_item`et`get_item`, qui chiffrent et déchiffrent de façon transparente vos éléments de table lorsque vous lisez à partir d'une table DynamoDB. Seule la méthode `update_item` n'est pas prise en charge.

Vous pouvez utiliser les classes d'annotations clientes au lieu d'interagir directement avec le chiffreur d'élément (p. 9) de bas niveau. Utilisez ces classes à moins que vous n'ayez besoin de définir des options avancées dans le chiffreur d'élément.

Les classes d'annotations clientes incluent les éléments suivants :

- EncryptedTablepour les applications qui utilisent leTableaudans DynamoDB pour traiter une table à la fois.
- EncryptedResourcepour les applications qui utilisent leRessource de servicedans DynamoDB pour le traitement par lots.
- EncryptedClientpour les applications qui utilisent leClient de bas niveaudans DynamoDB.

Pour utiliser les classes d'antécédents clientes, l'appelant doit avoir l'autorisation d'appeler DynamoDBDescribeTablesur la table cible.

Classe TableInfo

LeTableInfoclass est une classe d'andamoDB qui représente une table DynamoDB, avec des champs pour sa clé primaire et ses index secondaires. Elle vous permet d'obtenir des informations précises et en temps réel sur la table.

Si vous utilisez une classe d'annotations clientes (p. 46), elle crée et utilise un objet `TableInfo` pour vous. Sinon, vous pouvez en créer un explicitement. Pour voir un exemple, consultez Utilisation du chiffreur d'élément (p. 50).

Lorsque vous appelez le`refresh_indexed_attributes`sur une`TableInfo`, il renseigne les valeurs de propriété de l'objet en appelant DynamoDBDescribeTable. L'interrogation de la table est beaucoup plus fiable que le codage en dur des noms d'index. Le`TableInfo`comprend également une classe`encryption_context_values`qui fournit les valeurs requises pour laClient de chiffrement DynamoDB (p. 11).

Pour utiliser le plugin`refresh_indexed_attributes`, l'appelant doit avoir l'autorisation d'appeler DynamoDBDescribeTablesur la table cible.

Actions d'attribut en Python

Les actions d'attribut (p. 9) informent le chiffreur d'élément des actions à exécuter sur chaque attribut de l'élément. Pour spécifier les actions d'attribut en Python, créez un objet `AttributeActions` avec une action par défaut et les exceptions éventuelles pour des attributs particuliers. Les valeurs valides sont définies dans le type énuméré `CryptoAction`.

> **Important**
>
> Après avoir utilisé vos actions d'attribut pour chiffrer vos éléments de table, l'ajouter ou la suppression d'attributs de votre modèle de données peut provoquer une erreur de validation de signature qui vous empêche de déchiffrer vos données. Pour obtenir une explication détaillée, consultez Modification de votre modèle de données (p. 54).

```
DO_NOTHING = 0
```

```
SIGN_ONLY = 1
ENCRYPT_AND_SIGN = 2
```

Par exemple, l'objet `AttributeActions` établit `ENCRYPT_AND_SIGN` comme valeur par défaut pour tous les attributs, et spécifie les exceptions pour les attributs `ISBN` et `PublicationYear`.

```
actions = AttributeActions(
    default_action=CryptoAction.ENCRYPT_AND_SIGN,
    attribute_actions={
        'ISBN': CryptoAction.DO_NOTHING,
        'PublicationYear': CryptoAction.SIGN_ONLY
    }
)
```

Si vous utilisez une classe d'annotations clientes (p. 46), vous n'avez pas besoin de spécifier une action d'attribut pour les attributs de clé primaire. Les classes d'annotations clients vous empêchent de chiffrer votre clé primaire.

Si vous n'utilisez pas une classe d'annotations clientes et que l'action par défaut est `ENCRYPT_AND_SIGN`, vous devez spécifier une action pour la clé primaire. L'action recommandée pour les clés primaires est `SIGN_ONLY`. À des fins de simplification, utilisez la méthode `set_index_keys`, qui utilise SIGN_ONLY pour les clés primaires ou DO_NOTHING quand il s'agit de l'action par défaut.

