TLS1.3とは何か？

大津 繁樹
株式会社 インターネットイニシアティブ (IIJ)

PKI Day 2016
2016/04/22

自己紹介

- 株式会社インターネットイニシアティブ(IIJ) 経営企画本部 配信事業推進部

- Node.js Core Technical Committeeメンバー
  (crypto, tls, OpenSSL bindingを担当しています)

- IETF httpbis WGで HTTP/2の仕様策定に参画。
  TLS1.3にはまだ直接関わっていません。

- ブログ: http://d.hatena.ne.jp/jovi0608 でTLS周りのネタを色々書いています。

内容

1. TLSの簡単な歴史

2. Webプロトコルの進化

3. TLS1.3が求められる背景

4. TLS1.3とは何か (時間が足りないので主要部分のみ)

本発表は、4/18時点の最新TLS1.3ドラフト(draft-12)とIETF95の議論の結果を一部含んでいます。最終仕様で変わる可能性があります。

TLSの簡単な歴史

	SSL 1.0(未発表)
1994年	SSL 2.0 DROWN
1995年	SSL 3.0 POODLE
1996年	IETF TLS WG スタート
1999年	TLS 1.0 BEAST
2006年	TLS 1.1
2008年	TLS 1.2
2013年	TLS 1.3検討スタート
2016年	TLS 1.3仕様化完了？

