HTTP/2, QUICからTLS1.3へ

大津 繁樹
株式会社インターネットイニシアティブ(IIJ)

Internet Week 2015
2015/11/18

2015年11月19日更新版

自己紹介

- 大津 繁樹
- 株式会社 インターネットイニシアティブ(IIJ)
  - プロダクト本部 アプリケーション開発部サービス開発2課

- Node.JS Technical Steering Committee メンバー
  - (主にTLS/CRYPTO/OpenSSLバインディングを担当)
- IETF httpbis WG で HTTP/2相互接続試験等仕様策定に参画。しかしTLS1.3の標準策定作業にはこれまで直接かかわっていません。

- ブログ： http://d.hatena.ne.jp/jovi0608/ でTLS周りのネタをいろいろ書いています。

内容

現在仕様策定作業中のTLS1.3について解説します。

注意： 内容は2015年11月2日時点での draft (*)を元にしています。今後の仕様策定作業で本プレゼンの内容が変更になる場合がありますのでご注意ください。

- TLS1.3が求められる背景
  - インターネット環境の変化、TLS1.2の限界
- TLS1.3仕様について
  - TLS1.2と1.3の違いを中心して仕様の一部を詳しく解説します。

(* https://github.com/tlswg/tls13-spec/ の master HEAD)

TLSの簡単な歴史

現在の利用推奨

- SSL 1.0未発表
- 1994年 SSL 2.0
- 1995年 SSL 3.0

SSLは、旧ネットスケープ社の私的プロトコル

- 1996年 IETF TLS WGスタート

TLSと名前を変えて標準化

- 1999年 TLS 1.0

SSL3.0と基本設計は大きく変えず、内部バージョンは TLS1.0 =SSL 3.1

- 2006年 TLS 1.1
- 2008年 TLS 1.2

様々な拡張仕様を追加

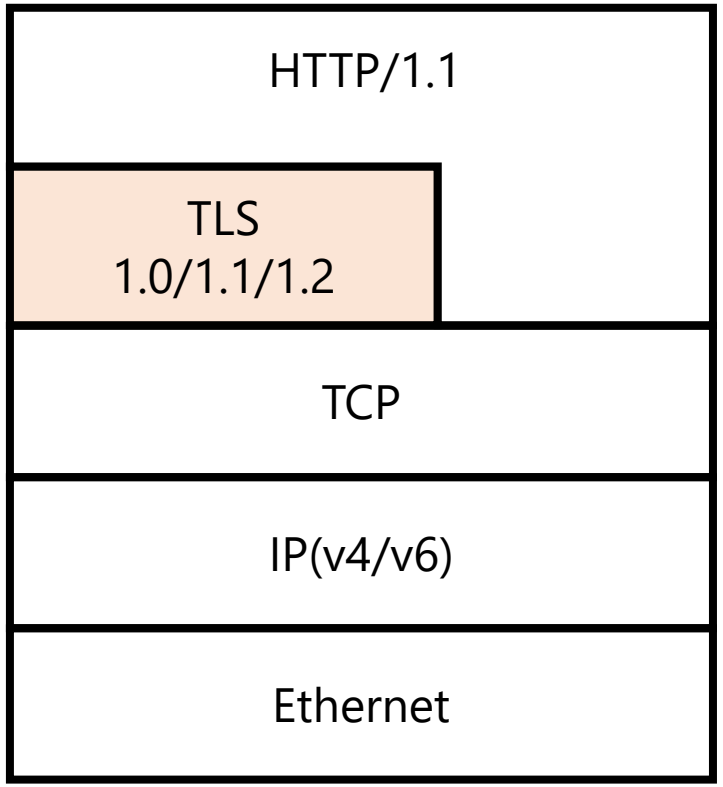
- 2013年 TLS 1.3検討スタート

2015年11月2日時点で draft10

- 2016年 TLS 1.3仕様化完了？？？

まだわかりません

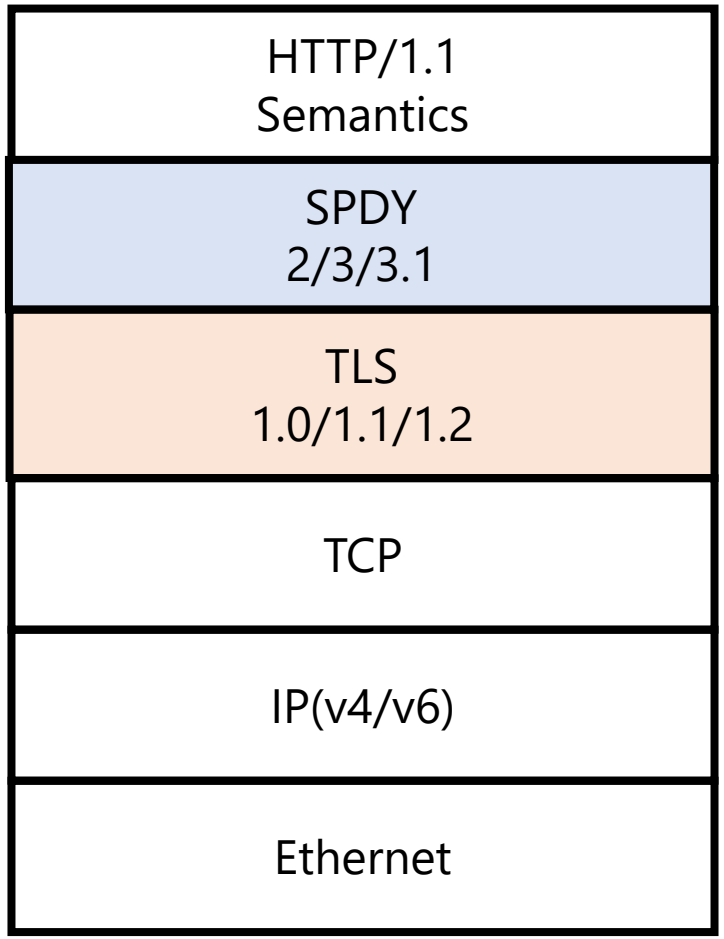
TLSの位置付け
HTTP/1.1 の時代 (1999～)



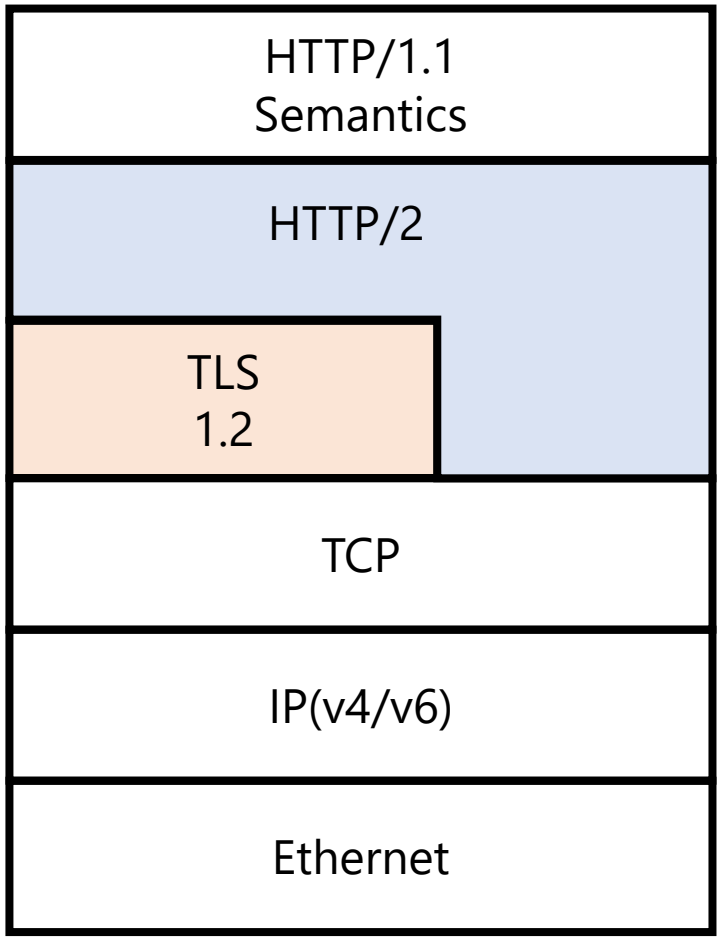
(* TLS1.1 2006～, TLS1.2 2008～)

TLSの位置付け
HTTP/1.1からHTTP/2へ

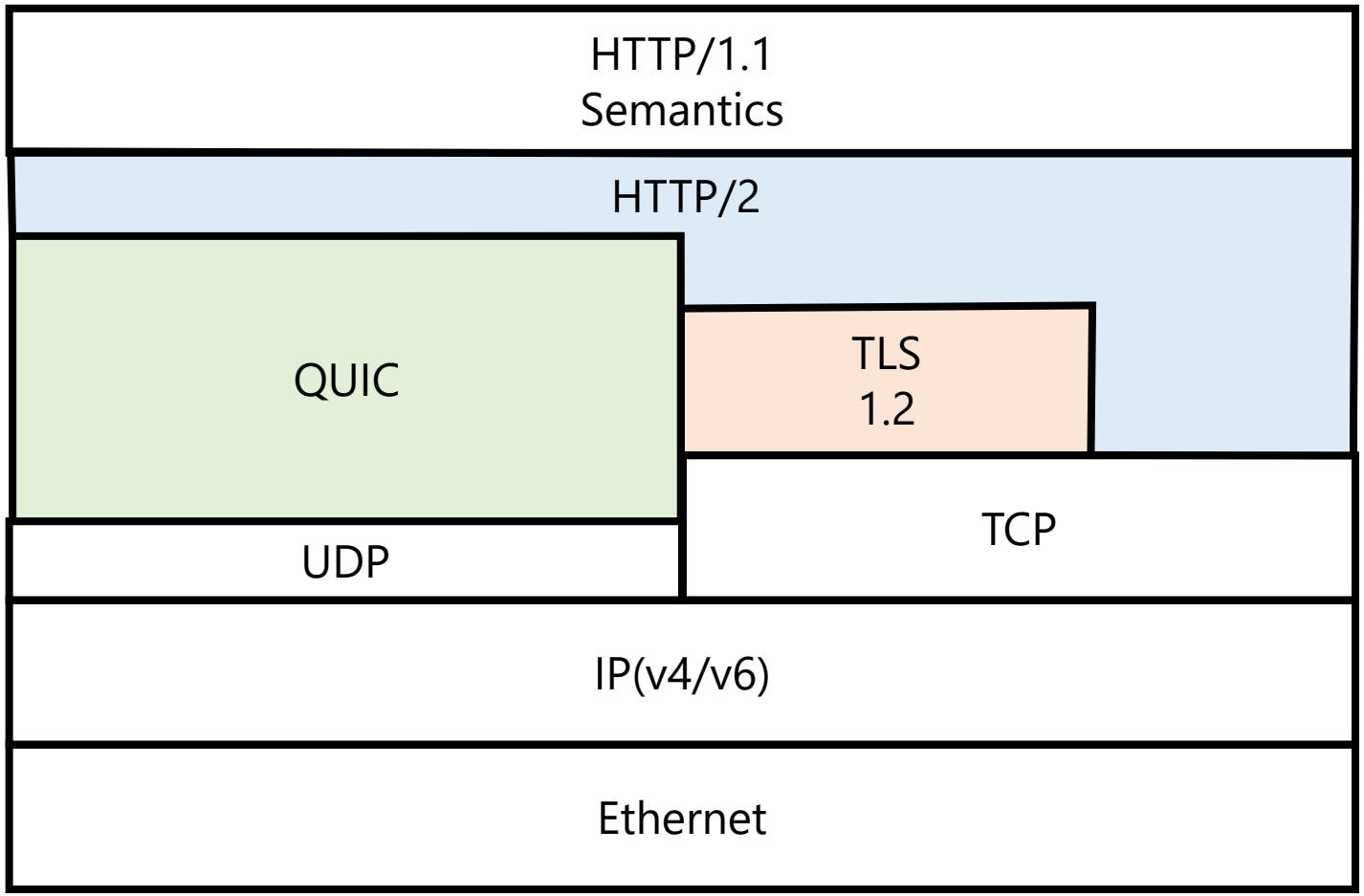
(2009～)



(2015～)