> Warning
>
> Ne chiffrez pas les attributs de la clé primaire. Ils doivent rester en texte brut pour que DynamoDB puisse trouver l'élément sans exécuter une analyse complète de la table.

```
actions = AttributeActions(
    default_action=CryptoAction.ENCRYPT_AND_SIGN,
)
actions.set_index_keys(*table_info.protected_index_keys())
```

Exemple de code pour DynamoDB Encryption Client pour Python

Les exemples suivants vous montrent comment utiliser le client de chiffrement DynamoDB pour Python afin de protéger les données DynamoDB dans votre application. Vous pouvez trouver d'autres exemples (et contribuer les vôtres) dans le répertoire exemples du référentiel aws-dynamodb-encryption-python sur GitHub.

Rubriques

- Utilisation de la classe d'annotations cliente EncryptedTable (p. 48)
- Utilisation du chiffreur d'élément (p. 50)

Utilisation de la classe d'annotations cliente EncryptedTable

L'exemple suivant montre comment utiliser le fournisseur KMS direct (p. 14) avec la `EncryptedTable` classe d'annotations cliente (p. 46). Cet exemple utilise le même fournisseur CMP (p. 9) que l'exemple Utilisation du chiffreur d'élément (p. 50) qui suit. Cependant, il utilise la classe `EncryptedTable` au lieu d'interagir directement avec le chiffreur d'élément (p. 9) de bas niveau.

En comparant ces exemples, vous pouvez voir le travail que la classe d'annotations cliente effectue à votre place. Cela inclut la création duClient de chiffrement DynamoDB (p. 11)et s'assurer que les attributs de clé

primaire sont toujours signés, mais jamais chiffrés. Pour créer le contexte de chiffrement et découvrir la clé primaire, les classes d'assistant clientes appelle DynamoDB.DescribeTable. Pour exécuter ce code, vous devez avoir l'autorisation d'appeler cette opération.

Consultez l'exemple de code complet : aws_kms_encrypted_table.py

Étape 1 : Création de la table

Commencez par créer une instance d'une table DynamoDB standard avec le nom de la table.

```
table_name='test-table'
table = boto3.resource('dynamodb').Table(table_name)
```

Étape 2 : Création d'un fournisseur de matériaux de chiffrement

Créez une instance du fournisseur CMP (p. 13) que vous avez sélectionné.

Cet exemple utilise le fournisseur KMS direct (p. 14), mais vous pouvez utiliser n'importe quel fournisseur CMP compatible. Pour créer un fournisseur KMS direct, spécifiez unAWS KMS key. Cet exemple utilise l'Amazon Resource Name (ARN) deAWS KMS key, mais vous pouvez utiliser n'importe quel identifiant de clé valide.

```
kms_key_id='arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab'
kms_cmp = AwsKmsCryptographicMaterialsProvider(key_id=kms_key_id)
```

Étape 3 : Création de l'objet actions d'attribut

Les actions d'attribut (p. 9) informent le chiffreur d'élément des actions à exécuter sur chaque attribut de l'élément. L'objet `AttributeActions` de l'exemple chiffre et signe tous les éléments à l'exception de l'attribut `test`, qui est ignoré.

Ne spécifiez pas d'actions d'attribut pour les attributs de clé primaire quand vous utilisez une classe d'annotations cliente. La classe `EncryptedTable` signe, mais ne chiffre jamais, les attributs de clé primaire.

```
actions = AttributeActions(
    default_action=CryptoAction.ENCRYPT_AND_SIGN,
    attribute_actions={'test': CryptoAction.DO_NOTHING}
)
```

Étape 4 : Création de la table chiffrée

Créez la table chiffrée à l'aide de la table standard, du fournisseur KMS direct et des actions d'attribut. Cette étape termine la configuration.

```
encrypted_table = EncryptedTable(
    table=table,
    materials_provider=kms_cmp,
    attribute_actions=actions
)
```

Étape 5 : Insère l'élément en texte brut dans la table

Lorsque vous appelez le`put_item`sur la méthode`encrypted_table`, vos éléments de table sont chiffrés de façon transparente, signés et ajoutés à votre table DynamoDB.

Tout d'abord, définissez l'élément de table.

```
plaintext_item = {
    'partition_attribute': 'value1',
    'sort_attribute': 55
    'example': 'data',
    'numbers': 99,
    'binary': Binary(b'\x00\x01\x02'),
    'test': 'test-value'
}
```

Puis, insérez-le dans la table.

```
encrypted_table.put_item(Item=plaintext_item)
```

Pour obtenir l'élément depuis la table DynamoDB sous sa forme chiffrée, appelez le`get_item`sur la méthode`table`objet. Pour obtenir l'élément chiffré, appelez la méthode `get_item` sur l'objet `encrypted_table`.

