# Signed HTTP Exchanges (SXG)とはなにか

## SXGの仕組みからAMP連携の導入まで

**大津 繁樹**

**SXG Study**
**2019年５月２８日**

# 内容

1. SXGのしくみ
2. SXG/AMPシステム事例
3. amppackagerについて
4. SXG導入振り返り

# SXGのしくみ

# 問題1：何が違う？

1. Signed HTTP Exchanges
2. Signed Exchange
3. SXG

# 解答1

1. Signed HTTP Exchanges: 仕様の正式名称
2. Signed Exchange: 話している時みんな使っている言葉
3. SXG： 拡張子もしくは略称表記

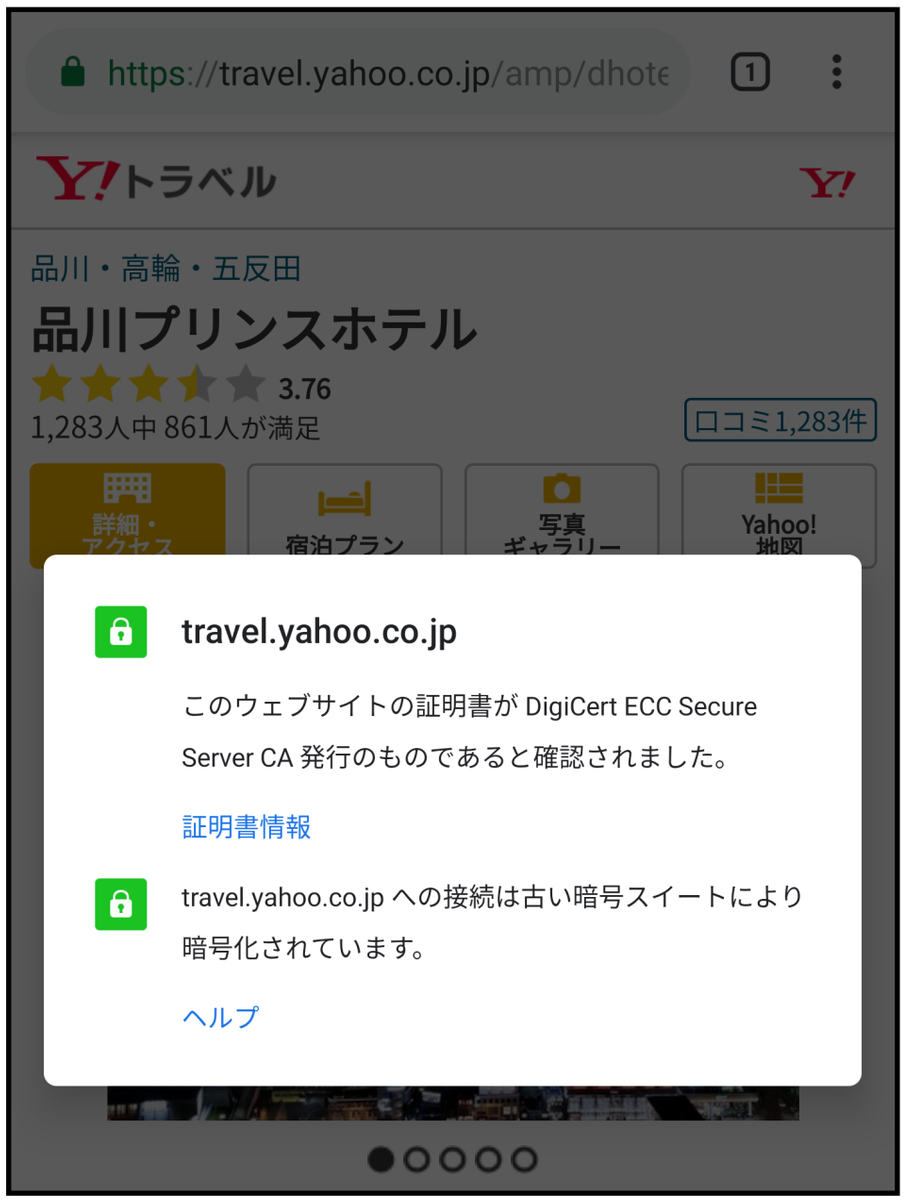
# 問題2: どっちがSXG?