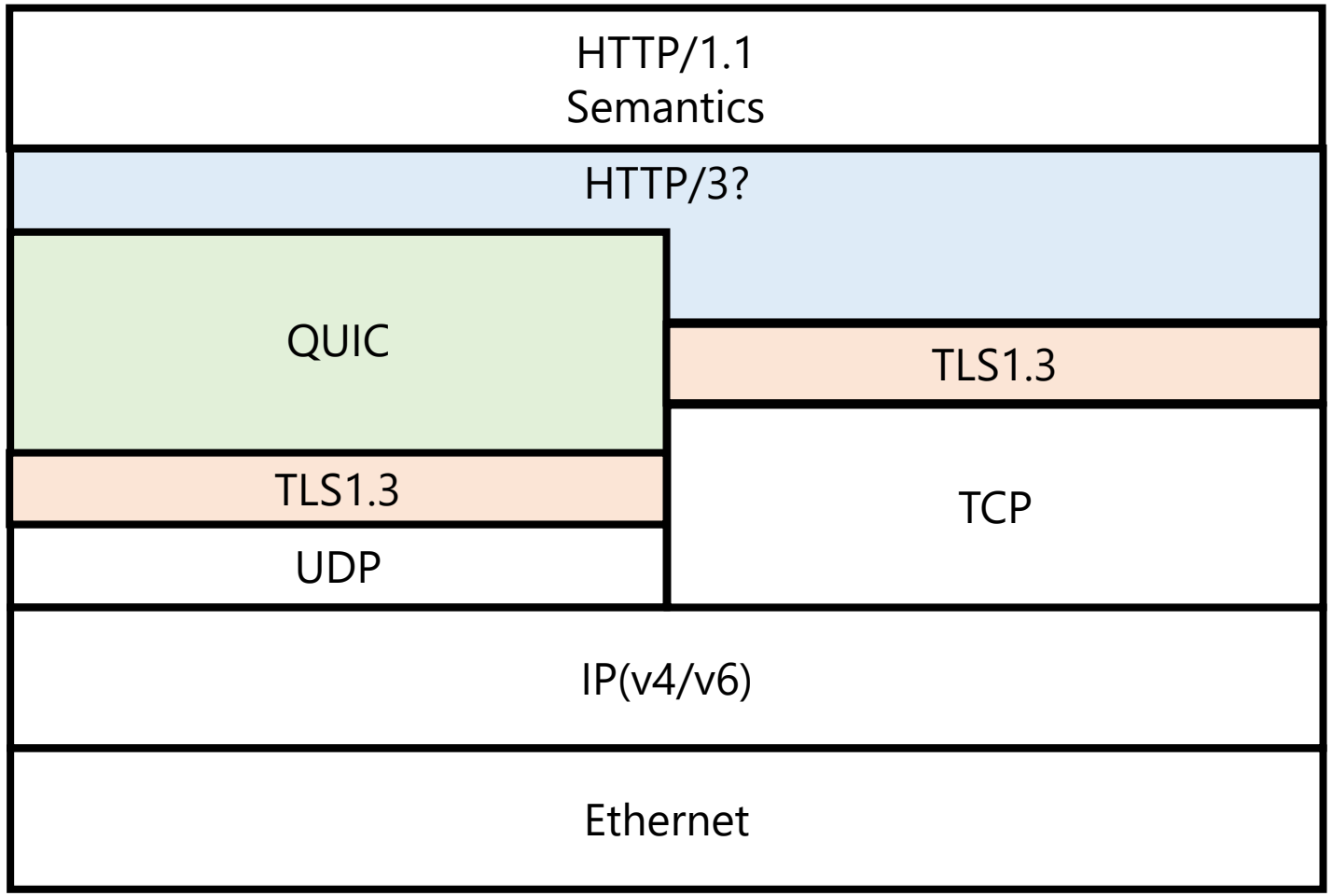
TLSの位置付け
HTTP/2からQUICへ (2013～)



(* HTTP/2 2015～)

TLSの位置付け
QUICからTLS1.3へ

(2016 or 2017?～)



TLS1.3が求められる背景

- インターネット環境の変化
  - スノーデン事件
  - 新プロトコル(HTTP/2, QUIC)
  - その他

- プロトコル上の問題点
  - TLS1.2の限界

インターネット環境の変化
スノーデン事件

- 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。
- Lavabitサービス停止→PFSの重要性
  - 米国司法当局によるメールサービスの秘密鍵の提供命令。
  - 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。→ PFS(Perfect Forward Secrecy) の導入が進む。
- もはやインターネット上の通信は安全ではなく、将来的なリスクも考慮して対策をたてるもの。→ よりセキュアのデフォルト

平文のhttp接続より、常時 https 接続を

インターネット環境の変化
新プロトコル(HTTP/2)

Appendix A. TLS 1.2 Cipher Suite Black List

An HTTP/2 implementation MAY treat the negotiation of any of the following cipher suites with TLS 1.2 as a connection error (Section 5.4.1) of type INADEQUATE_SECURITY:

- TLS_NULL_WITH_NULL_NULL
- TLS_RSA_WITH_NULL_MD5

HTTP/2で利用禁止の暗号方式ブラックリストが11ページ(275種類)掲載

- HTTP/2 仕様でTLSの利用条件を制限すべきかどうか大きな議論になったが、新しいプロトコルはよりセキュアな状態で提供すべきとの意見で合意。

- 9.2 (Use of TLS features)の節で、HTTP/2が利用できるTLSの設定条件を詳細に定義。 (後述)

nginxでデフォルトCipherのままhttp2を使うと Chromeでエラーになります。

HTTP/2がTLSに求める制限

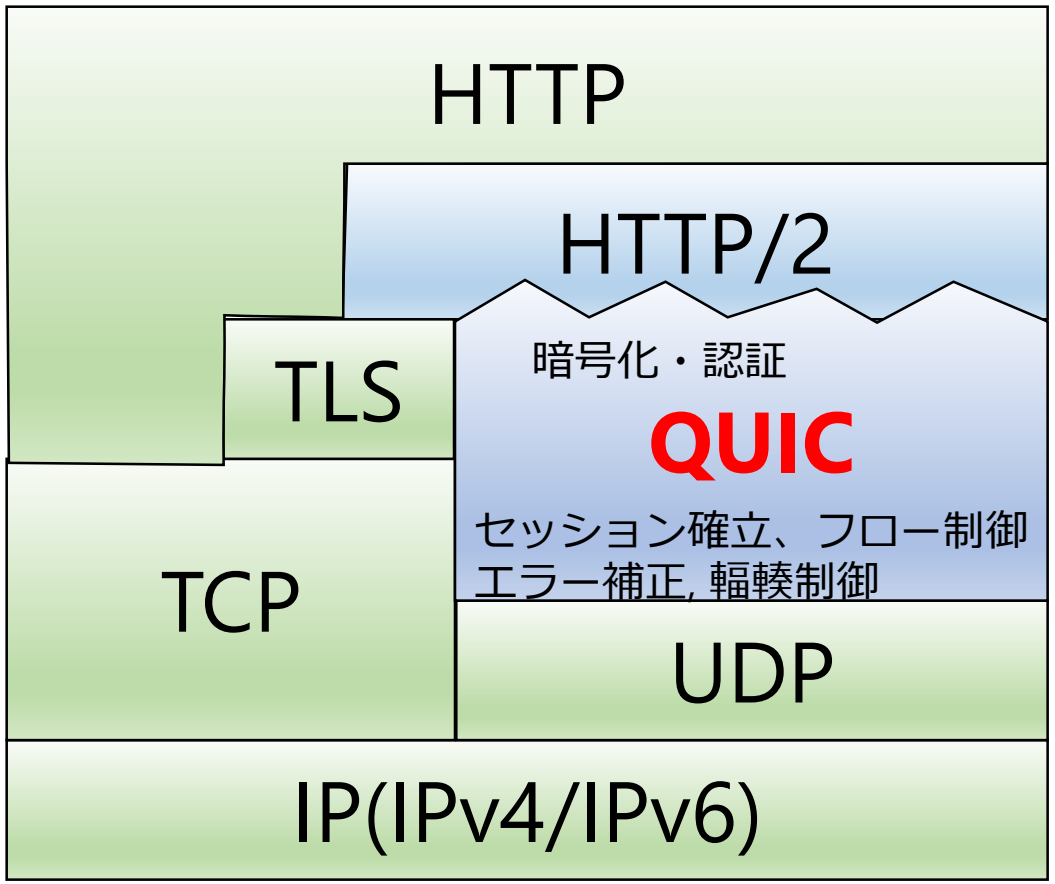
- TLSのバージョンは1.2以上
- プロトコル選択にALPN(RFC7301)を使う
- サーバ認証を共有できる接続は接続共有が可能
- クライアント認証の利用は初期接続時のみ可能
- SNI(Server Name Indicator)拡張必須
- TLS Compression禁止
- Renegotiation禁止
- 鍵長 (DHE 2048bit以上、ECDHE 224bit以上）サポート必須
- PFS必須 (DHE, ECDHE)
- AEAD以外の暗号方式をブラックリストとして利用禁止

TLS利用に関する様々な禁止事項。
どうしてこのような条件が必要なのか後述します

ほとんどがTLS1.3に取り込まれている条件

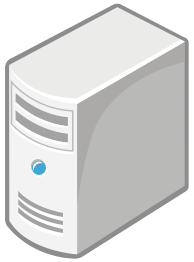
インターネット環境の変化
QUIC

- Googleが独自に開発しているプロトコル
- UDP上でTCP+TLS相当＋αの機能を実装
  - 最短0-RTTで再接続、暗号通信を必須化
  - TCPと同様の輻輳制御で帯域の公平化
  - 独自の誤り訂正と再送機能でパケットロスによるTCPのHead of Line Blockingを回避
  - HTTP/2のフレーム制御機能を取り込み
- Googleの全サービスで運用中



QUICがTLS1.3に求める機能
0-RTTによる再接続

Client



Server

前回の接続でサーバのConfigを入手し、既に自分のトークンも送信した。
その設定を使って暗号ハンドシェイクと同時にいきなり暗号化してアプリデータを送る。

CHLO(STK, SNO, SCID)

Encrypted Application Data

このクライアントは、正当なトークンとconfig idを送信してきているから、以前利用した設定を使ってアプリの復号化を行う。

TLS1.3では、接続性能向上のため0-RTT接続をサポート

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サービス→個人レベルのTLSの導入・普及
- 性能向上
  - AES-NI/AVX/AVX2等によるCPUの暗号処理普及→TLSの高速化
- 上位レイヤーの利用環境の変化
  - HTTP/2, SPDYの利用に伴う接続の集約→TLS利用時のインフラ負荷の低減
- 提供サービスの多様化
  - SNI(Server Name Indicator)を利用したCDNサービスのTLSマルチテナント化→より高度で複雑な環境でのTLSの利用が進む

TLS1.2の限界

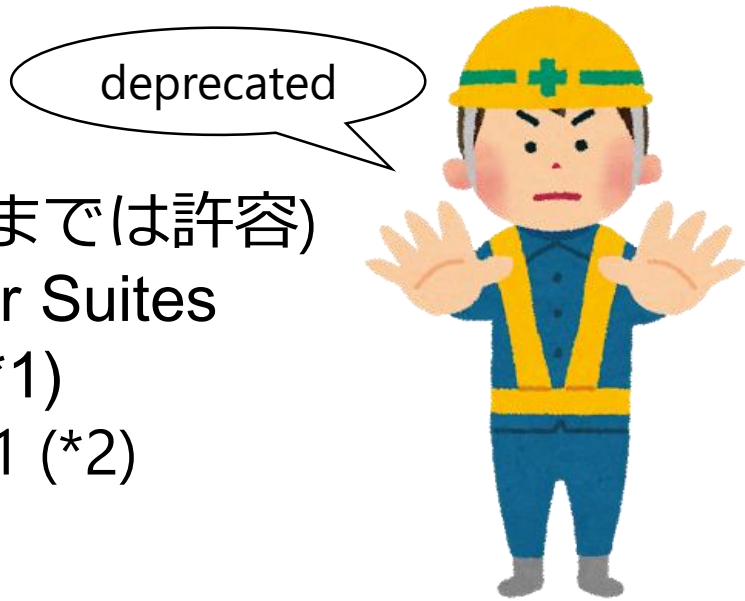
- 現在のTLS1.2仕様に含まれている幾つかの機能・項目は、現在何も考えずに無条件に利用すると危険である。

- そのため、RFC7525(TLS/DTLSの安全な利用ための推奨）が作成されている。

- 過去様々なTLSの攻撃手法や脆弱性が公開され、そのたび対策が取られてきた。しかし、機能の無効化や拡張機能を追加するようなパッチ的な対策も多く、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中に規定されており、更新が必要。

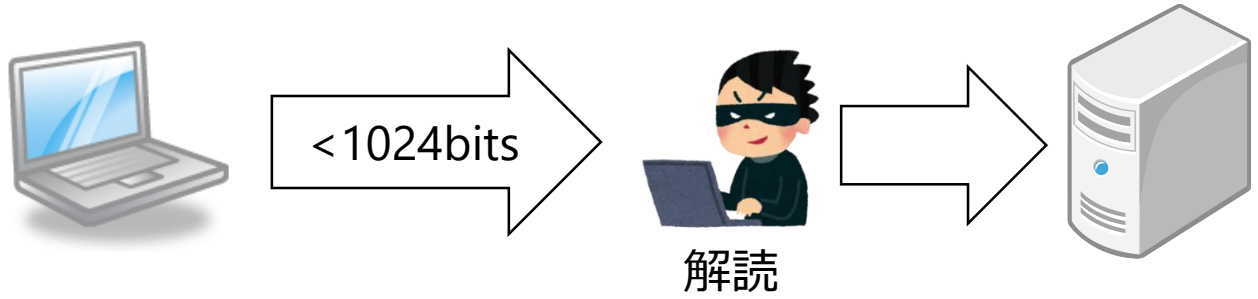


DES: 2005年 NIST FPS46-3規格の廃止(2030年までは許容)
RC4: 2015年 RFC7465: Prohibiting RC4 Cipher Suites
MD5: 2008年 MD5 considered harmful today (*1)
SHA-1: 2015年 Freestart collision for full SHA-1 (*2)

