iijlab セミナー

Encrypted ClientHello について

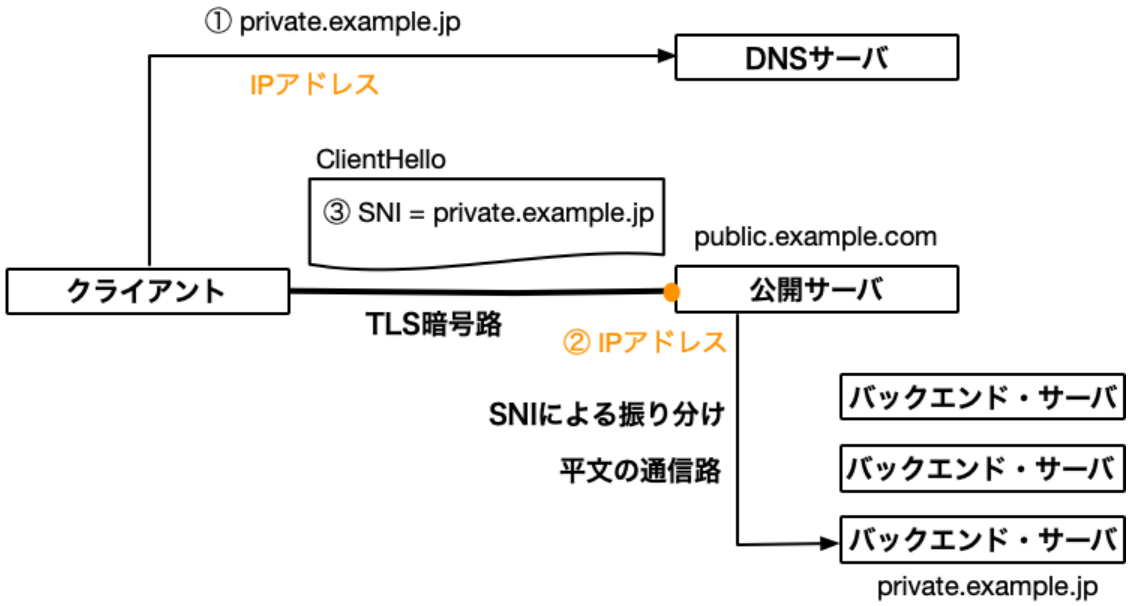
IIJ 技術研究所
山本和彦

2025.9.30 17:30

おしながき

- ECH(Encrypted ClientHello)の必要性
- ECHの仕様とその設計思想
- ECHの周辺技術
- ECHの普及戦略と普及状況
- ECHの実装の体験談

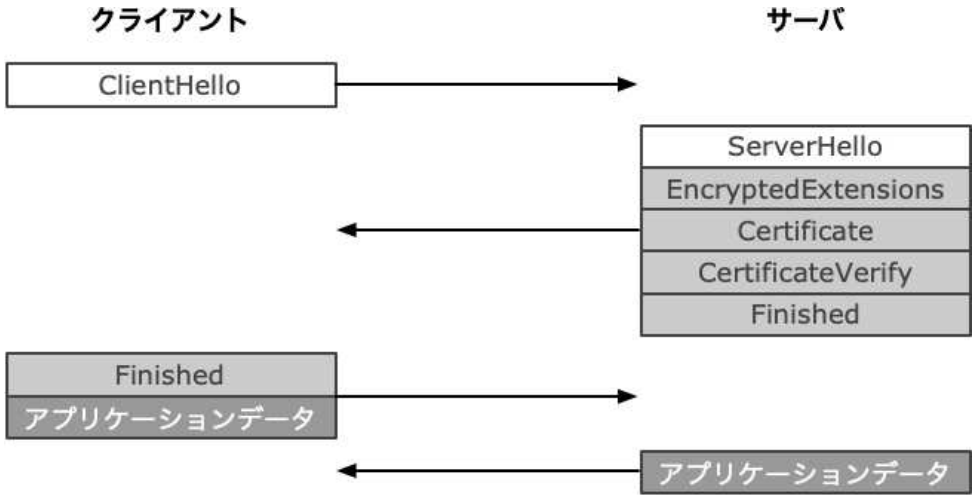
アクセス先情報の漏洩ポイント



- バックエンド・サーバの名前でDNSを検索すると
  公開サーバのIPアドレスが返る

TLS 1.3 のフルハンドシェイク

- EncryptedExtensions から暗号化
  - ハンドシェイク・トラフィック用の鍵
- アプリケーション・データから鍵が変わる
  - アプリケーション・トラフィック用の鍵