A


B

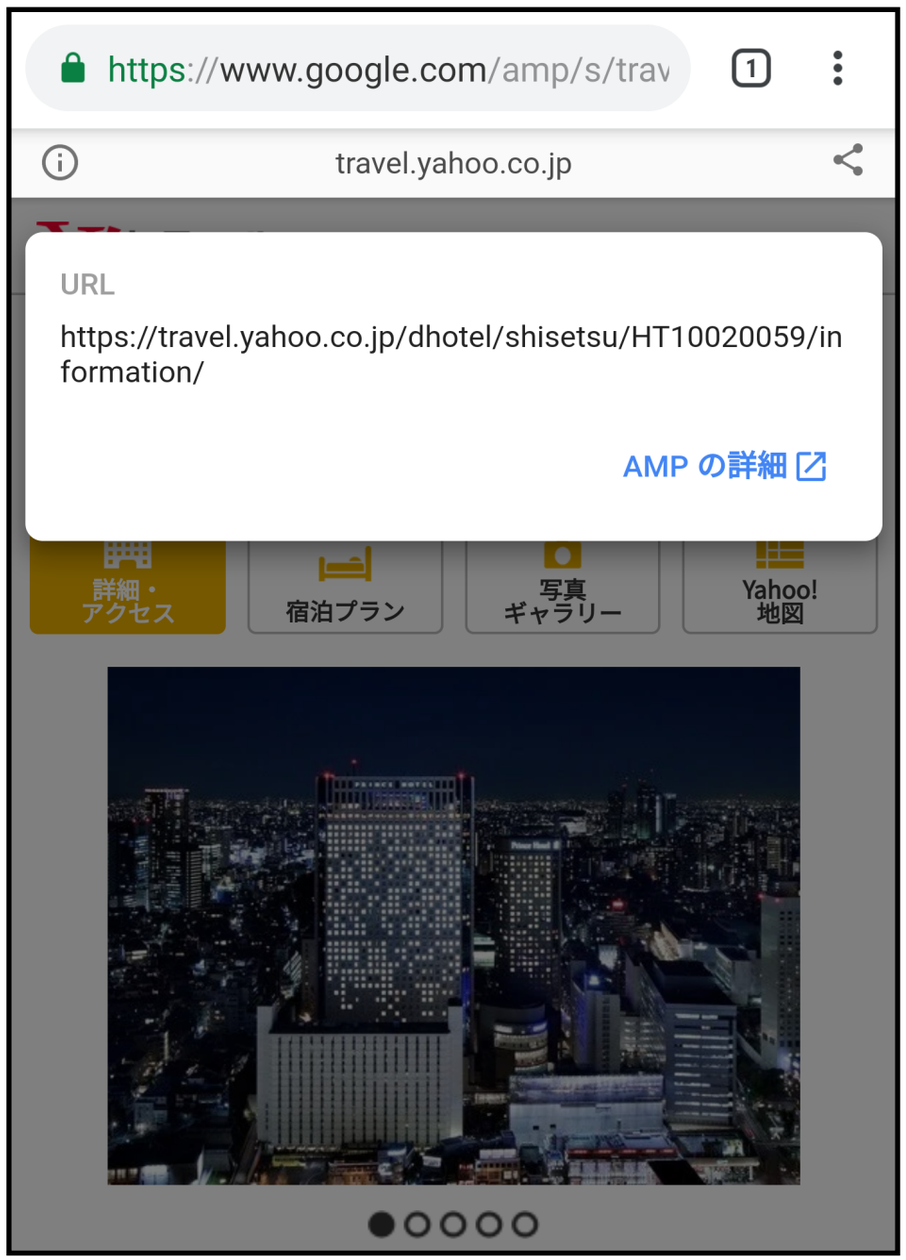
# 解答2

A




SXG

B




AMP

# 問題3: どっちがSXG?

A


B

# 解答3

A

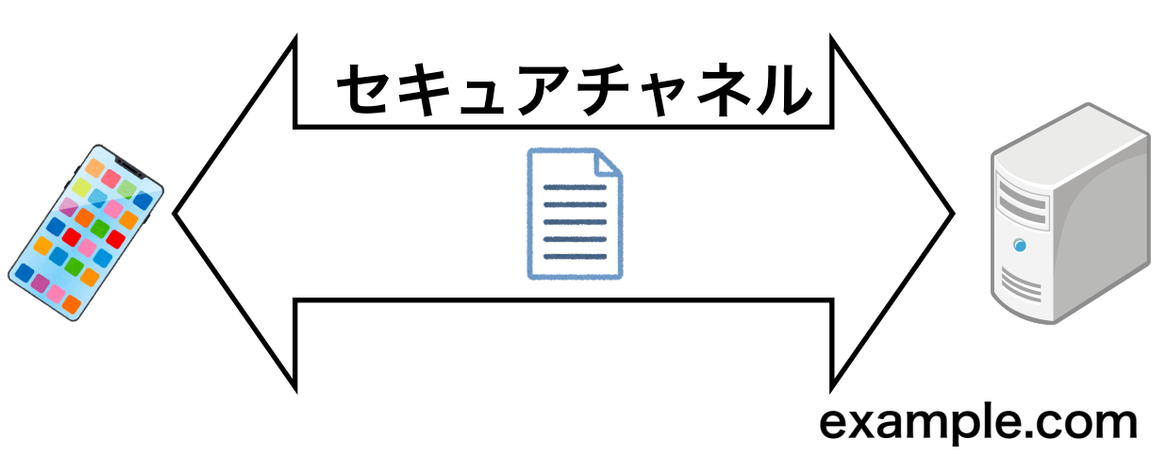

HTTPS

B


SXG

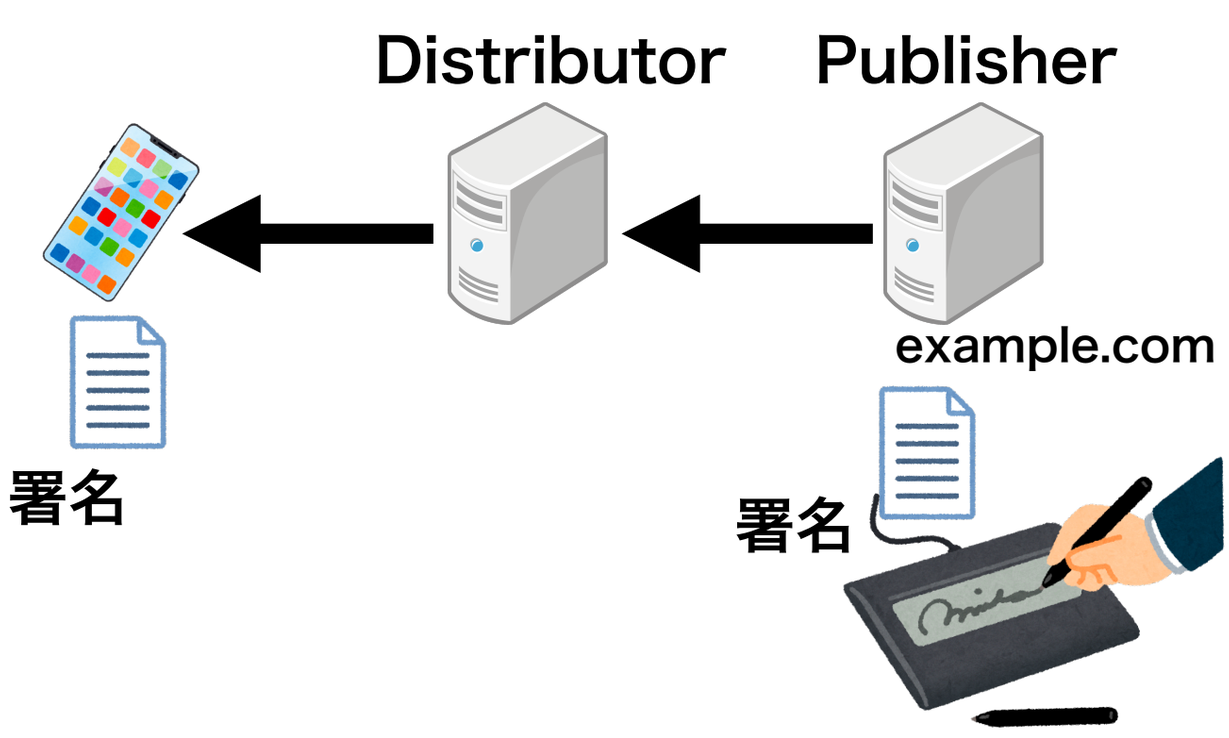
# HTTPSとSXGの違い#1

## HTTPS




通信を守る

## SXG




コンテンツを守る

publisherのoriginで表示

# HTTPSとSXGの違い#2

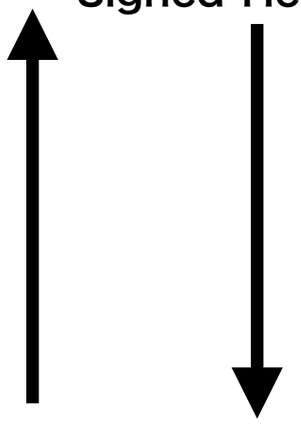
| | HTTPS | SXG |
|---|---|---|
| 機密性 | ✓ | N/A |
| 完全性 | ✓ | ✓ |
| 認証 | ✓ | ✓ |
| 否認防止 | N/A | ✓ |
| Origin | 接続先 | Publisher |
| 保護している時間 | 通信している間<br>(数百ミリ秒程度) | 署名の有効期間<br>(最大７日間) |
| HTTPの制限 | N/A | キャッシュ可能なコンテンツ限定<br>(GET限定、禁止ヘッダ有) |

# SXGのやり取り方法3種

今日の話

## 1. Same-Origin

Signature:
Signed-Headers:

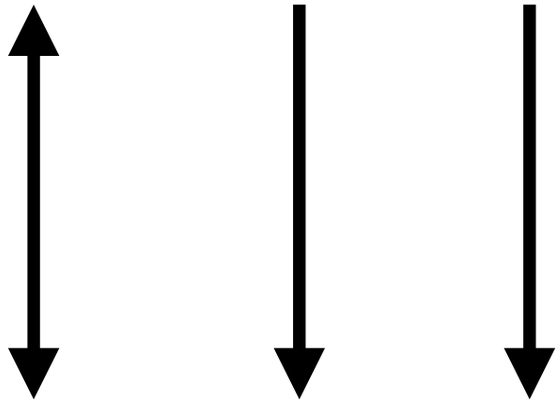


## 2. cross-origin Server Push

ENABLE_CROSS_ORIGIN_PUSH

Push Promise

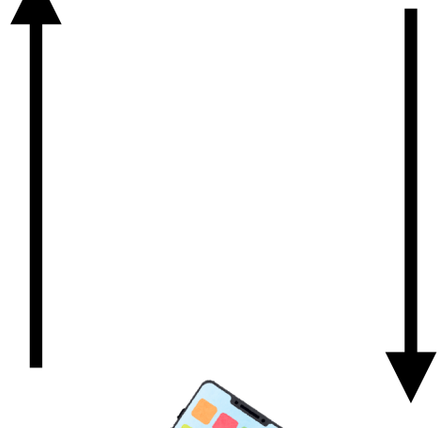
cross-origin ServerPush



ENABLE_CROSS_ORIGIN_PUSH

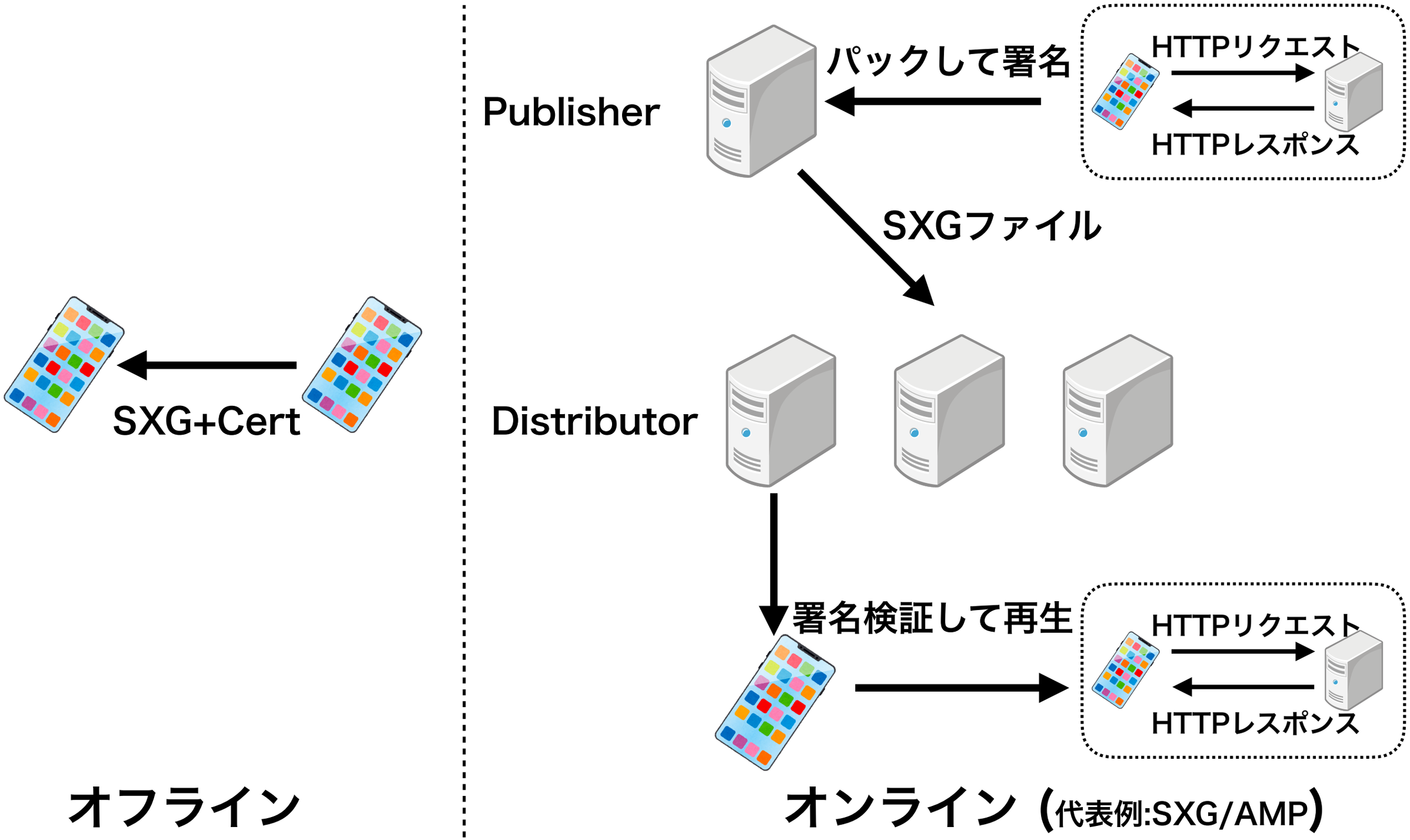
## 3. application/signed-exchange フォーマット