(*1 http://www.win.tue.nl/hashclash/rogue-ca/)

(*2 https://eprint.iacr.org/2015/967.pdf)

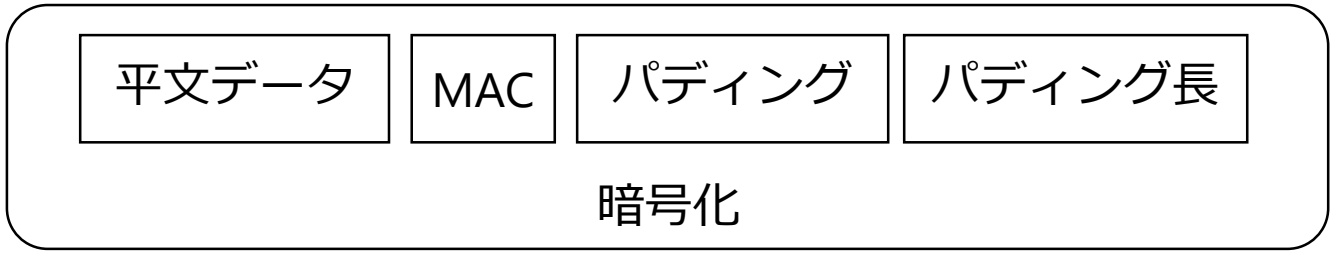
TLS1.2の限界:
暗号強度の危殆化



- 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。国家予算レベルの計算資源も脅威としてみなされる時代に（スノーデン事件）。

- 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は既に強度不足のものも含まれている。(224bit以下のECDH)

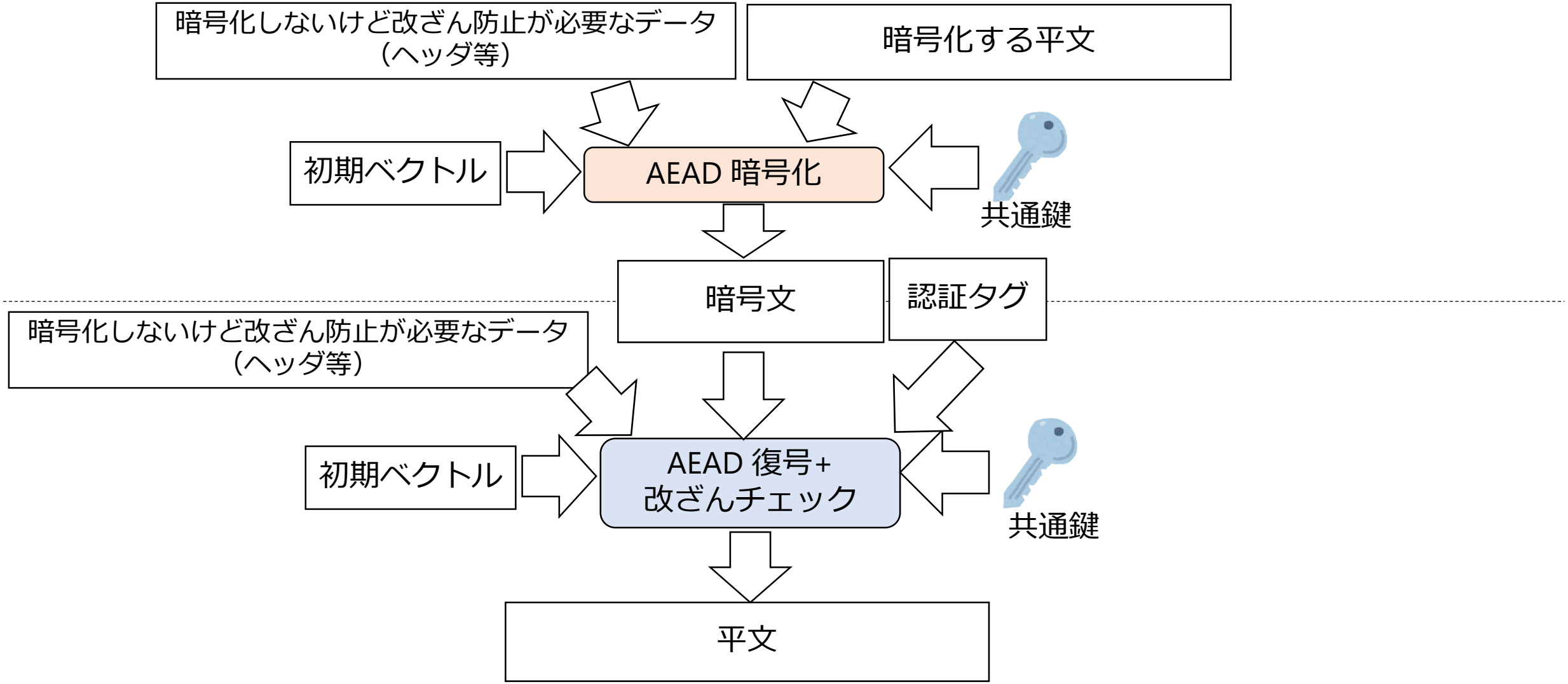
TLS1.2の限界: 暗号手法の安全性



- 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 などの攻撃対象となり、今後も新たな攻撃手法が出てくることが予想される。
- BlockCipher(CBC)の限界、Stream Cipher(RC4)の危殆化

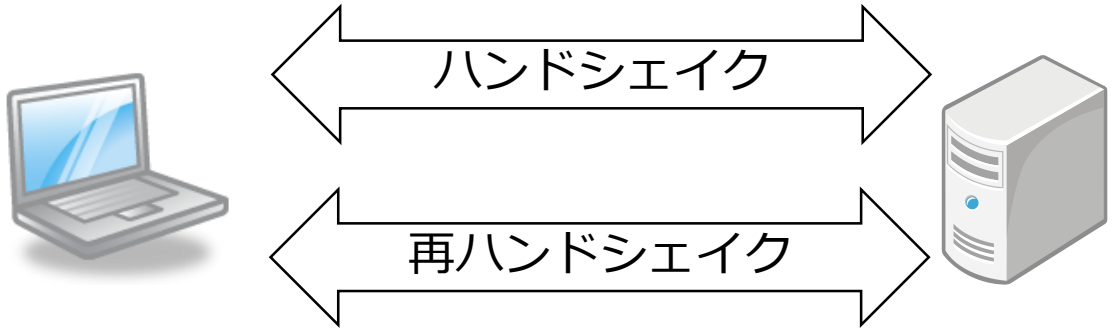
TLS1.3ではAEADのみサポート

AEAD（認証付き暗号）



AEAD: Authenticated Encryption with Associated Data

TLS1.2の限界: Renegotiation



- Renegotiation
  - TLSハンドシェイク完了後に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を行う場合に必要

- 過去Renegotiation脆弱性が公表され、SecureRenegotiation拡張で対応した。Triple Handshake攻撃（後述）などでも Renegotiationが利用された。
- Renegotiation前後でセッションの引継ぎが安全に行われないのが原因である。

TLS1.3では、Renegotiation を廃止

クライアント認証への遷移や鍵更新を別方式を検討

TLS1.2の限界: Compression

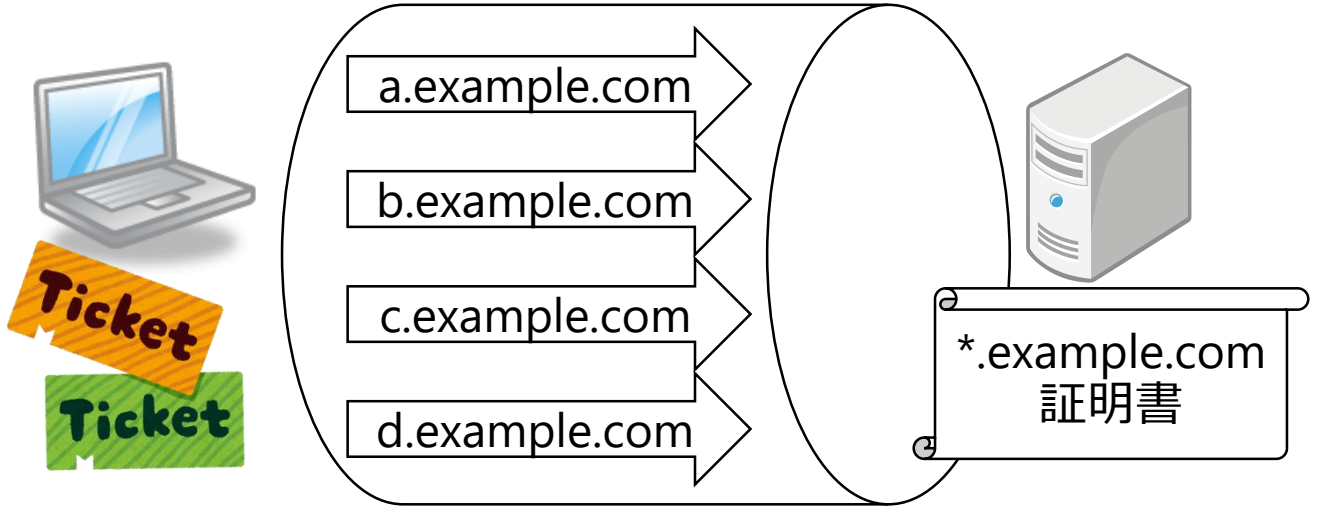
- TLSのデータを圧縮してから暗号化する機能。ハンドシェイク時に圧縮方式を合意する。

- 圧縮サイドチャンネル攻撃
  - CRIME/TIME/BREACHといったデータ圧縮によるサイズ変更を手掛かりとした攻撃が公表された。
  - データ圧縮によるサイズ収縮は避けられないため、対策するには圧縮機能を停止するしかない。

TLS1.3では圧縮機能を廃止

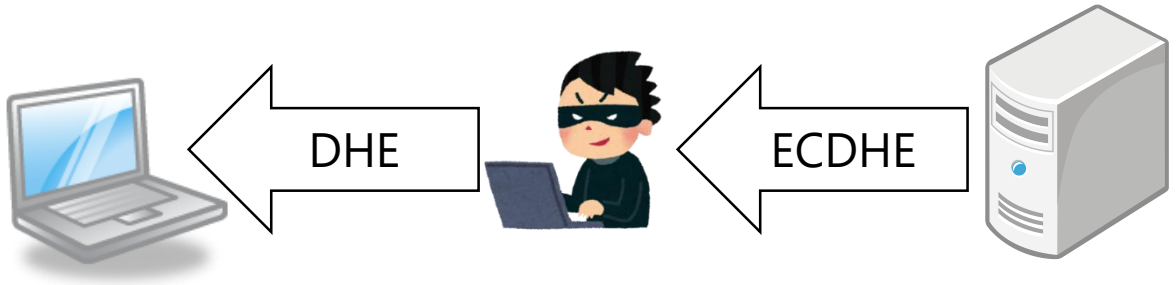
HTTP/2ではHPACKというヘッダ圧縮機能を開発、しかし完全に攻撃に対策できないため重要なデータヘッダのみ圧縮しないよう回避策が取られている。

TLS1.2の限界: Resumption



- 以前のTLSのセッション情報を引き継いで再接続を行う機能
- SPDYやHTTP/2では、サーバ証明書の認証が同一なものであれば異なるTLS接続を Resumptionとして１つのTLS接続に共有することができる。
- 共有の際の条件が甘いと認証bypass脆弱性が発生し、まだ十分対策が取られていない。

TLS1.2の限界: Cross-Protocol Attack



- Cross-Protocol Attack
  - TLS1.2の ServerKeyExchange でMiTM攻撃を行い、ECDHの鍵交換書式のデータをDH鍵交換をみなしてクライアントに送付させる攻撃手法
  - 1/2^40程度の確率で pre master secret の鍵解読が成功する

  - 対策としては explicit なECパラメータ/DHパラメータの利用を止め、named パラメータのみ利用する。ServerKeyExchangeの署名をセッションハッシュ(*)に変更する。

TLS1.3では named group(DH/ECDH)のみサポート 全ての署名はセッションハッシュ

(* セッションハッシュ：それまでやりと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

- Key Deviation(鍵導出): TLSの対称暗号の秘密鍵やMAC鍵を生成する方式
- TripleHandshake攻撃
  - master secretが client/server nonce と pre master secret で決定されることを突いた攻撃
  - MiTMで同じ nonce の２つのハンドシェイクを行い、同じ master secret を持つ２つのTLS接続を確立する。
  - 攻撃者はクライアント認証用Renegotiationを誘発させ、サーバとクライアント間のTLS接続のセキュリティパラメータを完全に同期させ、悪意のあるデータを送り込むことが可能になる。

