TLS 1.3 の標準化動向

2018.2.14

山本和彦
@kazu_yamamoto

TLS 1.2 は飛ばします

- TLSの動向
  - https://www.iij.ad.jp/dev/report/iir/031/03_01.html
  - 山本和彦
- TLS 1.3
  - http://seminar-materials.iijlab.net/iijlab-seminar/iijlab-seminar-20170110.pdf
  - 山本和彦
- いまなぜHTTPS化なのか？
  技術者が知っておきたいSEOよりずっと大切なこと
  - https://employment.en-japan.com/engineerhub/entry/2018/02/14/110000
  - 大津繁樹

TLS 1.3 の特徴

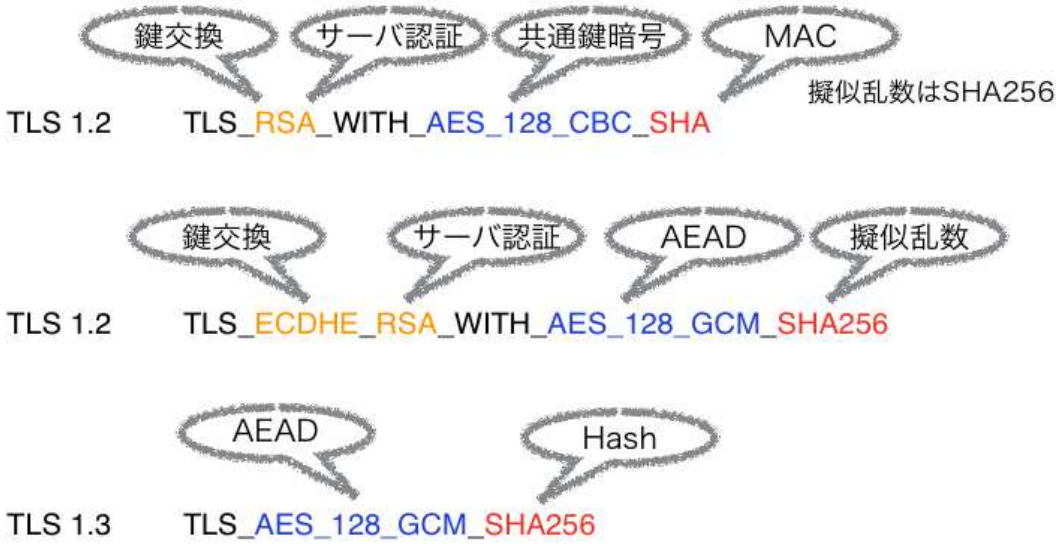
安全性の向上	老朽化した暗号技術の排除 再ネゴシエーションの排除 圧縮の排除 前方秘匿性 (サーバ認証と鍵交換の分離)
機密性の向上	クライアント証明書の暗号化 サーバからの拡張の暗号化
速度の向上	1RTT 再開 (前方秘匿性あり) 0RTT 再開 (前方秘匿性なし)

TLS のハンドシェイク

TLS 1.2	TLS 1.3
フルハンドシェイク	フルハンドシェイク
	再トライ
再開 (resumption)	
PSK (Pre-Shared Key)	再開 + PSK
	0RTT
再ネゴシエーション	