SXGファイル



Accept: application/signed-exchange;v=b3

# 代表的なSXGのユースケース



# SXG/AMPシーケンス図

UA

検索サービス

Distributor

AMP Cache

Publisher

Origin

Botクロール

Vary: AMP-Cache-Transform

Accept: application/signed-exchange;v=b3
AMP-Cache-Transform: google;v="1"

Botクロール

SXG

AMP最適化
(Transfomer)

GET

CertChain+OCSP

cert-url書き換え
index化

Search

SERP

Prefetch
LocalCache

SXG

GET
Verify

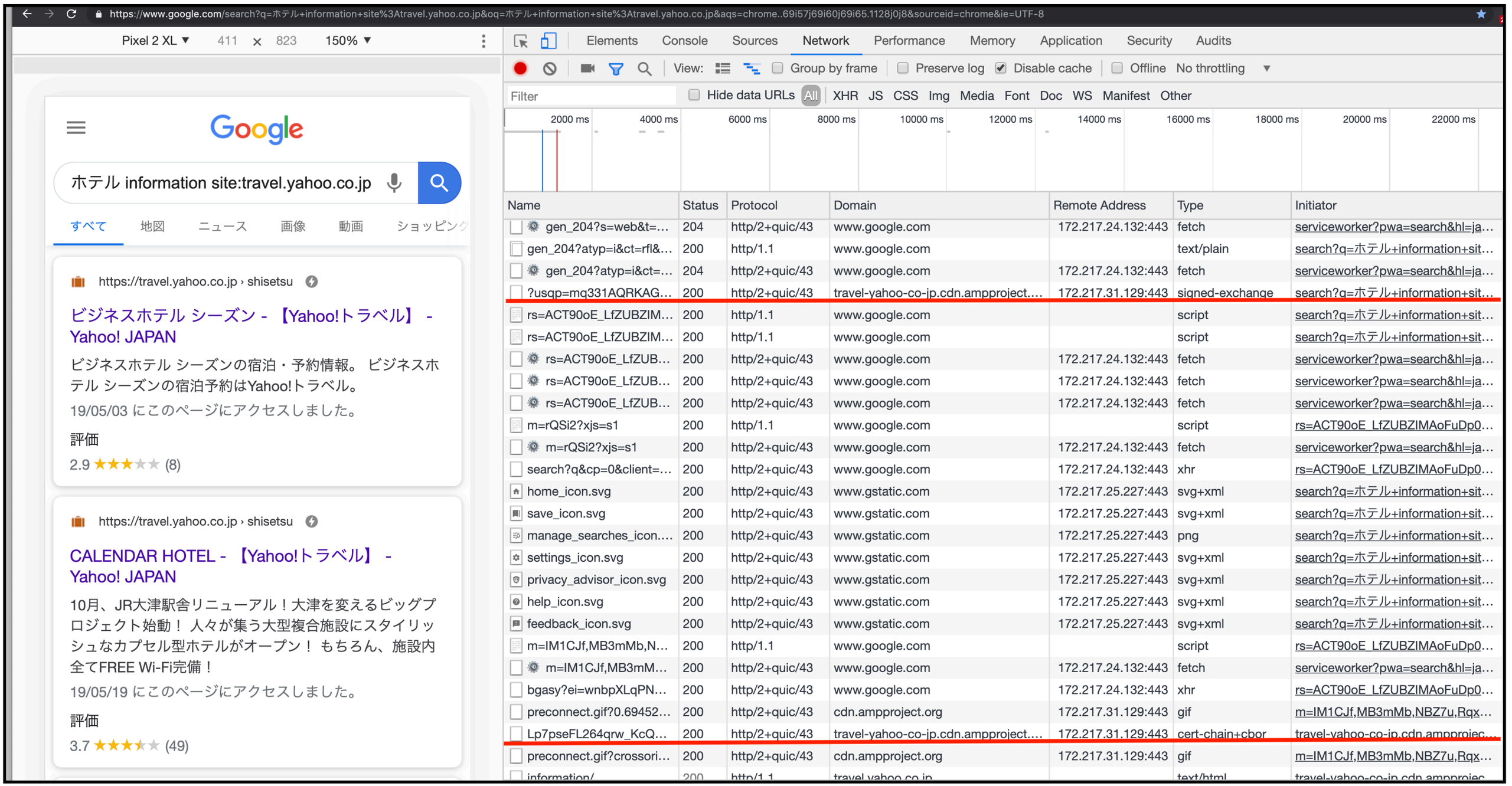
CertChain+OCSP

navigation

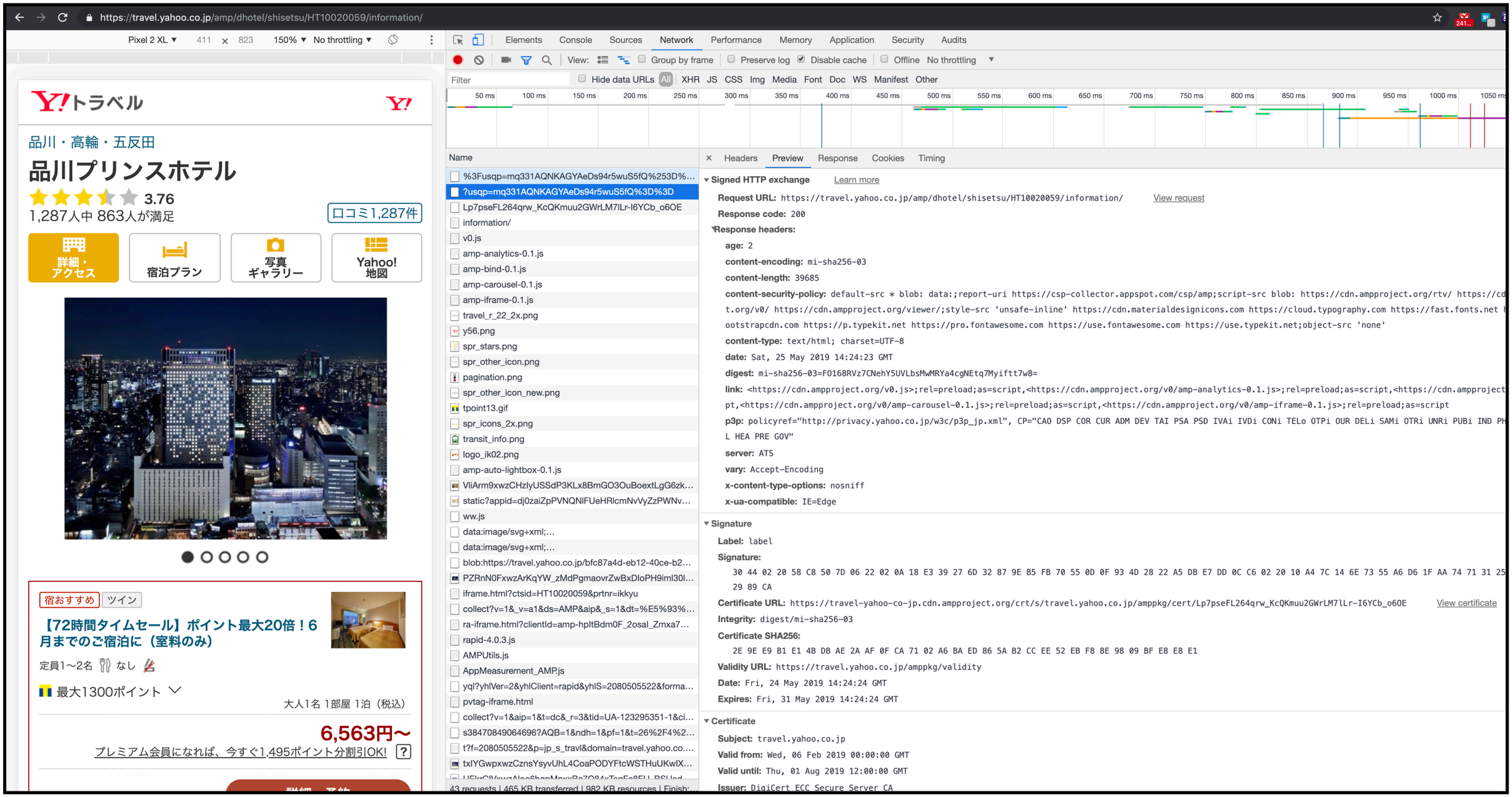
LocalCache

Originにアクセスしない
ことでPrivacyを保護

# 検索結果(SERP)から
# SXGをPrefetch



# SXG Verifyの状況




devtool上ネットワークパネルのPreviewで確認できます

# SXGの状況

- ChromeM73より default で enable (バージョン:b3)
- iOS版は非対応。現状モバイル環境でSXGが使えるのは Android Chromeのみ
- MacOS版IE Edge(Chromiumベース)でもSXG動作確認済
- Google Searchで4月中旬よりSXG/AMPを正式対応
- CloudflareもSXG機能提供開始(GUIベース)
- 標準化… まだわかりません。DISPATCH->ESCAPE Workshop

