９０分で分かるサーバ証明書の最新動向〜いまTLSとトラストが暑いんです〜

運用の観点から見た TLSプロトコルの動き

大津 繁樹
ヤフー株式会社

2018年５月31日
Internet Week ショーケース in 広島

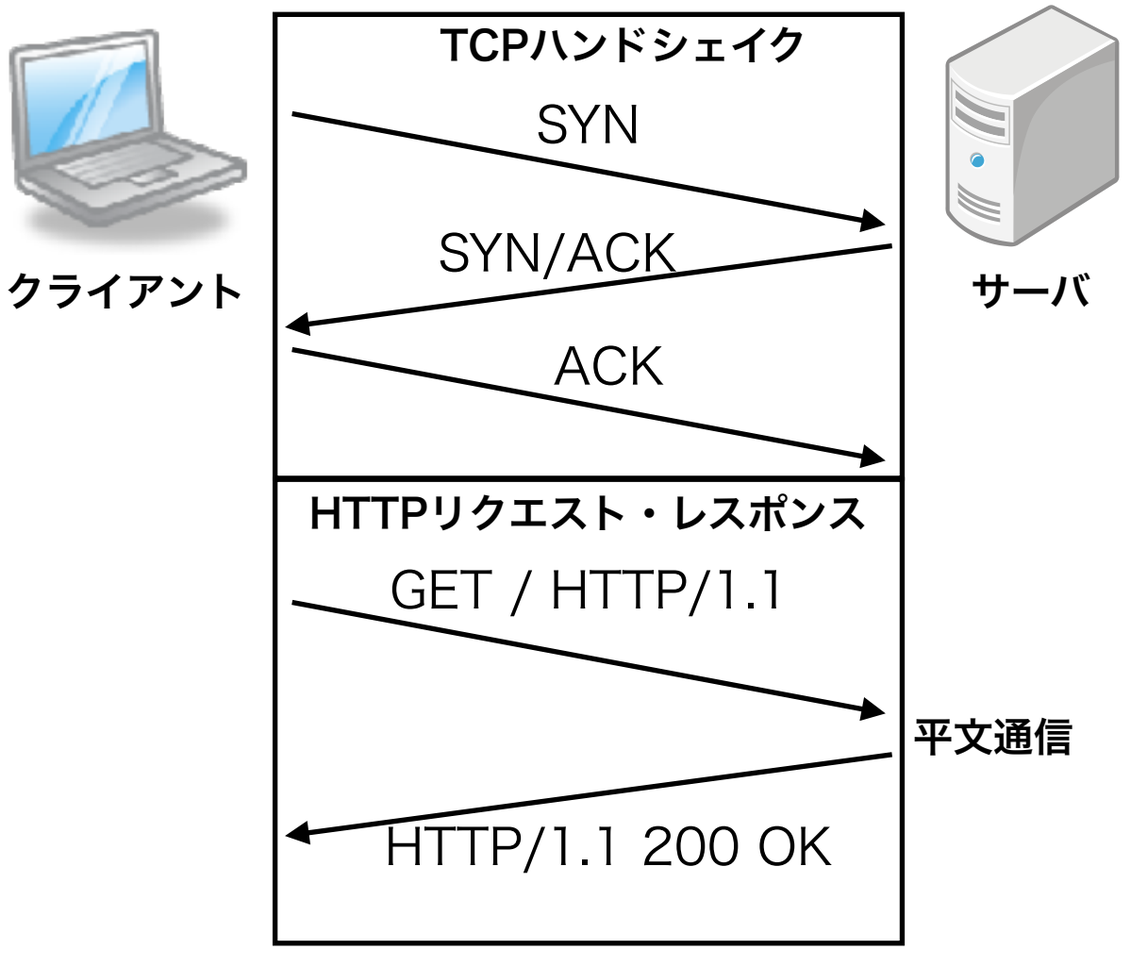
自己紹介

- ヤフー株式会社: CDNチーム所属、IETF標準化参加、黒帯（ネットワーク・セキュリティ）

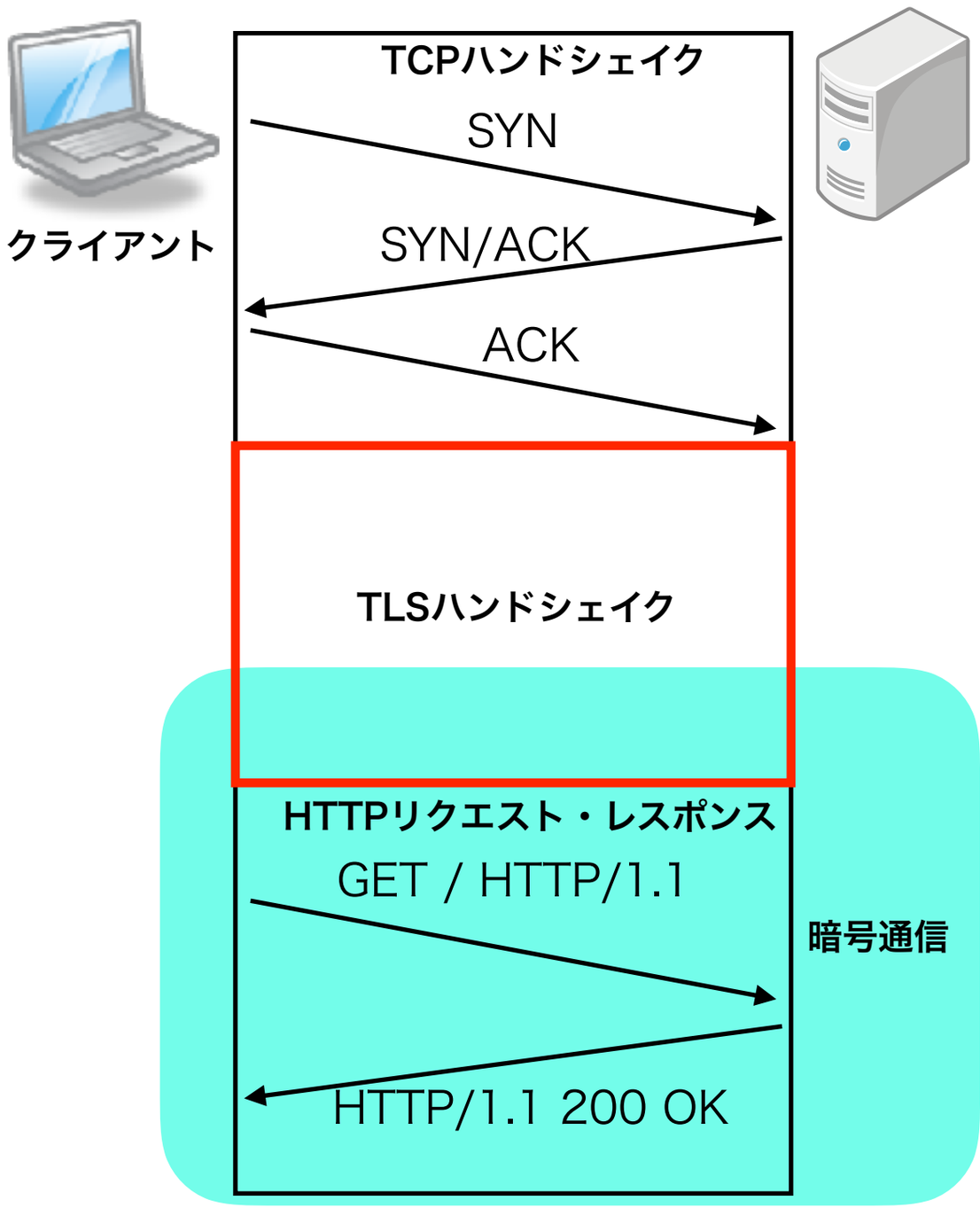
- Node.js コミッター: tls,crypto周りの実装、セキュリティチーム所属 (ヤフー株式会社 OSSデベロッパー認定者)