TLS1.3ではmaster secretの生成を nonce ではなくセッションハッシュを用いるように変更

TLS1.3の仕様について

再掲 注意： 内容は2015年11月2日時点での draft (*)を元にしています。今後の仕様策定作業で本プレゼンの内容が変更になる場合がありますのでご注意ください。

(* https://github.com/tlswg/tls13-spec/ の master HEAD)

TLS1.3の背景

- 2008年 TLS1.2仕様化から7年
- 古い暗号やプロトコルの危殆化に伴う、TLS機能の抜本的な見直しの必要性
- 常時TLS化を目指し、将来的に安心で強固なセキュリティの必要性(HTTP/2)
- モバイル環境の普及やWebアプリの高度化に伴う高パフォーマンス、低レイテンシー化要望の高まり (QUIC)

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
   - フルハンドシェイクは、 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)

これまでのTLS1.3の歩み

2013/03	IETF-86 TLS1.3の議論が始まる
2014/03	Charter化完了 マイルストーン2014/11
2014/04 draft00/01	TLS/1.2のコピー
2014/07 draft02	各種廃止(CBC,RC4,compression…) EarlyDataExtention導入
2014/10 draft03	RFC449(ECC)導入 HelloRetry導入
2015/01 draft04	CCS,Renegotiation廃止 session_hash導入
2015/03 draft05	SSL2.0/3.0との互換性を禁止

2015/06 draft06	IV廃止 RecordバージョンをTLS1.0に固定
2015/07 draft07	OPTLSの導入 PRF→HKDFに変更
2015/08 draft08	0-RTT用拡張を統合 0-RTTでも常にハンドシェイク署名 MD5/SHA-224のサポート廃止 MTI Extensionの指定
2015/10 draft09,10	MTI Cipherの決定 SHA-1,DSAの廃止 ハンドシェイクの署名をRSA-PSSに レコードPaddingの導入 鍵交換をKeyShare拡張に統一

TLS1.3仕様策定の進め方と今後の見込み

- IETF TLS WGでのメーリングリストの議論がメイン
- ドラフトは GitHub https://github.com/tlswg/tls13-spec/ でMarkdown形式で作成されている。
- Editorialな修正は、GitHub issue/pull requestで修正
- 設計に関連する課題は issue+メーリングリスト
- これまで6回のInterim(中間会議）を開催し、F2Fで集中的に議論、設計を行っている。
- 2016年2月にTRON Workshop(TLSv1.3 - Ready or Not?)(*)を開催
- Workshopまでに最終ドラフト直前まで行くのではないか。

(* https://www.internetsociety.org/events/ndss-symposium-2016/tron-workshop-call-papers)

TLS1.3の特徴

- 様々な機能・項目を見直し・廃止
  - 時代に合わなくなったもの、より効率的な仕組みに変更修正されたものなど、TLS1.2の機能・項目を数多く廃止。（後述）
- 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必須やパディング等将来的な攻撃に備える。
- 性能向上
  - QUICで使われているような 0-RTTの接続をサポートし、接続レイテンシーを下げられるようにしている。将来的にQUICはTLS1.3をサポート(*)する方向。

(* QUICとTLSでフレームフォーマットが異なるので全く同一にならない可能性があります。)

TLS1.3の廃止項目

これまでの機能の必要性を全面的に見直し

機能	Compression	CRIME/TIME等の圧縮タイミング攻撃対策
	Renegotiation	Triple Handshake等Renegotiation前後の状態引継ぎの不備を対策 廃止に伴いクライアント認証とRekeyを行う方式を変更
	SessionID resumption	PSKと同一方法で resumptionが可能になったため
暗号署名方式	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)	RC4危殆化、MtEを狙ったパディングオラクル攻撃の対策
	IVフィールド	AEADのみになったため nonce は seq no から生成
	AEADのadditional data	要検証なレコードヘッダ部はnonce/暗号データ内で改ざん検知可能
	custom DHE	Cross-Protocol対策で named groupに統一
	ECDHE compress format	uncompress書式だけで利用用途が十分
	anonymous DH	raw public key (RFC7250)を使うか、証明書の検証をしない

TLS1.3の廃止項目 cont'd

ハンドシェイク書式	record layerのバージョン	実質的に利用用途がないため。後方互換だけのために残されている。
	random中のgmt unix time	乱数生成が高度化され利用用途がなくなったため
	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	危殆化プロトコルのサポート廃止
	不要タイプの削除	ChangeCipherSpec,HelloRequest,Sever/ClientKeyExchange等(後述)
拡張	max_fragment_length	最大値は2^14 に固定
	truncated_hmac	AEADに利用できないし、安全でないため。
	srp	PSKに置き換え？
	encrypt_then_mac	AEAD利用のため
	extended_master_secret	TLS1.3仕様に取り込み
	SessionTicket	Resumption/PSKで利用するTicketに置き換えるため
	renegotiation_info	Renegotiationを禁止

TLS1.2と1.3の違い: MTI(必須暗号方式)

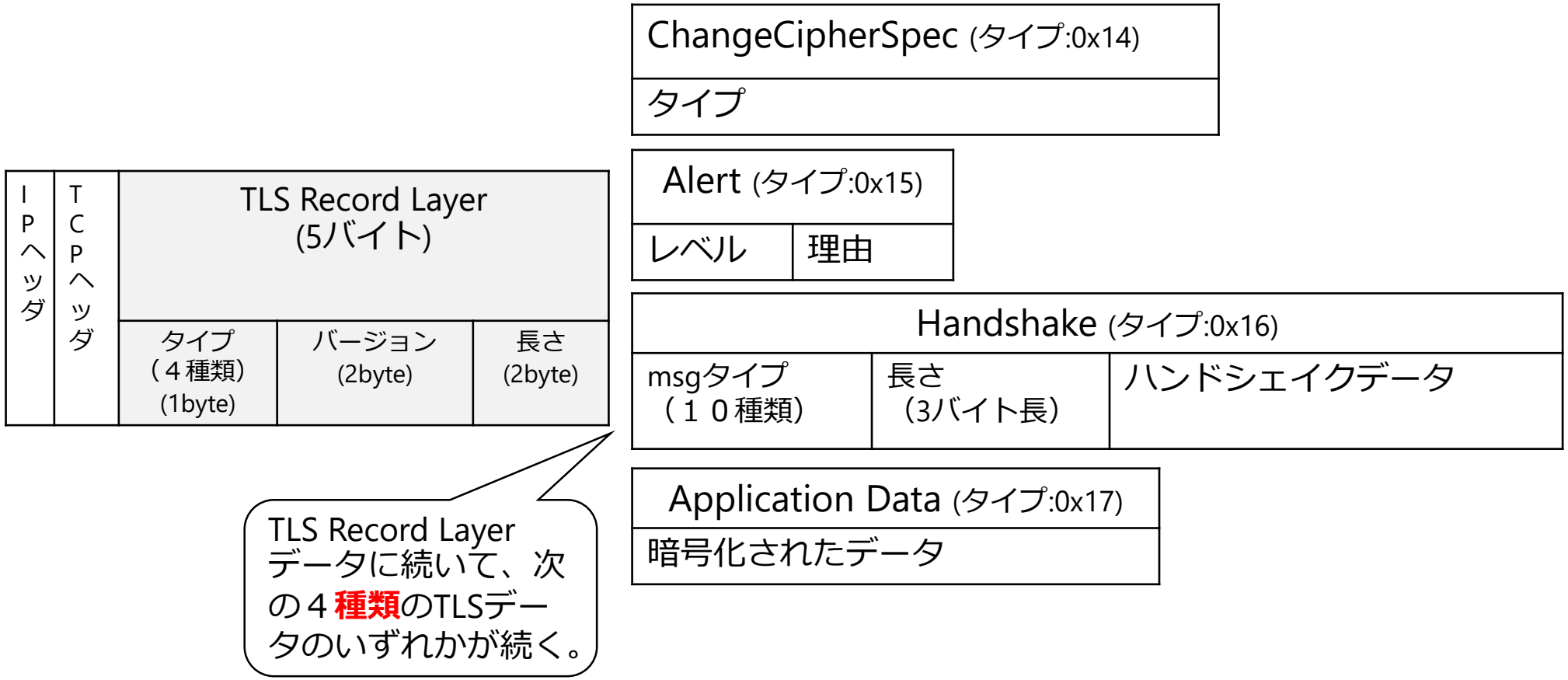
- TLS1.2
  - TLS_RSA_WITH_AES_128_CBC_SHA

- TLS1.3 (MUST)
  - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
  - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
  - NIST P-256 curve
- TLS1.3(SHOULD)
  - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305
  - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
  - TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305
  - X25591 curve

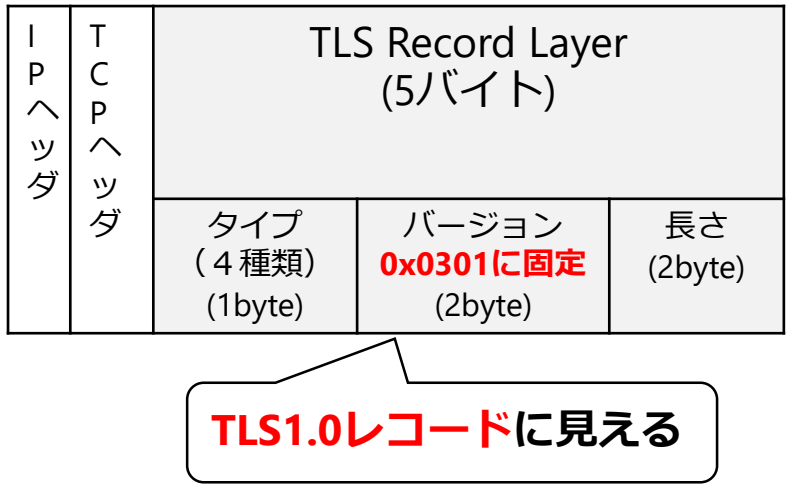
ChaCha20+Poly1305 (RFC7539)をサポート

TLS _ 鍵交換 _ デジタル署名 _WITH_ 対称暗号 _ 鍵長 _ 暗号モード _ HKDFに利用するハッシュ

TLS1.2と1.3の違い: フレーム書式
TLS1.2の構造



TLS1.3の構造



TLS1.0レコードに見える

Alert (タイプ:0x15)	
レベル	理由

ChangeCipherSpecを廃止

Handshake (タイプ:0x16)		
msgタイプ（１０種類）	長さ	ハンドシェイクデータ

Application Data (タイプ:0x17)
暗号化されたデータ

Early Handshake (タイプ:0x19)		
msgタイプ	長さ	ハンドシェイクデータ

レコードタイプ	TLS1.2	TLS1.3
0x00	N/A	予約
0x14	ChangeCipherSpec	廃止
0x15	Alert	
0x16	Handshake	
0x17	Application Data	
0x19	N/A	Early Handshake

Padding判定用

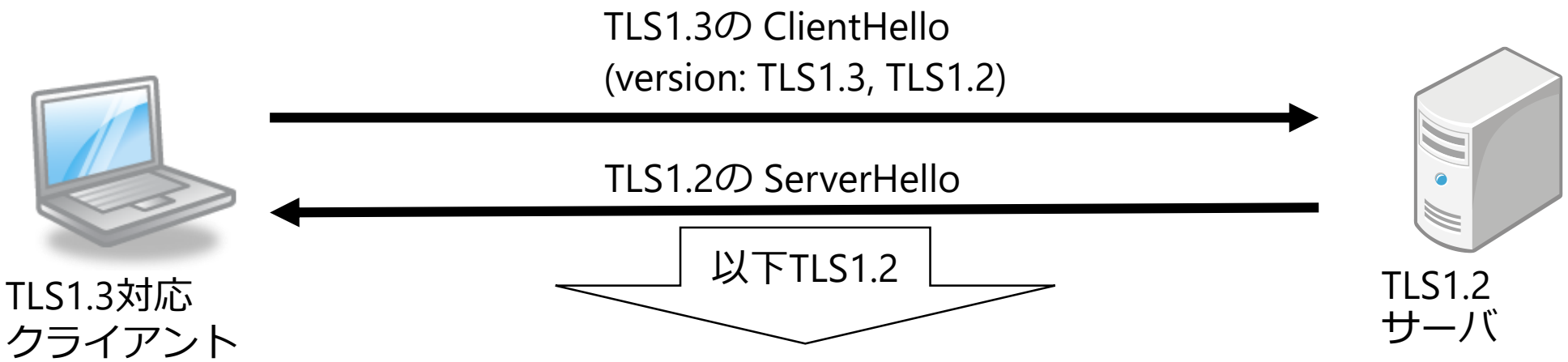
SeverHelloの拡張領域の途中で即暗号化なので不要

0-RTT用の情報を格納

TLS1.3のハンドシェイクデータ書式

Handshake (タイプ:0x16)		
msgタイプ (1バイト長)	長さ (3バイト長)	ハンドシェイクデータ

TLS1.2とTLS1.3共に同じ書式
→特にClientHelloの互換性確保



TLS1.2と1.3の違い:ハンドシェイクタイプ

ハンドシェイク	TLS1.2	TLS1.3
0x00	hello_request	廃止
0x01	client_hello	
0x02	server_hello	
0x04	—	session_ticket
0x06	—	hello_retry_request
0x08	—	encrypted_extensions
0x0b	certificate	
0x0c	server_key_exchange	廃止
0x0d	certificate_request	
0x0e	server_hello_done	廃止
0x0f	certificate_verify	
0x10	client_key_exchange	廃止
0x11	—	server_configration
0x14	finished	

renegotiation廃止のため不要

ServerHello拡張に移動

2-RTT接続廃止のため不要

ClientHello拡張に移動

TLS1.2と1.3の違い: ClientHelloのデータ

TLS1.2

Record Layer				Handshake (ClientHello)									
type	protocol version		length (2byte)	msg type	length (3byte)	client version		random		session id	cipher suite	compression	Extension
	major	minor				major	minor	gmt_unix_time	random				
0x16	0x03	0x01		0x01		0x03	0x03	4 bytes	28 bytes	可変	可変	可変	可変