ClientHello の構造

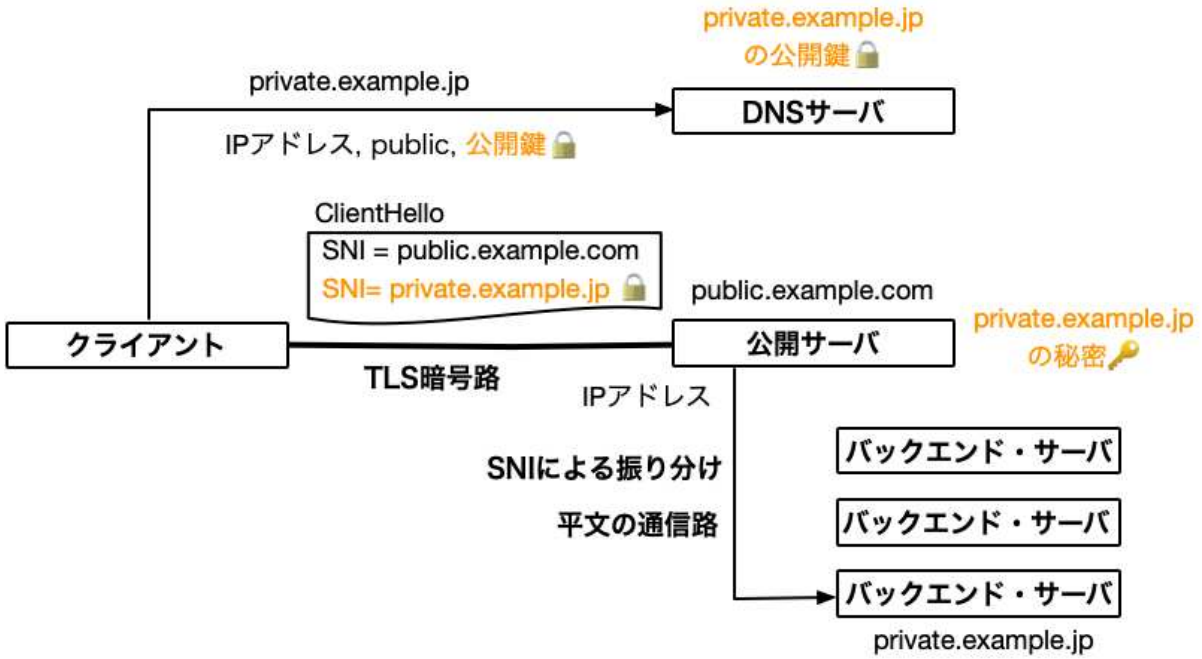
- ClientHelloの生命線は拡張フィールド
  - legacy_ で始まるフィールドは、TLS 1.3 ではダミー

```
struct {
  ProtocolVersion legacy_version = 0x0303;
  Random random;
  opaque legacy_session_id<0..32>;
  CipherSuite cipher_suites<2..2^16-2>;
  opaque legacy_compression_methods<1..2^8-1>;
  Extension extensions<8..2^16-1>;
} ClientHello;
```