SSLは旧ネットスケープ社の私的プロトコル

大人の事情で名称変更

SSL3.0と基本設計は大きく変えず改良

様々な機能拡張

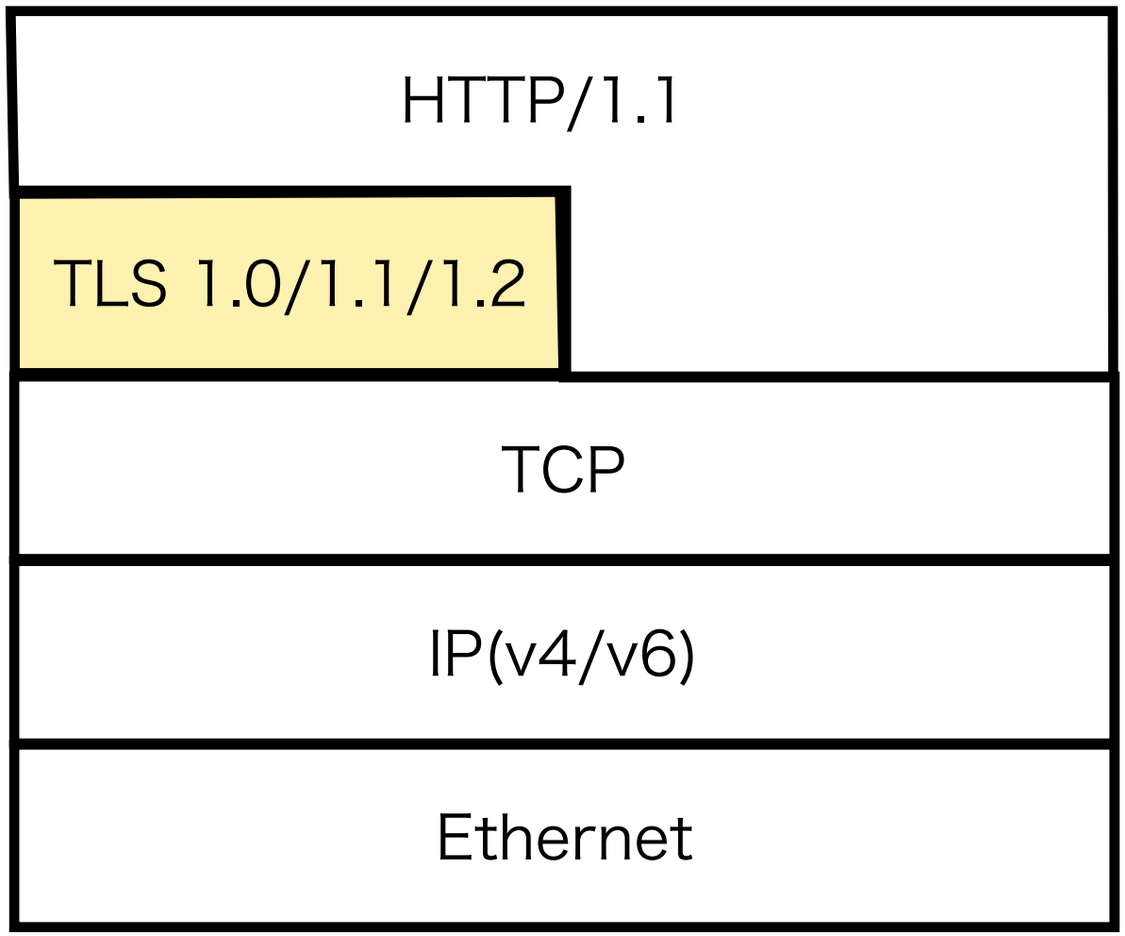
5月末WGラスト コール目標

Webプロトコルの進化

TLS1.3の位置付け

Webプロトコルの進化

HTTP/1.1の時代



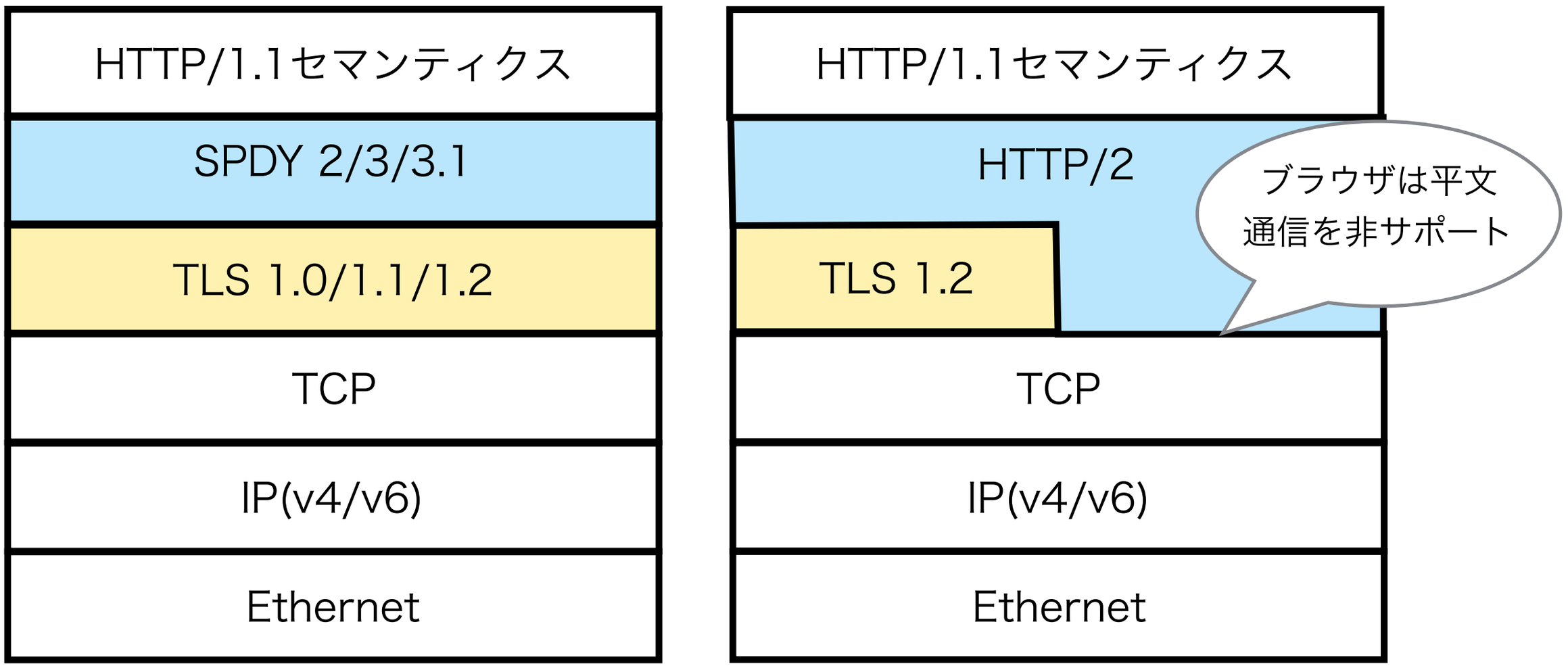
1999〜

(* TLS1.1 2006〜)

(* TLS1.2 2008〜)