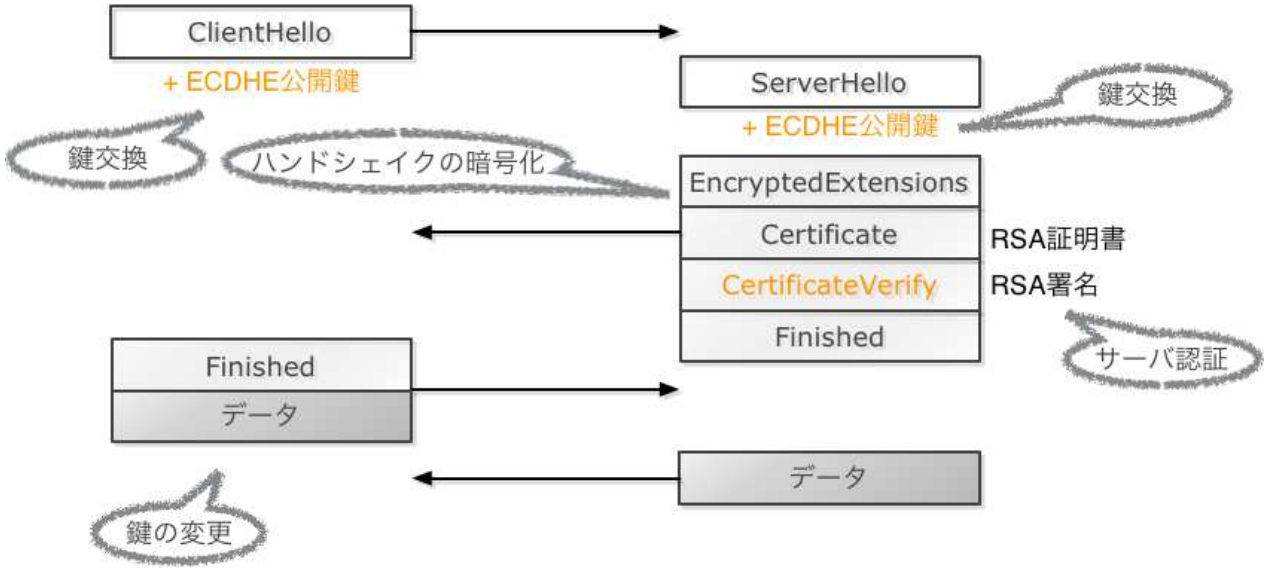
暗号スイート

- TLS 1.2 の最後の解釈は難しい
  - MAC と擬似乱数の意味
  - 擬似乱数は SHA256 以上 (SHAと書いてSHA256と読む)
  - AEAD だと MAC の意味はない
- TLS 1.3 では鍵交換とサーバ認証が削除された



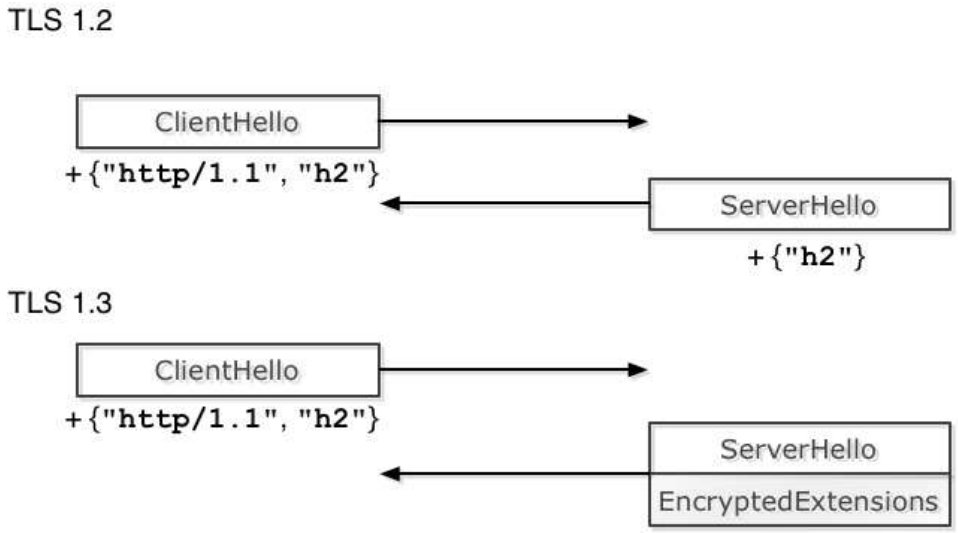
フルハンドシェイク

- 前方秘匿性のある 1RTT
  - (EC)DHEを使ったクライアントからの鍵交換
  - サーバ認証との分離
  - TLS 1.2 のフルハンドシェイクでも false start を使えば前方秘匿性のある 1RTT が実現できる



