

# プライバシ保護のための インターネットアーキテクチャの進化 セキュリティエンジニアリング特論 第10回

栗原 淳

兵庫県立大学大学院

2025-12-18

# はじめに

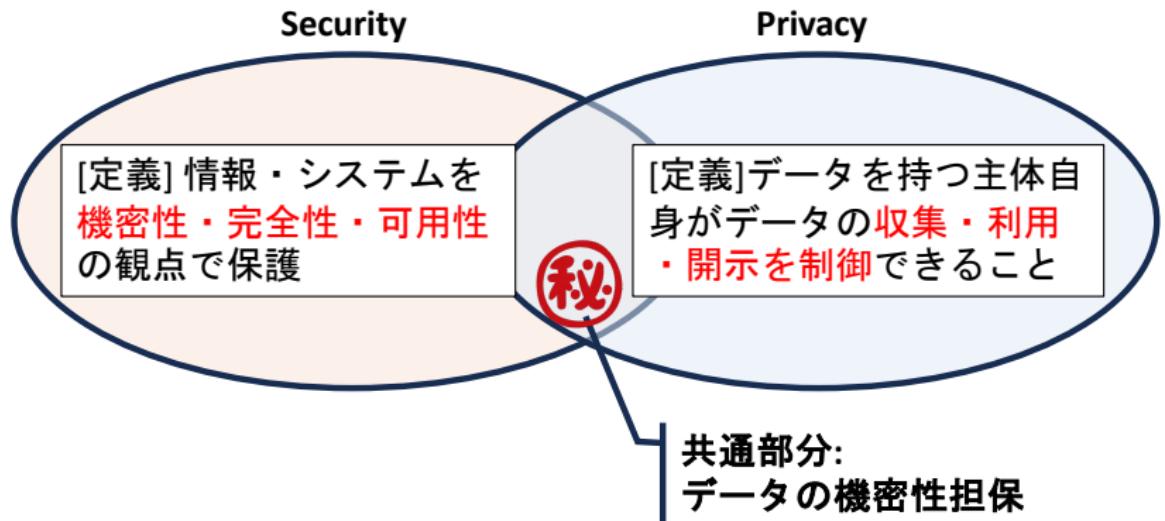
# この資料の位置付け

2025年7月現在での、インターネットにおけるプライバシの社会的・技術的課題と、その解決のためのアプローチについて解説する。

想定する聴衆: セキュリティ・ネットワークについて、基礎的な知識を持つ技術者や研究者、学生

# 社会と技術の背景

# セキュリティとプライバシ～違いと重なり～



目的： 盗聴・侵入・改ざんを防止  
代表技術： 従来の TLS, IPSec, etc.

目的： 匿名性・リンク不可能性等  
代表技術： ODoH, ECH, etc.

異なる概念ではあるものの、共通部分も多く、相互に補完  
⇒ 両者を同時に満たすことが重要

# 社会背景: プライバシ意識やプライバシ保護の歴史

インターネットの普及とともに、個人情報の収集・利用が増加

- ソーシャルメディア、オンラインサービスの普及
- データマイニング技術の進化
- ユーザトラッキング・広告ターゲティング技術の発展



プライバシ侵害の事例が急速に増加

- 個人情報の漏洩事件 (例: FB-Cambridge Analytica @2018)
- 国家による監視プログラム (例: 米国スノーデン事件 @2013)
- ソーシャルメディアでの個人情報の無断利用



これらの背景を受け、ユーザ意識・法律・技術の3つの側面で、プライバシ保護が近年活発化。

### ユーザのプライバシ意識の高まり

- プライバシ保護技術への関心増加
- プライバシ侵害に対する不安感

### 法律・規制の整備

- EUにおけるGDPR<sup>1</sup>の施行@2018年
- GDPRに続き、各国でのプライバシ保護法<sup>2</sup>の制定

### プライバシ保護技術に関する国際的な議論の活発化

- IETFやW3Cなどの標準化活動
- 国際会議の増加(例: Privacy Enhancing Technologies Symposium; PETs)

<sup>1</sup>General Data Protection Regulation

<sup>2</sup>日本: 改正個人情報保護法@2022年、ほかカリフォルニア・南米・オセアニア等

# 社会背景: インターネットのプライバシ危機の現在地

## すべての始まり スノーデン暴露 (2013)

NSA PRISM / Upstream での 国家レベル盗聴 が可視化

⇒ HTTPS 普及率が 5 年で 30% → 80% に急伸 (**Encryption by Default**)

近年、ナショナリズムの高まりと共に、**盗聴**に加え、**分断と監視**が急速に進行。

### デジタル主権 (EU)

- AWS European Sovereign Cloud (2025/6 開設)
- EU "クラウド調達規則"→域内クラウドを優先
- **課題**: 国境ごとにクラウド壁／運用コスト増

### データローカライゼーション (インド)

- インド RBI(中央銀行) 指令: 決済データを国内 DC に保管
- 国外転送は原則禁止 (2024/9 発効)
- **課題**: 国際サービスは暗号化スプリット (読めないデータしか出さない) 必須

### 政府監視強化 (米・英)

- 米 FISA § 702 : 2024 再認可
- 英 Online Safety Act : E2EE でも検査命令可
- **課題**: 通信内容だけでなくメタデータも対象

### 能動的サイバー防御に関する立法 (日本)

- 2025/5 サイバー対処能力強化法成立: 政府が IP メタデータ常時監視
- 国外サーバ無力化権限・違法利用は懲役刑
- **懸念**: 憲法 21 条「通信の秘密」と衝突

これら分断・監視の課題を払拭し、

通信メタデータをも守る**基盤プロトコルの「Privacy by Design」**が注目

# プライバシ保護 ≠ 脱法行為

[誤解のなきように]  
注目されているのは  
「脱法行為をするための技術」ではなく、  
「プライバシを保護するための技術」です！  
(私は、後者の研究開発をしています。)

# IETFの宣言と、インターネット基盤プロトコル更新の大きなうねり

エドワード・スノーデン氏による NSA の監視プログラム PRISM の暴露(2013年)を契機とし、IETFは以下のRFCを発行。

## RFC 7258: Pervasive Monitoring Is an Attack [18]

- 2014年5月、IETFが発表したRFC
- インターネット上の監視は攻撃であると明言
- プライバシ保護のための技術開発を促進



プライバシを保護するインターネット基盤プロトコルの設計・展開が活発化

- DNS: DoH [19], DoT [20], DoQ [21], ODoH [23]
- TLS: ECH [32], Post-Quantum Hybrid Key Exchange [35]
- etc.