# Web Packaging仕様概要

**Binary JSONの**
**シリアライズ**

RFC7049
CBOR

**HTTPボディの**
**Hash Digest**

draft-thomson-
http-mic

**より扱いやすいHTTP**
**ヘッダの記述書式**

draft-ietf-httpbis-
header-structure

WHATWG
Fetch

Loading Signed
Exchanges

**Fetchに対する**
**モンキーパッチ**

draft-yasskin-wpack-
bundled-exchanges

**Bundle**

draft-yasskin-http-origin-
signed-response

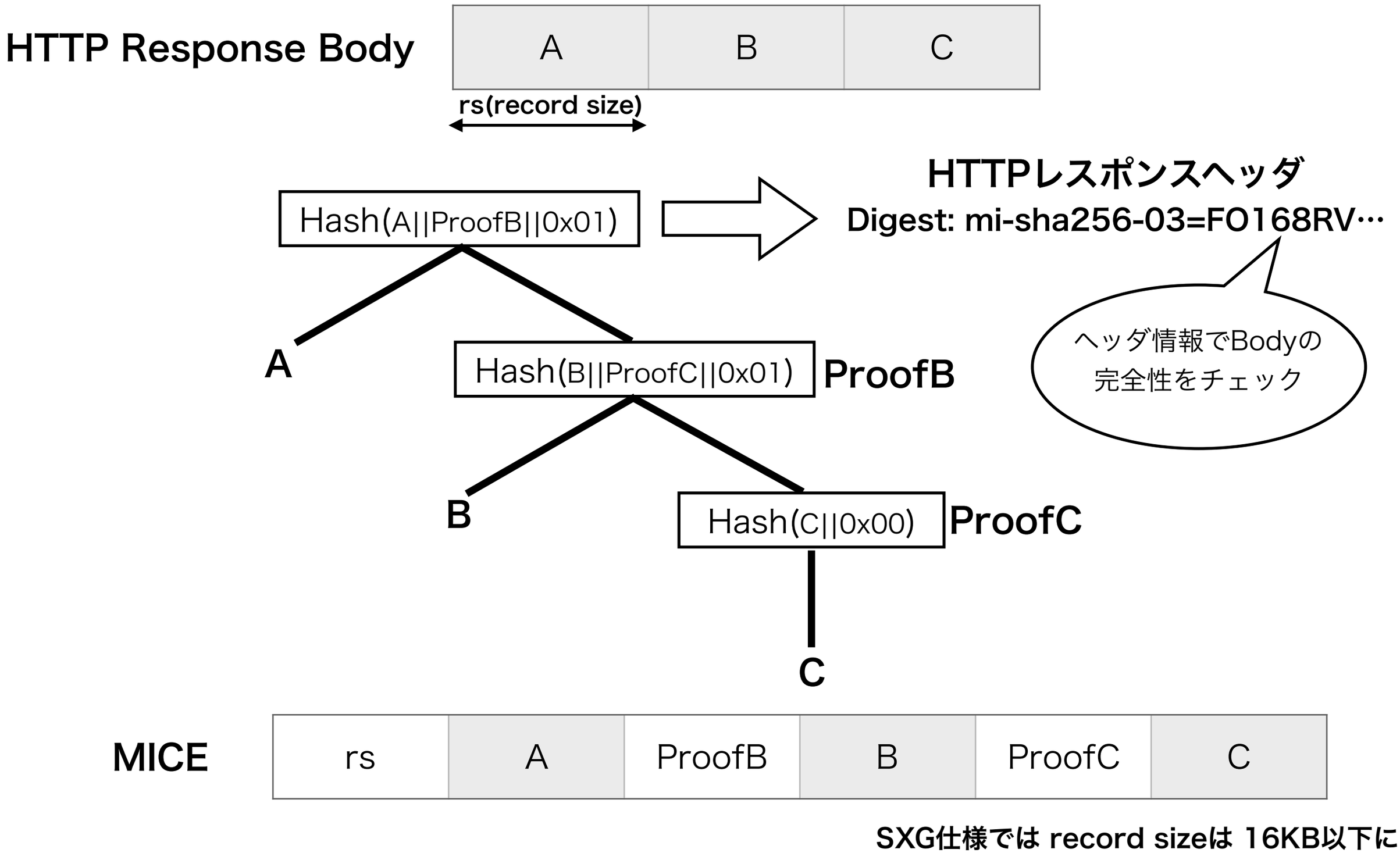
**SXG**

**(b3:2019年5月時点)**

**Web Packaging仕様**

**https://github.com/WICG/webpackage**

# Merkle Integrity Content Encoding




**SXG仕様では record sizeは 16KB以下に制限**

# SXGファイルの中身(b3)

| sxg1-b3 (8) | fallbackUrl length (2) | fallbackURL | sigLength (3) | headerLength (3) | signature | signedHeaders | payload body |
|---|---|---|---|---|---|---|---|

| | |
|---|---|
| sxg1-b3 | マジック |
| fallbackURL | SXG検証が失敗した場合にアクセスするURL |
| signature | 署名データ |
| signedHeaders | HTTPレスポンスヘッダ<br>(CBOR形式) |
| payload body | HTTPレスポンスのボディ<br>(MICE形式) |

| | |
|---|---|
| sig | 署名値 |
| integrity | HTTPレスポンスボディのハッシュアルゴリズム |
| validity-url | 新しい署名データを取得URL (現在未実装) |
| cert-url | 証明書データのURL (署名対象外) |
| cert-sh256 | 証明書データのハッシュ値 |
| ed25519key | 署名検証する公開鍵 (現在未実装) |
| date | 署名した時間 |
| expire | 署名の有効期限 (最大7日間) |

# SXGで署名しているデータ

| | |
|---|---|
| 0x20,0x20,0x20.....0x20 (64) | |
| HTTP Exchange 1(15) | |
| 0x00(1) | |
| cert-sha256(32) | |
| validity-url len (8) | validity-url |
| date(8) | |
| expires(8) | |
| requestUrl len(8) | requestUrl |
| responseHeaders len(8) | responseHeaders |

b3からrequestヘッダは除外

- SXG証明書の秘密鍵を使って ecdsa-with-SHA256(secp256r1) で署名
- SXG証明書の公開鍵を使って署名を Verify
- 署名のVerifyが成功してレスポンスボディのMICEチェックができたらSXGがvalid

# SXGの中身1

```
ohtsu@opro:~/github/shigeki/ohtsu_sxg$ node parser.js shinagawa_prince.sxg shinagawa_prince.cert
{ fallbackUrl:
   'https://travel.yahoo.co.jp/amp/dhotel/shisetsu/HT10020059/information/',
  sig:
   { sig:
      'MEUCIQDKYCYzUX/ia0KFsch/9T568s0dkcI39Ld8r5LhSnvaSAIgMIzfLG9ebiLe16jK+ifcO7PyhbhZ0clcYUAfTb47C2A',
     'validity-url': 'https://travel.yahoo.co.jp/amppkg/validity',
     integrity: 'digest/mi-sha256-03',
     'cert-url':
      'https://travel.yahoo.co.jp/amppkg/cert/Lp7pseFL264qrw_KcQKmuu2GWrLM7lLr-I6YCb_o6OE',
     'cert-sha256': 'Lp7pseFL264qrw/KcQKmuu2GWrLM7lLr+I6YCb/o6OE',
     date: '1558719559',
     expires: '1559324359' },
  header:
   '\u0000\u0000\u0000\u0000\u0000\u0000@\u0000<!doctype html><html i-amphtml-layout i-amphtml-no-boilerp
ontent="width=device-width,minimum-scale=1,initial-scale=1" name=viewport><meta content="telephone=no" na
ホテル宿泊予約はYahoo!トラベル。"                    eta content=品川プリンスホテル,品川・高輪・五反田,予
```