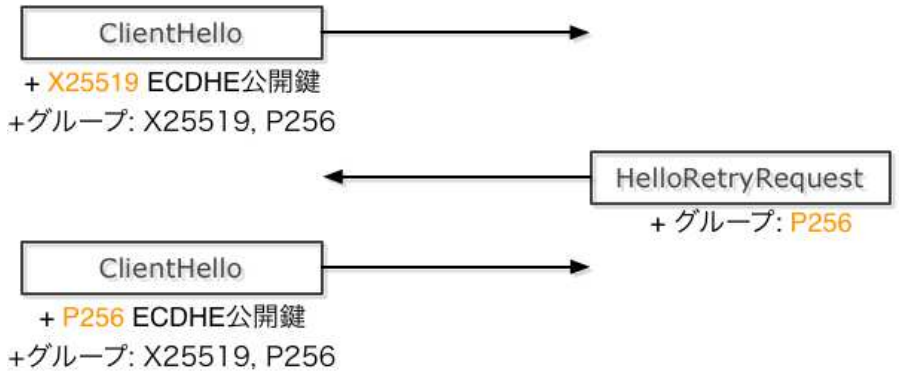
サーバからの拡張の暗号化

- ALPNの例
  - Application Layer Protocol Negotiation



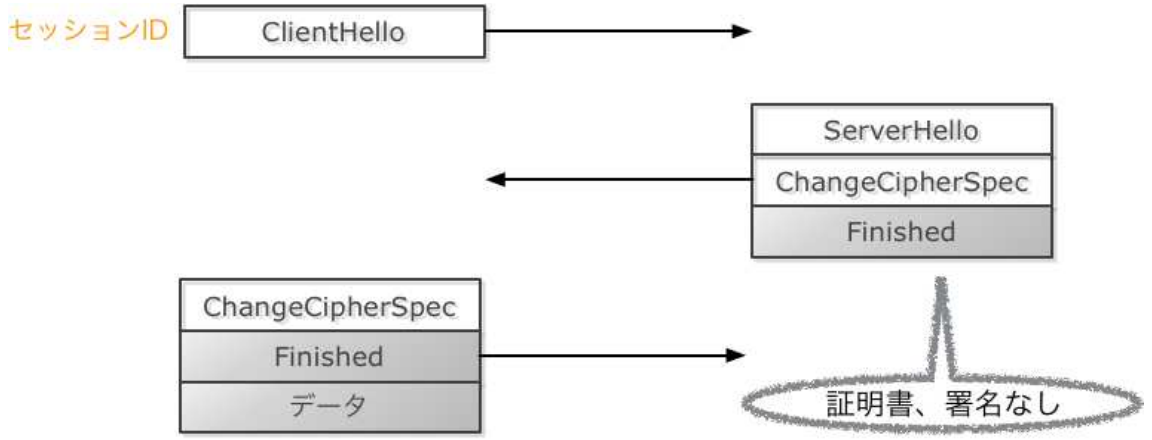
再トライ

- 鍵交換の拡張
  - 複数の公開鍵を送ってもよい
  - 帯域を節約するために1つの公開鍵だけを送ってもよい
- HRR: HelloRetryRequest
  - サーバが送られて来た公開鍵を受け入れられないが、NegotiatedGroup拡張に受け入れられる楕円曲線があれば、再トライを促せる