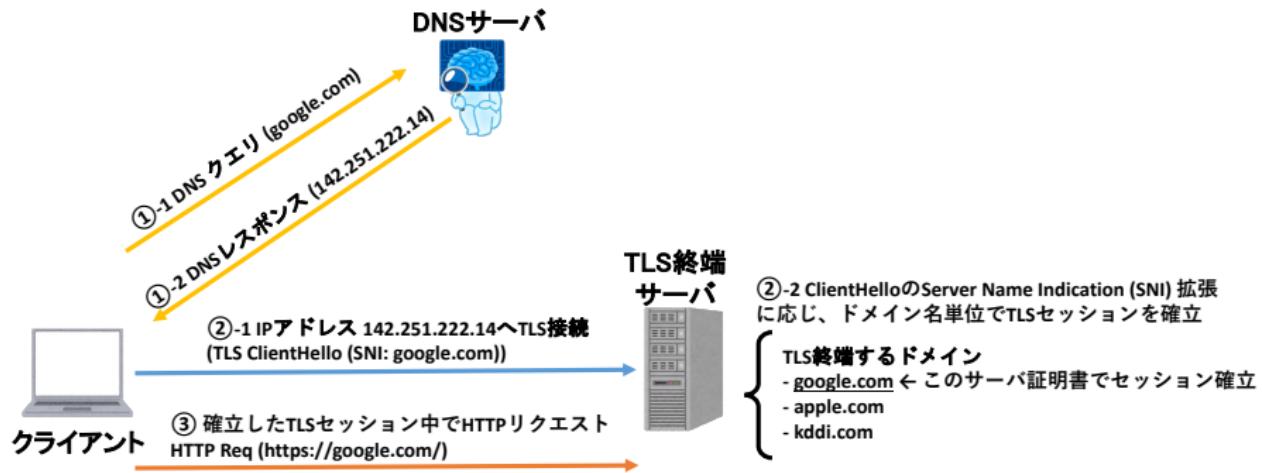
# 今日の「インターネットアクセス」の典型パターンと そのプライバシの課題

「インターネットアクセス」 ≈ 「HTTP(S) を用いた Web アクセス」



全てが Web に飲み込まれた現在における、典型的なアクセスパターン

① DNS 名前解決 → ② TLS ハンドシェイク → ③ HTTP リクエスト



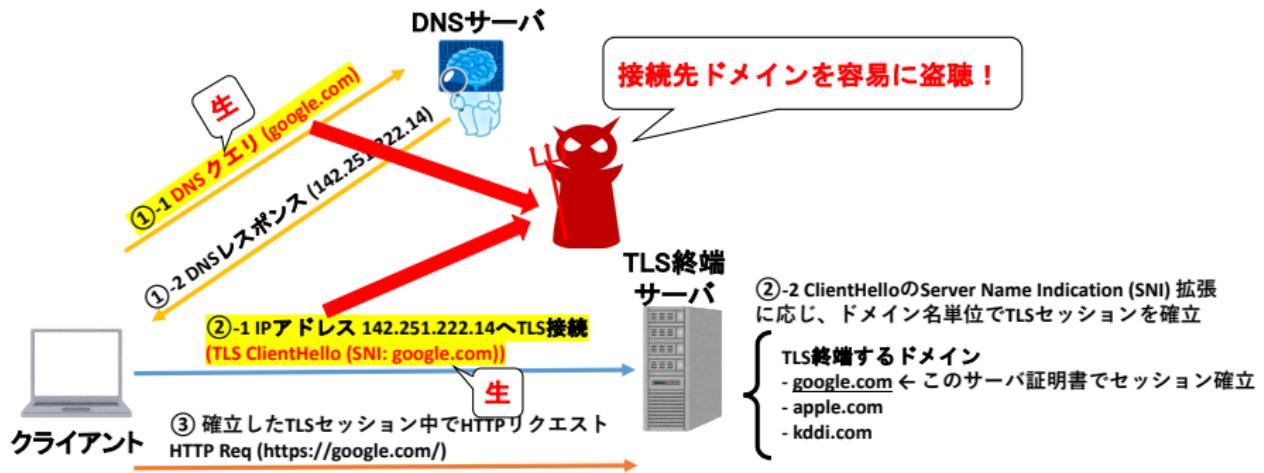
# 今日のインターネットアクセスにおけるプライバシ課題

## DNS 名前解決のプライバシ課題

- DNS メッセージは平文で送信
- 攻撃者は DNS メッセージを傍受し、ユーザの行動を追跡可能

## TLS ハンドシェイクのプライバシ課題

- TLS ClientHello 中の接続先ドメイン名 (SNI) は平文で送信
- 攻撃者はハンドシェイク情報をを利用してユーザを特定可能



※平文 HTTP リクエストは論外のため、今回は省略。

よって、今回は以下を取り上げる。

- (Topic 1) DNS 名前解決のプライバシ保護
  - 第一步 ⇒ **暗号化 DNS** (DoH, DoT, DoQ)
  - 更なるプライバシ ⇒ **匿名 DNS** (ODoH)
- (Topic 2) TLS ハンドシェイクのプライバシ保護  
⇒ **暗号化ハンドシェイク** (ECH; Encrypted ClientHello)

展開が進む**インターネットアクセスを支える基盤プロトコルの、暗号技術による進化**を紹介

(前スライド) DNS・TLS ハンドシェイクのプライバシ強化

インターネットアクセスを支える基盤プロトコルの、暗号技術による進化

### ここでふと疑問

TLS 等では元々暗号技術が用いられている。そういった、今日までのインターネットを支えてきた「暗号技術そのもの」は、今どうなのか？

**結論: 超危険。プロトコルを含めて早急な更新が必要。**

検討・展開が併せて進行中。

# インターネットにおける更なるプライバシの課題

## 差し迫った量子コンピュータの脅威: Harvest Now, Decrypt Later (HNDL) [29]

- 急速な量子コンピュータの発展により、**現在の暗号技術が近い未来に脅かされる可能性**
- TLS ハンドシェイクの鍵交換方式 (RSA, ECDH など) も、量子コンピュータに対して脆弱
- 量子コンピュータの発展を待って、保存した暗号化データを後から復号する攻撃 (**Harvest Now, Decrypt Later**) が現実の脅威に



## HNDL 自体は新しい概念ではない

- 米国政府機関が実際に SSL/TLS の暗号化データを収集・保存していたこと  
も、2013 年のスノーデン事件で暴露
- これを契機に「たとえ一部が解読されても、他の暗号化データは芋づる式で  
破られないようにする」考え方・実装<sup>3</sup> の普及  
⇒ しかし、それだけでは、量子コンピュータの発展に対して不十分

今現在、ネットワーク上を流れるデータを収集・保存している攻撃者がいた場合、それはいずれ解読される可能性が高い。

⇒ 量子コンピュータに対して耐性を持つ暗号技術<sup>4</sup> を用いた、基盤プロトコルの更新が進行中

(Topic 3) TLS ハンドシェイクのさらなる進化

⇒ 量子耐性を持つ鍵交換方式 (Post-Quantum Hybrid Key Exchange)

---

<sup>3</sup>完全前方秘匿性; Perfect Forward Secrecy (PFS)

<sup>4</sup>耐量子計算機暗号; 耐量子暗号; Post-Quantum Cryptography

# この後の発表の流れ

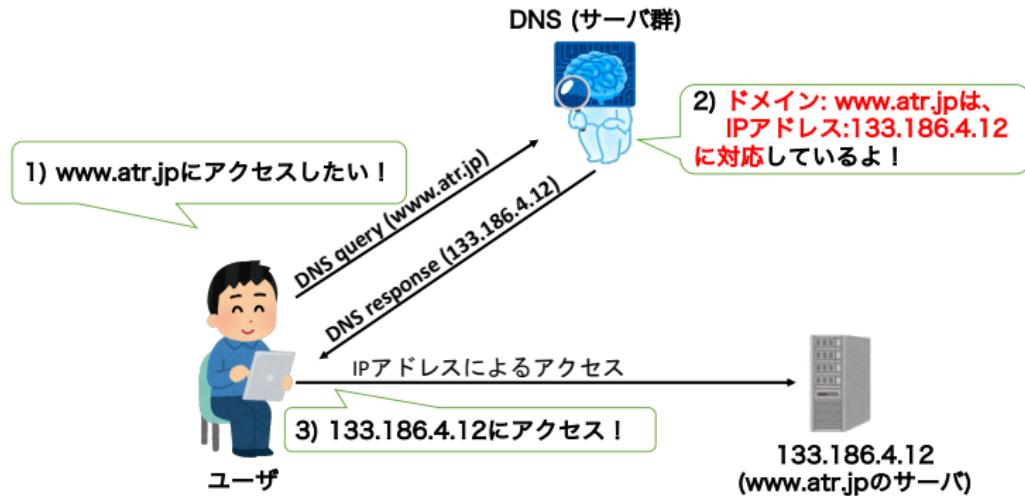
というわけで、インターネット基盤プロトコルとアーキテクチャの進化を、以下の 3 つの観点で紹介