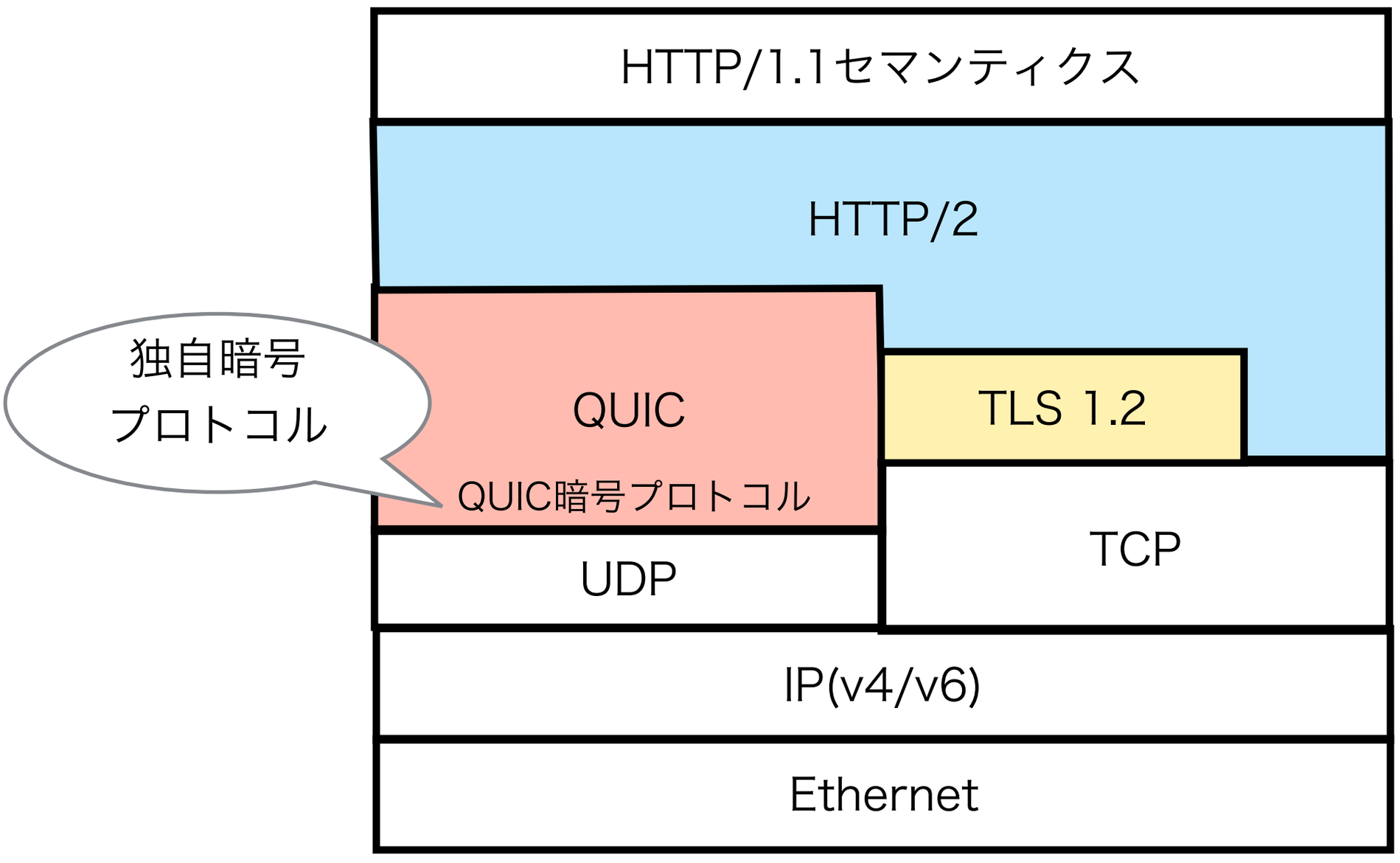
Webプロトコルの進化

HTTP/1.1からHTTP/2へ



Webプロトコルの進化

HTTP/2からQUICへ

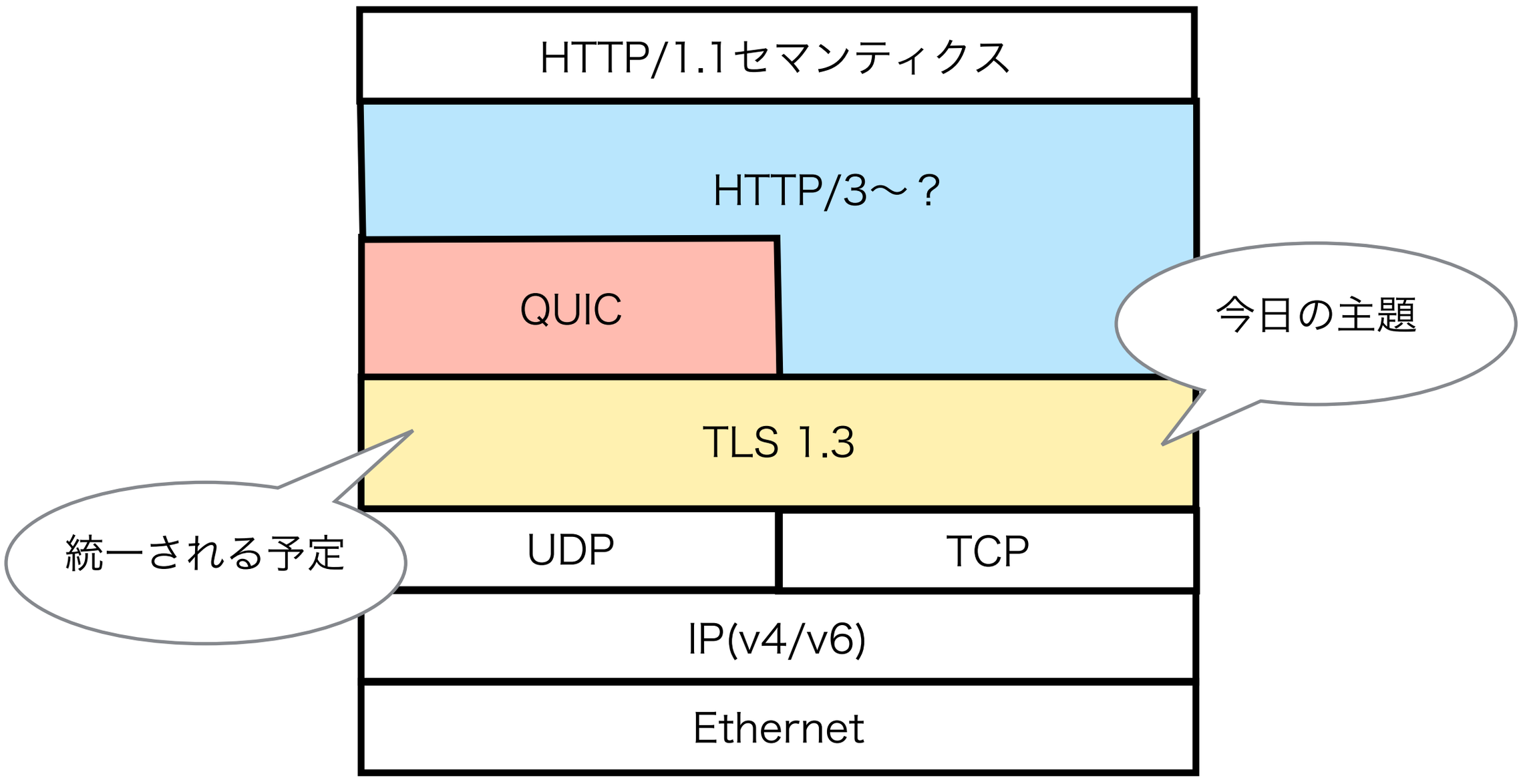


2013〜

(* HTTP/2 2015〜)

Webプロトコルの進化

QUICからTLS1.3へ



2016 or 2017〜

TLS1.3が求められる背景

常時TLS

1. インターネット環境の変化

・スノーデン事件、Lavabitサービス停止

・新プロトコル(HTTP/2, QUIC)

・その他(利用環境の変化、サービスの多様化)

2. TLS1.2の限界

・様々な技術負債の蓄積

インターネット環境の変化

インターネット環境の変化

スノーデン事件

- 米NSAによる pervasive surveillance(広範囲の盗聴行為)が明らかになる。

- 従来ここまでできないだろうと思われていた範囲まで実行。

- Logjam: 国家予算レベルで1024bits DHEを解読

インターネット環境の変化

Lavabitサービス停止

・スノーデン事件の余波で裁判所よりメールサービスのTLS秘密鍵の提出命令を受ける。

・暗号化されたデータであってもデータ保存＋秘密鍵開示命令によって復号化される恐れが現実に。

・Forward Secrecyの導入が一気に進む。

インターネット環境の変化

新プロトコル(HTTP/2)