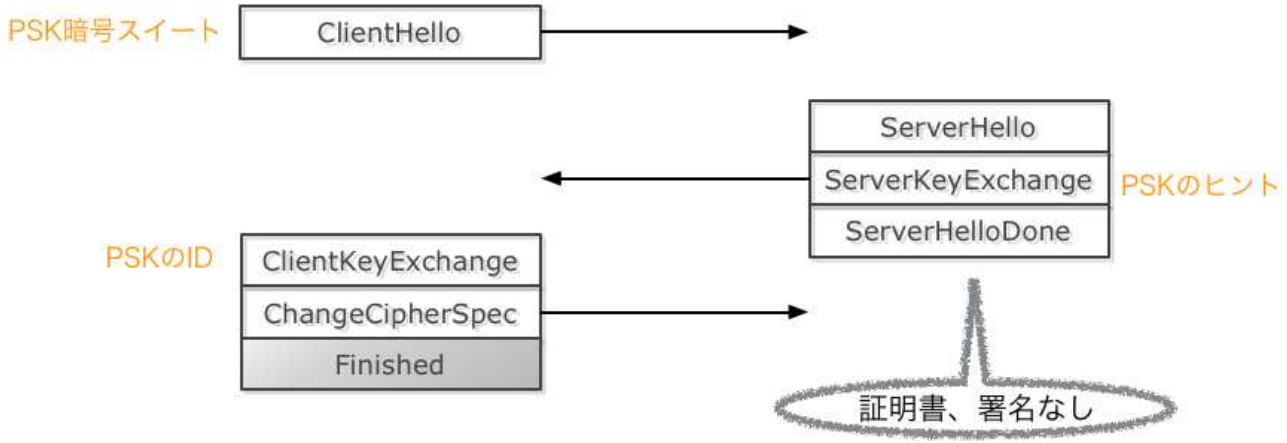
TLS 1.2のセッションの再開

- セッションの状態を再開できる
  - Resumption
  - 公開鍵暗号を使わずに前のセッションの通信相手を認証
  - セッションID: サーバが状態を保つ
  - セッションチケット: サーバが状態を持たない