TLS1.3

時刻入り乱数の廃止

セッションIDと圧縮の廃止

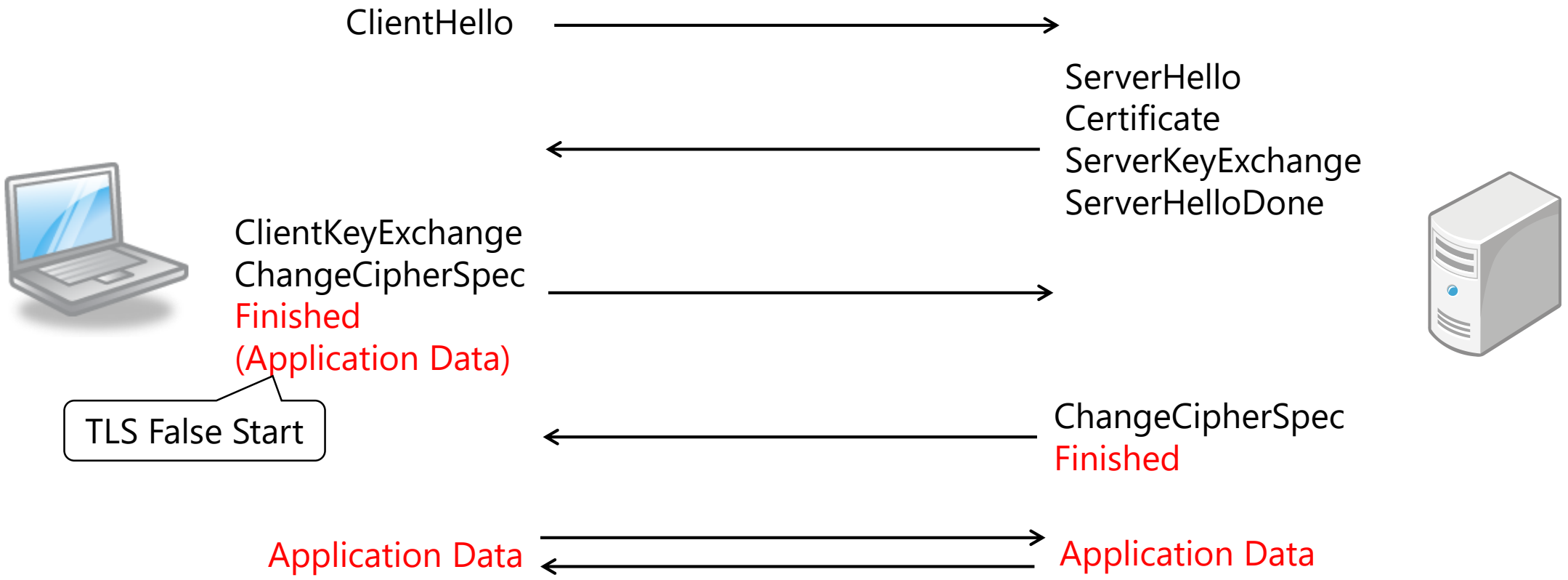
Record Layer				Handshake (ClientHello)								
type	protocol version		length (2byte)	msg type	length (3byte)	client version		random	session id	cipher suite	compression	Extension
	major	minor				major	minor					
0x16	0x03	0x01		0x01		0x03	0x04	**32 byte**	**0x00**	可変	**0x00**	可変

TLS1.2と1.3の違い：拡張機能のデフォルト

Extension Type	**TLS1.2**	**TLS1.3**
0x0a	DefaultではないがRFC4492で規定	supported_groups RFC4492(ECDH)と draft-tls-negotiated-ff-dheを取り込み named DHGroupを追加
0x0d	signature_algorithms (md5, sha1, sha224, sha256, sha384, sha512)	signature_algorithms (sha1, sha256, sha384, sha512) sha224:112bitレベルで廃止 sha1: TLS1.2フォールバック
TBD	ー	early_data 0-RTT接続で利用する拡張データを格納
TBD	ー	pre_shared_key RFC4279(TLS-PSK)を取り込み 拡張領域に移動
TBD	ー	key_share ClientKeyExchange/ServerKeyExchangeを 拡張領域に移動

（SNIやALPN等の拡張は従来のまま default外）

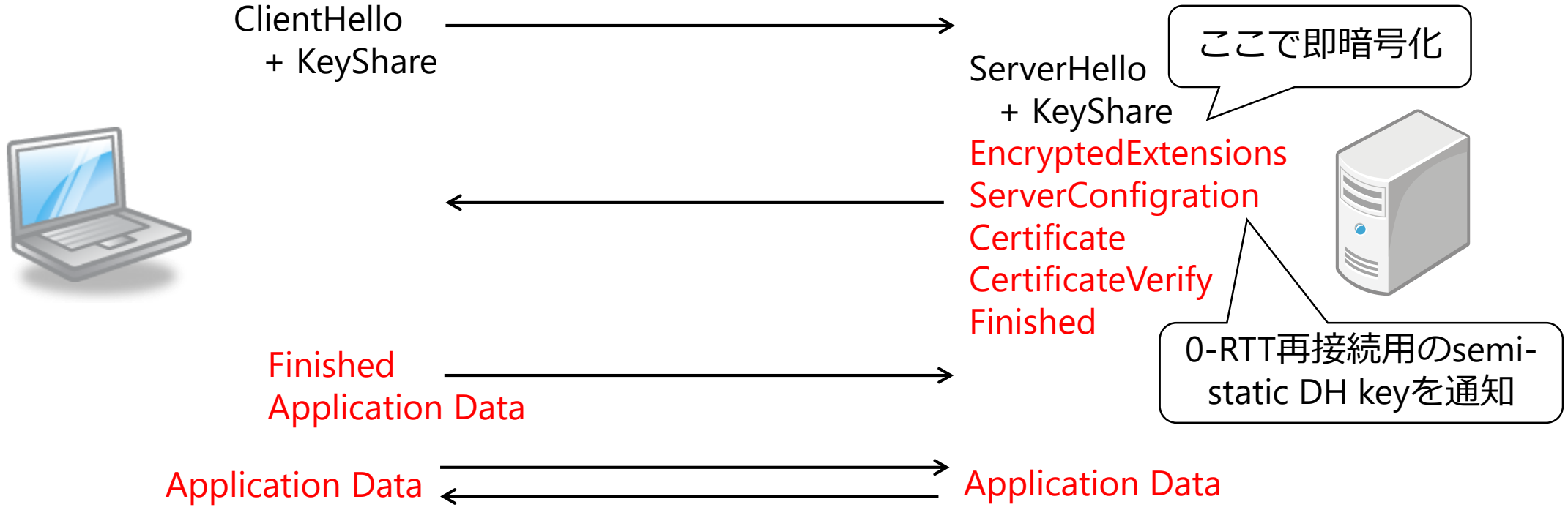
TLS1.2と1.3の違い: ハンドシェイク
TLS1.2の full handshake



2-RTT 又は1-RTT with False Start

(赤文字は暗号化されているデータ)

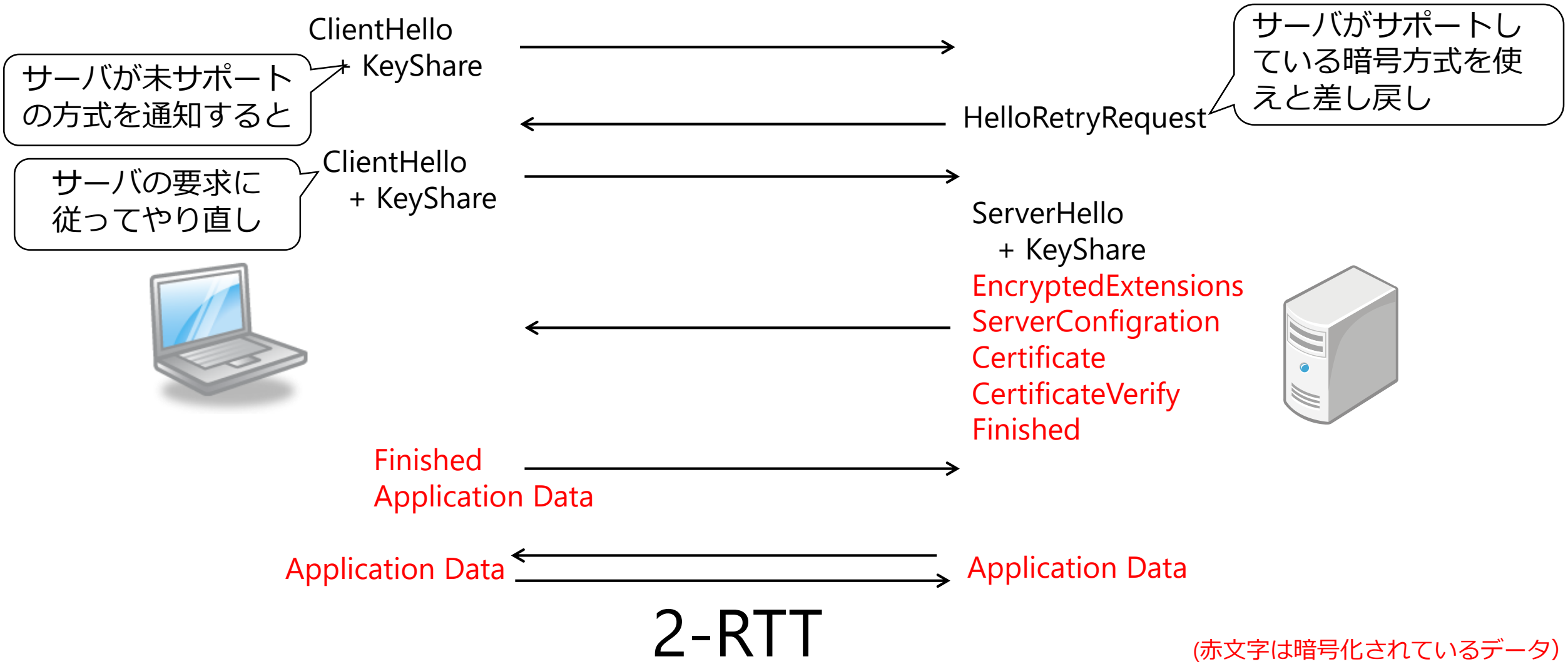
TLS1.2と1.3の違い: ハンドシェイク
TLS1.3の full handshake



1-RTT

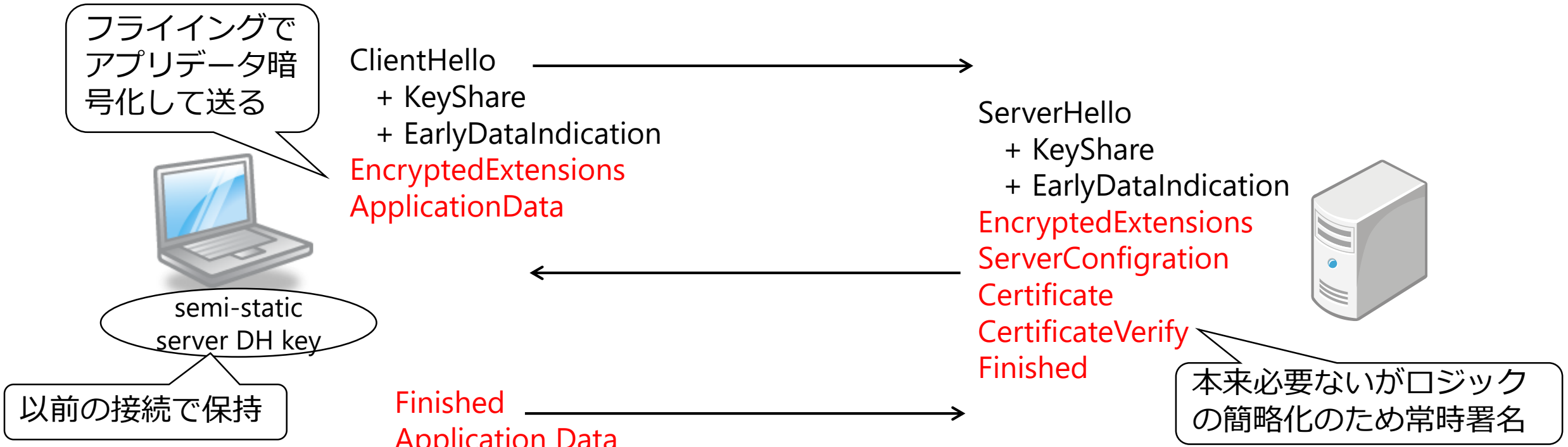
(赤文字は暗号化されているデータ)