ClientHello の拡張

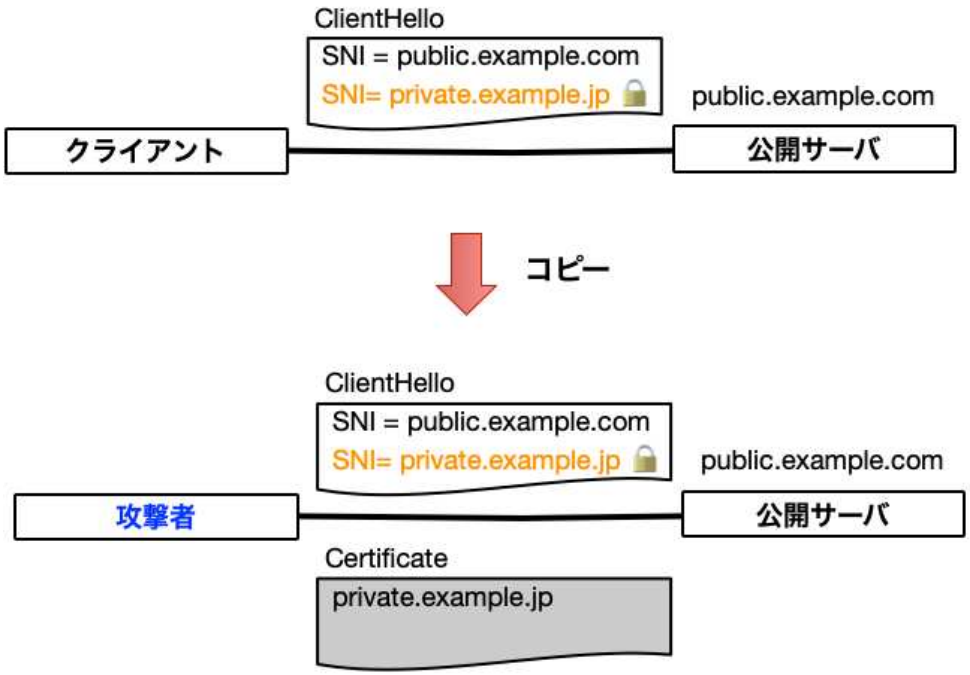
- 鍵交換
  - 楕円曲線DiffieHellman (ECDH)
- サポートしているTLSのバージョン
  - cf `legacy_version`
- サーバ名
  - SNI (ServerName Indication)
  - プライバシ性が高い
- アプリケーション層プロトコル
  - ALPN (Application Layer Protocol Negotiation)
    - HTTP/2: `"h2"`
    - WebRTC: `"webrtc"`
  - クライアントが提案 (平文、ClientHello)
  - サーバが選択 (暗号文、EncryptedExtensions)
  - プライバシ性が高い

暗号化SNI



- 公開鍵の解釈はどちらでもOK
  - ○ バックエンドサーバの公開鍵
  - ○ 公開サーバの公開鍵

カット＆ペースト攻撃 (1)

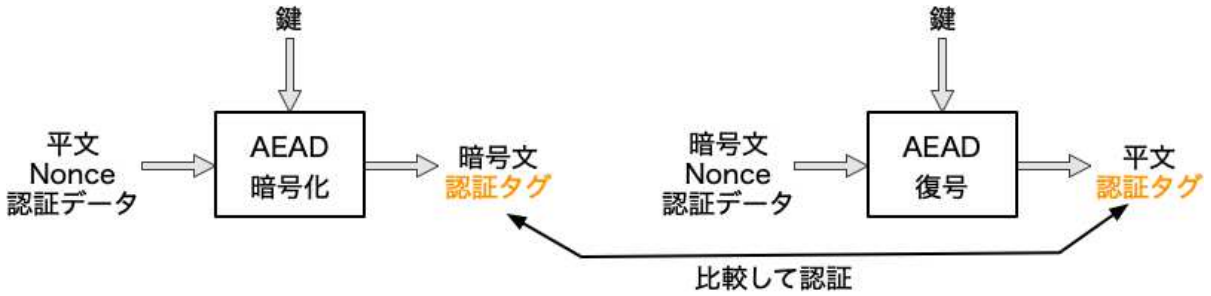

ClientHello
SNI = public.example.com
SNI= private.example.jp
public.example.com
クライアント
公開サーバ
コピー
ClientHello
SNI = public.example.com
SNI= private.example.jp
public.example.com
攻撃者
公開サーバ
Certificate
private.example.jp

カット＆ペースト攻撃 (2)