# SXGの中身2

```
ohtsu@opro:~$ dump-signedexchange -i shinagawa_prince.sxg
format version: 1b3
request:
  method: GET
  uri: https://travel.yahoo.co.jp/amp/dhotel/shisetsu/HT10020059/information/
  headers:
response:
  status: 200
  headers:
    P3p: policyref="http://privacy.yahoo.co.jp/w3c/p3p_jp.xml", CP="CAO DSP COR CUR ADM DEV TAI PSA PSD IV
    Content-Type: text/html; charset=UTF-8
    Content-Length: 39685
    X-Ua-Compatible: IE=Edge
    Server: ATS
    Content-Encoding: mi-sha256-03
    X-Content-Type-Options: nosniff
    Age: 0
    Date: Sat, 25 May 2019 17:39:18 GMT
    Link: <https://cdn.ampproject.org/v0.js>;rel=preload;as=script,<https://cdn.ampproject.org/v0/amp-anal
ject.org/v0/amp-carousel-0.1.js>;rel=preload;as=script,<https://cdn.ampproject.org/v0/amp-iframe-0.1.js>;r
    Vary: Accept-Encoding
    Digest: mi-sha256-03=FO168RVz7CNehY5UVLbsMwMRYa4cgNEtq7Myiftt7w8=
    Content-Security-Policy: default-src * blob: data:;report-uri https://csp-collector.appspot.com/csp/am
n.ampproject.org/viewer/;style-src 'unsafe-inline' https://cdn.materialdesignicons.com https://cloud.typog
ro.fontawesome.com https://use.fontawesome.com https://use.typekit.net;object-src 'none'
signature: label; sig=*MEUCIQDKYCYzUX/ia0KFsch/9T568s0dkcI39Ld8r5LhSnvaSAIgMIzfLG9ebiLe16jK+ifcO7PyhbhZ0cl
/travel.yahoo.co.jp/amppkg/cert/Lp7pseFL264qrw_KcQKmuu2GWrLM7lLr-I6YCb_o6OE"; cert-sha256=*Lp7pseFL264qrw/
payload [39757 bytes]:
@<!doctype html><html i-amphtml-layout i-amphtml-no-boilerplate lang=ja transformed="google;v=1" ⚡><head>
```

# CertChain+CBORの中身

U+F4DC 📜 Scroll
U+26D3 ⛓ Chains

```
[ [ '⛓',
    { cert:
       <Buffer 30 82 04 f1 30 82 04
      ocsp:
       <Buffer 30 82 01 34 0a 01 00
    { cert:
       <Buffer 30 82 03 ac 30 82 02
```

CBOR形式

```
ohtsu@opro:~$ dump-certurl -i shinagawa_prince.cert
Certificate #0:
  Subject: travel.yahoo.co.jp
  Valid from: 2019-02-06 00:00:00 +0000 UTC
  Valid until: 2019-08-01 12:00:00 +0000 UTC
  Issuer: DigiCert ECC Secure Server CA
  Embedded SCT:
    LogID: u9nfvB+KcbWTlCOXqpJ7RzhXlQqrUugakJZkNo4e0YU=
    LogID: h3W/51l8+IxDmV+9827/Vo1HVjb/SrVgwbTq/16ggw8=
  Has canSignHttpExchangesDraft extension
OCSP response:
  Status: 0 (good)
  ProducedAt: 2019-05-24 05:44:53 +0000 UTC
  ThisUpdate: 2019-05-24 05:44:53 +0000 UTC
  NextUpdate: 2019-05-31 04:59:53 +0000 UTC
Certificate #1:
  Subject: DigiCert ECC Secure Server CA
  Valid from: 2013-03-08 12:00:00 +0000 UTC
  Valid until: 2023-03-08 12:00:00 +0000 UTC
  Issuer: DigiCert Global Root CA
```

90日以内、現在有効？？

CT
ログ有？

SXG拡張有?

現在の時刻
で有効？

rootCAまで
署名検証

# SXG禁止HTTPヘッダ

## Hop-by-Hopレスポンスヘッダ

| |
|---|
| Connection |
| Keep-Alive |
| Proxy-Connection |
| Trailer |
| Transfer-Encoding |
| Upgrade |

## Statefulレスポンスヘッダ

| |
|---|
| Authentication-Control |
| Authentication-Info |
| Clear-Site-Data |
| Optional-WWW-Authenticate |
| Proxy-Authenticate |
| Proxy-Authenticate-Info |
| Public-Key-Pins |
| Sec-WebSocket-Accept |
| Set-Cookie |
| Set-Cookie2 |
| SetProfile |
| Strict-Transport-Security |
| WWW-Authenticate |

**SXG化するAMPサーバからのレスポンスに上記ヘッダが入っていないか注意する**
**amppkgではErrorOnStatefulHeadersの設定で制御できます。**

# amppackager について

github.com/ampproject/amppackager/

**release branch: Production用ブランチ**

**master branch: 開発ブランチ**

# amppackager 提供コマンド

- cmd/amppkg/　amppkg本体
- cmd/amppkg_dl_sxg/ SXGと証明書をダウンロード
- cmd/amppkg_test_cache/ SXGを提供する簡易キャッシュサーバ
- cmd/gateway_server/　gRPCでSXGを生成するゲートウェイサーバ
- cmd/transform/ AMP最適化を行うコマンド

# amppkg.toml(master)

| | | | |
|---|---|---|---|
| Port | Listenポート(default:8080) | | |
| LocalOnly | localhostでListenすることを強制 (default: false) | | |
| CertFile | SXG証明書のファイル(サーバ証明書,中間証明書, PEM形式 | | |
| KeyFile | SXG証明書の秘密鍵ファイル(PEM形式) | | |
| OCSPCache | OCSPデータのキャッシュファイルのパス | | |
| ForwardedRequestHeader | fetchする際に転送するリクエストヘッダ | | |
| [[URLSet]] | | | |
| | [URLSet.Sign] | 署名用の設定(必須) | |
| | | Domain | 署名するドメイン |
| | | PathRE | 署名を許可するPath(default: .*) |
| | | PathExcludeRE | 署名を許可しないPath |
| | | QueryRE | 署名を許可するQueryパラメータ (default: "") |
| | | ErrorOnStatefulHeader | statefulヘッダがあったらproxyさせるか削除するか |
| | | MaxLength | URLの最大長(default: 2000) |
| | [URLSet.Fetch] | AMPコンテンツをFetchする設定(オプション) | |
| | | Scheme | fetchを許可するschema (default: https) |
| | | SamePath | 署名と同じpathでfetchすることを強制させる |
| | | DomainRE | fetchを許可するドメインのパターン |
| | | Domain | fetchするドメイン |
| | | PathRE | fetchを許可するPATH (default:*.) |
| | | QueryRE | fetchを許可するQueryパラメータ (default: "") |

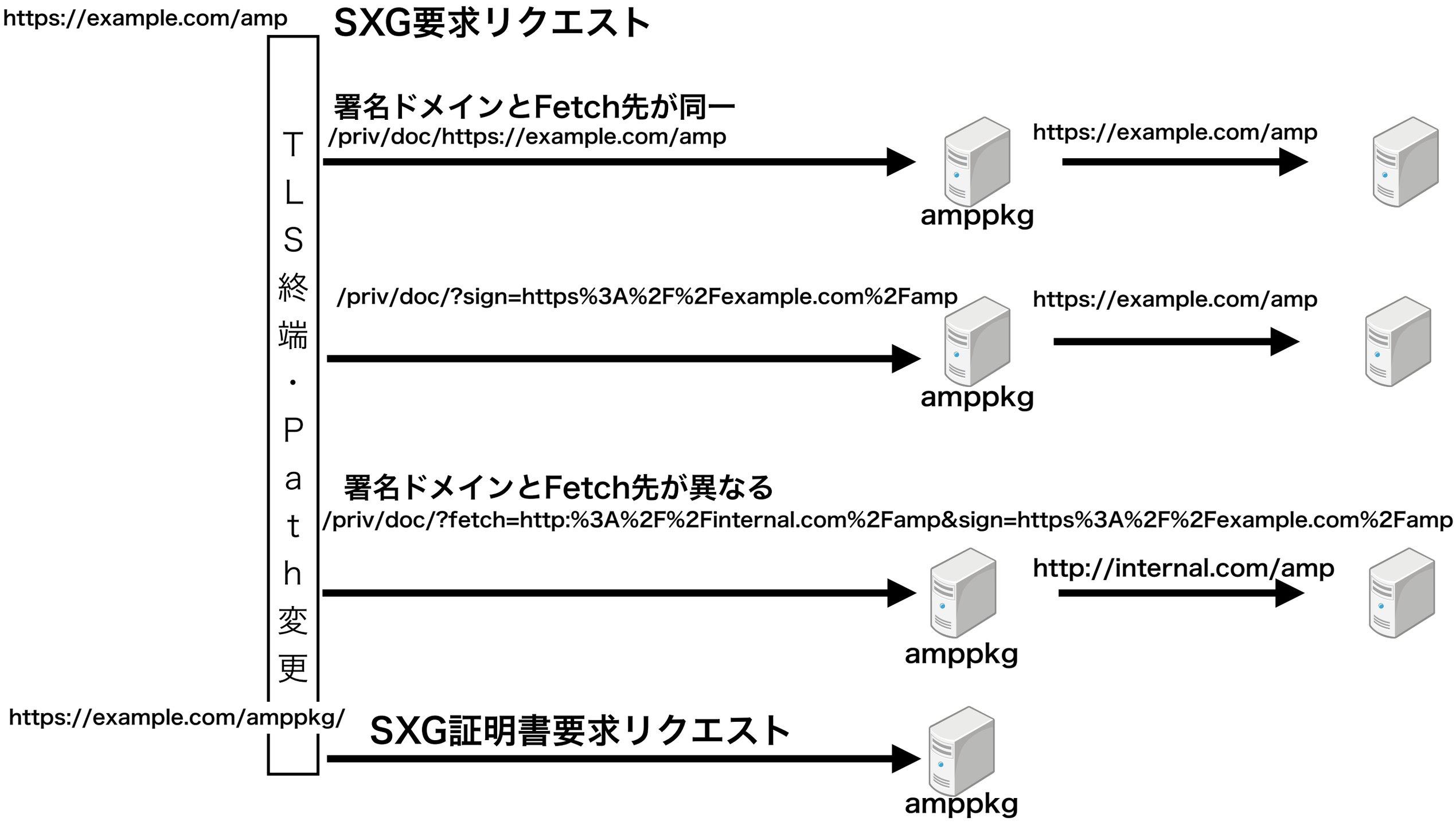
# amppkg

**コマンドラインフラグ**

| | |
|---|---|
| -config | configファイルの指定 |
| -developement | • CanSignHTTPExchange拡張がない証明書の利用を許可する<br>• localhost宛のURLだけ許可<br>• AMP-Cache-Transform/Acceptヘッダは必要ない |
| -invalidcert | CanSignHTTPExchange拡張がない証明書の利用を許可する |

