

3/15（金）まで！【Findy×Qiita】エンジニアの転職や働き方に関するアンケート



47



23



この記事は最終更新日から1年以上が経過しています。

▲ DMMグループ Advent Calendar 2020 12日目 ←

@daisukeoda (Daisuke Oda)

## AWSで動画コンテンツをDRMする&パッケージング&配信する仕組みを作る

AWS | 動画 | ストリーミング | MediaServices | MediaPackage

最終更新日 2023年02月02日 投稿日 2019年12月21日

### 概要

この記事はDMMグループ Advent Calendar 2020 12日目の記事です。

過去に、AWSを用いて動画配信基盤のディザスタークリアリストックを構築した記事を書いたことがあります！

もしもに備えた動画配信基盤のDRシステム - DMM inside

この記事ではその際の知見を使って、0からAWSで動画コンテンツのエンコード・DRM・パッケージング・配信をする仕組みを解説します。

今日は社会情勢も相まって、世界中で急激にライブ配信・リアルタイム通話・VOD・音声配信・ライブECなどのストリーミング関連サービスが勢いをつけてきていると思われます。0から自前でストリーミング機能を実装するようなケースも増えてきているのではないかでしょうか。

AWSな配信システムにDRMな仕組みを導入したい、開発工数に余裕がない、プロトタイプ的な配信システムをセキュアかつコストよく構築したい、というような場面で、この記事の内容が役に立てばと思います。

### 前提

- AWS Media Servicesの概要がわかる。
- その他AWSのメジャーなサービスある程度認知している。

### 技術スタック

AWS MediaLive、AWS MediaPackage(エンコード・パッケージング)、DRM情報生成API(API Gateway&Lambda)という構成で、HLS AES 128な動画を配信する仕組みを作ります。

詳しく後述しますが、AWSでDRMをかける部分で、AWS SPEKE というインターフェースを踏襲したAPIが必要になってきます。

### AWS SPEKEとは

公式ドキュメント <https://docs.aws.amazon.com/speke/latest/documentation/what-is-speke.html>

SPEKE = Secure Packager and Encoder Key Exchange

簡単にまとめると、動画のパッケージャ(AWS Elemental)とDRMエンクリプタ間で、DRMキーなどの情報をセキュアにやりとりできる仕様のことです。

SPEKEが定めるI/OインターフェースにのっとってDRM情報(キーやHLS AESのURL)を返すサーバを実装すると、AWS Elementalと連携して動画をDRMすることができます。Multi DRMに対応していたりと、AWSでDRMするハードルを下してくれる便利ツールです。

もうちょっとわかりやすくまとめる、パッケージングは今まで通りAWS Media Servicesに任せたDRMに必要な暗号化キーなどの情報の生成を、こっち側で作ったサーバに移譲できるということです。

また、DRMは各種ベンダー(Widevine, PlayReady, FairPlay)によって暗号化情報生成のインターフェースが違っていますが、AWS SPEKEを使うことで、1つのインターフェース経由でマルチDRMすることができます(合ってるはず..)。

これによって動画ごとにユニークなキーを生成できたり、キーのプロバイダをクラウド・オンライン好きな場所に構築することができるようになり、色々ハッピーなことが起こります。

### MediaPackageとSPEKEを使ってDRMをかけよう

[SPEKE API v1 - Standard payload components - Secure Packager and Encod...](https://docs.aws.amazon.com/speke/latest/APIReference/API_SecurePackagerAndEncoderKeyExchange.html)

SPEKEはDASH IFのCPIX(Content Protection Information eXchange)という仕様をラップしています。

DASH IF CPIXについての説明はこちら [https://docs.unified-streaming.com/documentation/drm/cpix\\_intro.html](https://docs.unified-streaming.com/documentation/drm/cpix_intro.html)

(公式のドキュメントではありませんが、分かりやすかったのでUnified Streaming社のドキュメントを貼ってます)