TLS1.2と1.3の違い: ハンドシェイク
TLS1.3 鍵合意できない時は最悪2-RTT



(赤文字は暗号化されているデータ)

TLS1.2と1.3の違い: ハンドシェイク
TLS1.3 0-RTT再接続



0-RTT

(赤文字は暗号化されているデータ)

TLS1.2と1.3の違い: ハンドシェイク

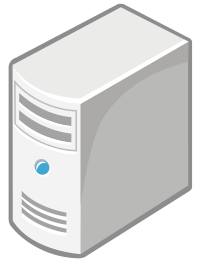
TLS1.3　Resumption, PSK

Resumption時は
SessionIDの替わりに
psk_identity を使う

ClientHello
+ KeyShare
+ PreSharedKeyExtension

PreSharedKey

ServerHello
+ KeyShare
+ PreSharedKeyExtension
EncryptedExtensions
Finished



PreSharedKey

Finished

Application Data

Application Data

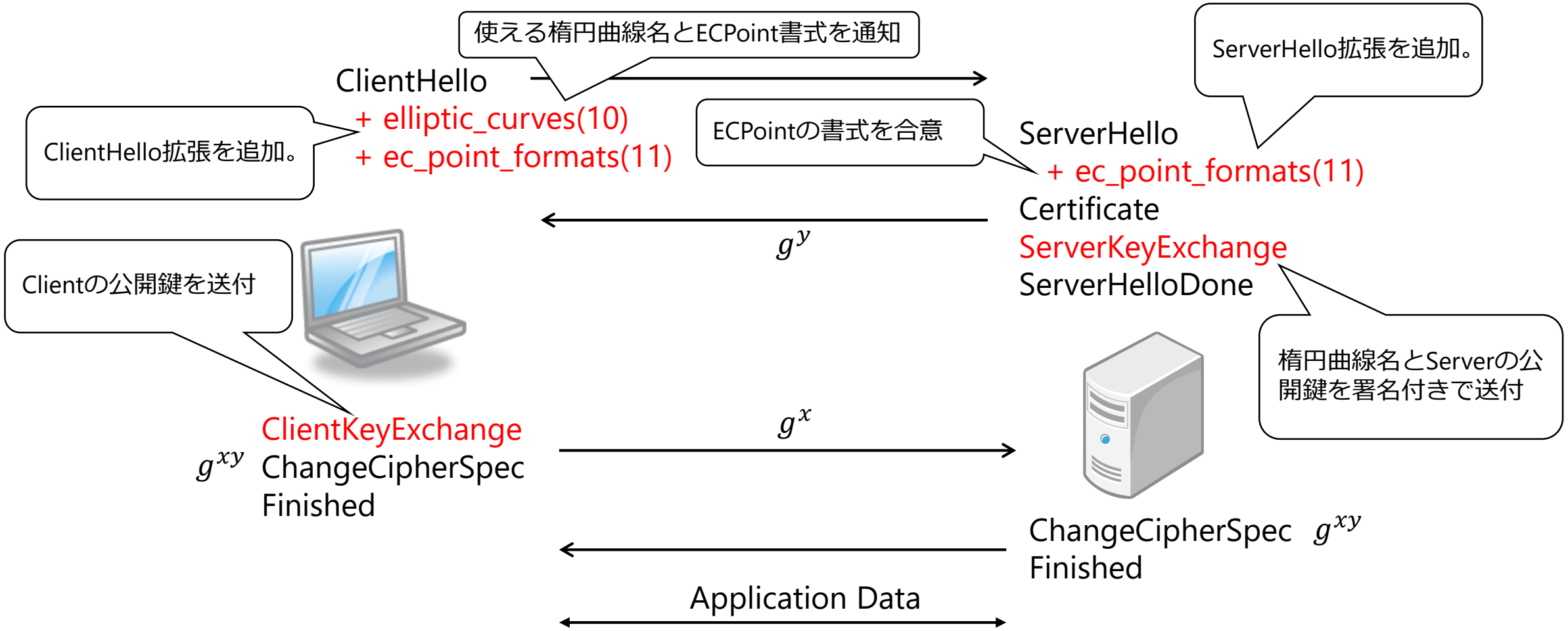
1-RTT

(赤文字は暗号化されているデータ)

TLS1.2と1.3の違い:
TLS1.3の新たな鍵交換方式と鍵スケジュール (OPTLSの採用)

- signature based handshake → DH based handshake にする提案
- 当初 DH証明書と公開鍵の offline signature(後述)の利用を想定していたため反対意見が多かった。
- プロトコルロジックや鍵生成が簡略化され、PSKの導入もしやすくなるのでTLS1.3に採用する方針となった。
- しかし offline signature機能を外し、online署名された semi-static DH鍵を利用することにした。

TLS1.2と1.3の違い:
TLS1.2のECDHEのハンドシェイク



TLS1.2と1.3の違い: TLS1.2の鍵スケジュール

pre master secret
（任意のバイト数:鍵交換による）
サーバ・クライアント間の鍵交換方式で生成し、秘密的に共有する

ECDHE/DHE: g^{xy}

PRF: Pseudo Random Functionを利用

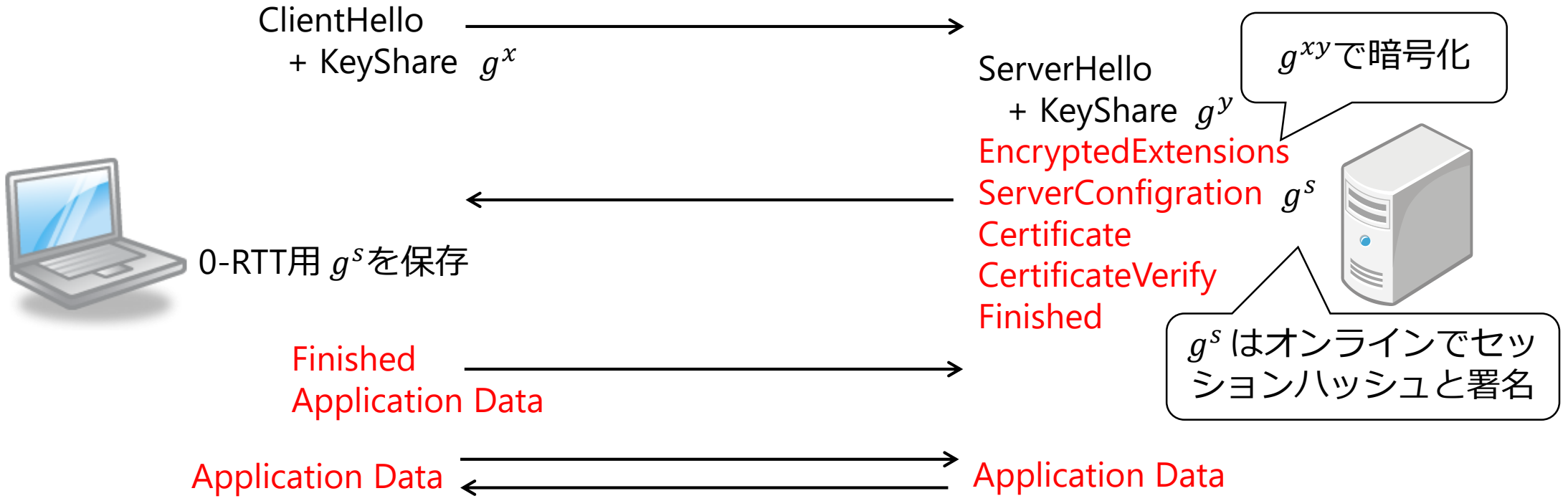
master secret
（48 bytes)
PRF(pre_master_secret, "master secret", client_random+server_random)

keyblock
（任意のバイト数:利用暗号方式による)
PRF(master_secret, "key expansion", server_random+client_random)

client_write_MAC	server_write_MAC	client_write_key	server_write_key	client_write_IV	server_write_IV

TLS1.3の鍵交換(full-handshake)

最初(full-handshake)は全部一時鍵(Ephemeral Key)で鍵スケジュールを行う。

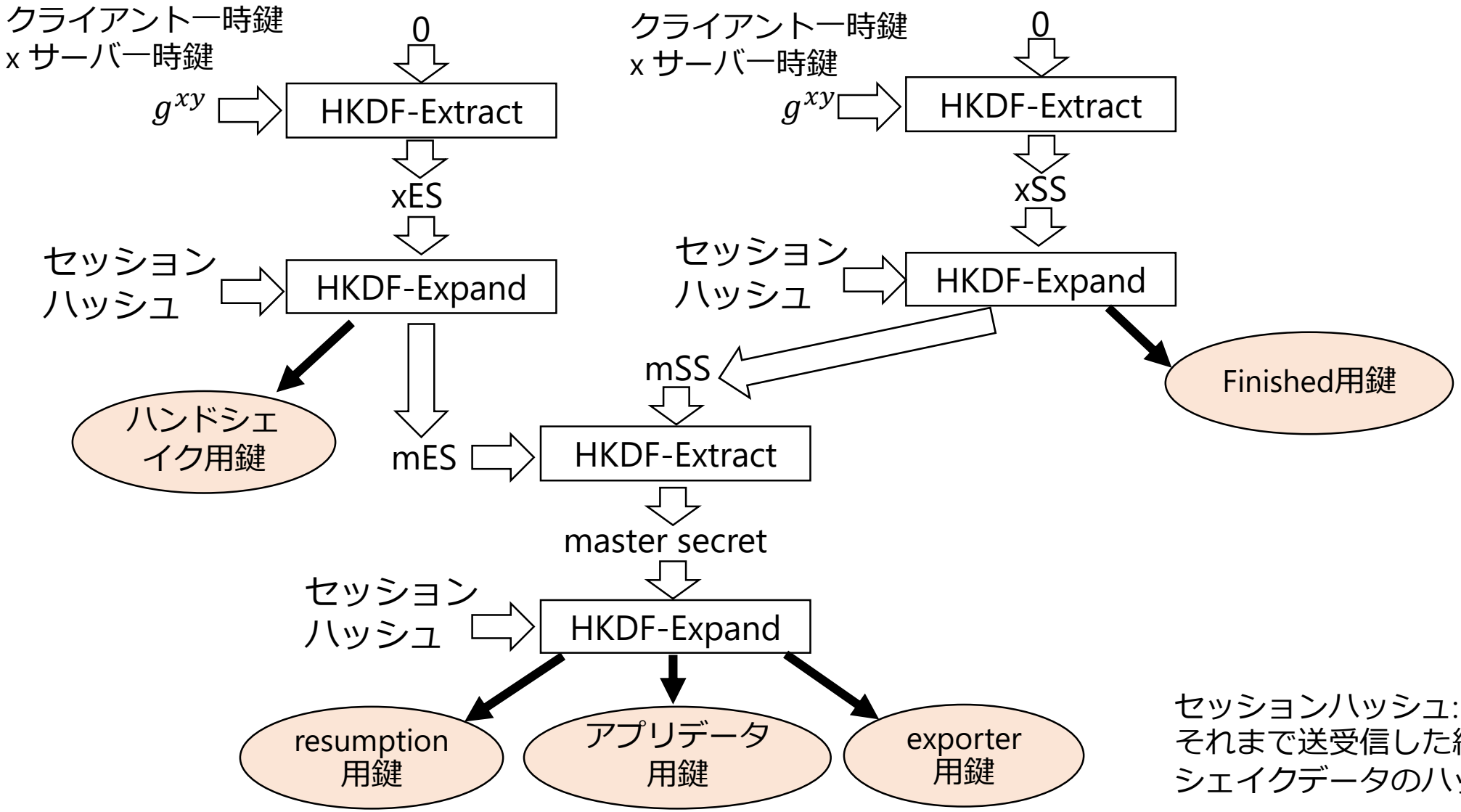


g^x: クライアント一時鍵
g^y: サーバ一時鍵
g^s: サーバ semi-static 鍵

(赤文字は暗号化されているデータ)