Utilisation du chiffreur d'élément

Cet exemple vous montre comment interagir directement avec lechiffreur d'élément (p. 9)dans DynamoDB Encryption Client lors du chiffrement des éléments de table, au lieu d'utiliser leClasses d'ancisse cliente (p. 46)qui interagissent avec le chiffreur d'élément pour vous.

Lorsque vous utilisez cette technique, vous créez le contexte de chiffrement DynamoDB et l'objet de configuration (`CryptoConfig`) manuellement. De même, vous chiffrez les éléments en un appel et les placez dans votre table DynamoDB dans un appel distinct. Cela vous permet de personnaliser votre`put_item`appelle et utilise le client de chiffrement DynamoDB pour chiffrer et signer des données structurées qui ne sont jamais envoyées à DynamoDB.

Cet exemple utilise le fournisseur KMS direct (p. 14), mais vous pouvez utiliser n'importe quel fournisseur CMP compatible.

Voir l'exemple de code complet : aws_kms_encrypted_item.py

Étape 1 : Création de la table

Commencez par créer une instance d'une ressource de table DynamoDB standard avec le nom de la table.

```
table_name='test-table'
table = boto3.resource('dynamodb').Table(table_name)
```

Étape 2 : Création d'un fournisseur de matériaux de chiffrement

Créez une instance du fournisseur CMP (p. 13) que vous avez sélectionné.

Cet exemple utilise le fournisseur KMS direct (p. 14), mais vous pouvez utiliser n'importe quel fournisseur CMP compatible. Pour créer un fournisseur KMS direct, spécifiez unAWS KMS key. Cet exemple utilise l'Amazon Resource Name (ARN) deAWS KMS key, mais vous pouvez utiliser n'importe quel identifiant de clé valide.

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

```
kms_cmp = AwsKmsCryptographicMaterialsProvider(key_id=kms_key_id)
```

Étape 3 : Utilisation de la classe d'assistant TableInfo

Pour obtenir des informations sur la table depuis DynamoDB, créez une instance deTableInfo (p. 46)Classe d'assistant. Lorsque vous travaillez directement avec le chiffreur d'élément, vous devez créer une instance `TableInfo` et appeler ses méthodes. Les classes d'annotation clientes (p. 46) le font pour vous.

L`erefresh_indexed_attributes`Méthode de`TableInfo`utilise leDescribeTableFonctionnement DynamoDB pour obtenir des informations précises et en temps réel sur la table. Elles incluent la clé primaire et ses index secondaires locaux et globaux. L'appelant doit avoir l'autorisation d'appeler `DescribeTable`.

```
table_info = TableInfo(name=table_name)
table_info.refresh_indexed_attributes(table.meta.client)
```

Étape 4 : Création du contexte de chiffrement DynamoDB

LeClient de chiffrement DynamoDB (p. 11)contient des informations sur la structure de la table et sur la façon dont elle est chiffrée et signée. Cet exemple crée explicitement un contexte de chiffrement DynamoDB, car il interagit avec le chiffreur d'élément. LeClasses d'ancisse cliente (p. 46)Créez le contexte de chiffrement DynamoDB pour vous.

Pour obtenir la clé de partition et la clé de tri, vous pouvez utiliser les propriétés de la classe d'annotations TableInfo (p. 46).

```
index_key = {
    'partition_attribute': 'value1',
    'sort_attribute': 55
}

encryption_context = EncryptionContext(
    table_name=table_name,
    partition_key_name=table_info.primary_index.partition,
    sort_key_name=table_info.primary_index.sort,
    attributes=dict_to_ddb(index_key)
)
```

Étape 5 : Création de l'objet actions d'attribut

Les actions d'attribut (p. 9) informent le chiffreur d'élément des actions à exécuter sur chaque attribut de l'élément. L'objet `AttributeActions` de l'exemple chiffre et signe tous les éléments à l'exception des attributs de clé primaire, qui sont signés, mais pas chiffrés, et de l'attribut `test`, qui est ignoré.

Lorsque vous interagissez directement avec le chiffreur d'élément et que votre action par défaut est `ENCRYPT_AND_SIGN`, vous devez spécifier une autre action pour la clé primaire. Vous pouvez utiliser la méthode `set_index_keys`, qui utilise `SIGN_ONLY` pour la clé primaire, ou `DO_NOTHING` s'il s'agit de l'action par défaut.