- ① DNS 名前解決のプライバシ保護～暗号化 DNS と匿名 DNS～
- ② TLS ハンドシェイクのプライバシ保護～暗号化 ClientHello～
- ③ TLS ハンドシェイクのさらなる進化～量子耐性を持つ鍵交換方式～

# DNS名前解決のプライバシ保護

# Domain Name System (DNS)

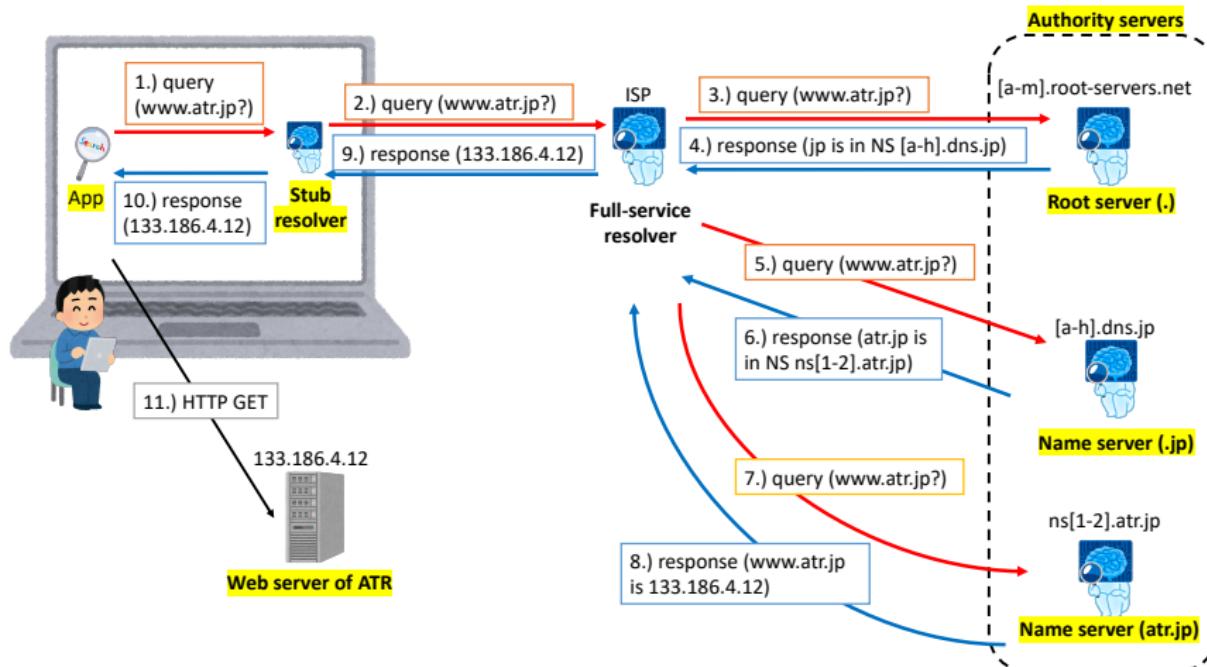
DNS: ドメイン名と IP アドレスの対応付けの管理システム



意識しない透過的な機構だが、現在のインターネットを支える重要なバックボーン。もし DNS が止まつたら URL による web アクセスもメール送受信も不可能。

# DNSの「名前解決」の仕組み

トップレベルドメイン (ex.: '.jp') からの「再帰的」な反復検索で実現



DNS レコード (IP アドレス・ドメインの関係) は、所望のドメイン名を直接管理するサーバに辿り着くまで繰り返す。

## 用語の解説:

- **リゾルバ (Resolver):**  
ドメイン・IP アドレスの対応を検索 (=解決)。
- **スタブリゾルバ (Stub resolver):**  
再帰検索を上流に要求。端末内や家庭内ルータ等。
- **フルサービスリゾルバ (Full-service resolver)<sup>5</sup>:**  
スタブリゾルバの再起検索要求に応じて反復検索。
- **権威サーバ (Authoritative server)<sup>6</sup>:**  
自管理ドメイン名空間 (ex.: '\*.atr.jp') 配下のドメイン状況を管理。
- **DNS レコード:**  
ドメイン名と IP アドレスの対応<sup>7</sup>、ドメインを管理する権威サーバの情報<sup>8</sup>など、当該ドメインに関する種々のデータ。権威サーバが発行。

---

<sup>5</sup>DNS キャッシュサーバ

<sup>6</sup>DNS コンテンツサーバ、単にネームサーバと呼ばれたりも

<sup>7</sup>A/AAAA レコード

<sup>8</sup>NS レコード

# 古の DNS の「セキュリティ」の問題

次のいずれも担保され「ない」

- **DNS パケットの機密性**  
⇒ 第三者に盗聴され放題。
- **DNS パケット・レコードの完全性**  
⇒ 通信路で改竄し放題。
- **DNS レコードの真正性**  
⇒ 各リゾルバは偽レコードを返し放題。<sup>9</sup>



DNS そのものの設計が古い(1983年)ので、致し方ない部分はあるものの、DNS はセキュリティに関して非常に脆弱。

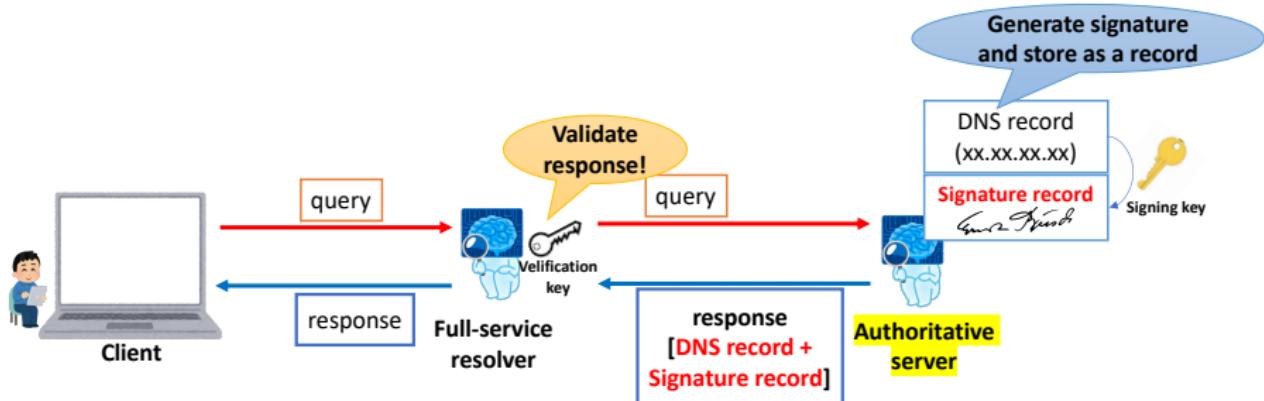
---

<sup>9</sup> ex.) 実際、米国 ISP はレコードを書き換え放題して(DNS 汚染)