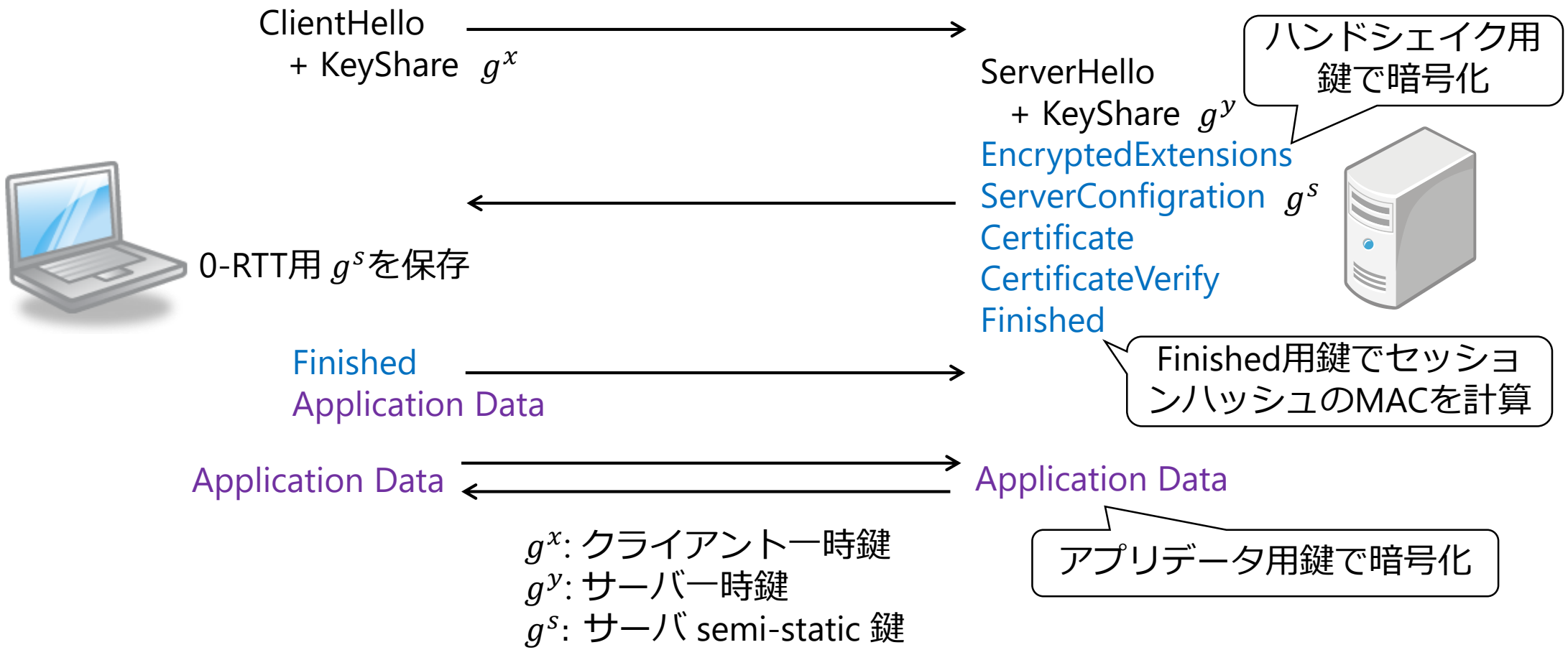
The OPTLS Protocol and TLS 1.3 (Hugo Krawczyk, HoeteckWee)を参考、改変

TLS1.3鍵スケジュール(full-handshake)

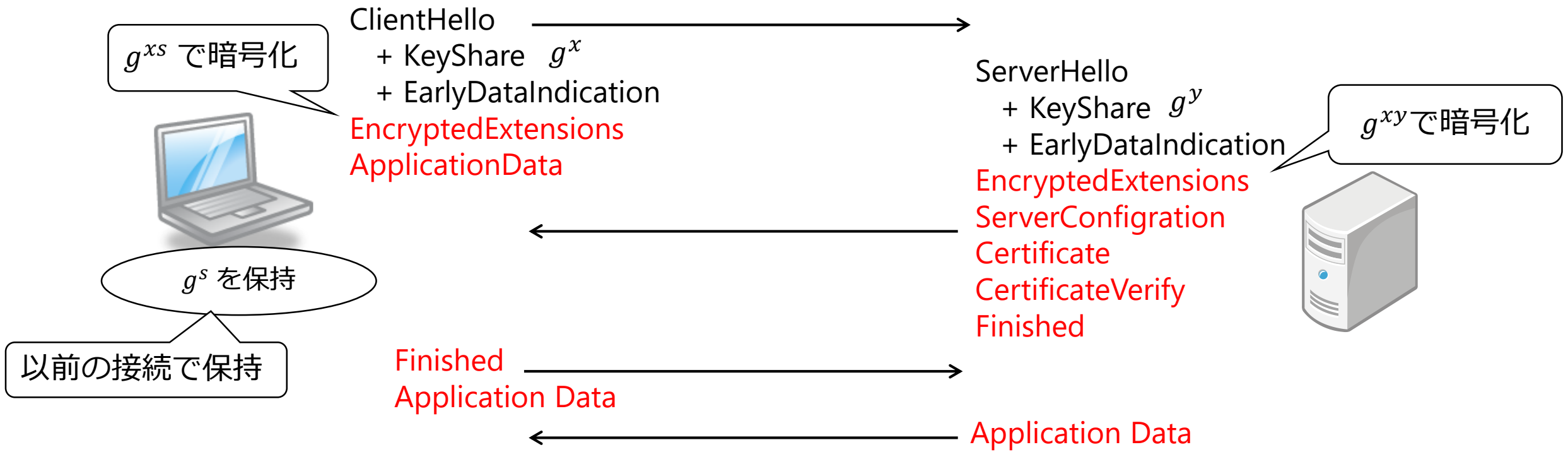


セッションハッシュ:
それまで送受信した総ハンドシェイクデータのハッシュ値

Full-Handshakeでは3種類の鍵を利用



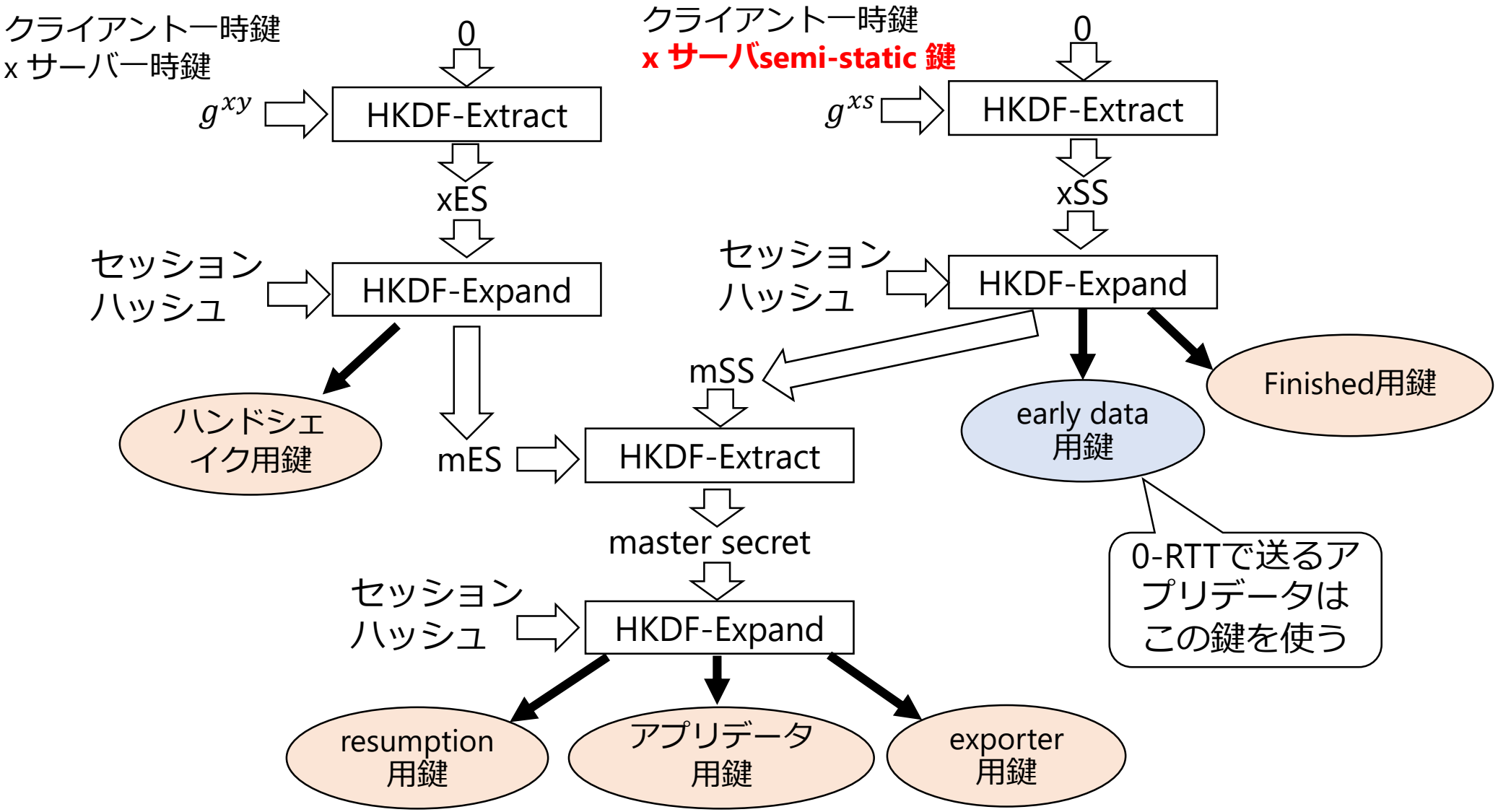
TLS1.3の鍵交換(0-RTT)



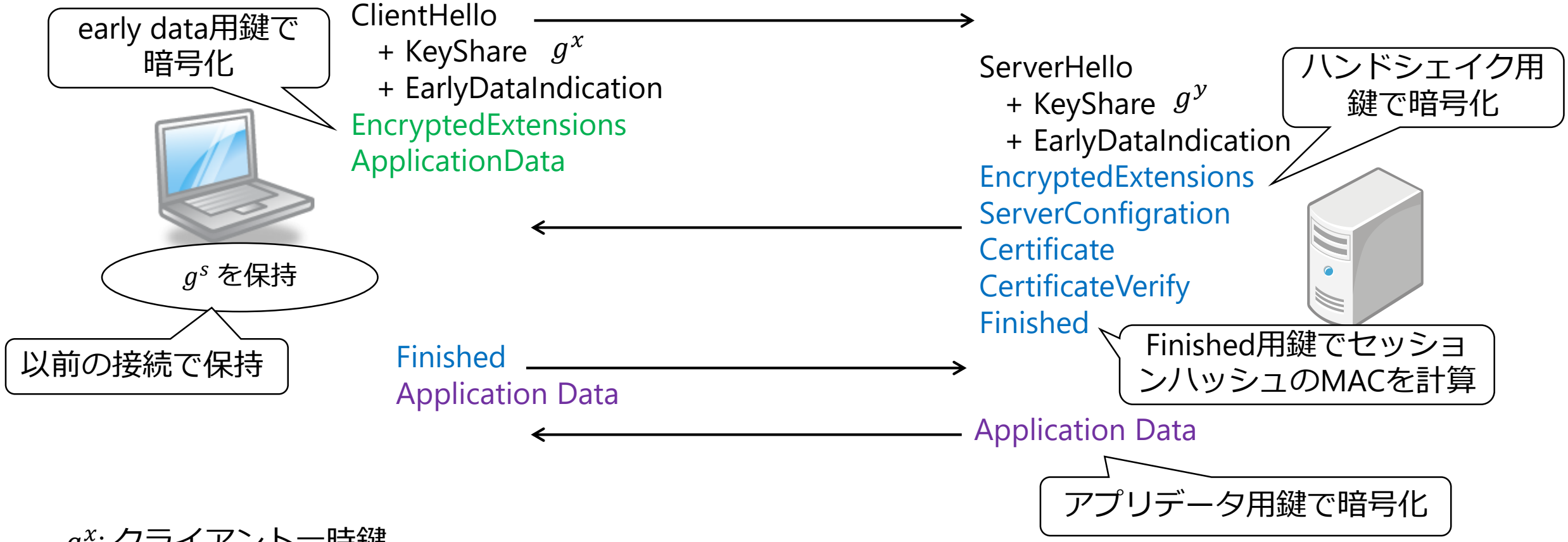
g^x: クライアント一時鍵
g^y: サーバ一時鍵
g^s: サーバ semi-static 鍵

(赤文字は暗号化されているデータ)

TLS1.3鍵スケジュール(0-RTT)