Pour spécifier la clé primaire, cet exemple utilise les clés d'index dans laTableInfo (p. 46)objet, qui est renseigné par un appel à DynamoDB. Cette technique est plus sûre que le codage en dur des noms de clé primaire.

```
actions = AttributeActions(
    default_action=CryptoAction.ENCRYPT_AND_SIGN,
    attribute_actions={'test': CryptoAction.DO_NOTHING}
)
```

```
actions.set_index_keys(*table_info.protected_index_keys())
```

Étape 6 : Création de la configuration pour l'élément

Pour configurer le client de chiffrement DynamoDB, utilisez les objets que vous venez de créer dans unCryptoConfigconfiguration pour l'élément de table. Les classes d'annotations clientes créent automatiquement la configuration CryptoConfig.

```
crypto_config = CryptoConfig(
    materials_provider=kms_cmp,
    encryption_context=encryption_context,
    attribute_actions=actions
)
```

Étape 7 : Chiffrer l'élément

Cette étape chiffre et signe l'élément, mais ne le place pas dans la table DynamoDB.

Lorsque vous utilisez une classes d'assistant cliente, vos éléments sont chiffrés et signés de façon transparente, puis ajoutés à votre table DynamoDB lorsque vous appelez le`put_item`de la classe d'assistant. Lorsque vous utilisez le chiffreur d'élément directement, les actions de chiffrage et de placement sont indépendantes.

Tout d'abord, créez un élément en texte brut.

```
plaintext_item = {
    'partition_attribute': 'value1',
    'sort_key': 55,
    'example': 'data',
    'numbers': 99,
    'binary': Binary(b'\x00\x01\x02'),
    'test': 'test-value'
}
```

Puis, chiffrez-le et signez-le. La méthode `encrypt_python_item` requiert l'objet de configuration `CryptoConfig`.

```
encrypted_item = encrypt_python_item(plaintext_item, crypto_config)
```

Étape 8 : Insère l'élément dans la table

Cette étape place l'élément chiffré et signé dans la table DynamoDB.

```
table.put_item(Item=encrypted_item)
```

Pour afficher l'élément chiffré, appelez la méthode `get_item` sur l'objet original `table`, et non sur l'objet `encrypted_table`. Elle obtient l'élément de la table DynamoDB sans le vérifier et le déchiffrer.

```
encrypted_item = table.get_item(Key=partition_key)['Item']
```

L'illustration suivante présente une partie d'un exemple d'élément de table chiffré et signé.

Les valeurs des attributs chiffrés sont des données binaires. Les noms et valeurs des attributs de clé primaire (`partition_attribute` et `sort_attribute`) et l'attribut `test` demeurent en texte brut. La sortie affiche aussi l'attribut qui contient la signature (`*amzn-ddb-map-sig*`) et l'attribut de description des matériaux (p. 10) (`*amzn-ddb-map-desc*`).

```
{
    '*amzn-ddb-map-desc*': Binary(b'\x00\x00\x00\x00\x00\x00\x00\x10amzn-ddb-env-
\x00\x00\x00\xe0AQEBAHhA84wnXjEJdBbBBylRUFcZZK2j7xwh6UyLoL28nQ
+0FAAAAH4wfAYJKoZIhvcNAQcGoG8wbQIBADBoBgkqhkiG9w0BBwEwHgYJYIZIAWUDBAEuMBEEDPeFBydr
izYl0R0C4M7wAK6E1/N/bgTmHI=\x00\x00\x00\x17amzn-ddb-map-signingAlg\x00\x00\x00\nHr
\x00\x00\x00\x11/CBC/PKCS5Padding\x00\x00\x00\x10amzn-ddb-sig-alg\x00\x00\x00\x0eE
\x00\x00\x00\x0faws-kms-ec-attr\x00\x00\x00\x06*keys*'),
    '*amzn-ddb-map-sig*': Binary(b"\xd3\xc6\xc7\n\xb7#\x13\xd1Y\xea\xe4.|^\xbd\xdf
    'binary': Binary(b'!"\xc5\x92\xd7\x13\x1d\xe8Bs\x9b\x7f\xa8\x8e\x9c\xcf\x10\x
    'example': Binary(b"'b\x933\x9a+s\xf1\xd6a\xc5\xd5\x1aZ\xed\xd6\xce\xe9X\xf0T
    'numbers': Binary(b'\xd5\xa0\\d\xcc\x85\xf5\x1e\xb9-f!\xb9\xb8\x8a\x1aT\xbaq\x
    'partition_attribute': 'value1',
    'sort_attribute': 55,
    'test': 'test-value'
}
```