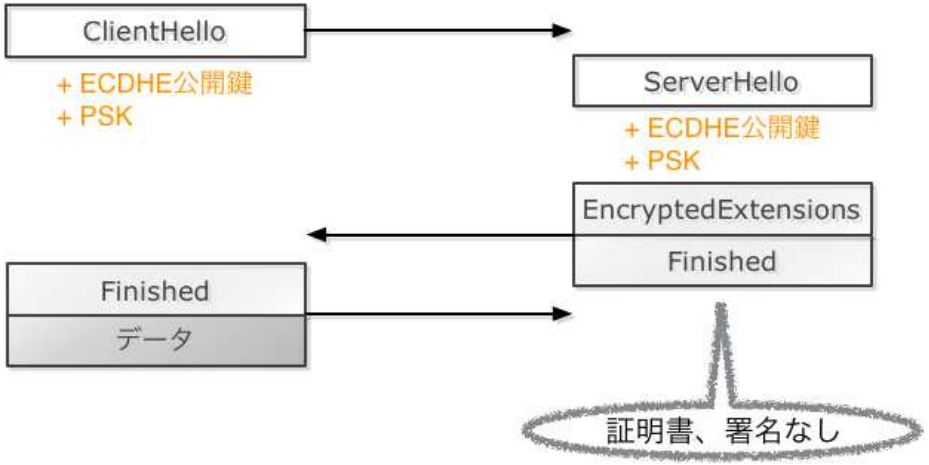
TLS 1.2の Pre-Shared Key

- 手で設定した秘密を使う
  - 公開鍵暗号を使わずに定められた通信相手を認証



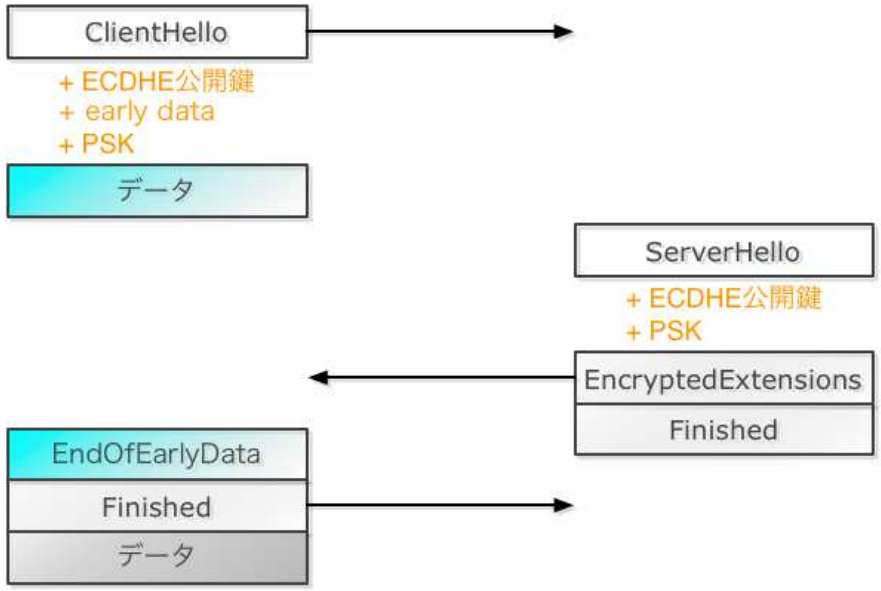
TLS 1.3 のPSK

- 2つの用途
  - 外部 PSK: TLS 1.2 での PSK (Pre-Shared Key)
  - 再開 PSK: TLS 1.2 での再開 (resumption)
- 前方秘匿性のある1RTT 再開



0RTT

- PSK を元に0RTTデータを暗号化
  - (EC)DHEによる共有鍵が得られる前なので前方秘匿性はない
  - リプレイ攻撃される可能性もある



標準化の歴史

- 2016.10.26 draft 18
  - 2016 11 IETF 97 Seoul
- 2017.03.10 draft 19 ← 2017.03.13 WGLC
  - 2017.03 IETF 98 Chicogo
- 2017.04.28 draft 20
- 2017.07.03 draft 21 ← 2017.07.03 WGLC 2
  - 2017.07 IETF 99 Prague
  - 2017.11 IETF 100 Singapore (best remote participant prize)
- 2017.11.29 draft 22
- 2018.01.05 draft 23 ← 2018.01.12 WGLC 3
  - 2018.03 IETF 101 London

draft 18 (1/2)

- 多くの実装が長期に渡り使用
- 4つのハンドシェイク
- 鍵交換に ECDHE X25519、X448 を利用
  - NIST P256 も使える
- RSA署名で RSASSA-PSS を利用
  - Probabilistic Signature Scheme
  - RSASSA-PKCS1-v1_5 の代わり
- 3段階の鍵スケジューリング
  - ealry traffic secret
  - handshake traffic secret
  - application traffic secret
  - ChangeCipherSpec はない

draft 18 (2/2)

- TLS 1.2 と互換性のない Server Hello
  - TLS 1.2 簡略化版

```
struct{
    ProtocolVersion server_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suite;
    CompressionMethod compression_method;
    Extension extensions<0..2^16-1>;
} ServerHello;
```

  - TLS 1.3

```
struct {
    ProtocolVersion version;
    Random random;
    CipherSuite cipher_suite;
    Extension extensions<0..2^16-1>;
} ServerHello;
```

draft 19 (1/2)

- 0RTT
  - early_data と ticket_early_data_info 拡張を統合

```
struct {} EarlyDataIndication;
struct {
    uint32 max_early_data_size; // early data を何バイト受け取るか
} TicketEarlyDataInfo; // NewSessionTicket用
```

↓

```
struct {
    select (Handshake.msg_type) {
        case new_session_ticket: uint32 max_early_data_size;
        case client_hello: Empty;
        case encrypted_extensions: Empty;
    };
} EarlyDataIndication;
```

draft 19 (2/2)