DNSの「セキュリティ」改善に向け、DNSSECは古くから展開。

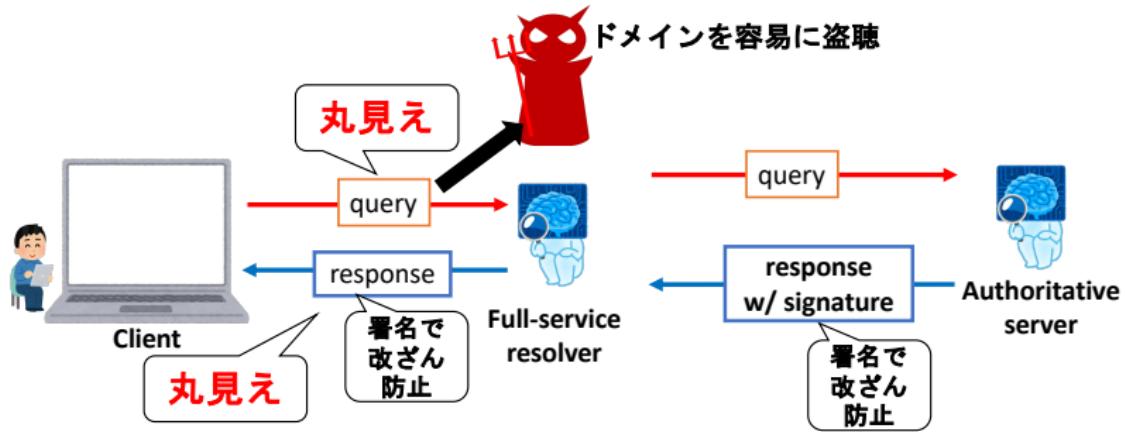
## DNS Security Extensions (DNSSEC) [5, 4, 3]

- オリジナルは 1999 年の RFC2535。20 年以上の歴史。
- 各レコードにデジタル署名を付与。
- 取得したレコードの署名を検証することで、正当な権威サーバで生成されたこと(真正性)、改竄されていないこと(完全性)を保証。



※ Validation can be done at the stub resolver as well.

DNSSEC でもメッセージは平文のまま。第三者に盗聴され放題。  
⇒ クエリの「機密性」は何も保護されないまま。



ネットワーク中継ノードが DNS クエリを容易に傍受可能

# DNS プライバシの構造的な問題

「機密性」の欠如を含め、DNS の機構にはプライバシの問題が存在。

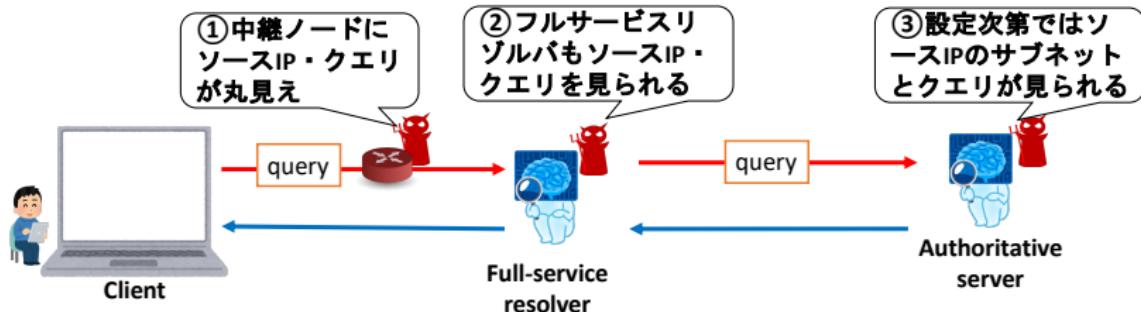
## ① 中継ノードが DNS クエリを傍受可能な問題

## ② フルサービスリゾルバに対するクエリプライバシの課題

反復解決の必要上、必ずユーザの DNS クエリの内容を知りうる。

## ③ 権威サーバへのクエリプライバシの課題

EDNS0 [11] の仕様では、エンドユーザの IP アドレス等の情報が含まれた DNS クエリが、権威サーバまで到達しうる。<sup>10</sup>



<sup>10</sup> EDNS Client Subnet (ECS; RFC7871 [10])。ECS 対応キャッシュサーバは、権威サーバへクエリ発信元 IP アドレスブロックを通知。

すなわち、DNS を通じ、エンドユーザあるいは組織の「行動履歴・行動パターン・業務上の秘密等」が第三者へ漏洩。

### DNS の構造的なプライバシの問題により発生した、社会的な課題の事例

- 米国では、ISP による DNS クエリの記録・収益化の歴史 [38]
- 各国において DNS クエリブロッキングや、政府によるクエリ監視<sup>11</sup>  
(代表例: 中国の Great Firewall, ロシアの国営 ISP による監視)

DNS プライバシは大きな社会的課題として捉えられるようだ。

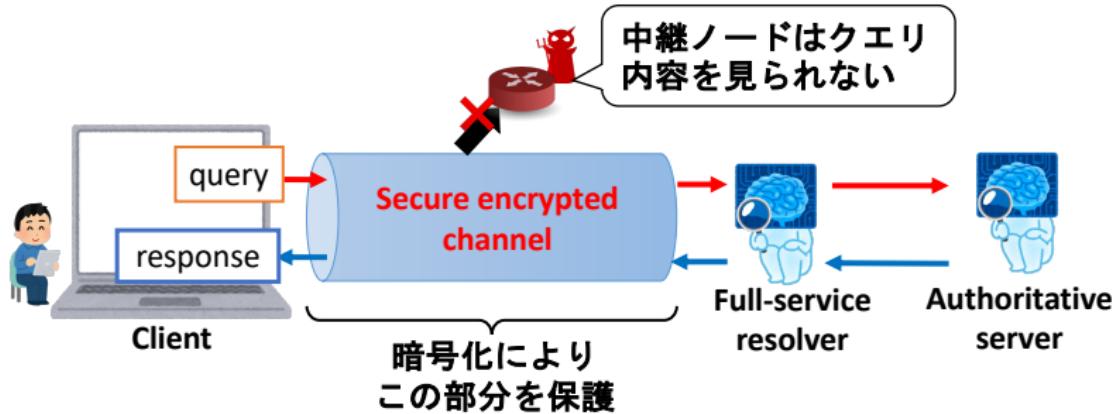
⇒ 2010 年代後半より、DNS プライバシ技術の研究・標準化が活発化。

- 暗号化 DNS: DoT (2016 [20]), DoH (2018 [19]), DoQ (2022 [21])
- 匿名 DNS: ODoH (2022 [23])

<sup>11</sup> 日本国内では通信の秘密により DNS ブロック等はある意味「アンタッチャブル」。

# 暗号化 DNS

DNS プロトコル自体をセキュアチャネル上で動作させることで、クエリの機密性を担保



代表的な技術は 3 種類

- **DoT (DNS over TLS) [20], DoQ (DNS over QUIC) [21]**
- **DoH (DNS over HTTPS) [19]**
- DNSCurve [17], DNSCrypt [16] (DNSCurve の後継) (今回は省略; Appendix 参照)

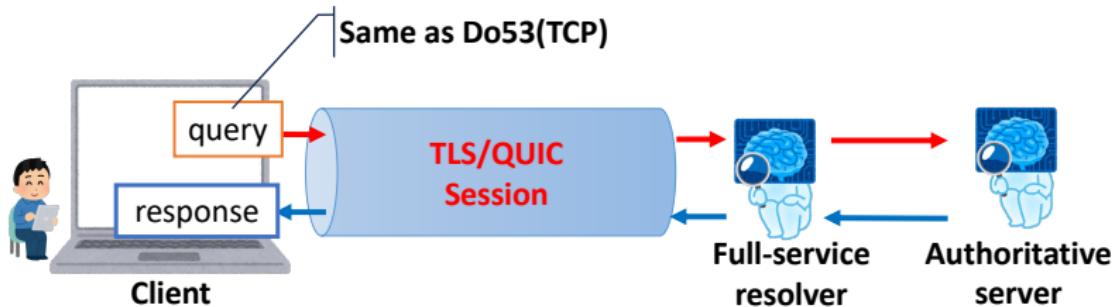
# DNS over TLS (DoT), DNS over QUIC (DoQ)

DoT: IETF RFC7858 [20] で標準化

- 素の DNS クエリ・レスポンスを、クライアント・サーバ間の **TLS のセキュアチャネル**上で送受信
- パケット構造は DNS over TCP port 53 のものと全く同一
- TCP port 853 を利用することが標準