HTTPとHTTPS通信の違い

HTTP通信



HTTPS通信



今年秋、HTTP通信に対する警告強化

2018年9月リリース予定のChrome69から

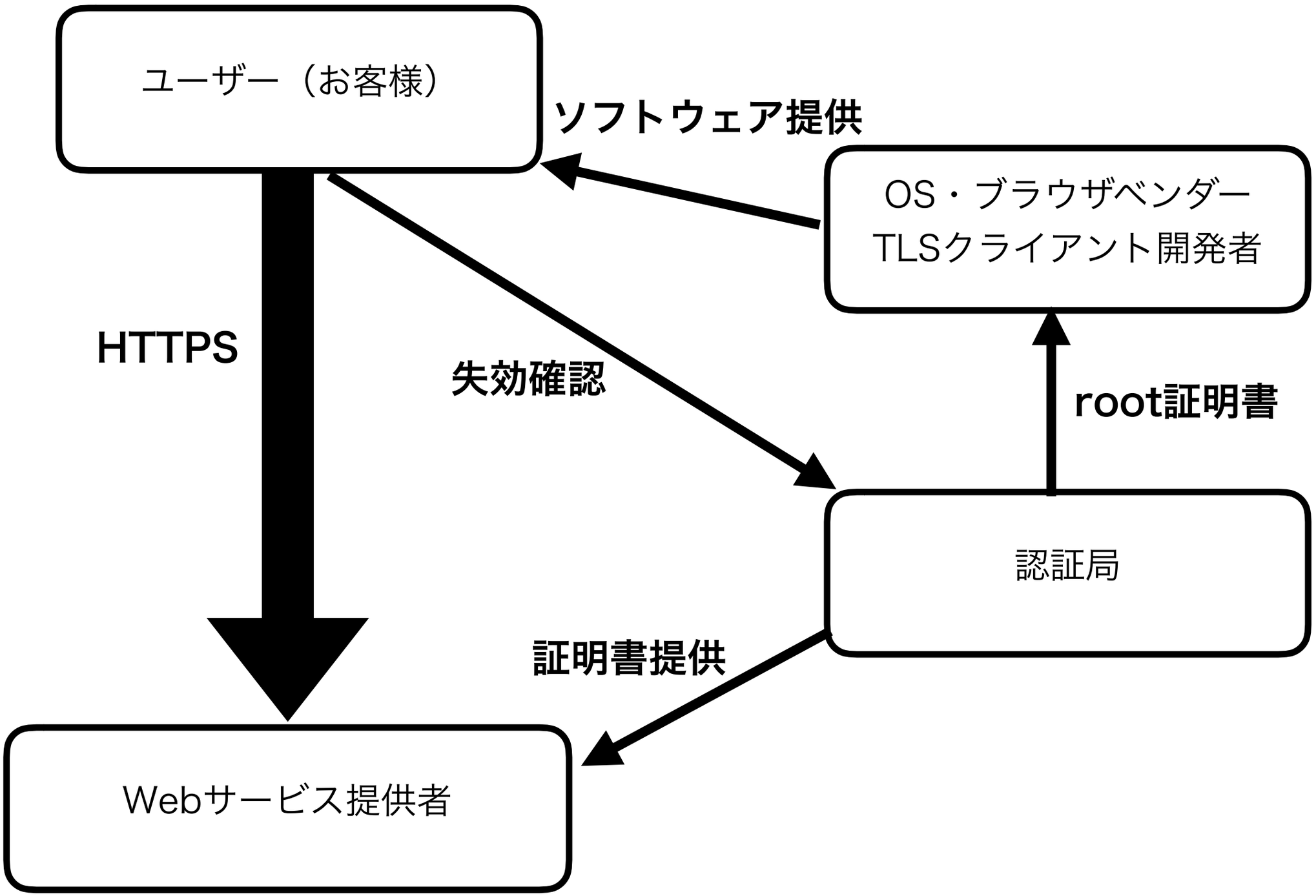
2018年10月リリース予定のChrome70から

本日の内容

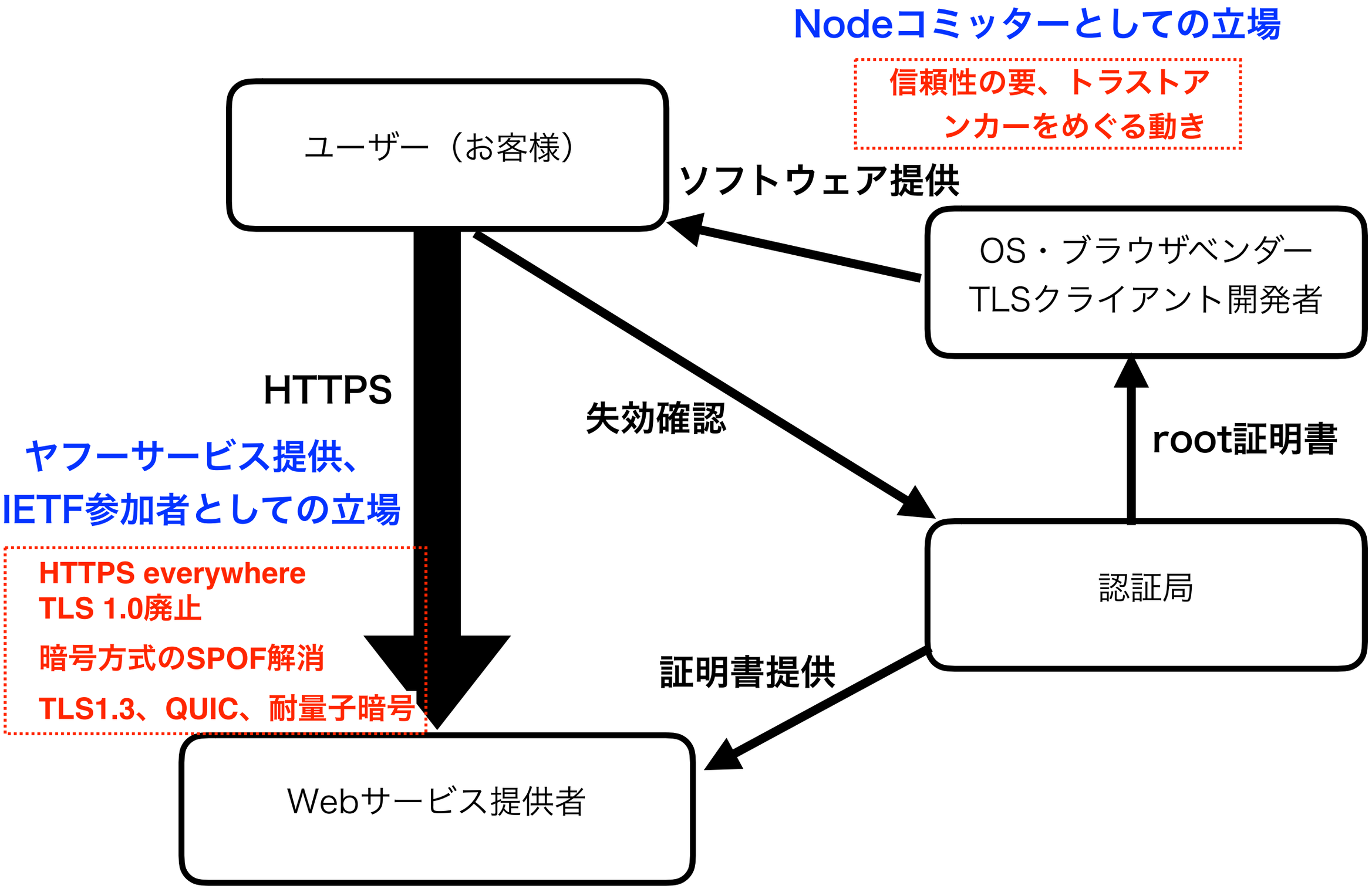
今後のHTTPSサービス運用で知っておくべきTLSプロトコルの現状の課題と最新動向について解説します。こんな疑問にお答えします。

- なぜ全部HTTPSにしないといけないの？
- ブラウザー、HTTPSサーバ、証明書(認証局)、どういう関係？
- うちのHTTPSサーバ、このままで大丈夫？ この後どうなるの？