セキュリティ分野でいち早くSASEを提供

JSOL セキュリティプロフェッショナルにインタビュー

Qiita Zine



広告 ⓘ

ITに携わる方必見のメディア

NTT DATA

### 概要

前提

技術スタック

AWS SPEKEとは

MediaPackageとSPEKEを使ってDRMをかけよう

1. SPEKE準備なキーサーバを実装

2. Role作成

3. MediaPackageでパッケージング

Ingest Asset

4. キーサーバの処理について

まとめ

CPIXはDRMに必要な情報をXMLベースでやりとりする仕様らしいです。正直あんまりよく分かってないんですがドキュメントに書いてある通りにデータを送ればだいたい動くので神です。

SPEKEにおいても、基本的にCPIXにのっとってXMLでデータをやりとりします。

公式にVOD動画をパッケージングするときのリクエスト・レスポンス例が載っているので今回はこれを参考に

HLS AES 128で暗号化してみたいと思います。

VOD Workflow Method Call Examples - <https://docs.aws.amazon.com/speke/latest/documentation/vod-workflow-methods.html>

## 1. SPEKE準拠なキーサーバーを実装

いきなりですが今回実装したコードを置いています。

<https://github.com/OdaDaisuke/aws-speke>

下記のリファレンス実装を参考にやって行きました。

[awslabs/speke-reference-server - https://github.com/awslabs/speke-reference-server](https://github.com/awslabs/speke-reference-server)

基本的には、リクエストで受け取ったCPIXのXMLを埋めていく処理を書くだけです。  
詳しい処理については後述のステップで解説していきます。

さて、キーサーバーのセットアップですが今回はLambda + API Gatewayで行なっていきます。  
上記コードをLambdaにアップしたら、以下のスクショにならって環境変数を設定しましょう。



KEY\_STORE\_BASE\_URLは、HLSのマニフェストファイルのext-x-keyのベースURLになります。

事前に空のバケットを作って外部からアクセスできるようにACLなど設定しましょう。

2番目のKEY\_STORE\_BUCKETはClearKeyを保存するバケットの名前です。

そしたら、Lambda用のロールを作成して設定してあげます。この時S3とMediaPackageとAPI Gatewayの操作ポリシーを指定してあげます。

また、同時にAPI Gatewayもセットアップして適当な名前のエンドポイントでキーサーバーを叩けるようにしていきますが、今回はリファレンス実装にならってcopy\_protectionというエンドポイントを作りました。ちなみにこのキーサーバーはAWS内部から叩けられればOKなので外部には公開しなくて大丈夫です。



## 2. Role作成

最初にMediaPackageのRoleを作成します。

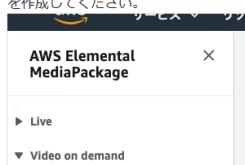
現在ロール作成時にデフォルトでMediaPackageを選択することができないので、最初にMediaConvertのロールとして作成した後

以下のように「信頼されたエンティティ」をmediapackage.amazonaws.comに変更、さらにMediaPackageのFullAccessポリシーをアタッチしてください。