- 攻撃を許す理由
  - リプレイ攻撃の一種
  - ClientHelloと暗号化されたSNIが結束されてないから

暗号化されたSNIとClientHelloを結束する

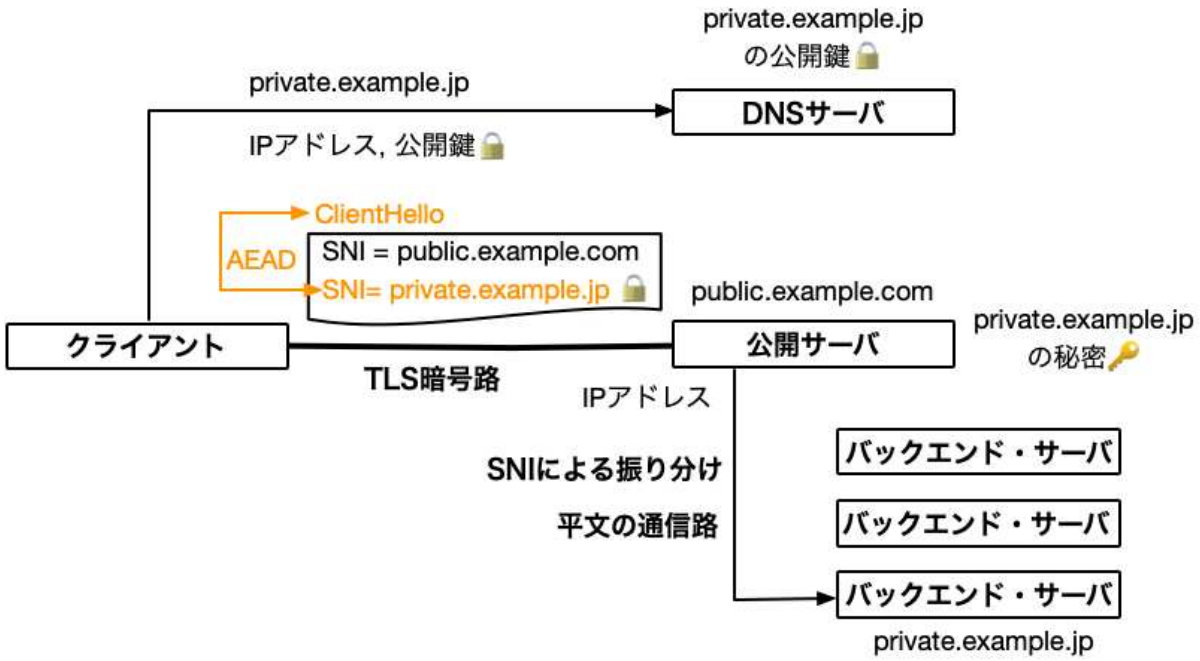
- AEAD
  - Authenticated Encryption with Associated Data
  - 暗号化と認証を同時にやる共通鍵暗号の暗号モード