Modification de votre modèle de données

Chaque fois que vous chiffrez ou déchiffrez un élément, vous devez fournirActions d'attribut (p. 9)qui indiquent à DynamoDB Encryption Client les attributs à chiffrer et à signer, ceux à signer mais à ne pas chiffrer, et ceux à ignorer. Les actions d'attribut ne sont pas enregistrées dans l'élément chiffré et le client de chiffrement DynamoDB ne met pas à jour vos actions d'attribut automatiquement.

> Important
>
> Le client de chiffrement DynamoDB ne prend pas en charge le chiffrement des données de table DynamoDB existantes non chiffrées.

Chaque fois que vous modifiez votre modèle de données, c'est-à-dire lorsque vous ajoutez ou supprimez des attributs de vos éléments de table, vous risquez une erreur. Si les actions d'attribut que vous spécifiez ne rendent pas compte de tous les attributs de l'élément, l'élément peut ne pas être chiffré et signé comme vous l'escomptiez. Surtout, si les actions d'attribut que vous fournissez lors du déchiffrement d'un élément diffèrent des actions d'attribut que vous avez fournies lors du chiffrement de l'élément, la vérification de la signature peut échouer.

Par exemple, si les actions d'attribut utilisées pour chiffrer l'élément lui disent de signer l'attribut `test`, la signature de l'élément comportera l'attribut `test`. Cependant, si les actions d'attribut utilisées pour déchiffrer l'élément ne tiennent pas compte de l'attribut `test`, la vérification échouera parce que le client essaiera de vérifier une signature qui n'inclut pas l'attribut `test`.

Il s'agit d'un problème particulier lorsque plusieurs applications lisent et écrivent les mêmes éléments DynamoDB car le client de chiffrement DynamoDB doit calculer la même signature pour les éléments dans toutes les applications. C'est également un problème pour toute application distribuée car les modifications dans les actions d'attribut doivent se propager à tous les hôtes. Même si vos tables DynamoDB sont accédées par un hôte dans un processus, l'établissement d'un processus de meilleures pratiques permettra d'éviter les erreurs si le projet devient de plus en plus complexe.

Pour éviter les erreurs de validation de signature qui vous empêchent de lire les éléments de votre tableau, suivez les instructions suivantes.

- Ajout d'un attribut (p. 55)— Si le nouvel attribut modifie vos actions d'attribut, déployez entièrement la modification d'action d'attribut avant d'inclure le nouvel attribut dans un élément.
- Suppression d'un attribut (p. 57)— Si vous arrêtez d'utiliser un attribut dans vos éléments, ne modifiez pas vos actions d'attribut.
- Modification de l'action : une fois que vous avez utilisé une configuration d'actions d'attribut pour chiffrer vos éléments de table, vous ne pouvez pas modifier en toute sécurité l'action par défaut ou l'action d'un attribut existant sans chiffrer à nouveau tous les éléments de votre table.

Les erreurs de validation de signature peuvent être extrêmement difficiles à résoudre, de sorte que la meilleure approche consiste à les prévenir.

Rubriques

Ajout d'un attribut

Lorsque vous ajouterez un nouvel attribut à des éléments de table, vous devrez peut-être modifier vos actions attributaires. Pour éviter les erreurs de validation de signature, nous vous recommandons d'implémenter cette modification en deux étapes. Vérifiez que la première étape est terminée avant de commencer la deuxième étape.

1. Modifiez les actions d'attribut dans toutes les applications qui lisent ou écrivent dans la table. Déployez ces modifications et confirmez que la mise à jour a été propagée à tous les hôtes de destination.
2. Écrire des valeurs dans le nouvel attribut dans vos éléments de table.

Cette approche en deux étapes garantit que toutes les applications et les hôtes ont les mêmes actions d'attribut et calculera la même signature, avant toute rencontre avec le nouvel attribut. Ceci est important même lorsque l'action de l'attribut est Ne rien faire (ne pas chiffrer ou signer), car la valeur par défaut de certains chiffreurs est de chiffrer et de signer.