DoQ: IETF RFC9250 [21] で標準化

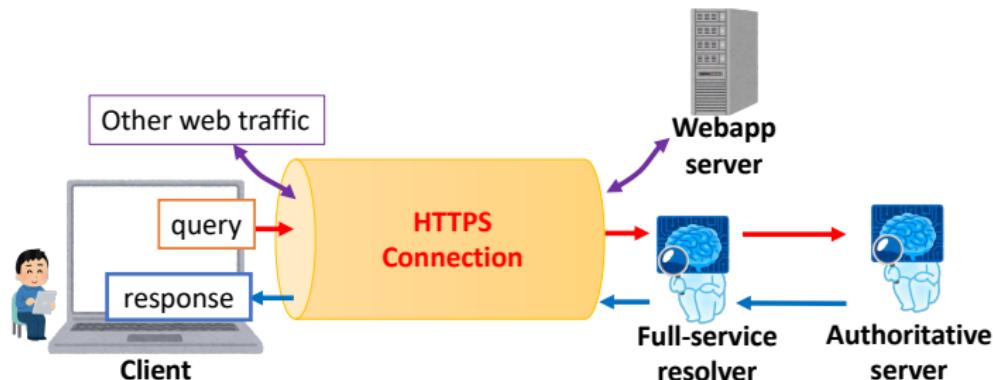
DoT の QUIC 版。UDP port 853 を利用。



# DNS over HTTPS (DoH)

DoH: IETF RFC8484 [19] で標準化

- DNS クエリ・レスポンスを、**クライアント・サーバ間の HTTPS コネクション上で送受信**  
⇒ TLS 上で HTTP のコンテキスト (POST/GET メソッド) を利用
- DoT と異なり、**他の Web トラフィックと同じポート 443 を利用**  
⇒ DoH のパケットのみを検閲・ブロックすることは困難<sup>12</sup>



<sup>12</sup>DoH により、DNS のアウトバウンドファイアウォールの設定は不可能に！

# 暗号化 DNS の対応・展開状況

## クライアント:

- ブラウザ: Firefox, Chrome, Edge 等は DoH をネイティブサポート
- OS: Windows 11, macOS 11, iOS 14, Android Pie 以降は DoT/DoH へネイティブ対応

## フルサービスリゾルバ:

- OSS リゾルバ: Unbound, Knot Resolver 等が DoT/DoH へ対応
- 公開リゾルバ: **Cloudflare, Quad9, Google DNS** 等が DoT/DoH を提供  
⇒ **殆どのユーザは、これら巨大企業の公開リゾルバを利用 [12]。**<sup>13</sup>

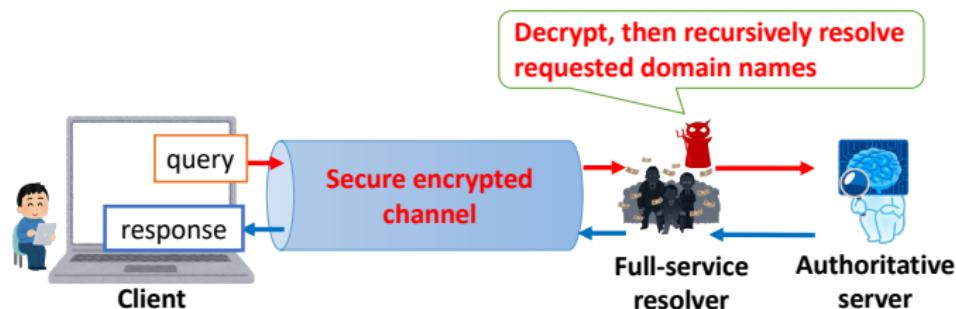
しかし、**巨大企業の公開リゾルバに DNS を集中させて良いのだろうか？**

---

<sup>13</sup>自分で対応するサーバを立てられない限り、実質これらを選択する以外の方法がない。

# 暗号化 DNS の限界と課題

- （当然だが）暗号化 DNS でも、リゾルバは名前解決のため、暗号化クエリを復号して平文のクエリを取得
- 暗号化 DNS クエリのソースアドレス＝クライアント IP アドレス



Both **plaintext query** and **user IP address** are known to the resolver!  
-> Every query is uniquely bound with the user who query issued.

フルサービスリゾルバは、**IP アドレス (誰)** と**クエリ (何)** を一意に紐づけ可能。  
ECS 対応の場合、権威サーバも正しくソース IP アドレスブロックを把握可能。

DNS プライバシ担保のために暗号化 DNS が普及するにつれ、逆説的に以下の課題が発生している。

- 巨大企業の公開リゾルバにユーザの DNS クエリが集中。
- その巨大企業へ、**ユーザの行動履歴・行動パターンが一元的に収集**。
- その結果、集約された情報が分析され、広告等に利用される懸念 (**プライバシリスク**)。<sup>14</sup>



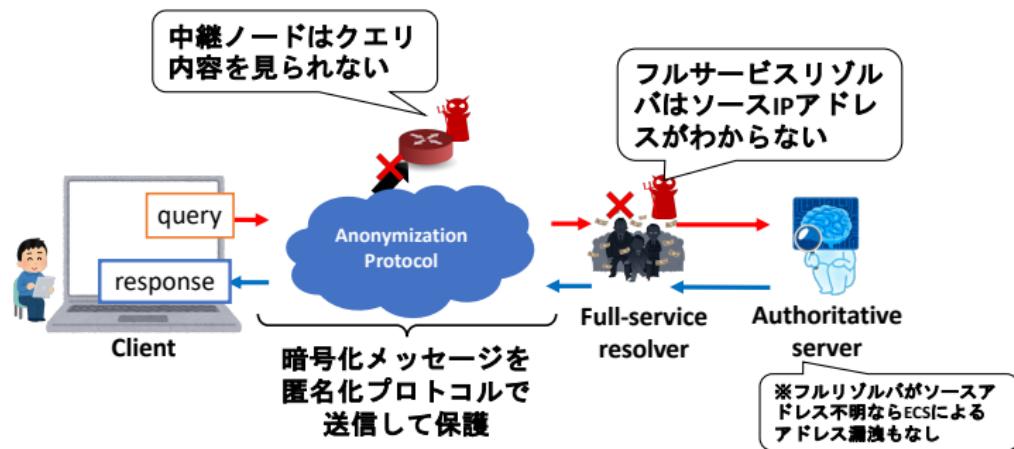
「誰」と「何」を分離する、すなわち  
**「匿名性を担保しつつ DNS 名前解決を行う手法(匿名 DNS)」**  
の研究開発・展開が活発化

---

<sup>14</sup>インターネットの中央集権化 (Internet Centralization) の文脈でも問題視されている。

# 匿名 DNS

暗号化 DNS を匿名化プロトコルを通じて実行することで、「ユーザの IP アドレス」と「クエリ内容」とを分離



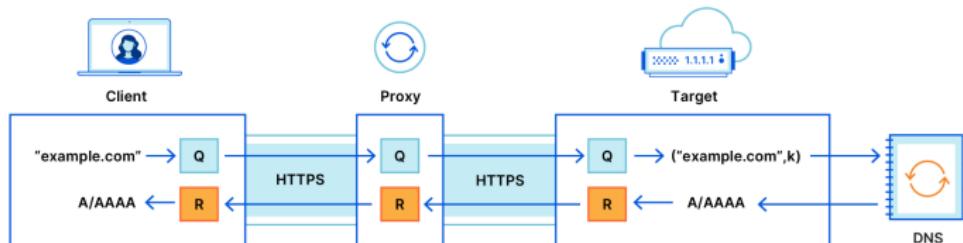
実際に展開されている手法は 3 種類

- ODoH (Oblivious DoH) [23]
- DoHoT (DNS over HTTPS over Tor) [37] (省略; Appendix 参照)
- DNSCrypt の Anonymized DNSCrypt [13, 14] (省略; Appendix 参照)