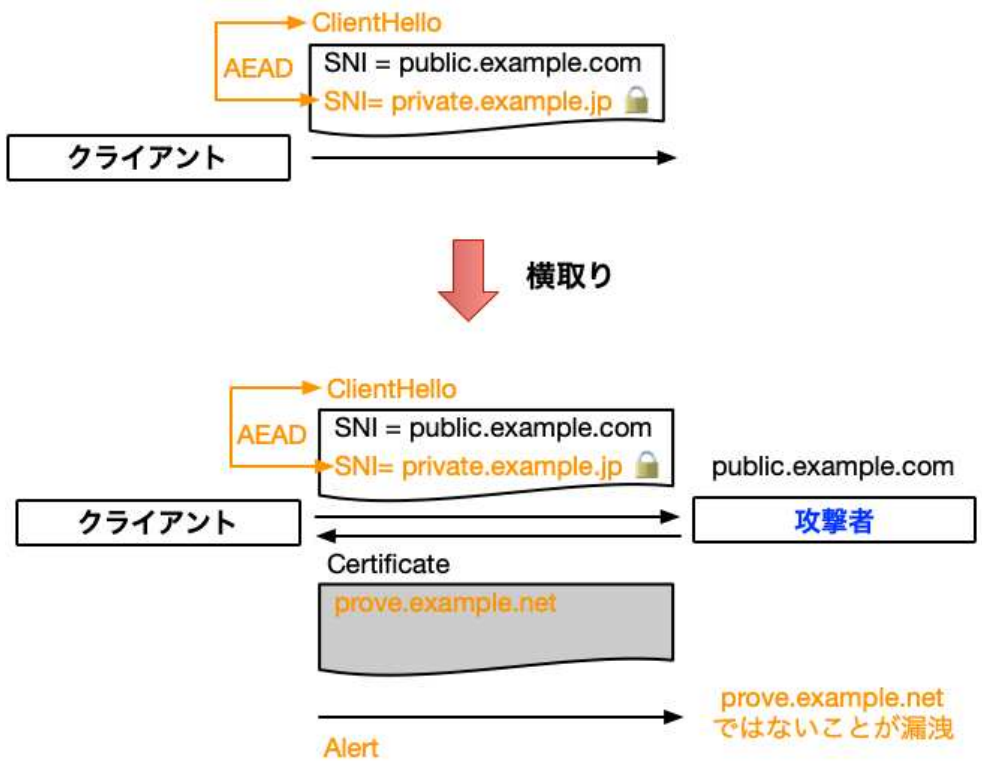
- 結束方法
  - 平文：SNI
  - 認証データ：ClientHello

ハイブリッド暗号化SNI

- 共通鍵暗号(AEAD)＋公開鍵暗号
  - 共通鍵暗号でSNIを暗号化、ClientHelloと結束
  - それをさらに公開鍵暗号で暗号化する

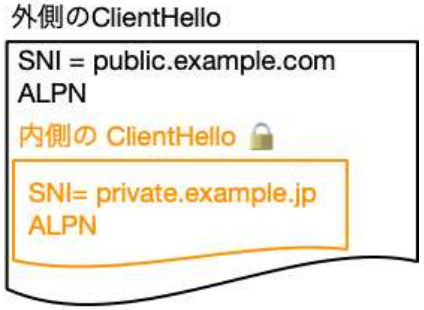


クライアント・リアクション攻撃



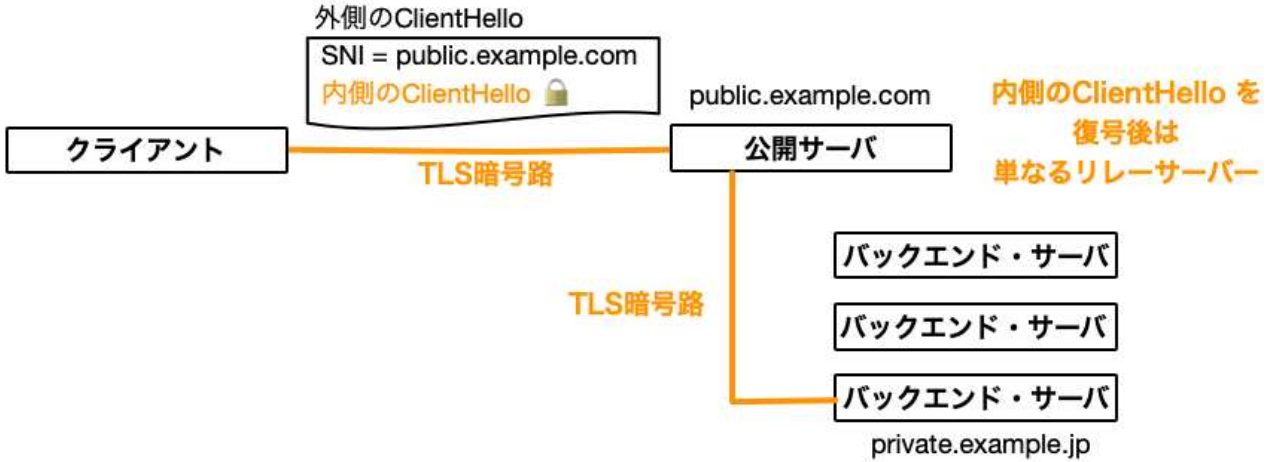
考察

- SNI 単体の暗号化
  - 暗号化できるのは SNI のみ
  - 攻撃の余地を残す
- ClientHello 全体を暗号化してみる
  - すべての拡張を暗号化できる
  - 攻撃者が作り得ない暗号路で証明書を返す



Encrypted ClientHello の正常系

- ECHのモデル
  - 公開サーバは外側のClientHelloを処理
    - 内側のClientHelloを復号
  - バックエンドサーバは内側のClientHelloでハンドシェイク



SVCB / HTTPS RR

- research.cloudflare.com

```
alpn=["h2"]
ipv4hint=[104.18.4.139,104.18.5.139]
ipv6hint=[2606:4700::6812:48b,2606:4700::6812:58b]
ech=[(
  4,
  DHKEM(X25519, HKDF-SHA256),
  "9976bac08ccc8aa8fb21ad21fd628982 \
   43a17a8ad732c7bf69ef0c2ea7635022",
  [(HKDF-SHA256,AES-128-GCM)],
  0,
  "cloudflare-ech.com",
  []
)]
```