Les exemples suivants montrent le code de la première étape de ce processus. Ils ajoutent un nouvel attribut d'élément `link`, qui stocke un lien vers un autre élément de table. Étant donné que ce lien doit rester en texte brut, l'exemple lui attribue l'action sign-only. Après avoir entièrement déployé cette modification, puis vérifié que toutes les applications et tous les hôtes possèdent les nouvelles actions attributaires, vous pouvez commencer à utiliser l'attribut `link` dans vos éléments de table.

Java DynamoDB Mapper

Lors de l'utilisation de `DynamoDB Mapper` et `AttributeEncryptor`, par défaut, tous les attributs sont chiffrés et signés à l'exception des clés primaires, qui sont signées mais pas chiffrées. Pour spécifier une action de signature uniquement, utilisez l'annotation `@DoNotEncrypt`.

Cet exemple utilise l'annotation `@DoNotEncrypt` du nouvel attribut `link`.

```
@DynamoDBTable(tableName = "ExampleTable")
public static final class DataPoJo {
  private String partitionAttribute;
  private int sortAttribute;
  private String link;

  @DynamoDBHashKey(attributeName = "partition_attribute")
  public String getPartitionAttribute() {
    return partitionAttribute;
  }

  public void setPartitionAttribute(String partitionAttribute) {
    this.partitionAttribute = partitionAttribute;
  }

  @DynamoDBRangeKey(attributeName = "sort_attribute")
  public int getSortAttribute() {
    return sortAttribute;
  }

  public void setSortAttribute(int sortAttribute) {
    this.sortAttribute = sortAttribute;
  }

  @DynamoDBAttribute(attributeName = "link")
  @DoNotEncrypt
  public String getLink() {
    return link;
  }
```

```
  public void setLink(String link) {
    this.link = link;
  }

  @Override
  public String toString() {
    return "DataPoJo [partitionAttribute=" + partitionAttribute + ",
        sortAttribute=" + sortAttribute + ",
        link=" + link + "]";
  }
}
```

Java DynamoDB encryptor

Dans le chiffreur DynamoDB de niveau inférieur, vous devez définir des actions pour chaque attribut. Cet exemple utilise une instruction switch où la valeur par défaut est `encryptAndSign` et des exceptions sont spécifiées pour la clé de partition, la clé de tri et le nouvel attribut `link`. Dans cet exemple, si le code d'attribut de lien n'était pas entièrement déployé avant son utilisation, l'attribut de lien serait chiffré et signé par certaines applications, mais seulement signé par d'autres.

```
for (final String attributeName : record.keySet()) {
    switch (attributeName) {
        case partitionKeyName:
            // fall through to the next case
        case sortKeyName:
            // partition and sort keys must be signed, but not encrypted
            actions.put(attributeName, signOnly);
            break;
        case "link":
            // only signed
            actions.put(attributeName, signOnly);
            break;
        default:
            // Encrypt and sign all other attributes
            actions.put(attributeName, encryptAndSign);
            break;
    }
}
```

Python

Dans DynamoDB Encryption Client for Python, vous pouvez spécifier une action par défaut pour tous les attributs, puis spécifier des exceptions.

Si vous utilisez une classe d'annotations clientes (p. 46), vous n'avez pas besoin de spécifier une action d'attribut pour les attributs de clé primaire. Les classes d'annotations clients vous empêchent de chiffrer votre clé primaire. Toutefois, si vous n'utilisez pas de classes d'annotations clientes, vous devez définir l'action SIGN_ONLY sur votre clé de partition et la clé de tri. Si vous chiffrez accidentellement votre partition ou votre clé de tri, vous ne pourrez pas récupérer vos données sans une analyse complète de la table.

Cet exemple spécifie une exception pour le nouvel attribut `link`, qui obtient l'action `SIGN_ONLY`.

```
actions = AttributeActions(
    default_action=CryptoAction.ENCRYPT_AND_SIGN,
    attribute_actions={
      'example': CryptoAction.DO_NOTHING,
      'link': CryptoAction.SIGN_ONLY
    }
)
```

Suppression d'un attribut

Si vous n'avez plus besoin d'un attribut dans les éléments qui ont été chiffrés avec DynamoDB Encryption Client, vous pouvez arrêter d'utiliser l'attribut. Toutefois, ne supprimez pas ou ne modifiez pas l'action de cet attribut. Si vous le faites, puis que vous rencontrez un élément avec cet attribut, la signature calculée pour l'élément ne correspondra pas à la signature d'origine et la validation de la signature échouera.