## 3. MediaPackageでパッケージング

まずは下記スクショのPackaging groupsというところから適当な名前でパッケージンググループを作成してください。



Packaging groups

Assets

次にHLS AES128のパッケージング設定を作成します。

General Settingsはこのように、

**General settings**

**Id info**: HLS\_AES\_128

**Segment duration (sec)**: 6

**Apple HLS**

Enable Encryptionにはチェックを入れ、追加で出てくる入力フォームはこのような設定に。

**Enable encryption** (checked)

**Encryption method**: AES 128-bit

**Constant initialization vector**: 0011233445566778899aabbccddeeff

**Key server URL**: https://xxxxxxxx.execute-api.ap-northeast-1.amazonaws.com/Prod/copy-protection

**Role ARN**: MediaPackageRoleARN

**System ID**: 81376844-f976-481e-a84e-cc25d39b0b33

Constant Initialization Vectorは、先述したIVのことと、適当な値をUUID形式で入れておけばOKです。

Key server URLには先ほど作成したキーバーのAPI GatewayのURLを、  
Role Arnには先ほど作成したMedia PackageのARNを入れ、  
最後にSystem IDを設定。このSystem IDですが、HLS AESは正確にはDRMではないため  
System IDは定義されていないので、適当な値をUUID形式で入力しておきます。今回はコー  
ド側 [https://github.com/OdaDaisuke/aws-speke/blob/master/src/server\\_response\\_builder.py#L13](https://github.com/OdaDaisuke/aws-speke/blob/master/src/server_response_builder.py#L13) で定義した 81376844-f976-481e-a84e-cc25d39b0b33 という値を設定してみました。

ここまでたらパッケージングの設定は完了です。Save Settingを押して次に進みます。

## Ingest Asset

再びMediaPackageのホーム画面に戻り、Asset -> Ingest Assetに進みます。  
パッケージングしたい動画があるバケットを選択し、IAM Roleには先ほど作成したMedia  
Packageのロールを選択します。

ページ中程にある Asset Details は、m3u8ファイルを選択しましょう。この時親子構造の  
マニフェストでなければパッケージングできないので注意です。詳しくは、<https://qiita.com/daisukeoda/items/1eecfd639aeae6f1026c> を参照。

**Asset 1**

**Filename**: hls/test.m3u8

**Resource ID - optional**: test

最後のPackaging settingsは、先ほど作成したパッケージング設定を選択。

**Packaging group**: test (HLS (1))

そして最後に「Ingest assets」を押せば、あとは勝手にDRMしつつパッケージングしてくれます。

ボタンを押したあとは下記のようにAssetsの項目が増えているはず。

**Assets (1) Info**

	<b>Id</b>	<b>Source ARN</b>
<input type="checkbox"/>	test	arn:aws:s3:::media-

最後に、項目のタイトルをクリックしてみるとこのようにPlayback details欄に再生URLが表示されているのでテスト再生してみましょう。

適当な動画プレーヤーに再生URLを入力して再生します。  
そして暗号化時に設定したライセンスサーバーのAPIに向かって復号キーがリクエストされ、動画が再生できていれば完璧です(再生URLは一部隠しています)。



#### 4. キーサーバの処理について

SPEKEキーサーバのログをみてみると、以下のようなリクエストを受け取っています。

```
<?xml version="1.0" encoding="UTF-8"?>
<cpx:CPIX xmlns:cpx="urn:dashif:org:cpx" xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
  <cpx:ContentKeyList>
    <cpx:ContentKey id="00112233-4455-6677-8899-aabbccddeeff"></cpx:ContentKey>
  </cpx:ContentKeyList>
  <cpx:DRMSystemList>
    <cpx:DRMSystem id="00112233-4455-6677-8899-aabbccddeeff" systemId="81376844-f976-481e-"
      <cpx:PSSH />
      <cpx:ContentProtectionData />
      <cpx:URIExtXKey />
      <speke:KeyFormat />
      <speke:KeyFormatVersions />
      <speke:ProtectionHeader />
    </cpx:DRMSystem>
  </cpx:DRMSystemList>
</cpx:CPIX>
```

そしてレスポンスは以下のようないXMLです。

```
<cpx:CPIX xmlns:cpx="urn:dashif:org:cpx" xmlns:pskc="urn:ietf:params:xml:ns:keyprov:pskc">
  <cpx:ContentKeyList>
    <cpx:ContentKey explicitIV="IV(UUID形式)のHEXバイナリをbase64した文字列" id="00112233-4455-6677-8899-aabbccddeeff">
      <cpx>Data>
        <pskc:Secret>
          <pskc:PlainValue>暗号化キー(UUID形式)のHEXバイナリをbase64した文字列</pskc:PlainValue>
        </pskc:Secret>
      </cpx>Data>
    </cpx:ContentKey>
  </cpx:ContentKeyList>
  <cpx:DRMSystem id="00112233-4455-6677-8899-aabbccddeeff" systemId="81376844-f976-481e-"
    <cpx:URIExtXKey>キーをbase64エンコードした文字列</cpx:URIExtXKey>
    <speke:KeyFormat>awRlbnp0Hk</speke:KeyFormat>
    <speke:KeyFormatVersions>MQ==</speke:KeyFormatVersions>
  </cpx:DRMSystem>
</cpx:DRMSystemList>
</cpx:CPIX>
```