HTTPS everywhere 時代のステークホルダー

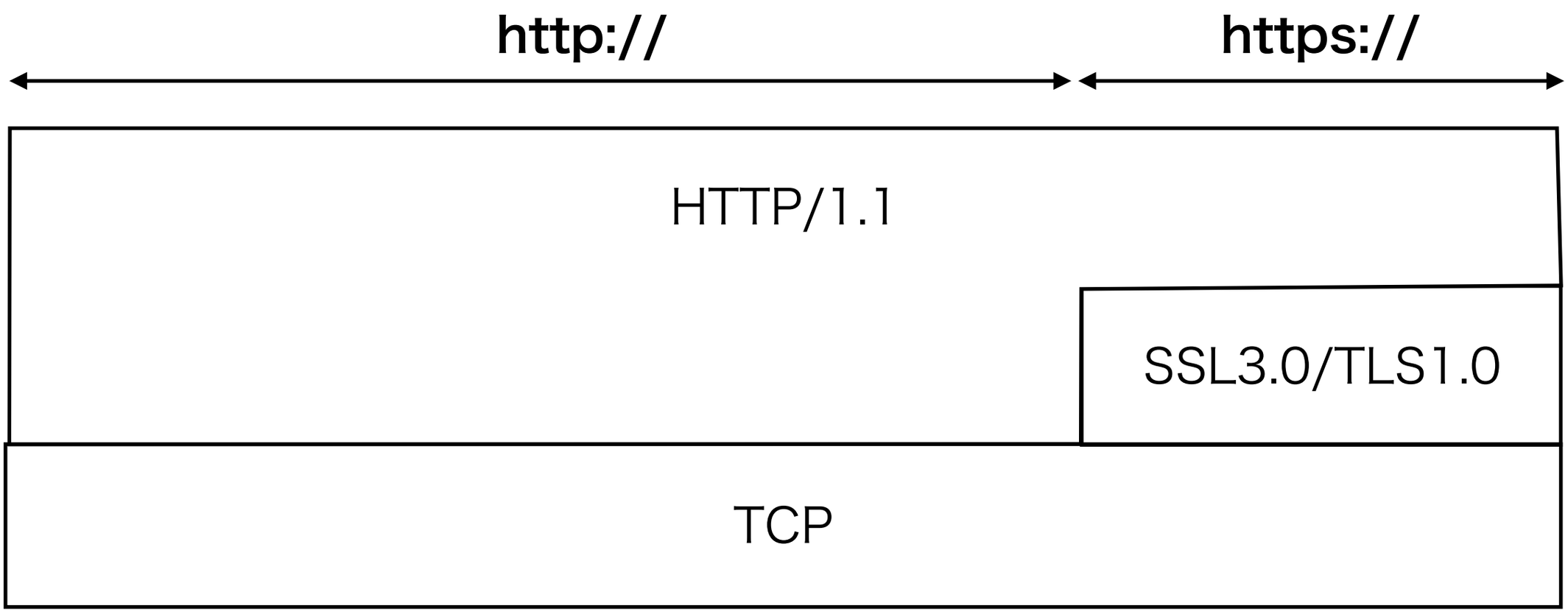


２つのポジションを使い分けして話をします



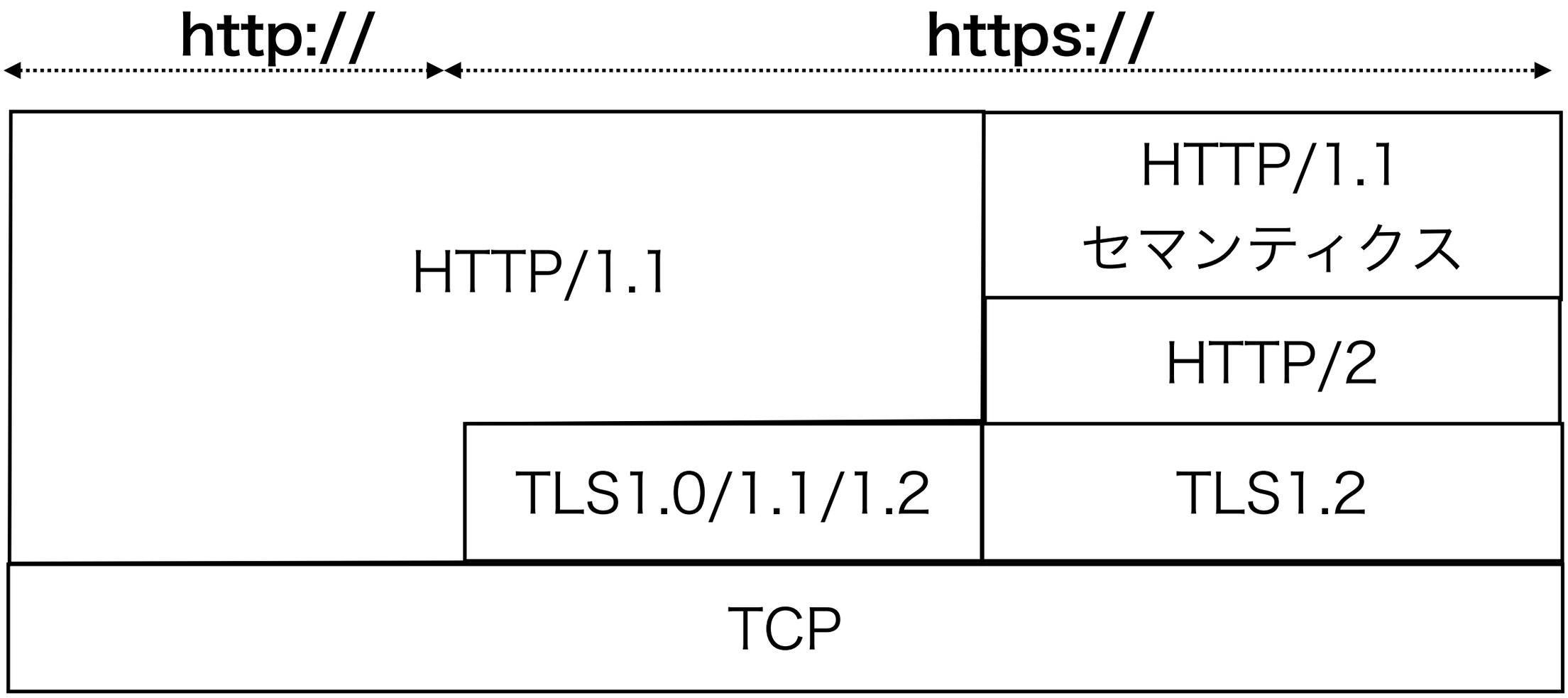
HTTPS Everywhere

1999年のWebプロトコルスタック図



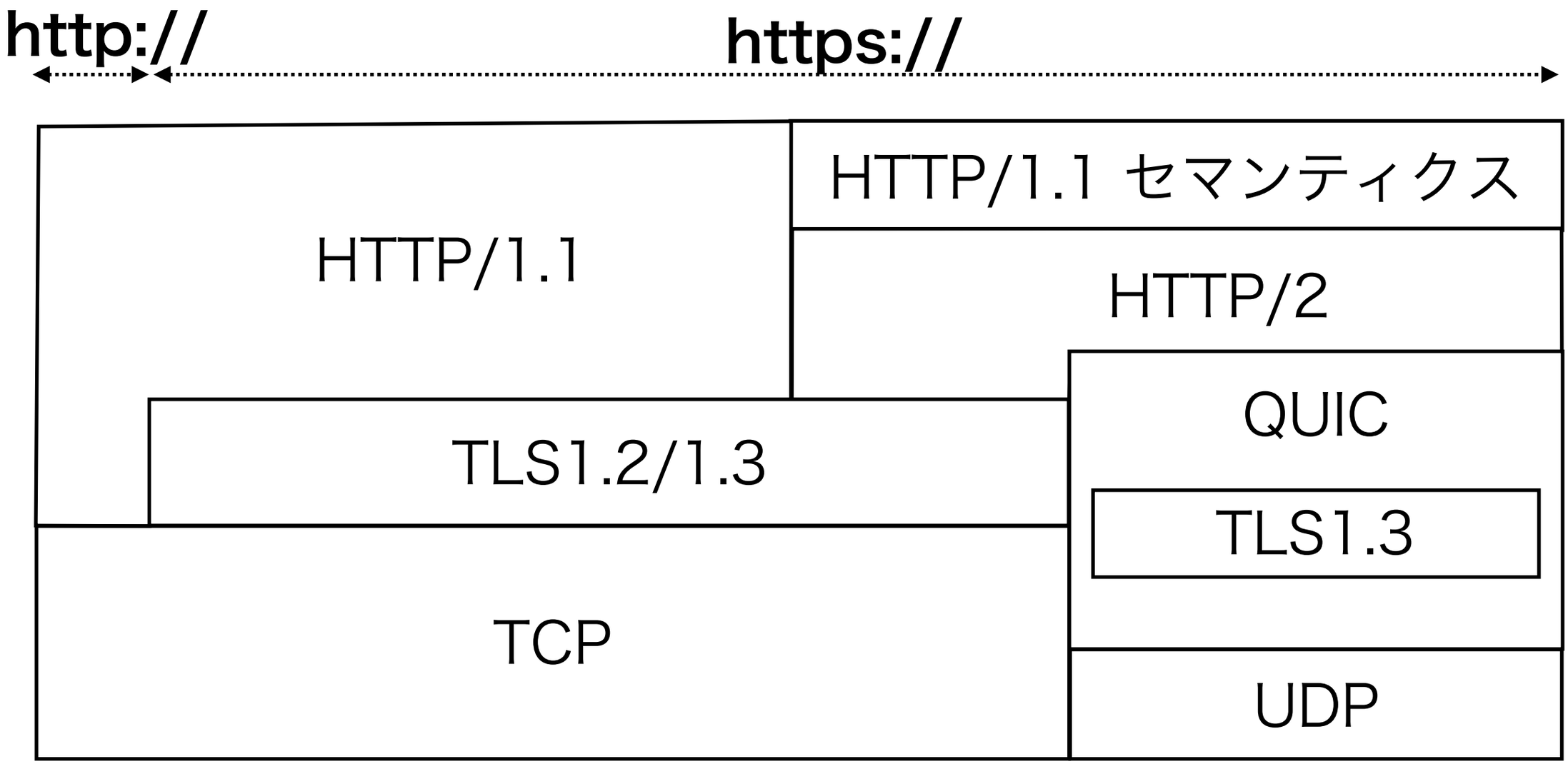
セキュリティ通信のHTTPSは, SSL3.0, TLS1.0を利用していた.