- 主要ブラウザーはTLS通信のみHTTP/2をサポート
- HTTP/2 (RFC7540)では、TLSの利用暗号を厳しく制限。275種類の暗号方式をブラックリストとして掲載して利用不可。

nginxのデフォルト暗号方式を使う
とHTTP/2接続はエラーになります。

- その制限の多くは、TLS1.3を前倒しで適応

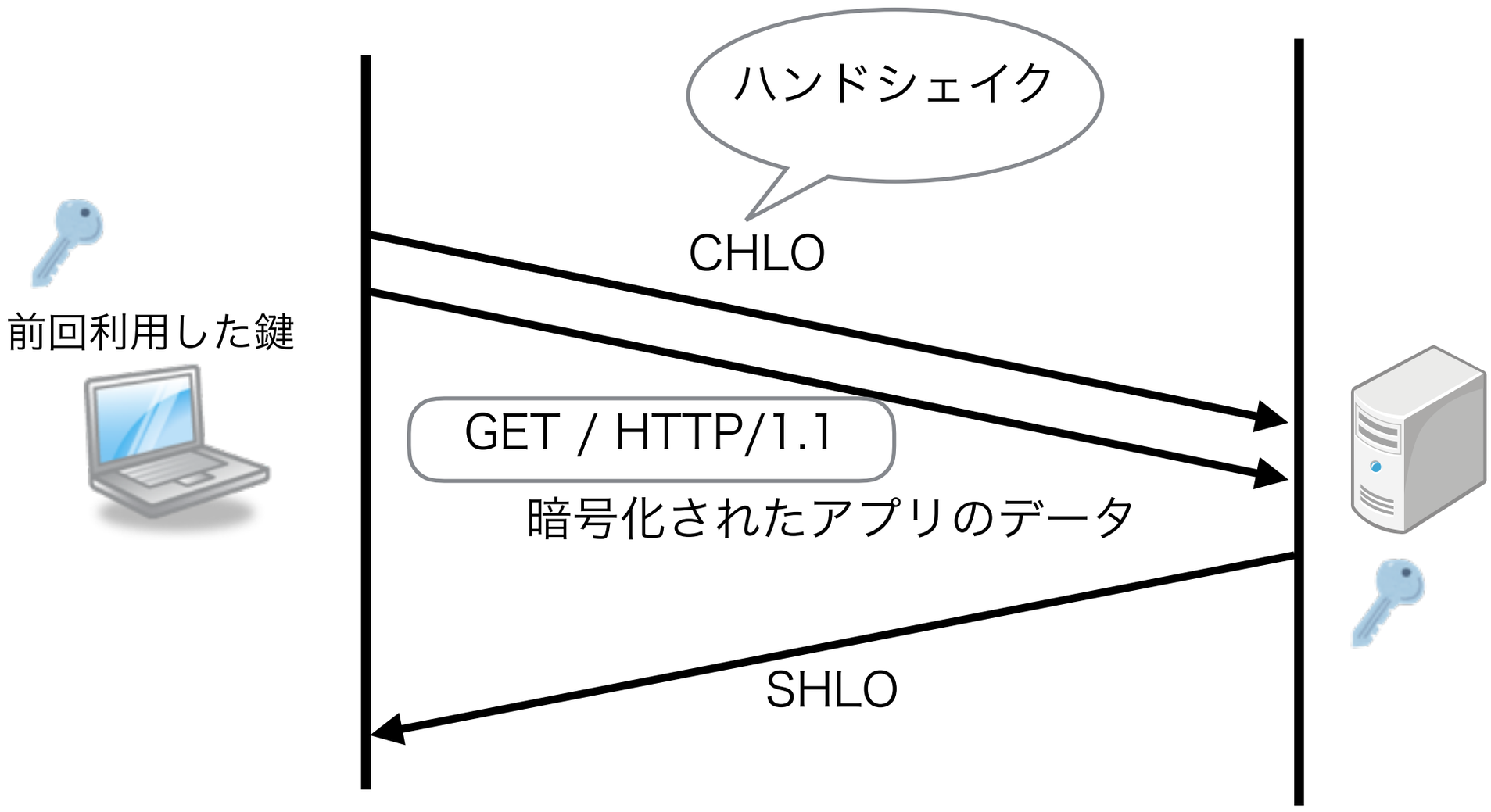
インターネット環境の変化

新プロトコル(QUIC)

- Googleが独自に開発しているプロトコル
- UDP上でTCP+TLS相当+αの機能を実現
- 0-RTTによる高速接続(Googleで75%割合)
- Googleの全サービスで運用中
- FacebookはモバイルアプリでTCP上のQUIC暗号を試験中
- LINEもQUIC利用を試験中

インターネット環境の変化

QUICによる0-RTTによる高速接続



再接続の際にはハンドシェイク完了を待たず、以前の接続で利用した鍵を使ってアプリデータを暗号化し送信する

インターネット環境の変化

その他

導入コストの低減	Let’s Encryptの無料DV証明書
性能向上	AES-NIによるCPU暗号処理能力の向上
インフラ負荷の変化	HTTP/2による接続集約
提供サービスの多様化	マルチテナントCDN
ブラウザの方針(Firefox)	非暗号化接続の段階的廃止、機能制限の方針

TLS1.2の限界

TLS1.2の限界

- 現在のTLS1.2で定義されている幾つかの機能・項目は、既に無条件で利用すると危険である。

- RFC7525(TLS/DTLSの安全な利用のための推奨)

- 過去様々なTLSの攻撃手法や脆弱性が公開され、その度対策が取られてきた。

- しかし機能の無効化や拡張機能の追加などad hocな対策で根本的・抜本的な対策になっていないものも多い。

TLS1.2の限界

暗号方式の危殆化

既に十分安全とは言えない暗号方式のサポートが仕様で規定されており、更新が必要。

DES	NIST FPS46-3の廃止(2030年まで許容)
RC4	RFC7645(Prohibiting RC4 Cipher Suites)
MD5	SLOTH攻撃(CVE-2015-7575)
SHA-1	Freestart Collision