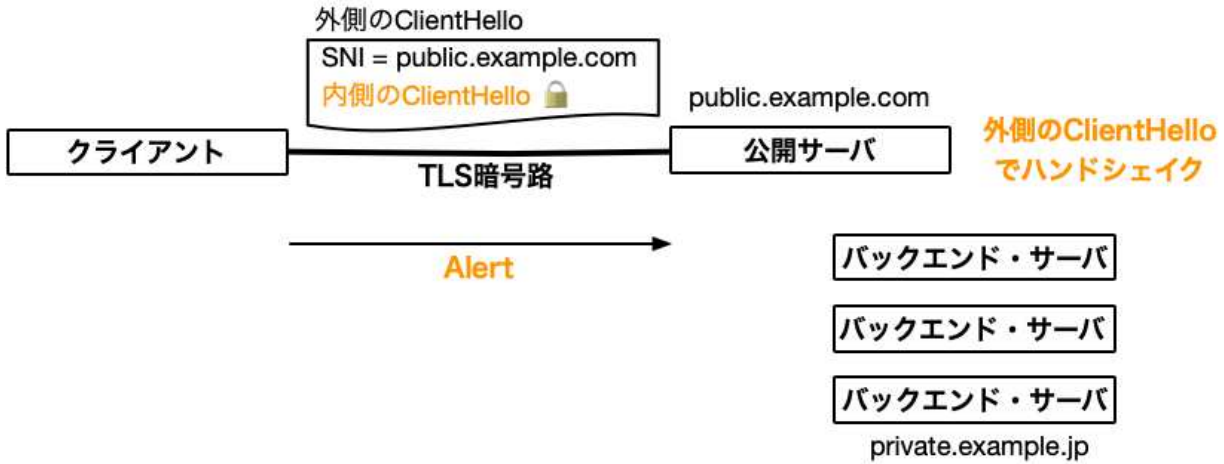
HPKE

- Hybrid Public Key Encryption
- 公開鍵暗号と共通鍵暗号のハイブリッド
  - 古典的な手法を今風に標準化
  - 鍵交換: 楕円曲線Diffie-Hellman (ECDH)
  - 鍵派生：HKDF
  - 共通鍵暗号：AEAD モード
- API を定義

```
def Seal<MODE>(pkR, info, aad, pt, ...):
  enc, ctx = Setup<MODE>S(pkR, info, ...)
  ct = ctx.Seal(aad, pt)
  return enc, ct
```

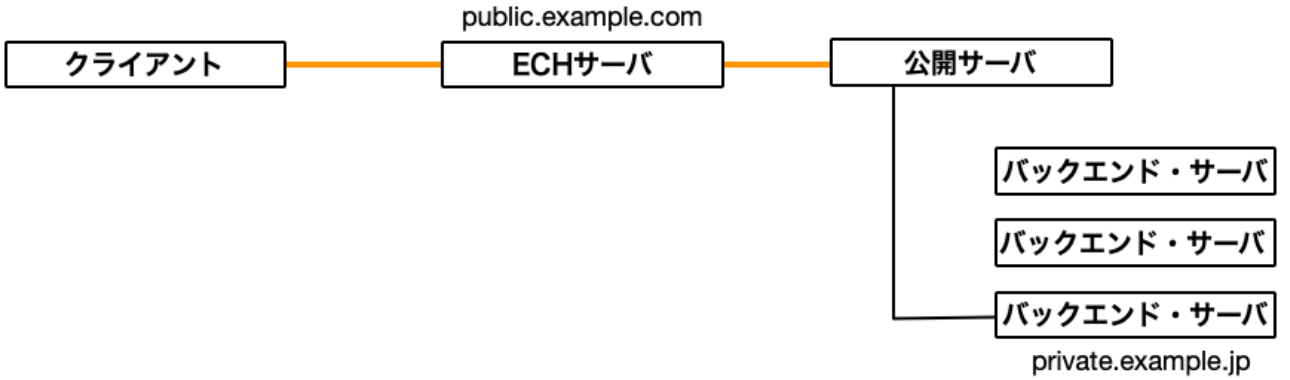
Encrypted ClientHello の異常系

- 公開サーバ
  - 内側の暗号化されたClientHelloの復号に失敗
  - 外側のClientHelloを使ってクライアントとハンドシェイク
    - ServerHello の random で失敗を通知
- クライアント
  - 暗号路の中で、コネクションを切る