**ログは標準出力のみ**

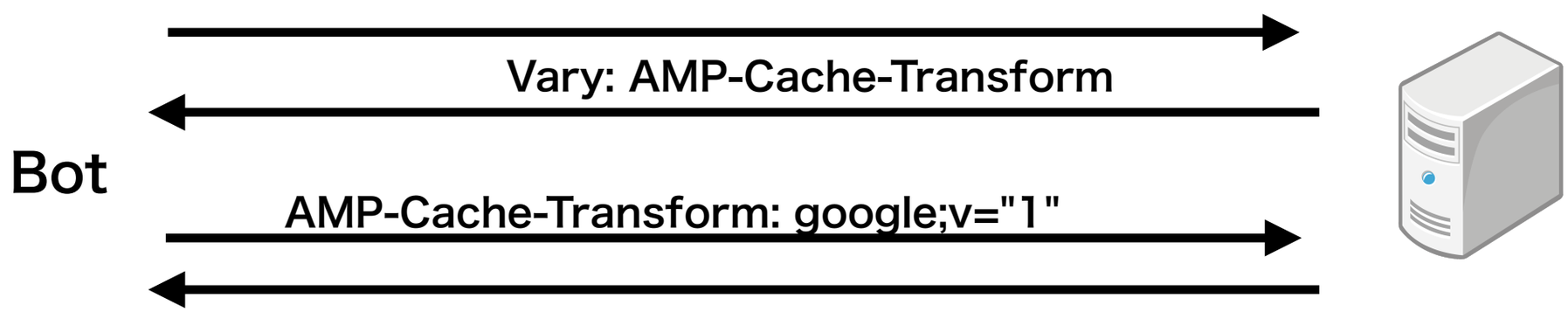
# amppkgへのリクエスト




amppkg前段でTLS終端をして、外部から直接/priv/docに接続させないこと

# Transformer

- これまでAMP CacheでGoogleがAMP最適化処理を行ってきた(AMPのrun time valueの入れ込みやinline化 etc)
- SXGでは署名した後では最適化が不可能。SXG化の前にやる必要がある。
- Transformerの指定をAMP-Cache-Transformerヘッダでネゴシエーションする。バージョンは6-8週間で上がる予定
  - AMP-Cache-Transform: google;v="1..3,5"
  - AMP-Cache-Transform: any

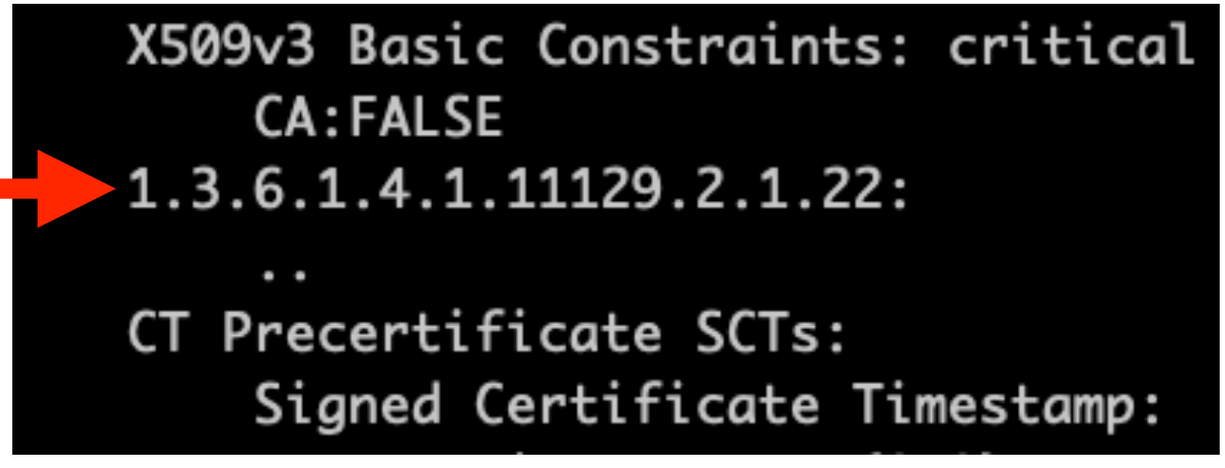



https://github.com/ampproject/amphtml/blob/master/spec/amp-cache-modifications.md
https://github.com/ampproject/amphtml/blob/master/spec/amp-cache-transform.md

# SXG証明書

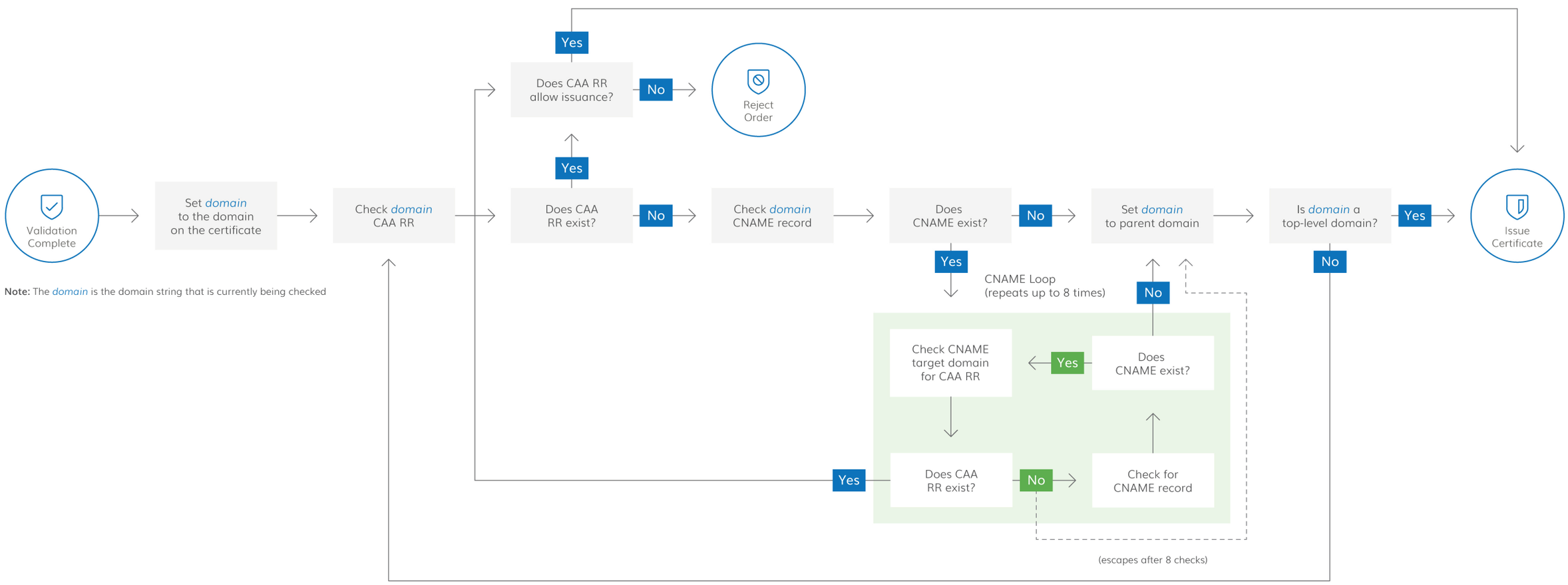
- SXGをサービスで提供するために必要不可欠
- 通常のECC証明書(secp256)にSXG拡張が付いたもの
- 通常のHTTPSサーバ用途の証明書に使うことは禁止
- 現在Digicert社のみ発行
- 2019年5月1日より
  - 証明書の有効期限を90日に制限
  - DNSCAAレコード登録を必須化

SXG拡張



# CAA導入の際の注意事項

example.com. CAA 0 issue "digicert.com;cansignhttpexchanges=yes"