TLS1.2の限界
暗号強度の危殆化

- FREAK, logjam： 解読手法の効率化、計算資源の高度化により十分安全とみなされる暗号強度が高くなった。スノーデン事件で国家予算レベルの計算資源も脅威となる時代。

- NIST SP 800-131A rev1, ECRYPT II推奨： 各種団体による指標とTLS1.2+RFC4492でサポートしている暗号の中に224bit以下のECDHなど既に強度不足のものが含まれている。

- NIST暗号への疑義：Dual_EC_DRBGのバックドア問題

TLS1.2の限界
暗号手法の安全性

- MAC-then-Encrypt(e.g. CBC mode)の安全性に対する疑問

- 過去 BEAST/Padding Oracle攻撃/Lucky Thirteen等の攻撃対象。今後も出てくることが予想される。

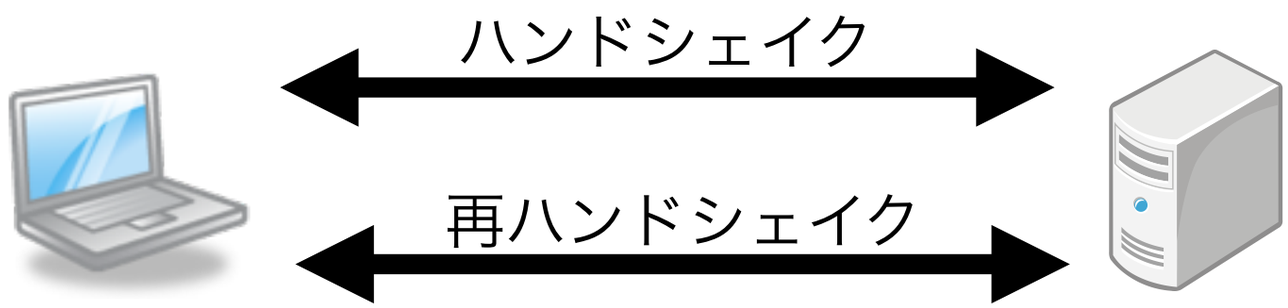
- BlockCipherの限界、Stream Cipher(RC4)の危殆化

暗号化
平文＋パディング+MAC

TLS1.3はAEAD(認証つき暗号)のみ

TLS1.2の限界
Renegotiation

- TLSハンドシェイク完了後に接続を維持したまま再度ハンドシェイクを行う機能。クライアント認証への遷移や鍵更新のために必要。
- 過去いろいろな攻撃対象となった。Renegotiation前後でセッションの引き継ぎが難しいことが起因



TLS1.3で廃止

TLS1.2の限界
Compression

- TLSのデータを圧縮してから暗号化する機能。ハンドシェイク時に圧縮方式を合意する。

- 圧縮サイドチャネル攻撃： データ圧縮によるサイズ変更を手がかりとして暗号化情報を漏洩させる手法(CRIME/TIME/BREACH)

TLS1.3で廃止

TLS1.2の限界

SessionID, Ticket

- 以前のTLSセッション情報を引き継いで再接続を行う機能

- 共有の際の条件が甘いと認証バイパスの脆弱性

TLS1.3で廃止

PSK(PreShareKey)方式に統一

TLS1.2の限界

鍵交換のCross-Protocol攻撃

- MiTMでTLS1.2のECDH鍵交換をDH鍵交換にみなしてクライアントに送付させる攻撃

- 1/2^40程度の確率でpre master secretの鍵解読が成功

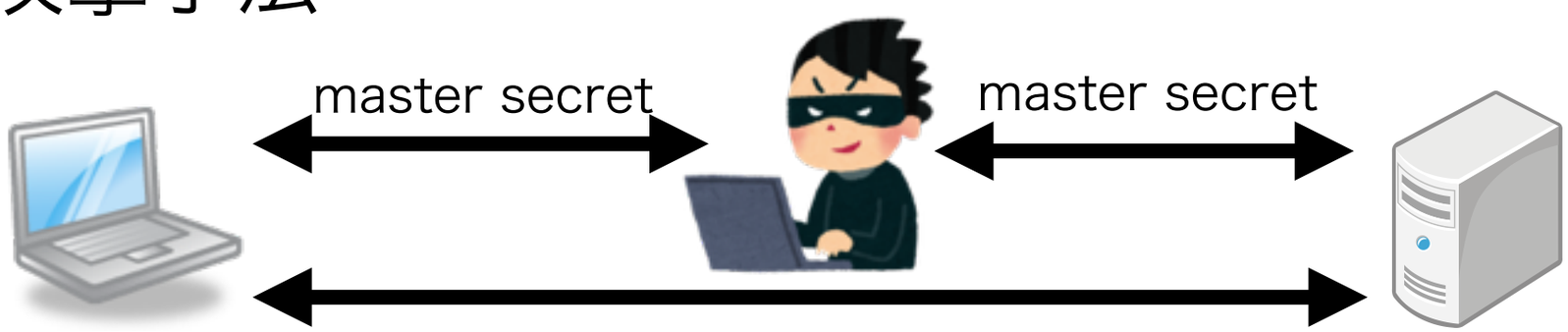
- TLS1.3ではパラメータによる鍵交換を廃止。名前付きパラメータ利用に制限、署名をセッションハッシュに変更した。

(* セッションハッシュ：それまでやり取りした全ハンドシェイクデータを利用してハッシュ値を生成する方法)

TLS1.2の限界
Key Deviation

- ハンドシェイクデータから対称暗号の秘密鍵やMAC鍵を生成する方式

- TripleHandshake攻撃：MiTMでnonceを操作して同じmaster secretを持つ２つのTLS接続を確立。Renegotiationを誘発させて悪意のあるデータを送り込む攻撃手法