ECHサーバの導入の実際

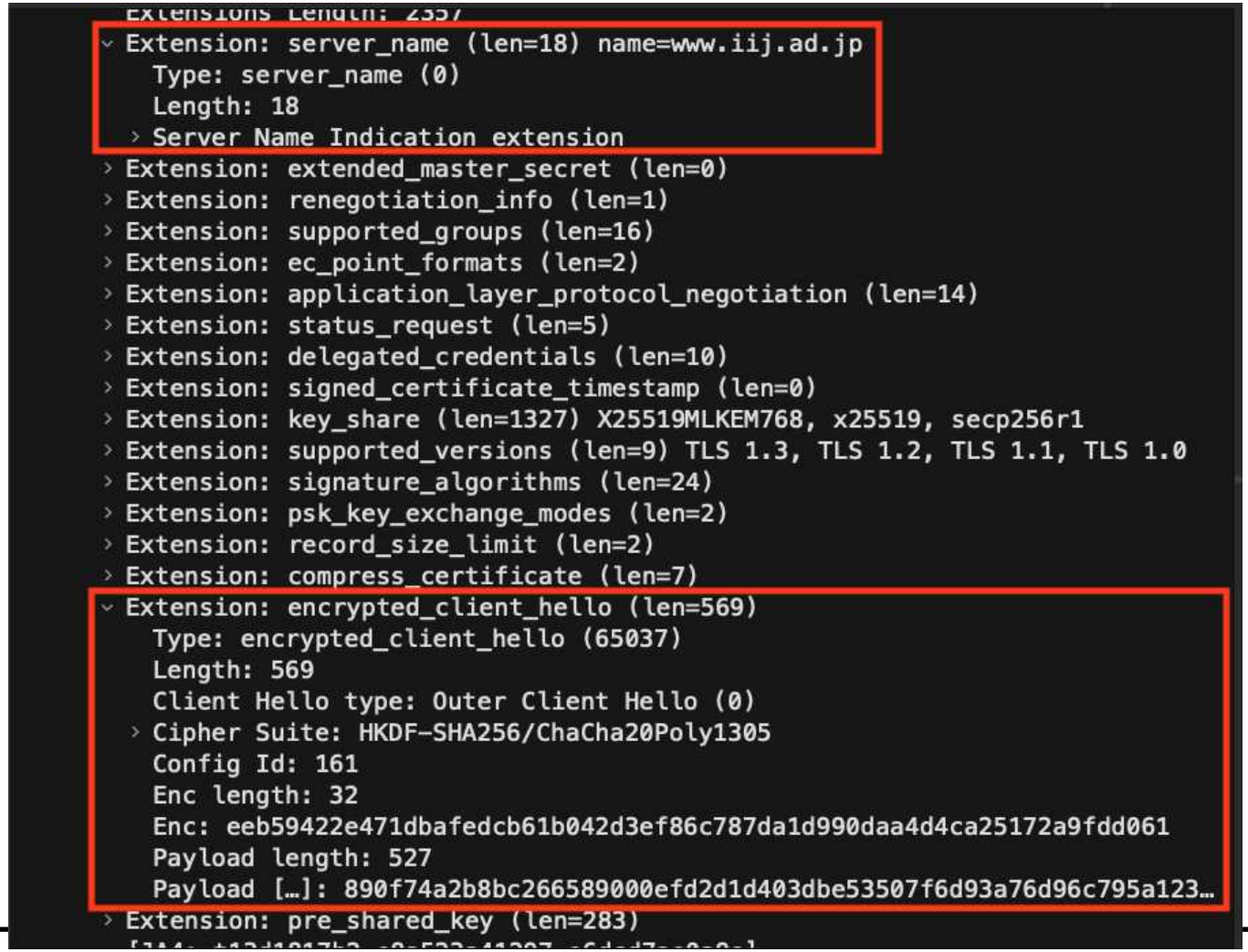
- 2つのモード
  - 分担モード
    - 公開サーバの前にECHサーバを設置
  - 共有モード
    - 公開サーバでECHをサポートする



普及戦略

- Do not stick out
  - 目立たない
- ECHをサポートしているクライアント
  - ECHに対応しているサーバ：適切なECH拡張
  - ECHに対応していないサーバ：デタラメなECH拡張
    - グリーシングと呼ばれる
- デタラメなECH拡張が普及
  - 適切なECH拡張は埋もれて目立たない
  - サーバがECHに対応すると
    - デタラメなECH拡張 → 適切なECH拡張
    - この変化も目立たない