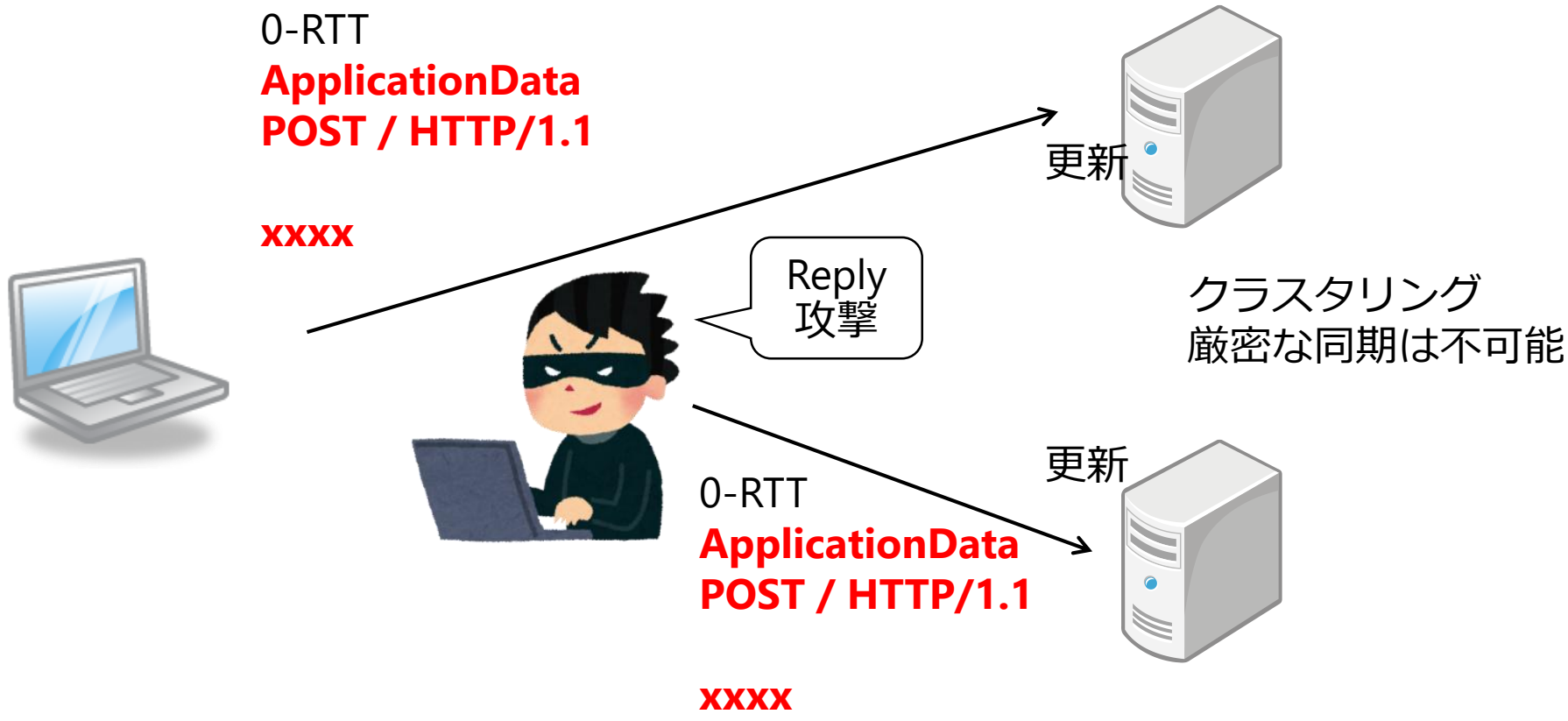


0-RTTでは4種類の鍵を利用



g^x: クライアント一時鍵
g^y: サーバ一時鍵
g^s: サーバ semi-static 鍵

TLS1.3 0-RTTの問題点



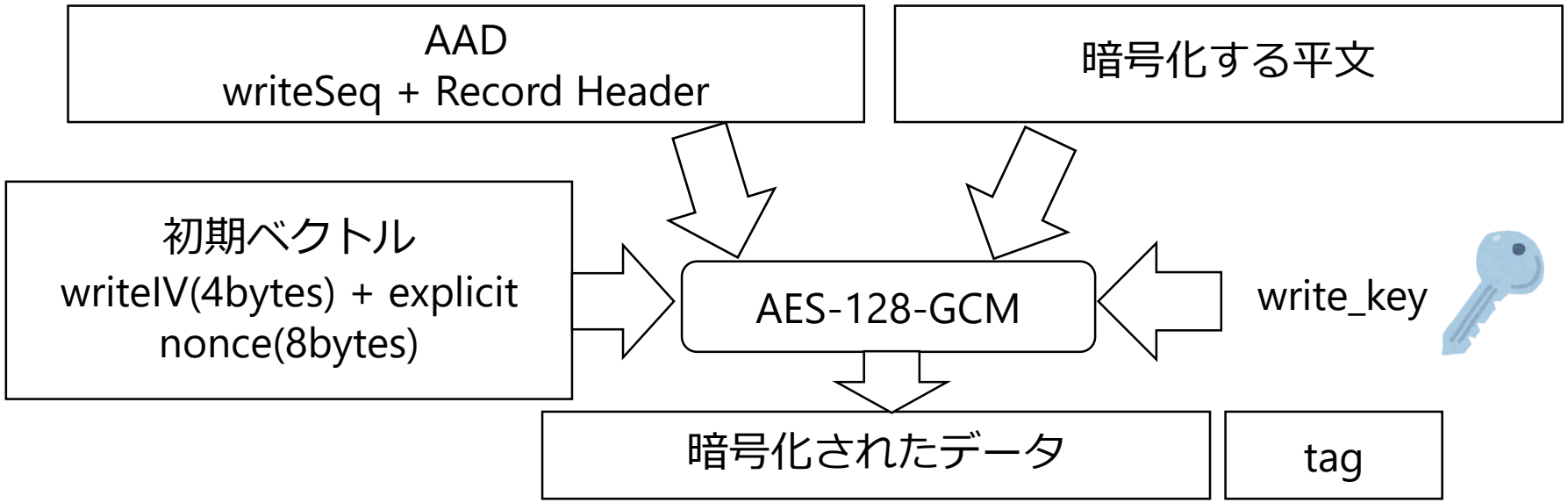
Reply攻撃に弱いため、最初のデータは冪等性のあるリクエストのみに制限しなければならない

TLS1.2のアプリケーションの暗号化

AES-128-GCMの場合

Record Header					**explicit nonce (8 byte)**	**暗号化されたデータ**	**tag (16 bytes)**
ContentType	Version		length (2byte)				
	major	minor					
0x17	0x03	0x03					

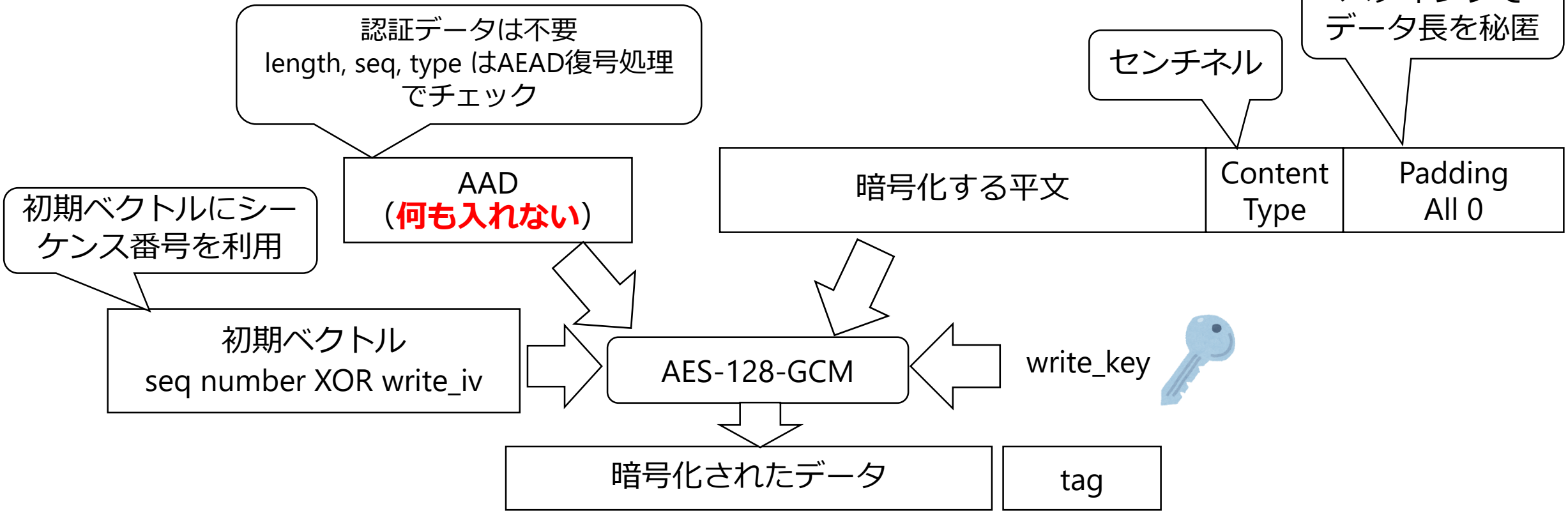
explicit nonce、暗号データ、
tag を合わせた長さに再計算



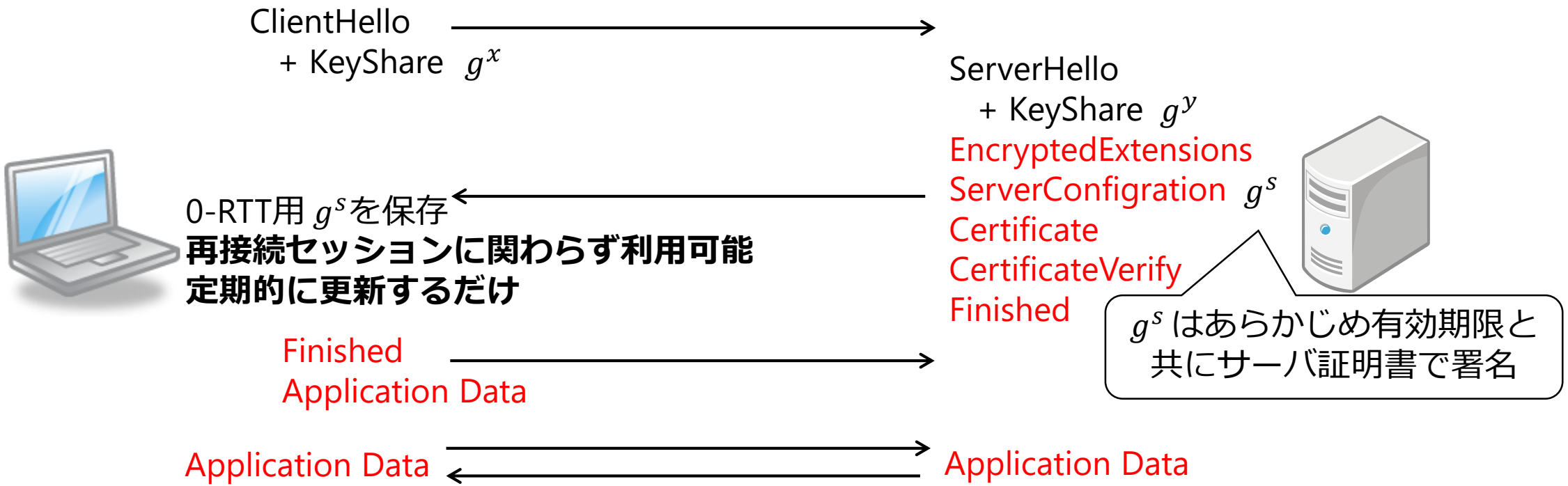
TLS1.3のアプリケーションの暗号化

AES-128-GCMの場合

Record Header					暗号化されたデータ	tag (16 bytes)
ContentType	Version		length (2byte)			
	major	minor				
0x17	0x03	0x01				



TLS1.3で見送られた機能(offline signature)



TLS1.3と並行して拡張仕様として検討を進める方針に

TLS1.3で見送られた機能(Ala-Carte暗号方式)

従来はセットメニュー方式

TLS _ 鍵交換 _ デジタル署名 _WITH_ 対称暗号 _ 鍵長 _ 暗号モード _ メッセージ認証(ハッシュ)

アラカルト方式にしてそれぞれ好きな組み合わせを選べるようにしたい

鍵交換 X デジタル署名 X 対称暗号 X 鍵長 X 暗号モード X メッセージ認証(ハッシュ)

従来の CipherSuiteの選択方式から大幅に変更になるため見送り

その他現在検討中、見送りの項目

- クライアント認証 （ドラフト作成中）
- 鍵更新（ドラフト作成中）
- SNIの暗号化
- ハンドシェイクのパディング

結局TLS1.3で何がどう変わるのだろうか？想定されること

- TLSライブラリ開発者
  - もうほとんどメジャーバージョンアップと同じぐらいの改変。
- TLSを利用するアプリ開発者、インフラ構築エンジニア
  - クライアント（ブラウザー）、サーバの対応状況は？
  - TLS1.2以前と本当に互換性を持てるのか？ミドルボックス、FWの影響はないか？
  - デフォルトが安全側になるので設定時の考慮は少なくなるのか？
  - 0-RTT用のAPIは、どうなるのか？ 本当に接続が速くなるのか？
- ユーザ
  - ブラウザーやサービス側のバージョンアップを待つだけ。特に気にしなくてもよい。切り替わったこともわからないだろう。
  - いつの間にか表示が速くなってた。
  - 不完全なサービス実装でユーザ側でTLS1.3機能を停止して回避ということも。