2017年のWebプロトコルスタック図



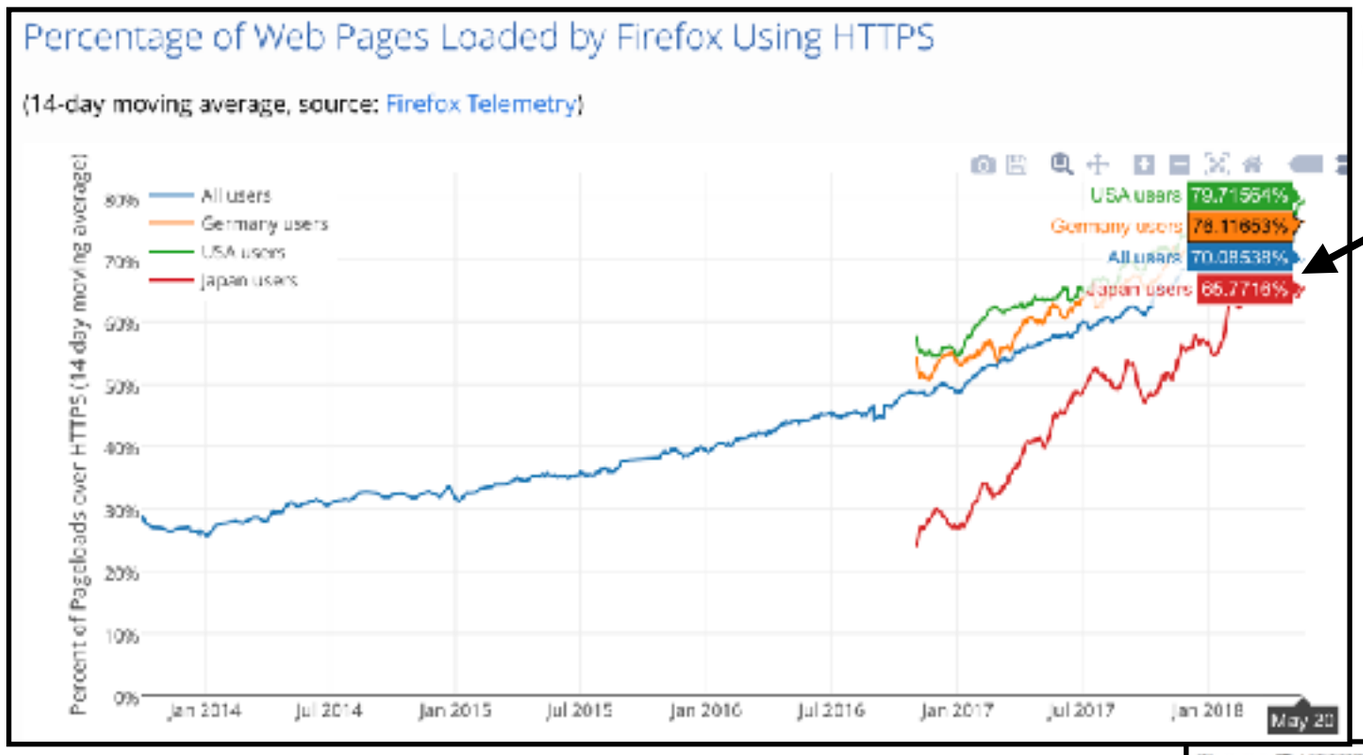
TLSの利用が1.0, 1.1, 1.2の3バージョンに広がる. 新しくTLS1.2上でHTTP/2の利用ができるようになった。

将来のWebプロトコルスタック図



TLS1.0,1.1が廃止され, TLS1.3が利用できる.UDP上で
TCP,TLS, HTTP/2の一部機能を実現するQUICが登場する.

HTTPSの導入状況(読み込みページ割合)



Firefox

Japan: 65.7% (2018/5/23)

2018/5/19 Chrome

日本は63%
10カ国中最低

https://letsencrypt.org/stats/#percent-pageloads

https://transparencyreport.google.com/https/overview

なぜ全部HTTPSにしないといけないの？

2013年6月： エドワード・スノーデン事件

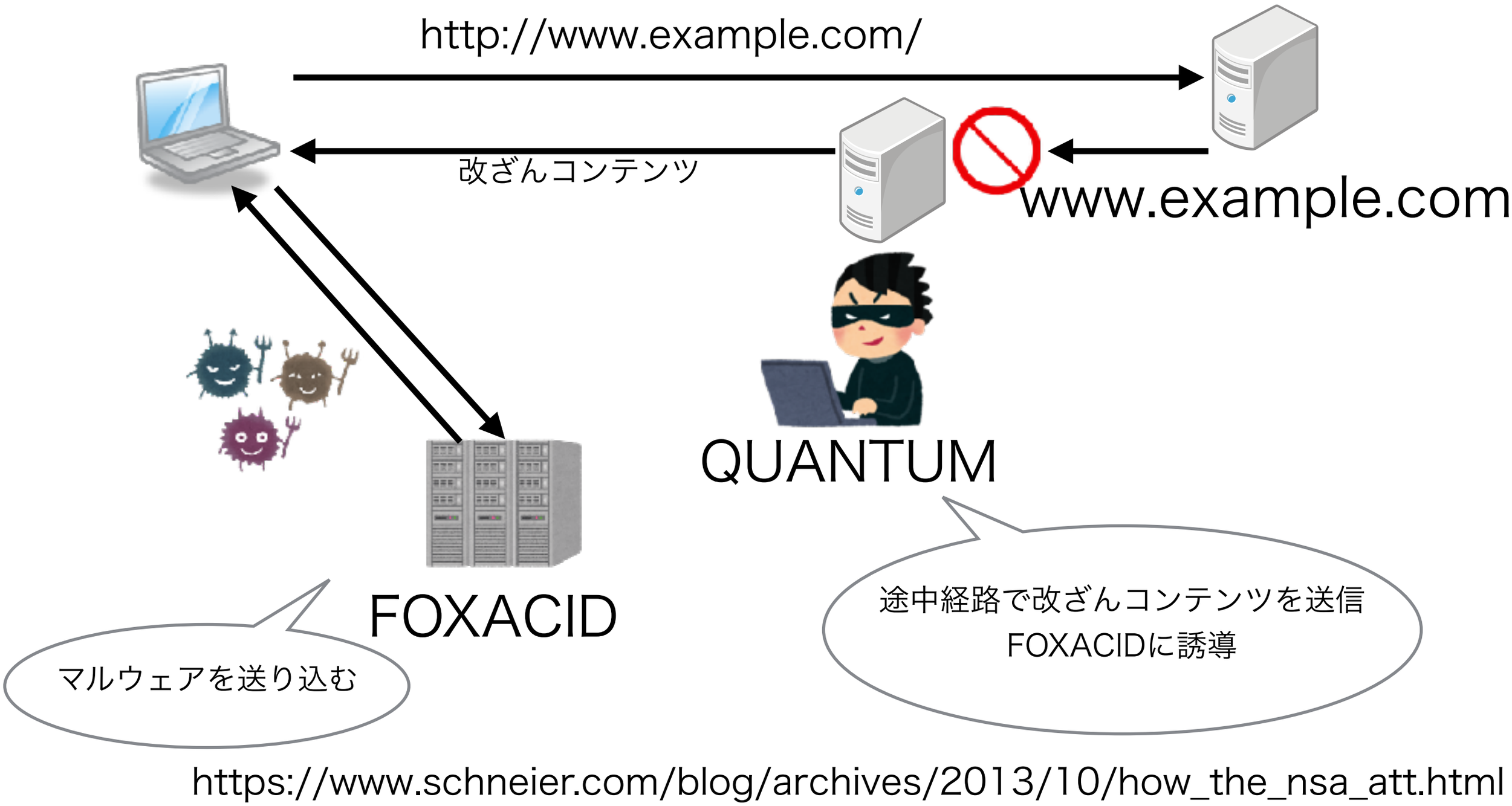
- NSA, CGHQによる国家レベルの広範囲な盗聴行為が明らかに

- 通信キャリアと協力し、DC内やインターネットバックボーンに盗聴・改ざんの仕掛けを配置

- 平文通信をMiTMで改ざんし未知のマルウェアを感染させる

- 暗号の標準化仕様にバックドアを仕掛けているとの話も

NSAによるサイバー攻撃の一例



https://www.schneier.com/blog/archives/2013/10/how_the_nsa_att.html

IAB(*)によるインターネットの信頼性に関する宣言(2014/11)

- 新しくプロトコルを設計する際には、**暗号化機能を必須**とすべき。

- ネットワーク運用者やサービス提供者に**暗号化通信の導入を推進**するよう強く求める。

- コンテンツフィルターやIDS等平文通信が必要な機能については将来的に代替技術の開発に取り組む。

(* Internet Architecture Board)
https://www.iab.org/2014/11/14/iab-statement-on-internet-confidentiality/