記事の最初の方に書いた、CPIX準拠のXMLです。

意味を軽く説明してみますと、

cpx:ContentKeyListのexplicitIVは、AES128またはSAMPLE AESでの暗号化の時に使うもので、CBCブロック暗号化の際に使う初期化ベクトルです。例えばMediaPackageの設定欄にIVを設定する項目がありますが、そこに設定した値が explicitIV属性に設定されリクエストが飛んできます。ここで受け取ったexplicitIVは、リクエストで受け取った値とは別な値でレスポンスすれば、値を上書きすることも可能です。この仕様をうまく利用して、キーローテートを実現することも可能。

pskc:PlainValueはコンテンツの暗号化キーが入ります。

cpx:DRMSystemListの配下の要素には、DRMに使う情報を格納します。今回の場合(AES)だと URIExtXkey、KeyFormat、KeyFormatVersionsだけ必要なのでそれらの情報を埋めてあげます。

cpx:PSSHとcpx:ContentProtectionDataとspeke:ProtectionHeaderはHLS AESの際には必要ないので、要素ごと削除してレスポンスしている感じです。これらはWidevineやPlayReadyで DRMする際に共通して必要になってきますが、今回は解説しません。

以上、SPEKEインターフェースを用いた動画配信システムの最低限の説明でした。

## まとめ

HLS AES 128における処理を紹介してみましたが、SPEKEは他にもWidevineやPlayReady、

FairplayなどのメジャーDRMに対応しています。

またDASH(Widevine, PlayReadyが対応)などでの配信も可能です。

AWSを使った動画配信システムの解説記事は多々ありますが、DRMまで対応したものはなかなか少ないと思うので、少しでも役に立てばと思います。

明日の投稿は、@tanikoさんです！



新規登録して、もっと便利にQiitaを使ってみよう

1. あなたにマッチした記事をお届けします
2. 便利な情報をあとで効率的に読み返せます
3. ダークテーマを利用できます

[ログインすると使える機能について](#)



[新規登録](#) [ログイン](#)



@daisukeoda (Daisuke Oda)

フォロー



## 関連記事 Recommended by LOGLY

[【初心者】AWS Elemental MediaLive + MediaPackage を使ってみる...](#)  
by mkksamba

[無料でできる MPEG-DASH + Widevine ストリーミング テスト配信](#)  
by tomopyon-sama

[AWS Media Servicesやってみた](#)  
by tin-machine

[AWS MediaLiveを使って見た](#)  
by khagi

[世界一給料が高い町工場を作る。社長と社員の「普段の会話」が本質すぎた](#)  
PR ビズヒント

[セブン鈴木元会長に「叱られなかつた」ことで気付けた。「相手に正しく伝える」ための2つの条件](#)  
PR ビズヒント

☞ この記事は以下の記事からリンクされています

[⑩ HLS暗号化に使うSPEKEサーバーのChalice実装を行う](#) からリンク 2 years ago

## コメント

この記事にコメントはありません。

いいね以上の気持ちをコメントで

[ログイン](#)

[新規登録](#)

Qiita Conference 2024 4月17日(水)~19(金)開催！

Qiita Conference 2024 | Qiita最大規模のエンジニアカンファレンス

記事投稿キャンペーン開催中

Qiita+Findy記事投稿キャンペーン「今の開発組織でトライしたこと・トライしていること・トライしようとしていること」