Firefox の ECH グリーシング

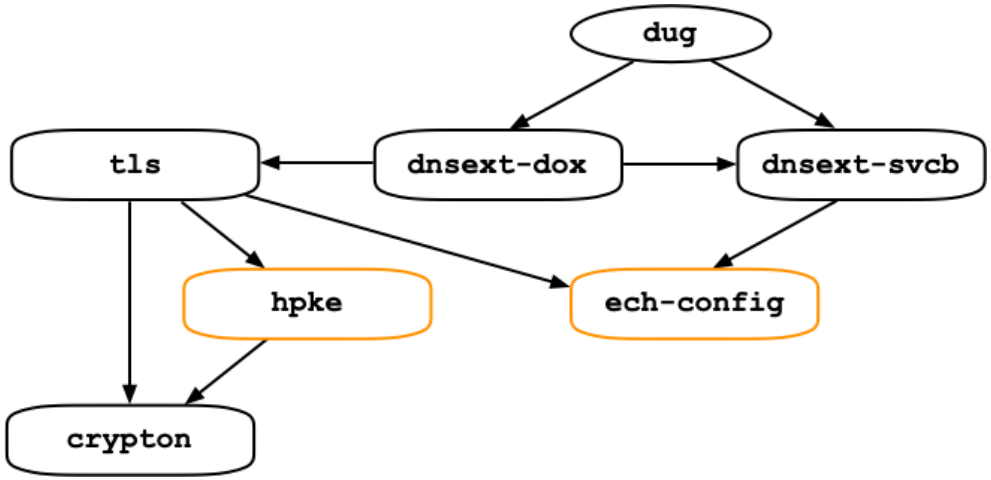


ECHの普及状況

- ブラウザ
  - Firefox
  - Chrome、Edge、Opera (ChatGPT 調べ)
- サーバ
  - A RR か CNAME RR を持つ2601個の www.〇〇〇.com の内 HTTPS RR の ech パラメータを持つのは 288個 (11.1%)
  - Cloudflare が中心

実装の体験談(1)

- ライブラリの構成
  - tls と dnsext-svcb の独立性を保つこと
  - ech-config が crypton に依存しないこと
  - hpke が拡張可能であること



実装の体験談(2)

- サーバの実装
  - 共有モードのみ
    - 内側のClientHelloが復号できたら、それを使う
    - そうでなければ、外側のClientHelloを使う
  - Full ハンドシェイクの実装は面倒だが簡単だった
  - HRR(Hello Retry Request) で沼にハマった
    - 原因：トランスクリプト・ハッシュの計算ミス
    - トランスクリプト・ハッシュ：ハンドシェイク・メッセージのハッシュ値
    - TLSセッションや「鍵」とハンドシェイク・メッセージを結合する
    - 最終的に Finished で検証
    - HRR だけは、ハッシュのハッシュを取る
    - トランスクリプト・ハッシュトレースする機能を追加
- ローカルでの相互接続性
  - picotls、boringssl、OpenSSL(DEfO)、NSS

実装の体験談(3)

- クライアントの実装
  - サーバが外側と内側のどちらのClientHelloを使うか分からない
  - 両方を保存する仕組みが必要
- ローカルでの相互接続テスト
  - picotls、boringssl、OpenSSL(DEfO)、NSS
  - サーバの公開鍵(ECH設定)の書式は定義されている
  - サーバの秘密鍵の書式は決まっていない
    - サーバの実装ごとにバラバラ
    - ech-gen を作成: 僕が知りうる限りの書式で秘密鍵を生成
- 公開サーバへの接続
  - あらかじめ HTTPS RR を検索し ech パラメーターを抽出する必要
    - ech-lookup を作成

ブログ

- Encrypted Client Hello の仕様
  - あどけない話
  - https://kazu-yamamoto.hatenablog.jp/entry/2025/03/12/193201
- Encrypted Client Hello の実装
  - あどけない話
  - https://kazu-yamamoto.hatenablog.jp/entry/2025/03/14/150120
- Encrypted ClientHelloの仕組み
  - IIJ エンジニアブログ
  - https://eng-blog.iij.ad.jp/archives/32414