なぜ全部HTTPSにしないといけないの？

- インターネットの健全性を確保するため。ここが揺らぐと大手のWebサービスの信頼性がなくなる。

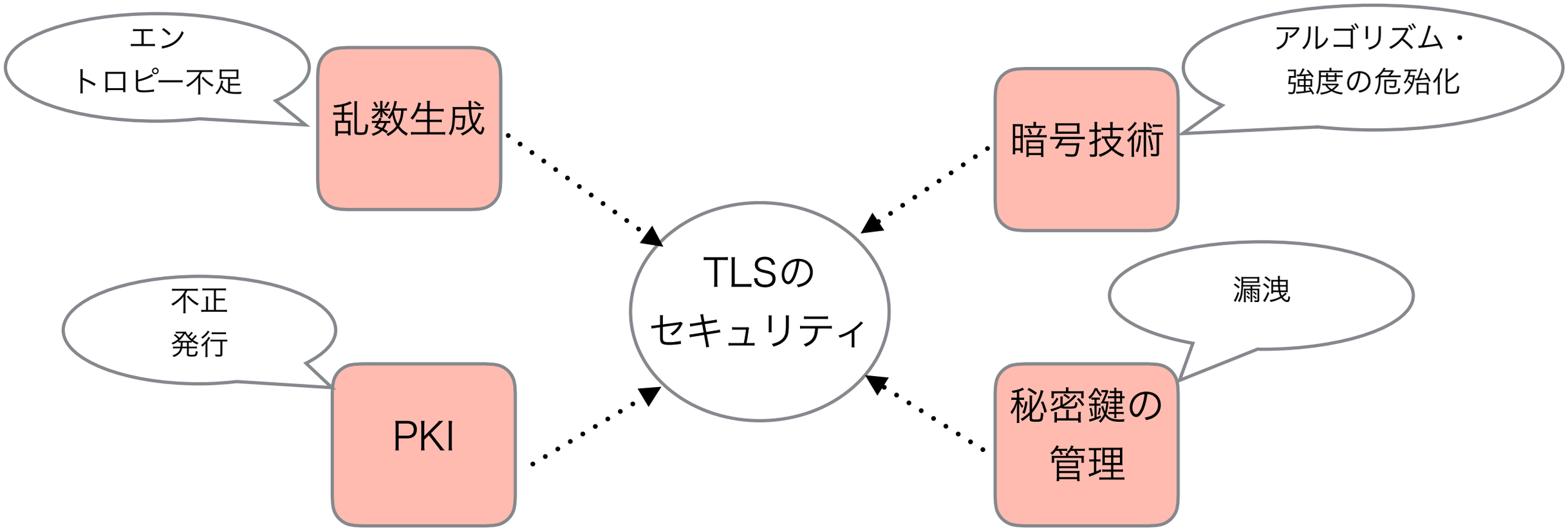
- 一部でも平文通信が残っているとそこを突かれる可能性がある。

- 自分は関係ないと思っていても知らないうちに仕込まれて利用される恐れも。

- 長期的な視点では http:// はフェーズアウトの方向です。

トラストアンカーをめぐる熱き戦い

TLSセキュリティの土台

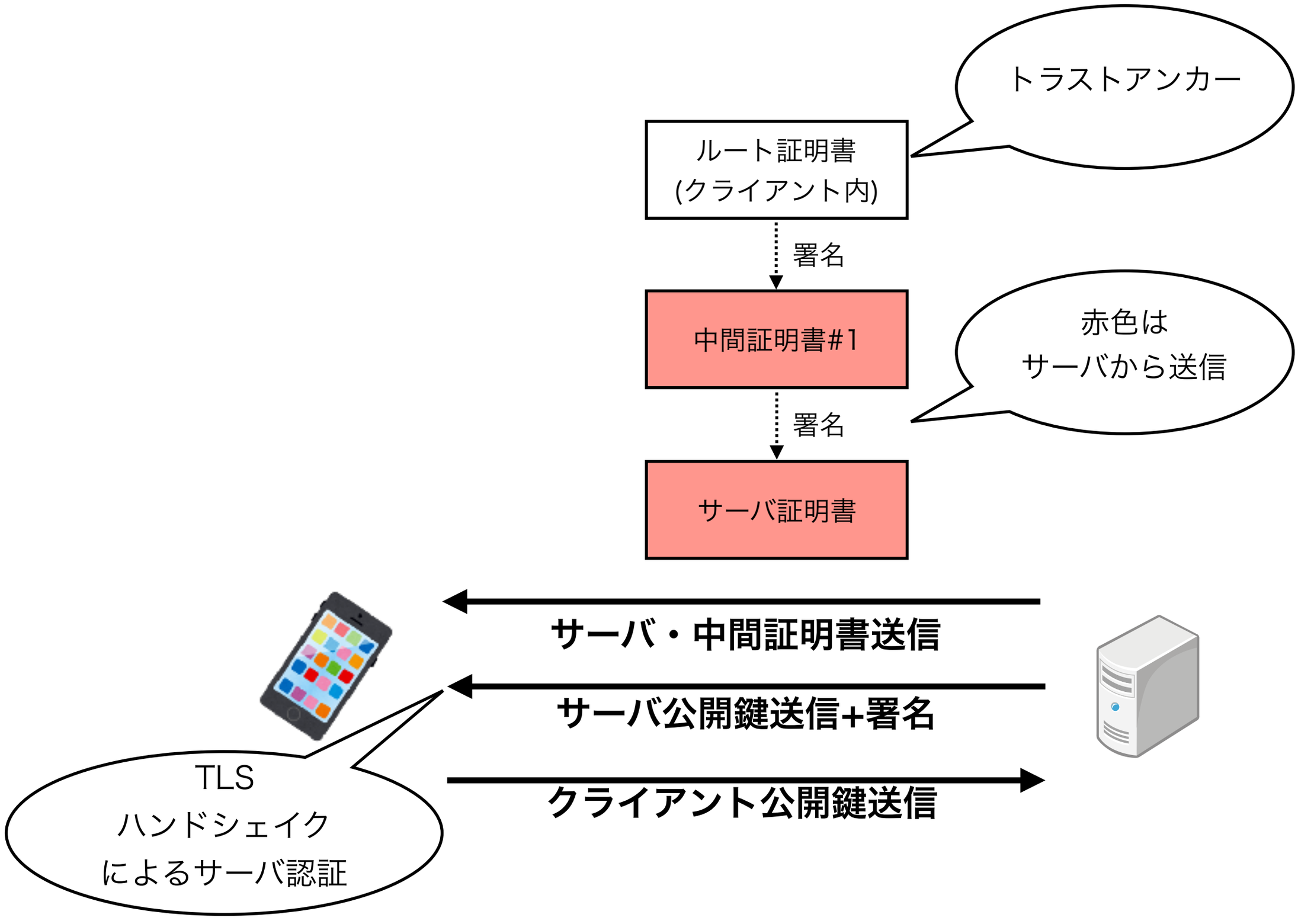


TLSは、この４つの外部要素の上でインターネットで安全な通信を提供する仕組みである。

逆に言えば、どれほど完璧なTLSプロトコルを作っても

この４つの外部要素が破られたら安全を確保できない。

信頼の要、トラストアンカー



TLSクライアントとOSで変わるrootCAの参照先

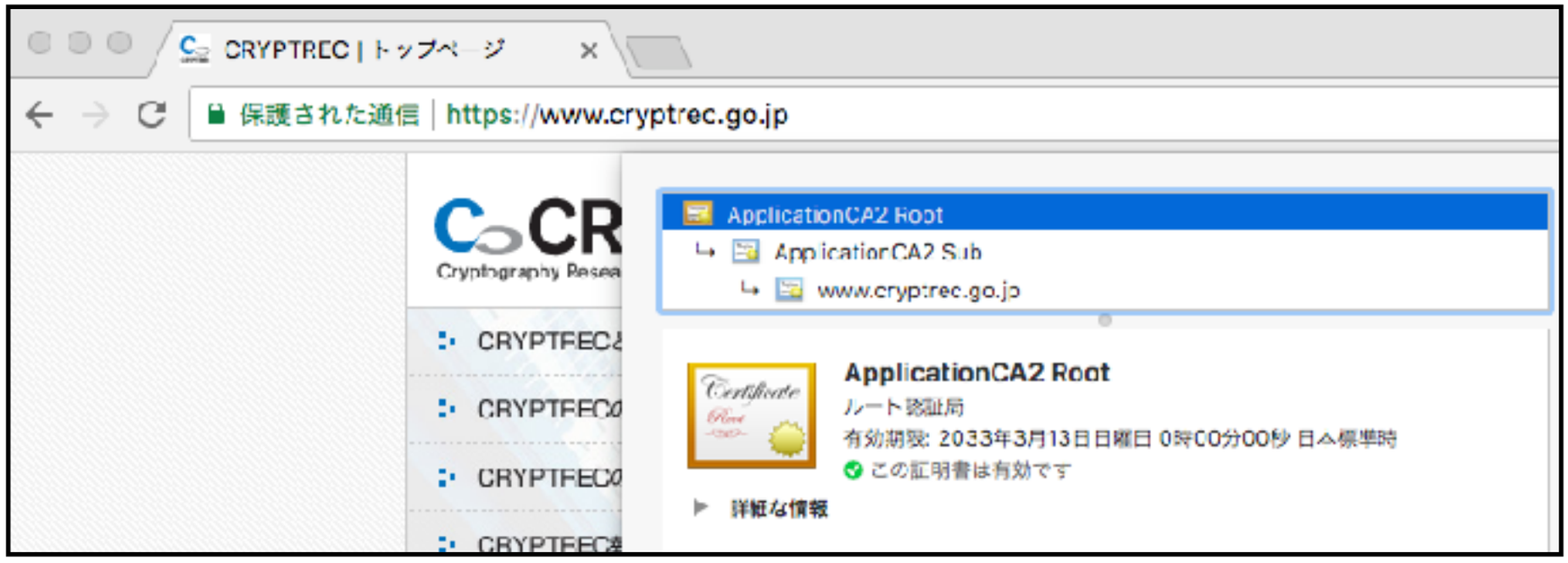
	Windows	MacOS	Linux	Android
IE/Edge	OSのrootCA	N/A	N/A	N/A
Safari	N/A	OSのrootCA	N/A	N/A
Chrome	OSのrootCA	OSのrootCA	Mozillaの rootCA(*)	Mozillaの rootCA(**)
Firefox	Mozillaの rootCA	Mozillaの rootCA	Mozillaの rootCA	Mozillaの rootCA
Node.js	Mozillaの rootCA	Mozillaの rootCA	Mozillaの rootCA	N/A