Bien que vous soyez tenté de supprimer toutes les traces de l'attribut de votre code, ajoutez un commentaire indiquant que l'élément n'est plus utilisé au lieu de le supprimer. Même si vous effectuez une analyse complète de table pour supprimer toutes les instances de l'attribut, un élément chiffré avec cet attribut peut être mis en cache ou en cours de traitement quelque part dans votre configuration.

Résolution des problèmes de votre application DynamoDB Encryption Client

Cette section décrit les problèmes que vous pouvez rencontrer lors de l'utilisation du client de chiffrement DynamoDB et propose des suggestions pour les résoudre.

Pour fournir des commentaires sur le client de chiffrement DynamoDB, soumettez un problème dans leaws-dynamodb-encryptionjavaouaws-dynamodb-encryption-Python GitHubrepository.

Pour émettre des commentaires sur cette documentation, utilisez le lien des commentaires sur n'importe quelle page. Vous pouvez également déposer un problème ou contribuer àaws-dynamodb-encryption-docs, le référentiel open source de cette documentation surGitHub.

Rubriques

Accès refusé

Problème : Votre application se voit refuser l'accès à une ressource dont elle a besoin.

Suggestion : En savoir plus sur les autorisations requises et les ajouter au contexte de sécurité dans lequel votre application s'exécute.

Détails

Pour exécuter une application qui utilise une bibliothèque client de chiffrement DynamoDB, l'appelant doit avoir l'autorisation d'utiliser ses composants. Sinon, l'accès aux éléments requis lui est refusé.

- Le client de chiffrement DynamoDB n'a pas besoin d'Amazon Web Services (AWS) ou dépendent d'unAWSservice. Toutefois, si votre application utiliseAWS, vous avez besoinunCompte AWSetutilisateurs disposant d'une autorisationpour utiliser le compte.
- Le client de chiffrement DynamoDB ne nécessite pas Amazon DynamoDB. Cependant, si l'application qui utilise le client crée des tables DynamoDB, place les éléments dans une table ou obtient les éléments à partir d'une table, l'appelant doit avoir l'autorisation d'utiliser les opérations DynamoDB requises dans votreCompte AWS. Pour plus d'informations, consultez lerubriques de contrôle d'accèsdans leAmazon DynamoDB Developer Guide.
- Si votre application utilise unclasse d'assistance client (p. 46)Dans DynamoDB Encryption Client for Python, l'appelant doit avoir l'autorisation d'appeler DynamoDBDescribeTable.
- Le client de chiffrement DynamoDB n'a pas besoinAWS Key Management Service(AWS KMS). Toutefois, si votre application utilise unFournisseur de matériaux KMS direct (p. 14), ou il utilise unÀ propos du fournisseur le plus récent (p. 23)avec un magasin de fournisseurs qui utiliseAWS KMS, l'appelant doit avoir l'autorisation d'utiliser leAWS KMS GenerateDataCléetDecryptopérations.

Échec de la vérification de la signature

Problème : Un élément ne peut pas être déchiffré en raison de l'échec de la vérification de la signature. L'élément peut aussi ne pas avoir été chiffré et signé comme vous l'escomptez.

Suggestion : Assurez-vous que les actions d'attribut que vous fournissez représentent tous les attributs de l'élément. Lors du déchiffrement d'un élément, veillez à fournir les actions d'attribut qui correspondent aux actions utilisées pour chiffrer l'élément.

Détails

LeActions d'attribut (p. 9)que vous fournissez, indiquez au client de chiffrement DynamoDB les attributs à chiffrer et à signer, les attributs à signer (mais pas à chiffrer) et ceux à ignorer.

Si les actions d'attribut que vous spécifiez ne rendent pas compte de tous les attributs de l'élément, l'élément peut ne pas être chiffré et signé comme vous l'escomptiez. Plus important encore, si vos actions d'attribut ne rendent pas compte de tous les attributs de l'élément, l'élément peut ne pas être chiffré et signé comme vous l'escomptiez.

Si les actions d'attribut que vous fournissez lors du déchiffrement d'un élément diffèrent des actions d'attribut que vous avez fournies lors du chiffrement de l'élément, la vérification de la signature peut échouer. Il s'agit d'un problème particulier pour les applications distribuées dans lesquelles les nouvelles actions d'attribut peuvent ne pas avoir été propagées sur tous les hôtes.