TLS1.3はセッションハッシュを用いた master secret に変更

TLS1.3とは何か？

時間の都合上網羅的・詳細な仕様内容まで踏み込みません

TLS1.3の目的

1. ハンドシェイクデータをできるだけ暗号化して秘匿する

2. ハンドシェイクレイテンシーの削減
    - 再接続の場合は 0-RTT
    - 新規接続の場合は、0.5-RTT, 1-RTT

3. ハンドシェイクで交換する項目の見直し、簡素化

4. レコード層暗号化の見直し(RC4廃止、CBC不採用等)

TLS1.3の特徴

1. 様々な機能、項目の見直し・廃止

   時代に合わなくなったもの、より効率的に変更修正できるものを
   TLS1.2から機能・項目を数多く廃止

2. よりセキュアに

   平文通信が必要な部分を極力少なくして情報を秘匿

   AEADやパディングなど将来的な攻撃に備える

3. 性能向上

   0-RTT, 0.5-RTT接続による性能向上

もうメジャーバージョンアップと同じ改変
TLS1.3の主な廃止項目 #1

機能	Compression	CRIME/TIME対策
	Renegotiation	PostHandshakeに置き換え
	SessionID, Ticket	PSKに置き換え
暗号方式	MD5, SHA-1, SHA-224	危殆化、低強度対策
	DSA	利用用途の減少
	non-AEAD(CBC, RC4)	危殆化、攻撃リスクの低減
	IVフィールド	nonce再利用リスクの排除
	custom DHE	CrossProtocol攻撃対策
	ECDHE compress format	利用用途の見直し、簡素化
	anonymous DH	RFC7250(raw pub key)を代用

もうメジャーバージョンアップと同じ改変
TLS1.3の主な廃止項目 #2

ハンドシェイク	record layerのバージョン	後方互換のためだけに維持
	random中の時刻データ	乱数生成が高度化し意味がなくなった
	PFS/PSK以外の鍵交換	秘密鍵の危殆化対策
	SSL2.0/3.0のサポート	危殆化したプロトコルのサポート廃止
	ChangeCipherSpec	鍵交換直後に暗号化開始
	Server/ClientKey Exchange	Client/ServerHello拡張に移動
拡張機能	max_fragment_length	2^14に最大値を固定
	truncated_hmac	AEADでの利用不可、安全ではない

なぜTLS2.0じゃないのか

- SSLLabの調査では、TLSレコード層のバージョンが、1.3の場合は10%、2.0の場合は70%以上のTLSサーバが接続を拒否する。

OpenSSL-1.0.2のソース

github.com/openssl/openssl/blob/OpenSSL_1_0_2-stable/ssl/s23_srvr.c#L352-L358

```
             * if major version number > 3 set minor to a value which will
             * use the highest version 3 we support. If TLS 2.0 ever appears
             * we will need to revise this....
             */
            if (p[9] > SSL3_VERSION_MAJOR)
                v[1] = 0xff;
            else
```

クライアントがTLS2.xがオファーしたらとりあえずTLS1.255として扱う

TLS1.2 vs 1.3

初期接続形態の違い、高速化

接続	TLS 1.2	TLS 1.3
0-RTT	N/A	再接続PSK (クライアント認証は不可)
0.5-RTT	N/A	新規接続(サーバ側発信)
1-RTT	再接続	新規接続
2-RTT	新規接続	暗号方式の不合意時

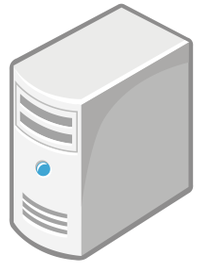
高速化

高速化

RTTが比較的大きいモバイル環境の性能改善

TLS1.2のハンドシェイク

2-RTT



0RTT

ClientHello

ServerHello
Certificate
ServerHelloDone

1RTT

暗号化範囲

ClientKeyExchange
ChangeCipherSpec
Finished

ChangeCipherSpec
Finished

2RTT

Application Data

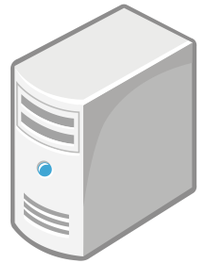
2－RTT

3RTT

Application Data

TLS1.3のハンドシェイク

0.5,1-RTT



0RTT

ClientHello →

暗号化範囲

ServerHello
EncryptedExtension
ServerConfiguration
Certificate
CertificateVerify
Finished
ApplicationData