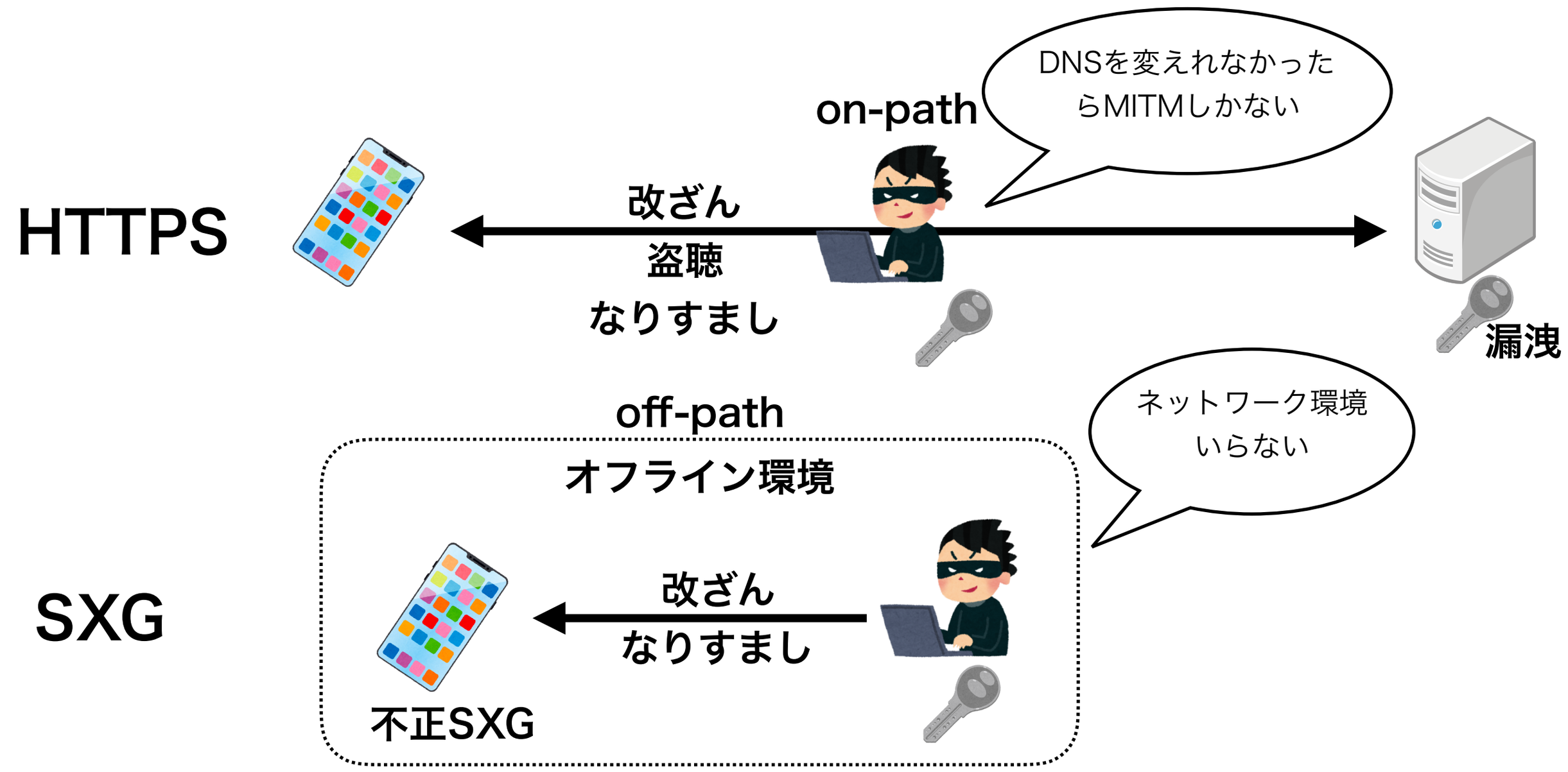



https://www.digicert.com/dns-caa-rr-check.htm

- CNAME付けているレコードには他レコード付けれない
- CNAME先のCAAを見に行くがCNAMEの親ドメインを辿らない(RFC6844 Errata)
- CNAME先のCAAがなければ親ドメインのCAAを参照しに行く。