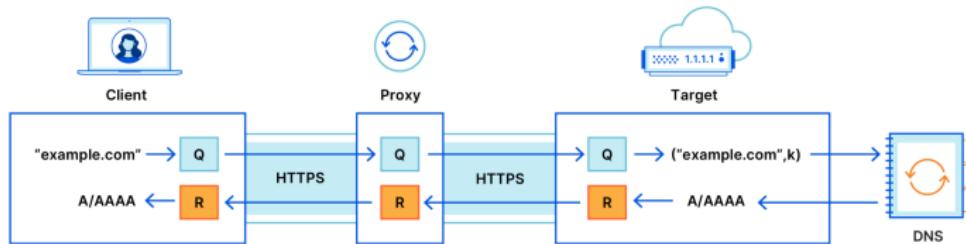
# ODoH (Oblivious DoH)

ODoH: IETF RFC 9230 [23] で標準化 (2022年2月)

- プロキシ(リレー)を介して、暗号化 DNS メッセージをフルサービスリゾルバと送受信。  
⇒ フルサービスリゾルバに対してユーザの IP アドレスを秘匿。
- DNS メッセージ自体は、TLS/HTTPS ではなく、HPKE [7] により暗号化。
- クライアントは、メッセージを HPKE 暗号化し、Oblivious Proxy (OP) を介して、Oblivious Target (OT) へ送信。
- Apple Private Relay として実装され、広く利用可能。UX を阻害しないシンプル・高速な構成。



ODoH の仕組み <https://blog.cloudflare.com/oblivious-dns/>



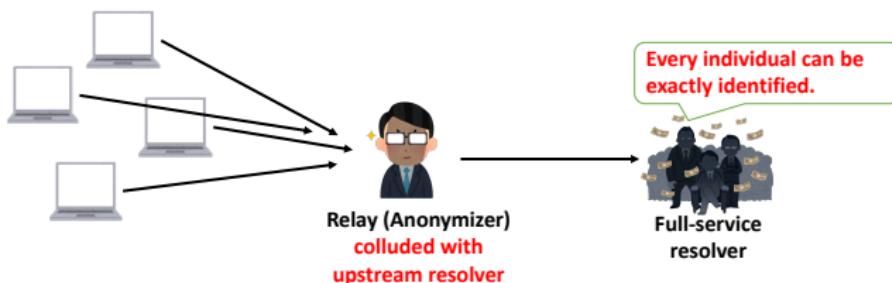
ODoH の仕組み <https://blog.cloudflare.com/oblivious-dns/>

- Client-OP、OP-OT で別の HTTPS コネクションを構築。Client-OT 間で HPKE 暗号化された DNS メッセージを送受信。
  - OP:** ユーザの IP アドレスを秘匿するプロキシ。  
⇒ ユーザの IP アドレスは OT にしか知られない。
  - OT:** 暗号化メッセージを復号し、名前解決を実行<sup>15</sup>。  
⇒ クエリ内容は OT にしか知れない。
- ただし OT と OP が結託したら匿名性が崩壊 (現状、どちらも限られた米国の CDN 事業者のみが提供しており、懸念がある)。

<sup>15</sup>通常、フルサービスリゾルバ兼ねる

# DNS 名前解決のプライバシ保護のまとめ

- **暗号化 DNS:** DNS メッセージの機密性を担保  
⇒ 中継ノードは傍受不可能に。フルサービスリゾルバに対するプライバシリスクは残存。
- **匿名 DNS:** DNS メッセージとユーザの IP アドレスを分離  
⇒ 行動履歴等が、リゾルバに一元的に収集されることを防止。
- 匿名 DNS の残る課題は「匿名化プロキシとリゾルバの結託への対策」。



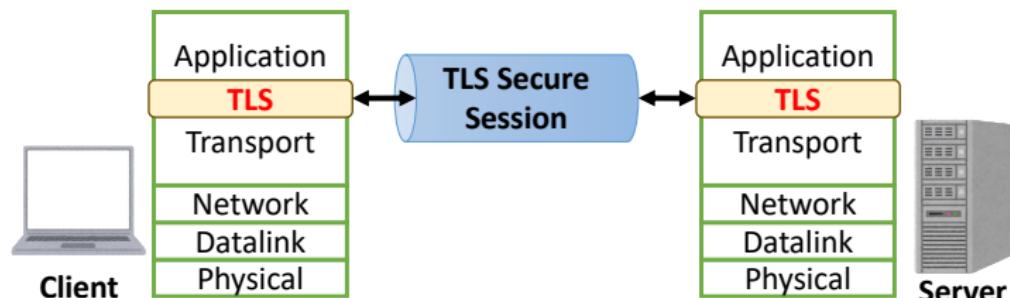
(おまけ) μODNS [24] のような、結託耐性を有し、実用に耐えうる匿名 DNS の研究開発が進行中。[栗原の研究]

# TLSハンドシェイクの プライバシ保護

# Transport Layer Security (TLS)

## TLS 概要

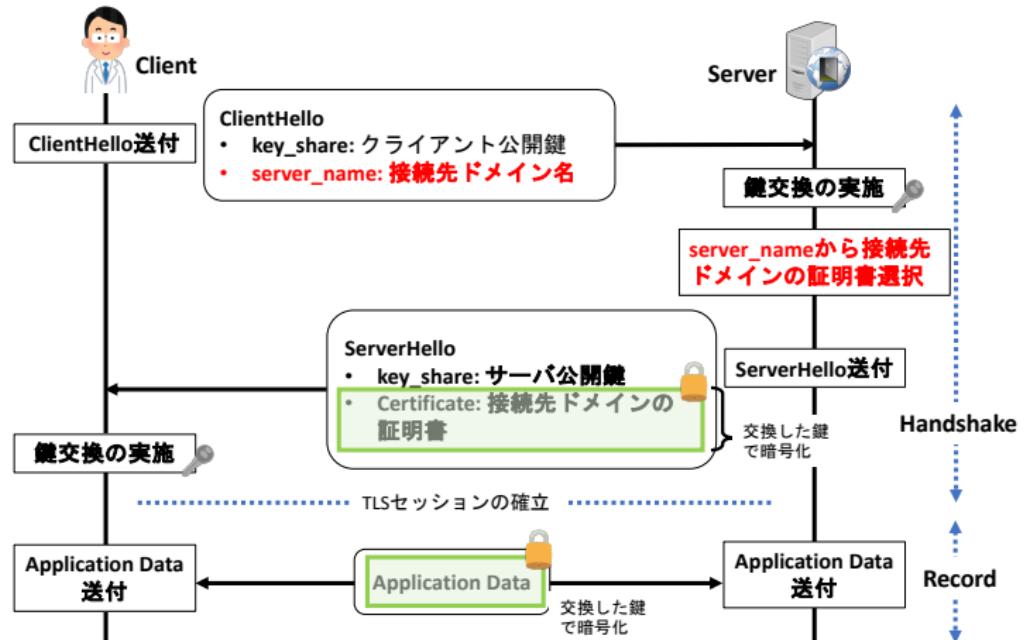
- 最新版: 2018 年発行の TLS v1.3 [31]
- トランスポート層とアプリケーション層の間のセキュリティプロトコル (TCP 上で動作<sup>16</sup>)
- クライアント・サーバ間でセッションを構築し、**サーバ認証**(あるいは相互認証)、セッション内通信の暗号化、暗号化通信の改ざん検出を実現



<sup>16</sup>UDP 上の QUIC [22] は TLS v1.3 がそのまま組み込まれている

# TLS v1.3 のセッション構築の流れ

- Handshake: 最短で 1 往復 (1RTT) で鍵交換を行い、セッションを構築
- Record: 交換した鍵を用いて、暗号化してアプリデータを送受信



TLS v1.3 のセッション構築 超概略 (RFC 8446 [31])