1RTT

←

Finished
Application Data

→

0.5-RTT

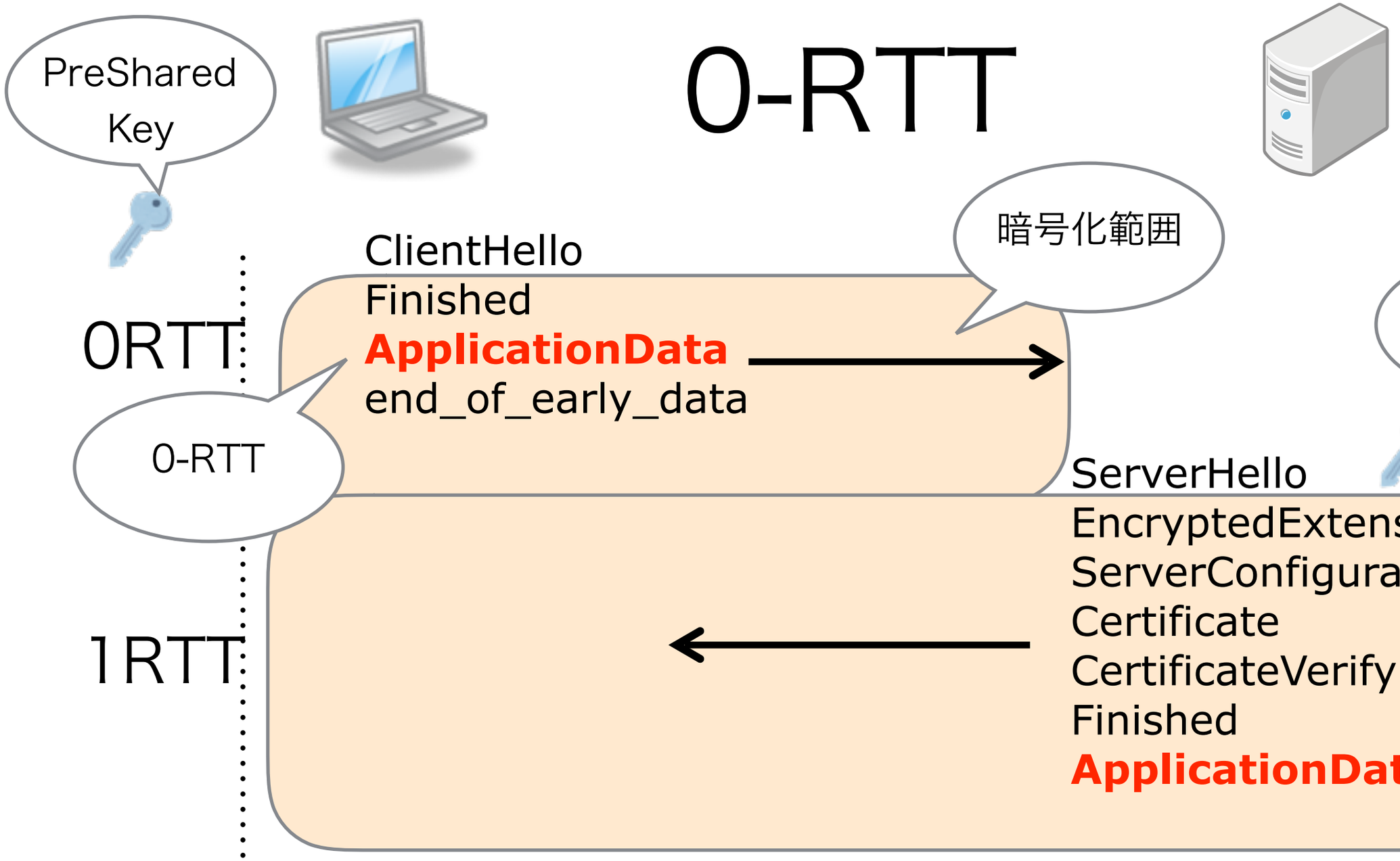
1-RTT

2RTT

← **Application Data**

TLS1.3のハンドシェイク

0-RTT



0RTTのデータはReply攻撃対策のためデータの冪等性が必要。
専用APIを用意してアプリ側で担保する。

TLS1.3の暗号方式

鍵交換	ECDHE, DHE, PSK	**secp256**/384/521r1 **x25519**, x443 ffdhe2048-8192
認証	RSA, ECDSA, PSK	rsa_pkcs1_sha256/384/512 ecdsa_secp256r1_sha256/384/512 rsa_pss_sha256/384/512 ed25519, ed448
対称暗号	AEAD	**AES128/256-GCM** **ChaCha20-Poly1305** AES128/256-CCM
HKDF ハッシュ	SHA256/384	SHA-3はまだです。

DJB curve

ハンドシェイク署名はPSSのみ

エドワード曲線

赤字はMTI(Must To Implement)

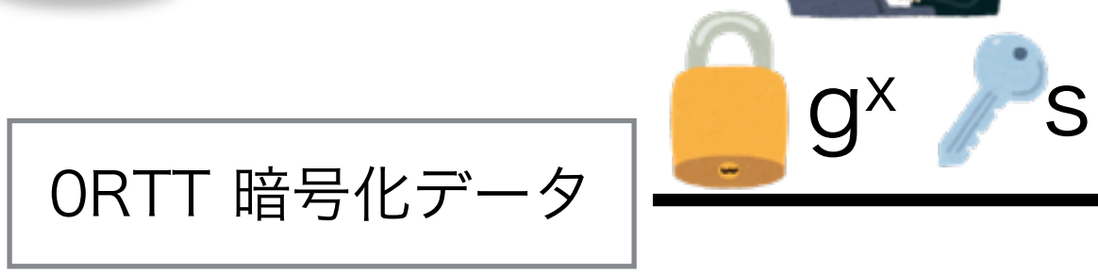
TLS 1.3 ホットな議論
(EC)DHE 0-RTTの不採用

- QUICではサーバからsemi-static鍵を予めクライアントに送付し、0-RTT接続時にデータを暗号化する。

- サーバのsemi-static鍵はクライアントでの管理が楽

- DNSでsemi-static鍵を公開すれば初期接続でも0-RTTできる。

TLS 1.3 ホットな議論
(EC)DHE 0-RTTの不採用



- semi-static秘密鍵が危殆化した場合にForward Secrecyが破られる
- 議論の結果、(EC)DHE based 0-RTTは不採用に
- PSKのResumptionは毎回master secretを更新するため 0-RTTはPSKモードのみ利用可能に

TLS 1.3のホットな議論

RSA: PSS vs PKCS1.5

- 10年以上経つのに証明書の署名方式は、RSA-PKCS1.5からより安全なRSA-PSSへの移行が全然進んでいない。

- TLS1.3では、ハンドシェイク署名をRSA-PSSのみに制限。証明書については言及せず。

- このままだとRSA-PSSとPKCS1.5の併用が続く。

- 将来的に必須かできるか？クライアントーサーバ間で拡張情報を交換してRSA-PSSだけ使えるようにできないか、引き続き検討。

今後の見込み

- 2016/02 TRON(TLS1.3 Ready Or Not)ワークショップでレビュー
- 2016/04 IETF95 B-A でラストコールに向けて各種課題の判断
- 2016/05末ぐらいに最終仕様ドラフト公開。 WGラストコール開始を目指す。