(* distribution提供のca-bundle pkgを参照する場合もあり), (** 変更、修正されている可能性あり)

MozillaのrootCAリスト https://hg.mozilla.org/mozilla-central/raw-file/tip/security/nss/lib/ckfw/builtins/certdata.txt

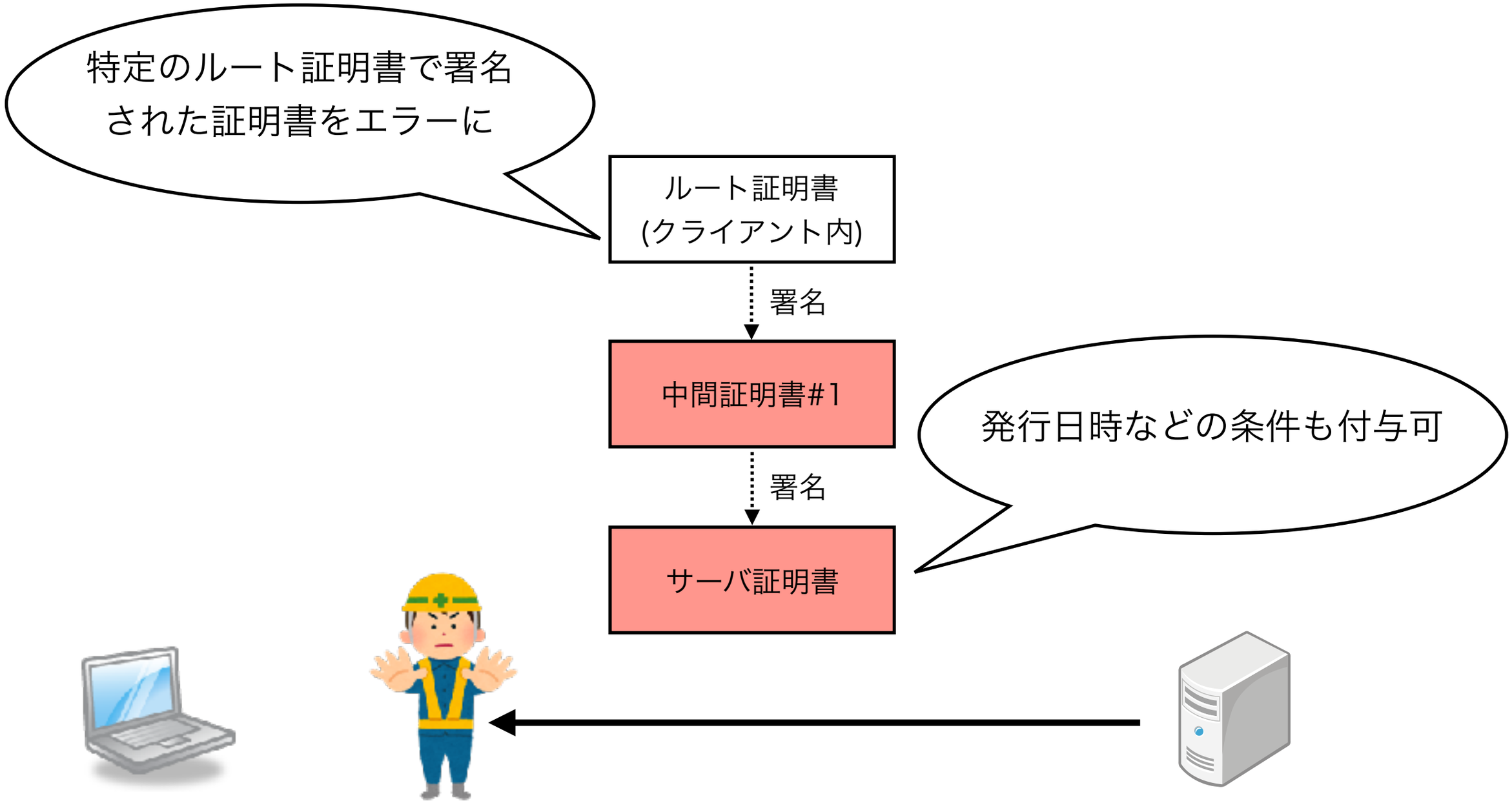
ブラウザによって異なる例

MacOS上のChrome



MacOS上のFirefox

ブラウザー vs 認証局



OSのroot証明書の管理とは別にクライアント独自の判断でフィルターできる。

HTTPSサーバの運用

技術負債

今後TLS1.0/1.1をどうしたらいいのか？

まずは
今の構成をしっかりチェックする。

最近のOSSはdefaultが安全側になるようにしています。

最新のLTSバージョンに随時アップデートしましょう。

ちゃんとした暗号方式の設定

鍵交換	RSA	Forward Secrecy			
		DHE	ECDHE NIST-P256	ECDHE X25519	
デジタル署名	RSA	DSS (DSA)	ECDSA	EdDSA	
対称暗号	DES/RC4	DES3	AES	ChaCha20	その他
暗号モード	CBC	AEAD			
		CCM	GCM	Poly1305	
メッセージ認証 PRF	MD5	SHA-1	SHA256	SHA384	

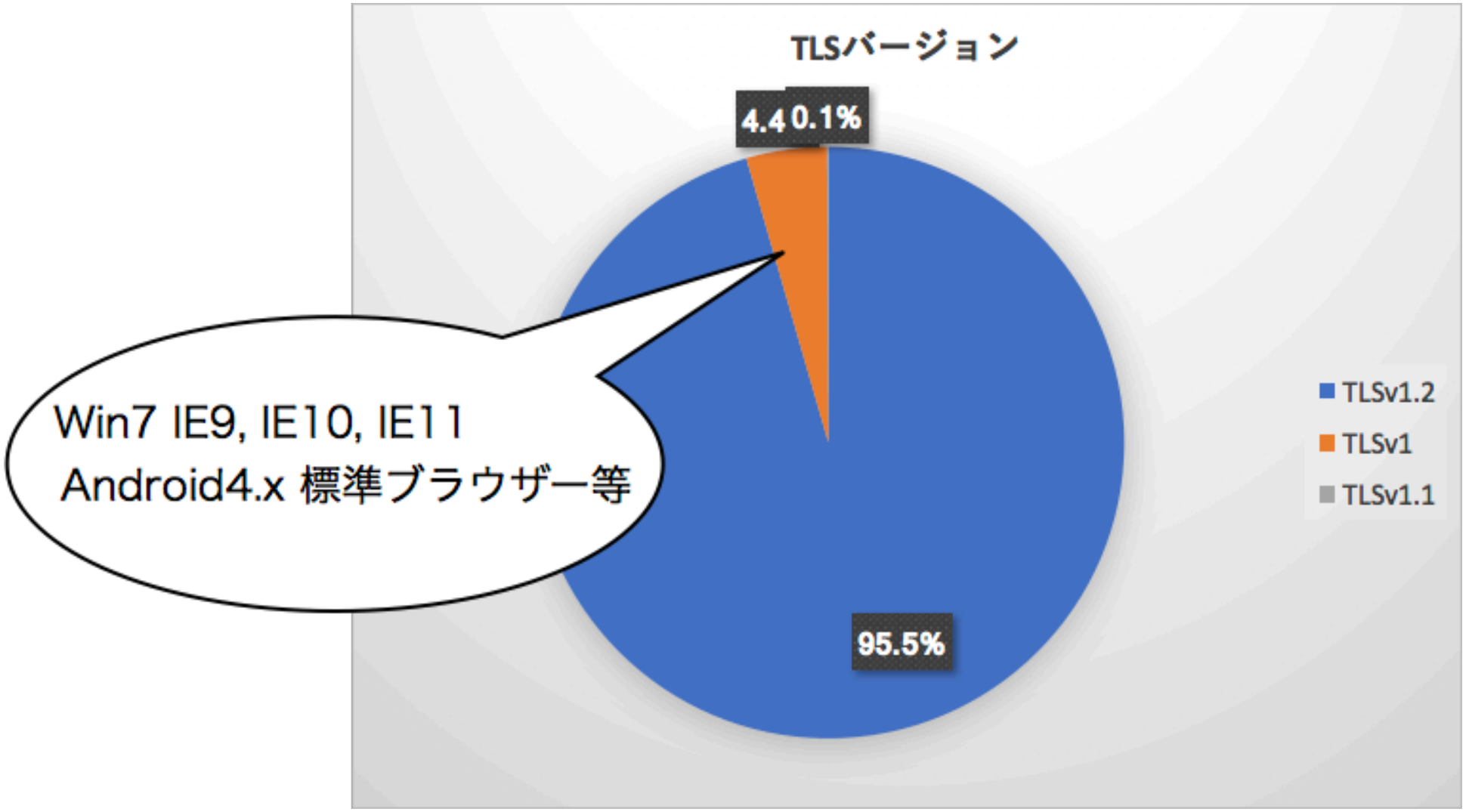
赤: 危険, 黄: 注意, 緑: 安全, 青: これから

(注意は、暗号学的注意と将来的に普及が見込まれない注意も含まれます)