# Server Name Indication (SNI)

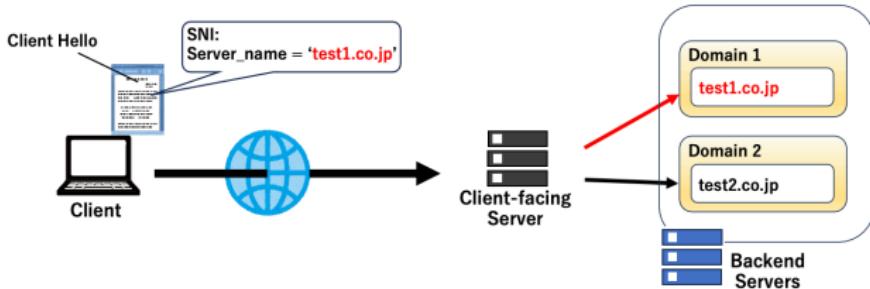
SNI: TLS 拡張 (Optional) のひとつ。現代の TLS では事実上必須。

## ■ SNI の役割

- 1 IP = 複数ドメイン (仮想ホスティング)
- TLS 接続先ドメイン名をサーバへ通知
- CDN/リバースプロキシで証明書選択に利用

## ■ プライバシリスク

- 訪問サイトが傍受者に露出
- DoH/DoT を回避し検閲可能
- 時刻・IP と結合し相関攻撃



TLS 接続先選択のために、TLS 暗号化の開始前に SNI を通知。

⇒ TLS ハンドシェイクの一番はじめ (ClientHello) に平文で送信される。

⇒ TLS 接続に必須な SNI だが、プライバシリスクを伴う

## 平文 SNI のプライバシリスクと現実の社会課題

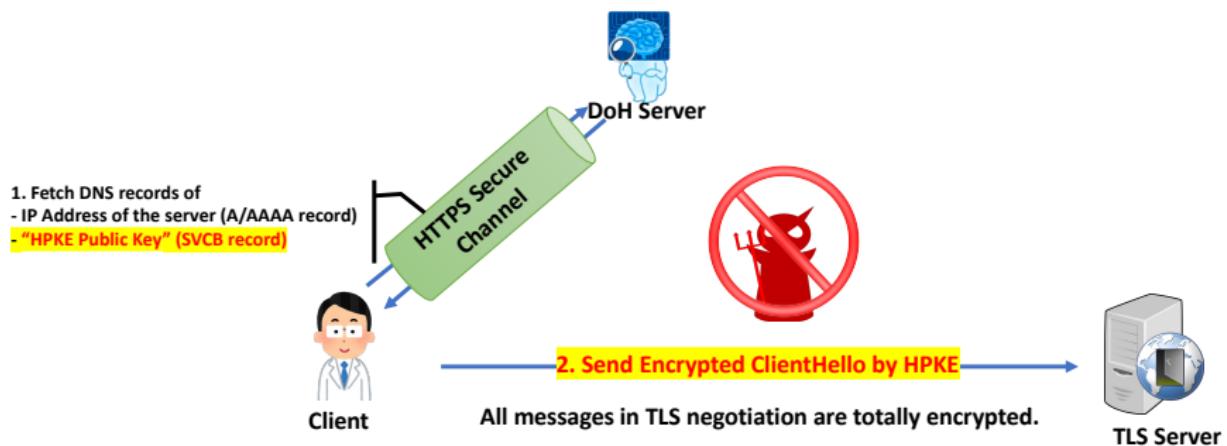
中国の Great Firewall やロシアの国営 ISP では、**DNS クエリに加えて、TLS SNI を監視**。特定ドメインへのアクセスをブロック。

DoH/DoT に続いて、2018 年より IETF を中心に SNI を秘匿する技術  
**Encrypted ClientHello (ECH)** の研究開発・展開が進行

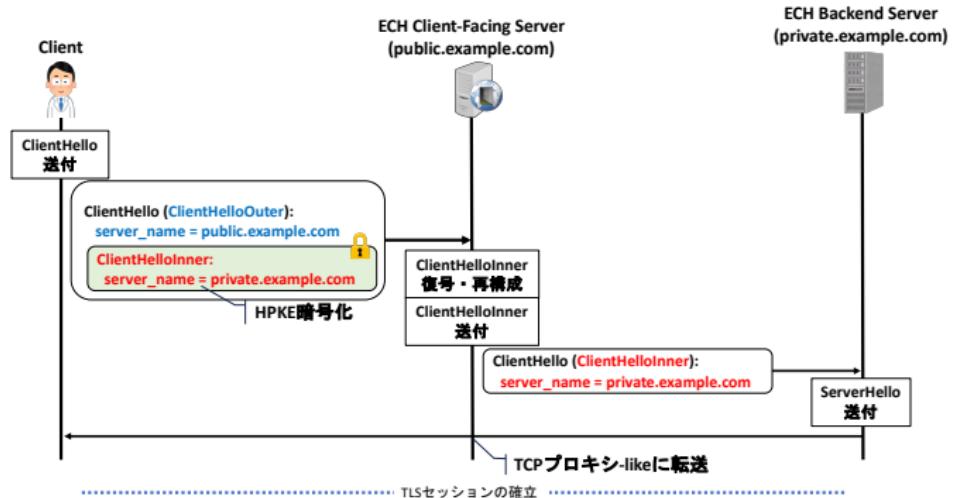
# Encrypted ClientHello (ECH): SNI の暗号化

ECH: IETF ドラフト提案 [32] (2025 年中に RFC 標準化見込)

- ClientHello 自体を暗号化、別 ClientHello の拡張フィールドに含める  
⇒ SNI に加えて、他のメタデータも秘匿
- ECH の鍵交換は、HPKE [7] (Appendix 参照) を利用。サーバ公開鍵は DNS SVCB レコード [33] で取得。  
⇒ (O)DoH/DoT 経由で取得が前提



- **ClientHelloInner:**  
秘匿する「秘密の宛先 **private.example.com**」へのオリジナルな接続要求
- **ClientHelloOuter:**
  - 「ダミー宛先 **public.example.com**」への接続要求 [平文 SNI]
  - HPKE で暗号化された ClientHelloInner を、ECH 拡張フィールドに含む



クライアントは ClientHelloOuter を送信  
 サーバは ClientHelloInnner を復号、本来の接続先 (**private.example.com**) へ転送  
 ⇒ フォワードプロキシ経由のような構成で、プライバシーを保護しつつ TLS 接続。

# ECH の展開状況

- ブラウザ: Chrome, Firefox, Edge 等は ECH をネイティブサポート
- CDN 事業者: Cloudflare が大規模に展開中<sup>17</sup>。
- TLS ライブラリ:
  - BoringSSL (Google): ECH をネイティブサポート<sup>18</sup>
  - OpenSSL ベース: DEfO プロジェクト [36]
  - Rustls: クライアント側のみ対応<sup>19</sup>

---

<sup>17</sup><https://blog.cloudflare.com/encrypted-client-hello/>

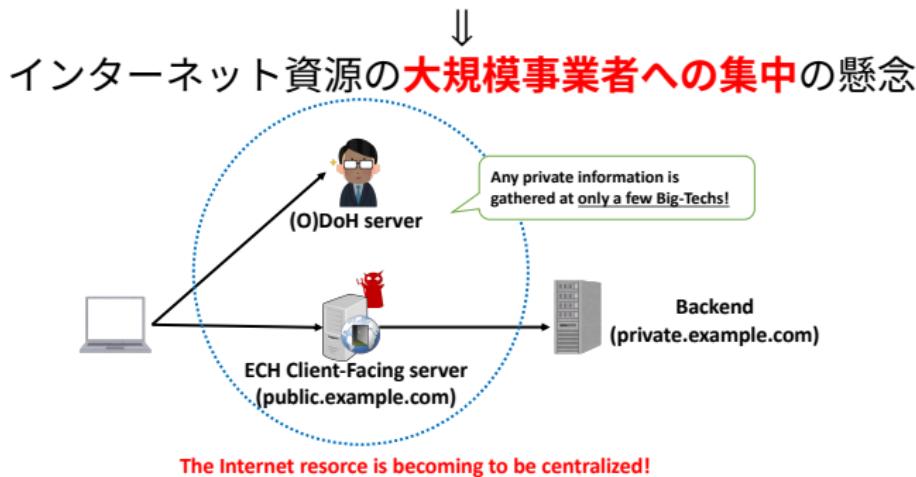
<sup>18</sup>[https://boringssl.googlesource.com/boringssl/+/refs/heads/master/ssl/encrypted\\_client\\_hello.cc](https://boringssl.googlesource.com/boringssl/+/refs/heads/master/ssl/encrypted_client_hello.cc)