# HTTPSとSXGのセキュリティの違い

## 秘密鍵が漏洩した場合

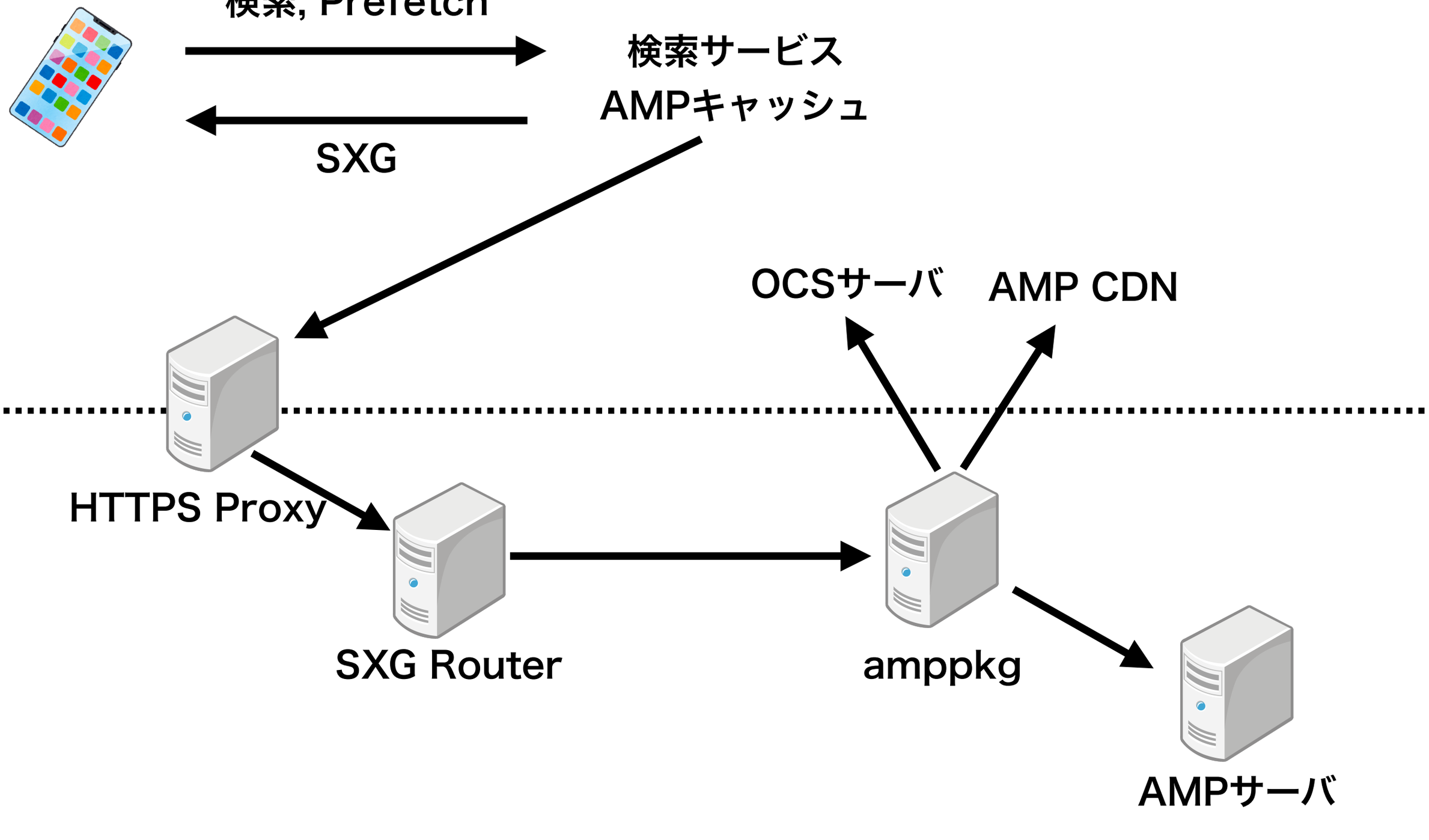



**CAAで証明書の不正発行・誤発行を防ぎ**

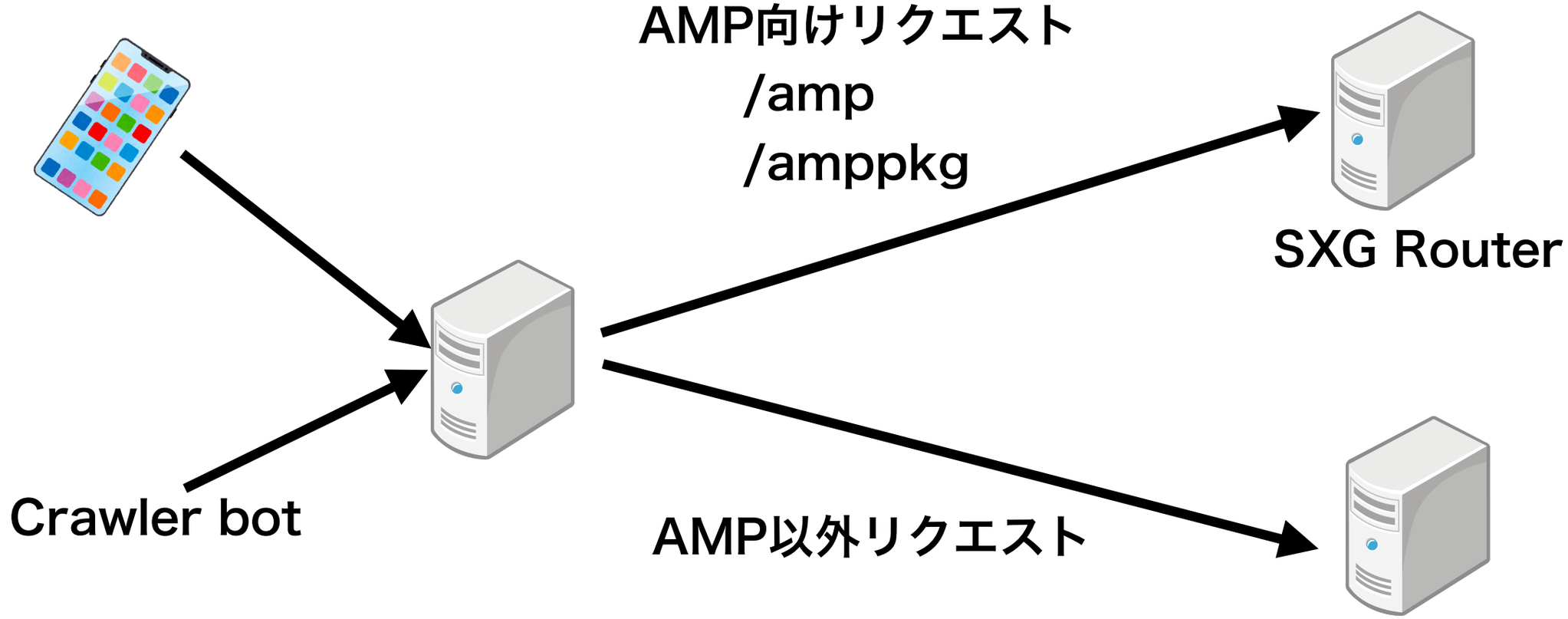
**短期間(90日)のSXG証明書で危殆化時のリスクを低減**

# SXG/AMP
# システム事例

# AMP/SXGシステム概要

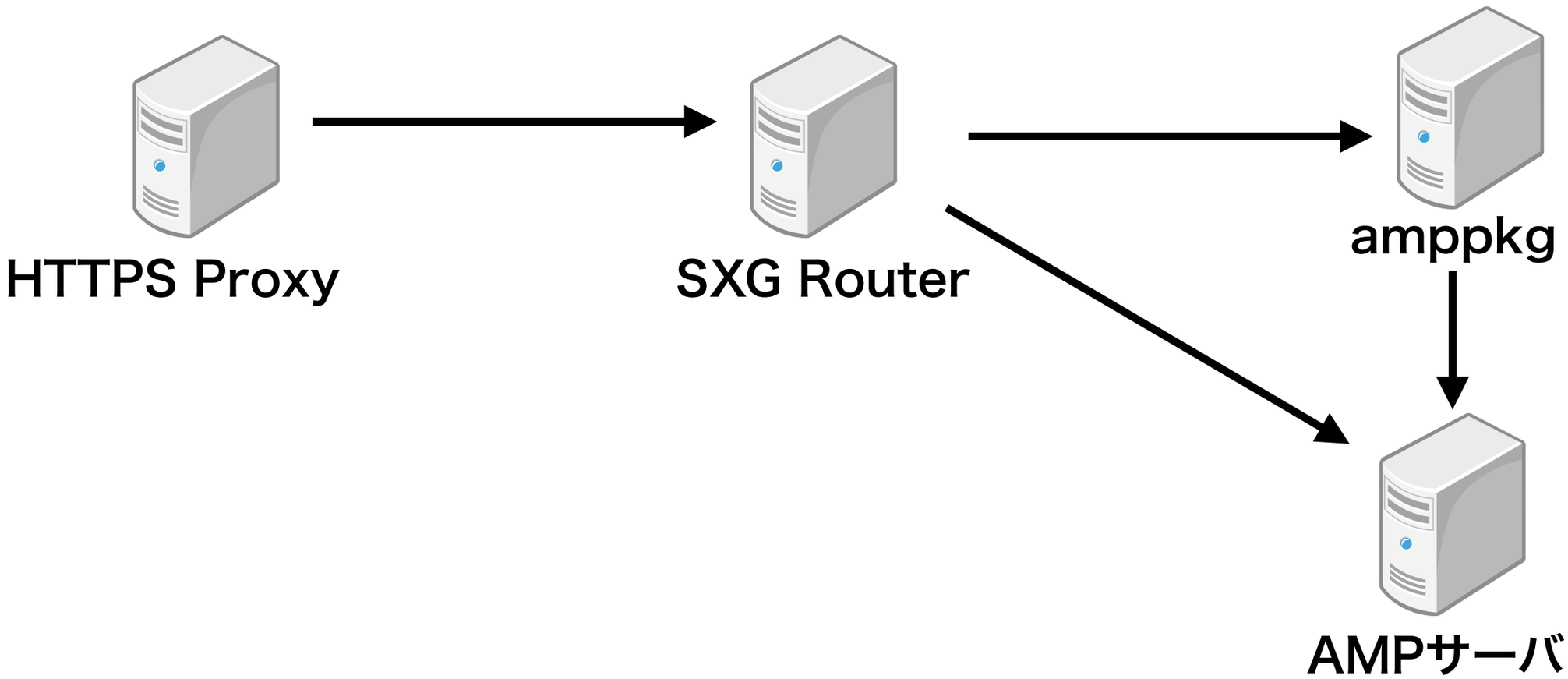


# HTTPS Proxy




- インターネットからのHTTPS接続の終端
- キャッシュ機能
- 内部サービスへの振り分け(/amp AMPサービス, /amppkg SXG証明書取得)

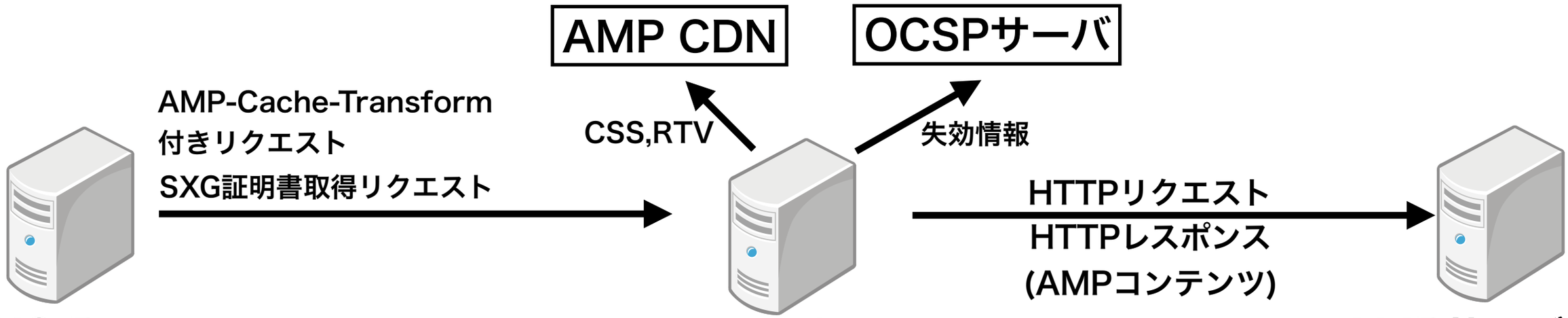
# SXG Router




- CrawlerにSXG対応であることを知らせる(Vary: AMP-Cache-Transform)
- SXG取得リクエスト(AMP-Cache-Transform: google;v=..)が来たらamppkにふる。
- リクエストを amppkg用のquery parameterに変更

# amppkg

https://github.com/ampproject/amppackager




- AMPコンテンツからSXGを生成し提供する(最大4MB)
- クライアントの時間ずれを考慮して署名期間は１日前から６日後
- SXG生成時にAMP最適化(Transform)を行う
- SXG証明書とOCSPをまとめたデータ(cert+cbor)を生成し、提供する
- パブリックキャッシュに適さないヘッダのチェックや削除を行う
- statusCode 200以外は基本スルー(proxy)

# SXG導入振り返り

# SXG/AMPシステム導入、振り返り

1. 既存インフラへの影響を最小限にして、導入が可能だった。
2. ユーザへの直接リクエストの影響をあまり気にせずにすんだ(主要リクエストがBotのクローリングであったため)
3. 事前にStaging環境でのテストが限定的だった。Botクロールや AMPキャッシュ化の試験はリリース後にしかできない。

# SXG/AMPシステム導入に関する注意事項

- キャッシュ可能なコンテンツか？ (SXG禁止ヘッダ)
- 個人情報やセキュリティ情報が入っていないか？
- Same Originになったらエラーにならないか？ (e.g. AMP-iframe)
- Soft404(not foundを200で返す)とかしていないか？

# まとめ

originを入れ替えるSigned HTTP Exchangesは、これまでのWebプロトコルの概念を根本的に変える技術です。

この技術によって Publisher/Distributor それぞれの関係が将来大きく変わる可能性があります。

今後、SXGが天使となるのか、それとも悪魔となるのか、
皆さんでその影響をよく考えた上でSXGを判断して下さい。