注意: CRYPTREC/IPAの「SSL/TLS暗号設定ガイドライン 第2.0版」と評価が異なります。

技術負債の返済に備える

オワコンプロトコル(TLS1.0)



出典 https://www.slideshare.net/techblogyahoo/web-html5j-html5jb

TLS 1.0は、なぜオワコン？

- 1999年に仕様策定。もともとはSSLの標準化を目指したもの
- AES選定(2000年)前だったので当初はRC4とDES3のみサポート。RC4は既に危殆化、DES3はSWEET32攻撃を受けることが知られている
- ブロック暗号を使う際の初期ベクトルでBEAST攻撃を受けることが知られている(クライアント側で要対策)
- CBCモードの実装は過去数多くの脆弱性を生み出したのでAEAD(認証付暗号)への移行が主流に(*)
- MAC/PRF(鍵生成)ではMD5とSHA-1を組み合わせたもの。これがある限り危殆化されたMD5のコードをなくすことはできない(*)

(* TLS 1.1と共通項目)

TLSプロトコルの比較

	策定	ブロック暗号の 初期ベクトル	CBC モード	AEAD	MAC/PRF
TLS1.0	1999年	なし	有	なし	SHA1, MD5
TLS1.1	2006年	有	有	なし	SHA1, MD5
TLS1.2	2008年	有	有	有	SHA256以上
TLS1.3	2018年	廃止	廃止	必須	HKDF, SHA256以上

PCI DSS3.1
禁止 2018/6〜

推奨

注意: CRYPTREC/IPAの「SSL/TLS暗号設定ガイドライン 第2.0版」と評価が異なります。

プロトコルの廃止は難しい

- もともとPCI DSS 3.1は2016年6月30日にTLS1.0を廃止予定だった。それを２年延期。

- 大手(SalesForce, IBM, Oracle)のサービスは既に対応済。だが移行作業はどこも一度は失敗してロールバックしている。

- Android4.xの標準ブラウザーが鬼門。API利用などはクライアント側の改修とか必要。

- 事前にユーザへの認知が難しい。廃止後はエラー画面に。

- SSL3.0はPOODLE脆弱性の公表でガラケー対応を含め廃止できた。TLS1.0は同じようになるのか、ならないのか。

ではどうすれば？

- まずはモニター
- TLSバージョンをアクセスログに出す
- 割合やUA、src ip等の傾向を把握
- インシデント発生時に影響度を把握できるよう準備が大事。