- 0RTT
  - end_of_early_dataがアラートからハンドシェークメッセージへ

```
end_of_early_data(1)
  ↓
struct {} EndOfEarlyData;
```

    - Client Finished に EndOfEarlyData を含めていない不備
  - 仕様の不備：メッセージに格上げしたら Finished に含めるべきなのに含めていない！

draft 20

- 鍵スケジューリングで使う文字列を短くした
  "client early traffic secret" → "c e traffic"

draft 21 (1/2)

- Security Review of TLS1.3 0-RTT
  - `https://github.com/tlswg/tls13-spec/issues/1001`
- NewSessionTicket に nonce

```
struct {
    uint32 ticket_lifetime;
    uint32 ticket_age_add;
    opaque ticket_nonce<1..255>;
    opaque ticket<1..2^16-1>;
    Extension extensions<0..2^16-2>;
} NewSessionTicket;
```

  - ticket が同じでも、nonce のおかげで、異なるPSK鍵が生成される

```
PSK = HKDF-Expand-Label(resumption_master_secret,
                        "resumption", ticket_nonce, Hash.length)
```

    - ticket が異なることが保証できるなら nonce は空でもよい
    - 仕様に不備あり

draft 21 (2/2)

- 0RTT に対するリプレイ攻撃への対策
  - early-data はリプレイ攻撃にさらされる
  - Single-use チケット
    - チケットの有効期間、情報を保持しないといけない
  - Client Hello recording
    - 比較的短い期間、情報を保持すればいい
- チケットの鮮度検査
  - 有効期限、age、RTT などの関係が明記された

draft 22 (1/2)

- Middlebox を騙せ！
  - ServerHelloの書式をTLS 1.2と同じにする
    - バージョンはTLS 1.2を指定する
      - サーバーが選択したバージョンは、supported_versions拡張で
        クライアントへ伝える
    - セッションIDが復活
    - 圧縮方式が復活。常に0

```
struct {
    ProtocolVersion legacy_version = 0x0303;  /* TLS v1.2 */
    Random random;
    opaque legacy_session_id_echo<0..32>;
    CipherSuite cipher_suite;
    uint8 legacy_compression_method = 0;
    Extension extensions<6..2^16-1>;
} ServerHello;
```

  - HelloRetryRequestを廃止する
    - ServerHelloを利用し、Randomに特定の値を持つServerHelloを
      HelloRetryRequestとみなす

draft 22 (1/2)

- Middlebox を騙せ！
  - レコードのバージョンは、TLS 1.2とする
  - ChangeCipherSpecを復活させる
    - ChangeCipherSpecを受け取ったら単に無視する
    - 互換モードの場合、適切なタイミングでChangeCipherSpecを送ってよい

draft 23 (1/2)

- key_share 拡張の値を変更
  - Canon のプリンタが 40 を利用していた

  `key_share(40)` → `key_share(51)`
- 不変条件の加筆
  - クライアントは提案したものは必ず実装してないといけない
  - サーバはわからないものは単に無視(異常終了してはいけない)
  - TLSを終端するミドルボックスはその両方を満たせ
  - 単にリレーするミドルボックスは中身を触るな

draft 23 (2/2)

- SignatureScheme の値の細分化

```
- rsa_pss_sha256(0x0804),
- rsa_pss_sha384(0x0805),
- rsa_pss_sha512(0x0806),

+ /* RSASSA-PSS algorithms with public key OID rsaEncryption */
+ rsa_pss_rsae_sha256(0x0804),
+ rsa_pss_rsae_sha384(0x0805),
+ rsa_pss_rsae_sha512(0x0806),


+ /* RSASSA-PSS algorithms with public key OID RSASSA-PSS */
+ rsa_pss_pss_sha256(0x0809),
+ rsa_pss_pss_sha384(0x080a),
+ rsa_pss_pss_sha512(0x080b),
```