Les erreurs de validation de signature sont difficiles à résoudre. Pour vous aider à les prévenir, prenez des précautions supplémentaires lorsque vous modifiez votre modèle de données. Pour plus d'informations, consultez Modification de votre modèle de données (p. 54).

Problèmes avec les tables globales des versions antérieures

Problème : Les éléments d'une ancienne table globale Amazon DynamoDB ne peuvent pas être déchiffrés en raison de l'échec de la vérification de la signature.

Suggestion : Définissez les actions attributaires afin que les champs de réplication réservés ne soient ni chiffrés ni signés.

Détails

Vous pouvez utiliser le client de chiffrement DynamoDB avecTables globales DynamoDB. Nous vous recommandons d'utiliser les tables globales avec unClé KMS multi-régionsset répliquez la clé KMS dans tous lesRégions AWSoù la table globale est répliquée.

En commençant par les tables globalesversion 2019.11.21, vous pouvez utiliser des tables globales avec DynamoDB Encryption Client sans configuration spéciale. Toutefois, si vous utilisez des tables globalesversion 2017.11.29, vous devez vous assurer que les champs de réplication réservés ne sont ni chiffrés ni signés.

Si vous utilisez les tables globales version 2017.11.29, vous devez définir les actions attributaires des attributs suivants sur`DO_NOTHING`dansJava (p. 37)ou`@DoNotTouch`dansPython (p. 47).

- `aws:rep:deleting`
- `aws:rep:updatetime`
- `aws:rep:updateregion`

Si vous utilisez une autre version des tables globales, aucune action n'est requise.

Mauvaise performance du fournisseur le plus récent

Problème : Votre application est moins réactive, surtout après la mise à jour vers une version plus récente de DynamoDB Encryption Client.

Suggestion : Ajustement dutime-to-livevaleur et taille du cache.

Détails

Le fournisseur le plus récent est conçu pour améliorer les performances des applications qui utilisent le client de chiffrement DynamoDB en permettant une réutilisation limitée des matériaux cryptographiques. Lorsque vous configurez le fournisseur le plus récent pour votre application, vous devez équilibrer les performances améliorées et les problèmes de sécurité liés à la mise en cache et à la réutilisation.

Dans les versions plus récentes de DynamoDB Encryption Client, letime-to-live(TTL) détermine la durée pendant laquelle les fournisseurs de matériel cryptographique (CMP) mis en cache peuvent être utilisés. Le TTL détermine également la fréquence à laquelle le fournisseur le plus récent recherche une nouvelle version du CMP.

Si votre TTL est trop long, votre application peut enfreindre vos règles métier ou vos normes de sécurité. Si votre TTL est trop bref, les appels fréquents vers le magasin de fournisseurs peuvent entraîner la limitation des demandes de votre application et d'autres applications partageant votre compte de service. Pour résoudre ce problème, ajustez le TTL et la taille du cache à une valeur qui répond à vos objectifs de latence et de disponibilité et conforme à vos normes de sécurité. Pour plus d'informations, consultez Définition d'une valeur de durée de vie (p. 27).

Historique des documents pour le manuel Amazon DynamoDB Encryption Client Developer Guide

Le tableau suivant décrit les modifications importantes apportées à cette documentation. En plus de ces principales modifications, nous mettons fréquemment à jour la documentation pour améliorer les descriptions et les exemples, et pour répondre aux commentaires que vous nous envoyez. Pour recevoir une notification concernant des modifications importantes, abonnez-vous au flux RSS.

update-history-change	update-history-description	update-history-date
Modification de la documentation	Remplacez leAWS Key Management Servicetermeclé principale client (CMK)avecAWS KMS keyetClé KMS.	30 août 2021
Nouvelle fonction	Ajout de la prise en charge deAWS Key Management Service(AWS KMS) Clés multi-régions. Les clés multi-région sontAWS KMSclés dans différentesRégions AWSqui peuvent être utilisés de manière interchangeable car ils ont un ID de clé et de matériel de clé identiques.	8 juin 2021
Nouvel exemple	Ajout d'un exemple d'utilisation de DynamoDBMapper dans Java.	6 septembre 2018
Prise en charge de Python	Ajout de la prise en charge de Python, en plus de Java.	2 mai 2018
Première version	Première version de la présente documentation.	2 mai 2018

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.