将来に備える

TLS1.3

TLS1.3が求められる背景

1. 常時TLS時代を迎えるにあたって、しっかりしたプロトコルが必要

2. TLS1.2の限界

    - 様々な技術負債の蓄積

長期に使えるより安全で高性能なTLSプロトコルを作る

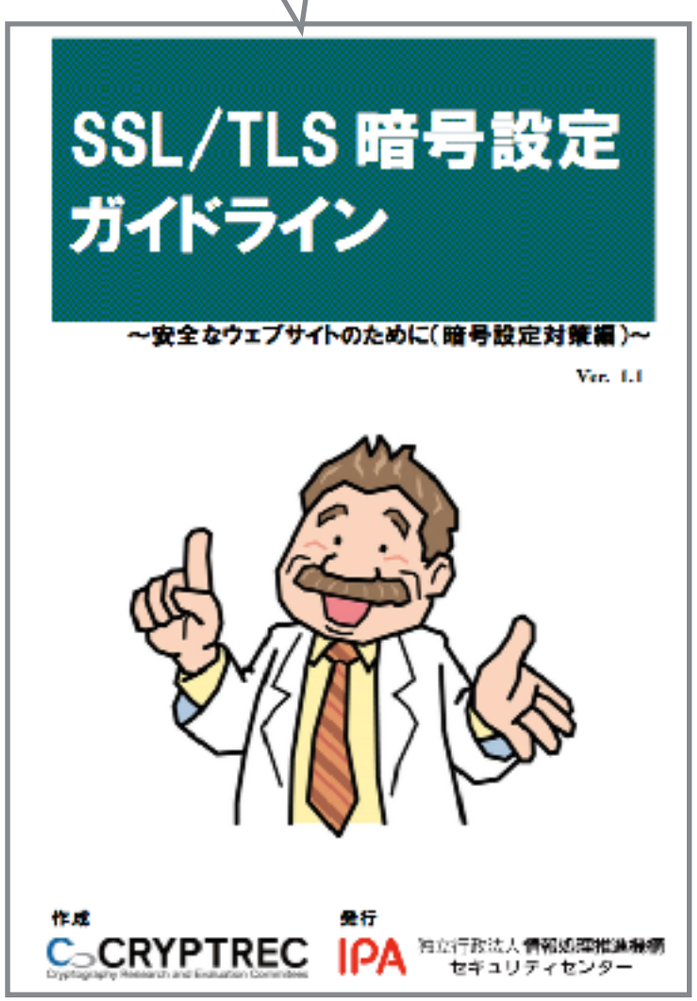
TLS1.2の限界

- 現在のTLS1.2で定義されている機能の一部は、既に利用すると危険である。

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

- しかし一時的な対応で根本的・抜本的な対策になっていないものも多い。

本来このようなガイドラインがなくて済むのが望ましい



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

   これまで攻撃対象となった機能を極力排除し将来的な攻撃に備える

3. 性能向上

   初期接続の短縮による性能向上

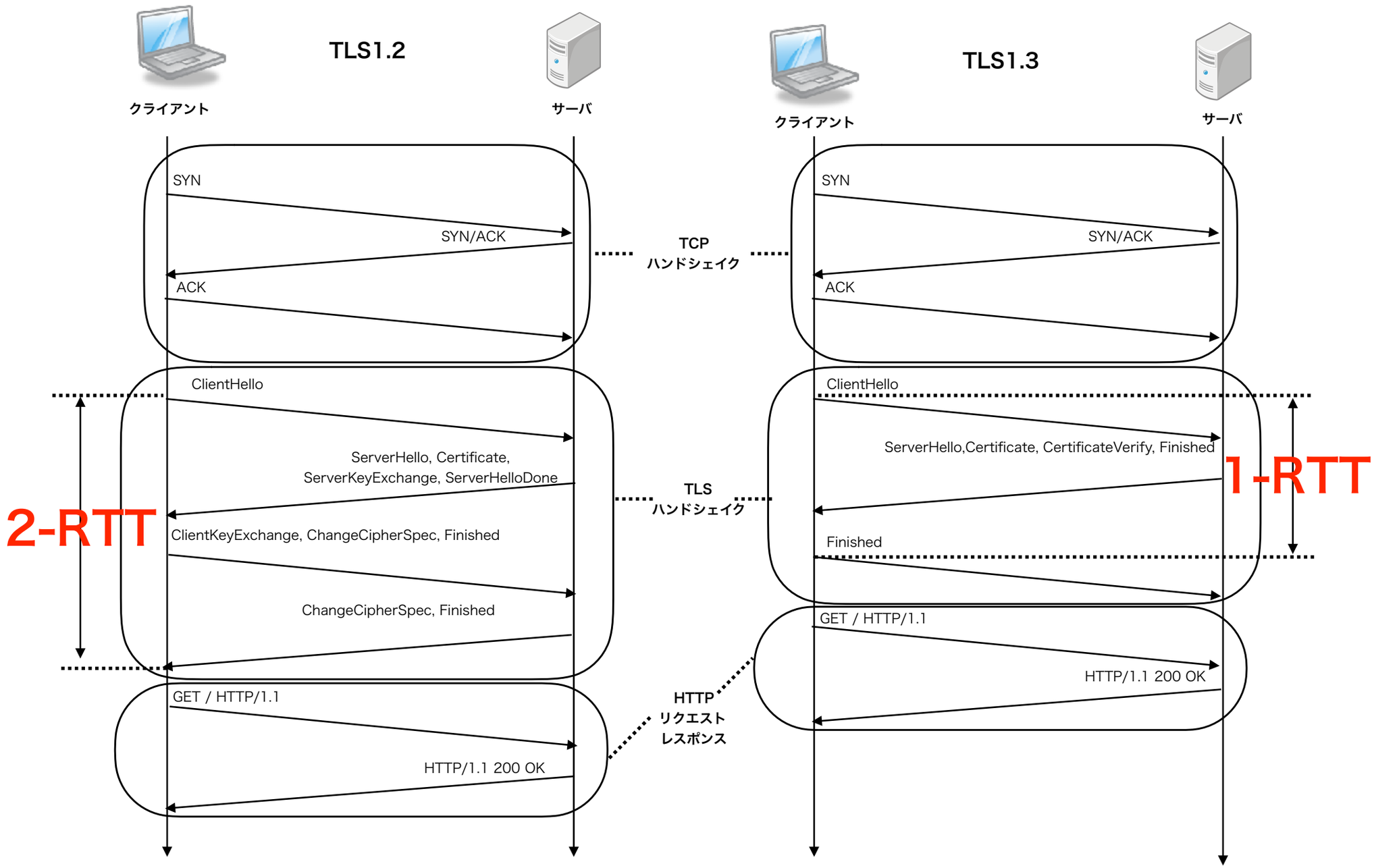
中身的にはTLS2.0レベルの大変更

2018年、TLS1.3仕様化完了

RFC発行待ち

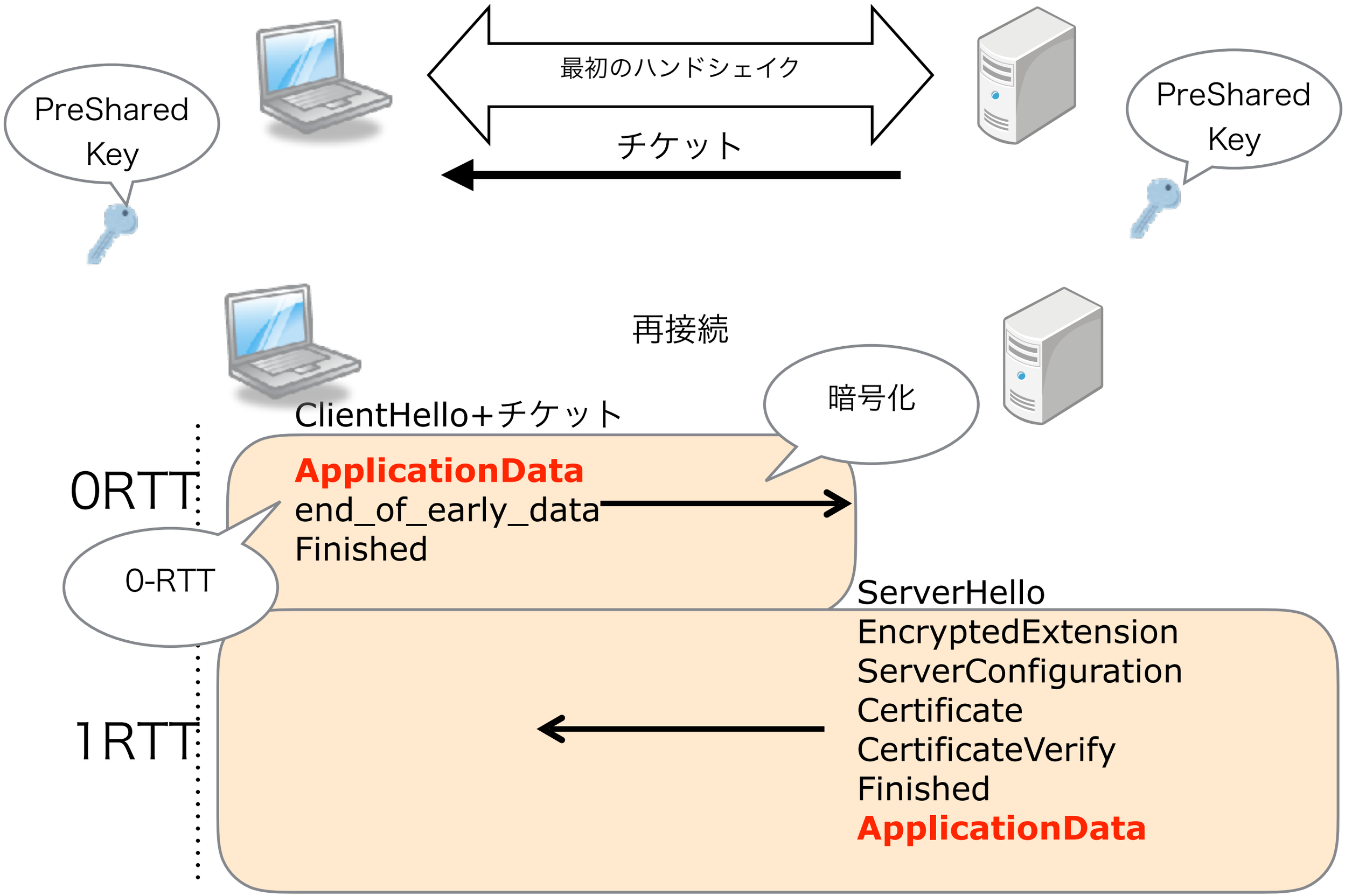
皆さん、知らないうちにTLS1.3を使っています。

TLS1.3とTLS1.2の違い

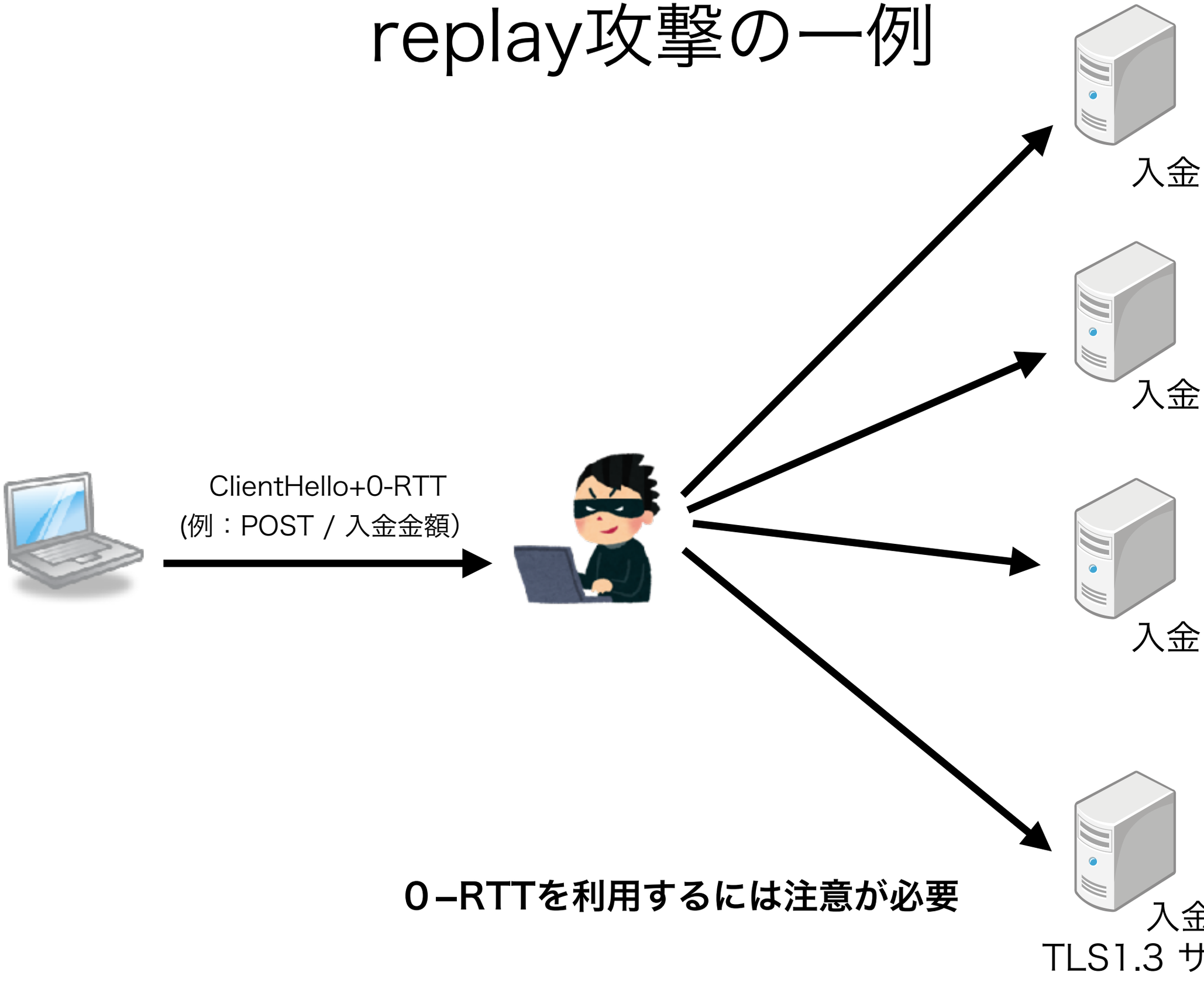


早いハンドシェイク

TLS1.3のハンドシェイク 0-RTT



TLS 1.3 0-RTTの脅威
replay攻撃の一例



0–RTTを利用するには注意が必要

アプリ開発者やサーバ管理者が何を気をつければいいのか？

- 運用的な課題
  - TLS1.3が有効になることでミドルボックス、FW、IDS/IPSなど中間通信装置の障害や機能停止などの影響はないか？
  - TLS1.3で廃止になる機能(特にRSA鍵交換)を前提としたシステムの機能を使っていないか？
- 技術的な課題
  - 0-RTTを本当に安全に利用することができるのか？
  - 0-RTTを使わなかった場合に問題は発生しないのか？

TLS1.3 移行のタイミングは？

- TLS1.3の仕様化によってTLS1.2が廃止されることはない。TLS1.2を継続して使い続けるのも一つの選択肢。TLS1.0/1.1はタイミングを見て廃止されるかもしれない。

- OpenSSL-1.1.1(近日リリース予定)ではTLS1.3がdefaultで有効

- 現在のLTS OpenSSL-1.0.2は、2019年12月末でEoL

- 今年末から来年にかけてバージョンアップは必須。自然にTLS1.3になってしまうことも。TLS1.2だけの暗号設定だとエラーになる可能性もあります。

- NISTは米国政府系のHTTPSサーバは2020/01まで(0-RTTを使わない)TLS1.3対応を求める予定。