- Firefox(NSS), Chrome(BoringSSL), CloudFlareのプロトタイプが出てくる。MS/Appleも取り組む予定。OpenSSLは時期的に遅くなりそう。
- 再度TRONや相互接続試験を行い、最終的なレビューの機会を設けたい。
- 2016/07のIETF96でLastCallの合意にもっていけないか。ラストコール期間は通常より長くなりそう。
- 非常に順調に行けば11月過ぎにTLS1.3の仕様化完了か？ まだまだわかりません。

TLS1.3で何が変わるか

TLSライブラリ 開発者	設計・実装し直し
アプリ開発者	専用APIを使った０–RTT対応
サーバ管理者	クライアントの普及を見ながらいつ導入するか判断
ネットワーク 管理者	ミドルボックス(FW, IDS, Proxy等)の影響を確認
ユーザー	知らないうちに使っている。なんか早くなった気がする。

TLS1.3デモ(時間が余れば)

NSS
サーバ

draft-11/(一部draft12)
https://developer.mozilla.org/ja/docs/NSS

1. screen

```
ohtsu@obu:~/tmp$ ./dist/Linux3.19_x86_64_cc_glibc_PTH_64_DBG.OBJ/bin/selfserv -
d tests_results/security/obu.1/ssl_gtests -V tls1.3:tls1.3 -n 'server' -p 4430
```

2. ssh

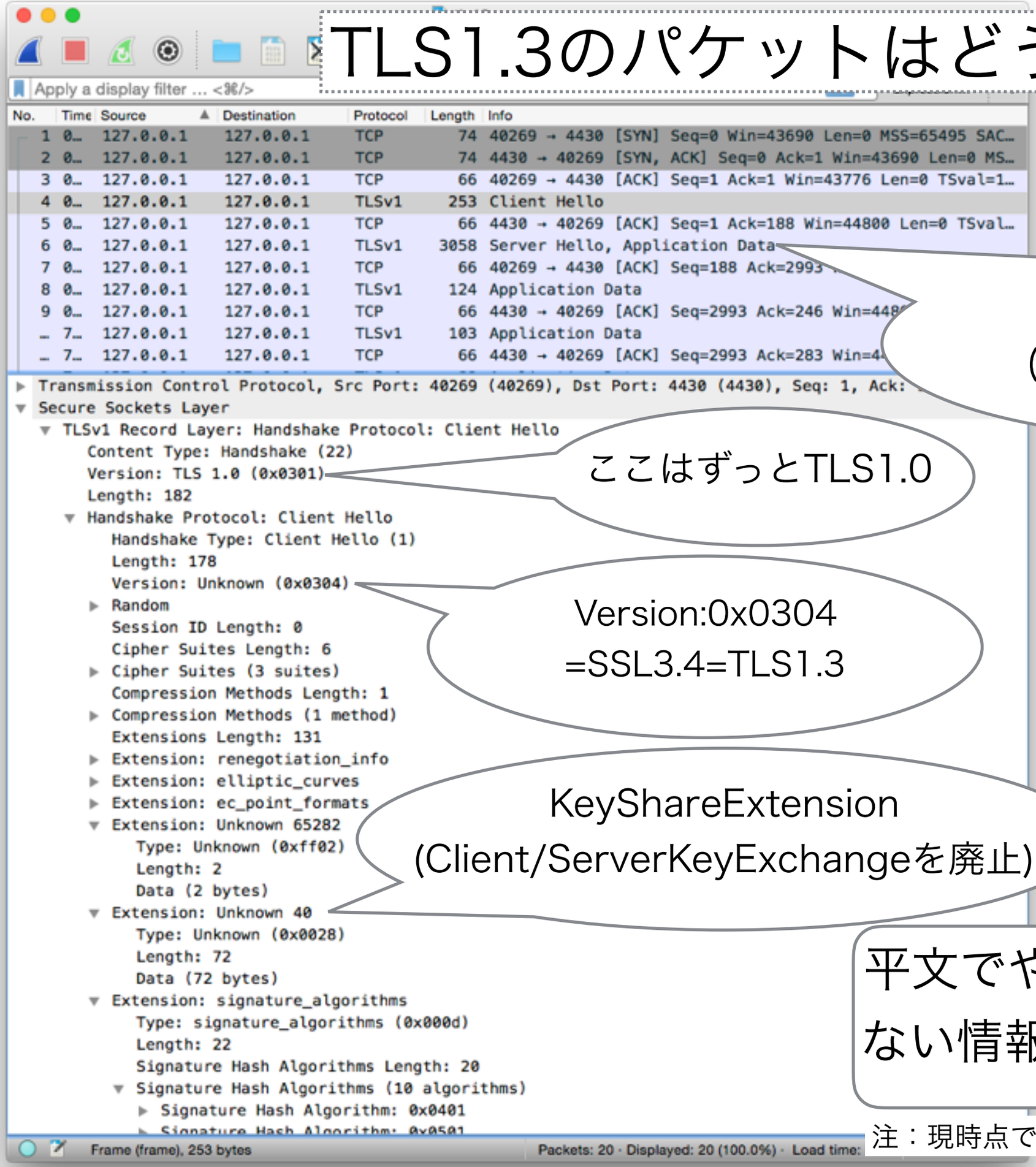
```
ohtsu@obu:~$ go run go/src/github.com/bifurcation/mint/bin/mint-client-https/m
in.go
GET / HTTP/1.1
Host: localhost:4430
User-Agent: Go-http-client/1.1

EOF
```

Go TLS1.3
クライアント

https://github.com/bifurcation/mint

TLS1.3のパケットはどう見える？



暗号化されたTLS拡張
(ALPN)や証明書その他

ここはずっとTLS1.0

Version:0x0304
=SSL3.4=TLS1.3

KeyShareExtension
(Client/ServerKeyExchangeを廃止)

でもSNIはまだ
見えます。

平文でやり取りしないといけない情報以外は**全て暗号化**

注：現時点でwiresharkはTLS1.3に対応していません。