<sup>19</sup><https://github.com/rustls/rustls>

# ECH の課題

利用面: (O)DoH 同様、**大規模事業者に全てのプライバシを握られる懸念**

- 強固なプライバシ保護のためには、ECH 復号可能なダミー宛先 (public.example.com) は、多数のバックエンドを持つ必要<sup>20</sup>  
⇒ **CDN 事業者のような大規模事業者以外では利用困難。**
- DNS ((O)DoH/DoT) と連携した鍵の定期更新・即時反映が必須  
⇒ **大規模な DNS インフラを持つ事業者の利用が前提。**



<sup>20</sup>木を隠すなら森の中。バックエンドが 1 つだけでは、結局 public-private が紐づいてしまう。

## 運用面

- ECH の利用有無を悟らせないために、**ECH 非利用時でもダミーの ECH 拡張フィールドを含める運用を推奨**<sup>21</sup>  
⇒ 未知の拡張フィールドにより、L4 ロードバランサ等の**ネットワーク機器の誤作動誘発の可能性**

---

<sup>21</sup> GREASE: Chrome, Firefox では対応済。常に ECH が利用されているように見える。

# TLSハンドシェイクのプライバシ保護のまとめ

- **ECH**: ClientHello を暗号化、接続先ドメイン名 (SNI) の機密性を担保  
⇒ DoH/DoT と組み合わせ、中継ノードに対して「ネットワークアクセス全体のプライバシ」が保護可能に
- (O)DoH 同様に大規模事業者へのインターネット資源の集中の懸念が残る

(おまけ) ECH に対応した L4 リバースプロキシの実装をリリース<sup>22</sup>

---

<sup>22</sup><https://github.com/junkurihara/rust-rpxy-14>

# TLSハンドシェイクの さらなる進化

# 公開鍵暗号と量子コンピュータ

古典的な公開鍵暗号 (例: RSA, Diffie-Hellman, ECDH):

- TLS をはじめとするインターネット基盤の根幹に利用
- 素因数分解問題・離散対数問題の求解困難性に安全性が依存

## 量子コンピュータ

- 素因数分解・離散対数問題は、Shor のアルゴリズム [34] で求解可能
- $N$  量子ビットコンピュータは、1回の演算で  $2^N$  回の演算を実行<sup>23</sup>  
→ 1量子ビットの増加で計算速度が2倍

古典コンピュータは引き続き発展していく、すなわち現在の公開鍵暗号の強化も進むだろうが、**量子コンピュータの発展速度は、その安全性強化のペースを上回ると予測。**<sup>24</sup>

<sup>23</sup> 古典コンピュータは  $2^N$  回の演算は  $2^N$  回の逐次実行が必要

<sup>24</sup> NIST は、2048 ビット RSA 暗号は 2030 年頃に危険化と予測。現在は 4096 ビットの利用が主だが、2030 年には一体何ビットを使えばいいのか？現実的に超長ビットの RSA なんて動くのか？

これまで解説した (O)DoH/DoT や ECH は、**全て既存の TLS を前提**

⇒ 量子コンピュータに対して脆弱な公開鍵暗号を用いており、**量子コンピュータの発展により、これらの技術は脆弱化**

⇒ **量子コンピュータに対して安全な公開鍵暗号による、TLS の進化が必須**

では、量子コンピュータが現実的になりそうになったら、そのとき対応すればいいのか？

⇒ **NO!**

## (再掲) Harvest now, decrypt later (HNDL) 攻撃

「暗号化データを収集・保存しておき、あとで復号」という、リアルタイム性はないものの、(量子)計算機の発展を待った有効な攻撃の思想

国家規模では今現在、ネットワーク上を流れるデータを収集・保存している攻撃者が存在。今流通している暗号化データは、近い将来解読される。



量子コンピュータの発展を待たずして、早急な対策が必須

⇒ 量子アルゴリズムでも効率的に解けない 耐量子公開鍵暗号 の開発・標準化が一気に進むことに。TLS にもその組み込みが進む。

というわけで、待ったなし。

# 耐量子公開鍵暗号の標準化: ML-KEM

米国 NIST により、耐量子公開鍵暗号の公募が開始 (2016 年)。2018 年 1 月～Round 1、2019 年 1 月～Round 2、2020 年 7 月～Round 3。

2022 年 7 月: Round 3 の Finalist が選出。それらをベースに 2024 年 8 月に以下の 3 つ<sup>25</sup> を NIST 標準として発表。

- **Module-Lattice-Based Key-Encapsulation Mechanism Standard (FIPS 203: ML-KEM)** [27] → 鍵カプセル化アルゴリズム
- **Module-Lattice-Based Digital Signature Standard (FIPS 204: ML-DSA)** [26] → 電子署名アルゴリズム
- **Stateless Hash-Based Digital Signature Standard (FIPS 205: SLH-DSA)** [28] → 電子署名アルゴリズム

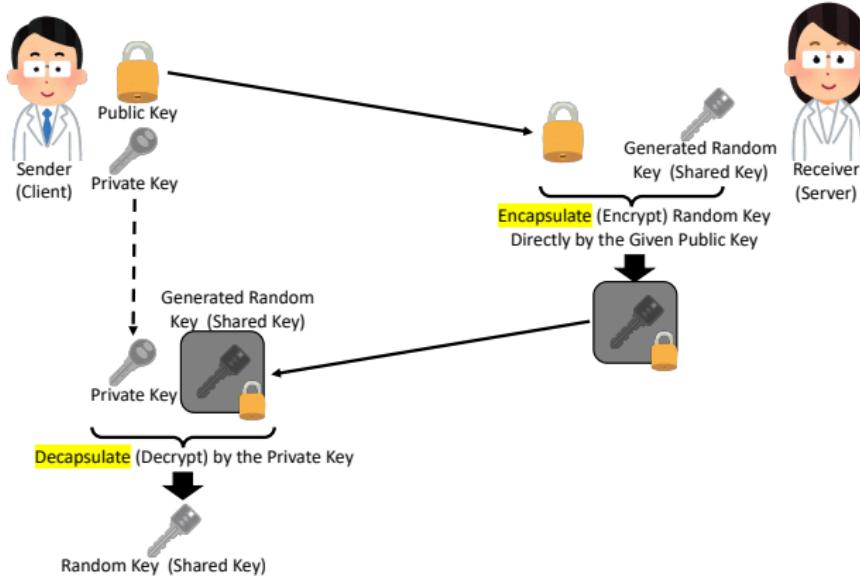
---

<sup>25</sup>Round 3 の Finalist は 4 つだが、4 つ目 (FALCON) は現在文書策定中 (FIPS 206: FN-DSA 予定)

# 鍵力プセル化 (Key Encapsulation Mechanism; KEM)

KEM: 共通鍵を暗号化するための、公開鍵暗号方式。<sup>26</sup>

受信者が生成したランダム共有鍵を、送信者の公開鍵で暗号化して送付。



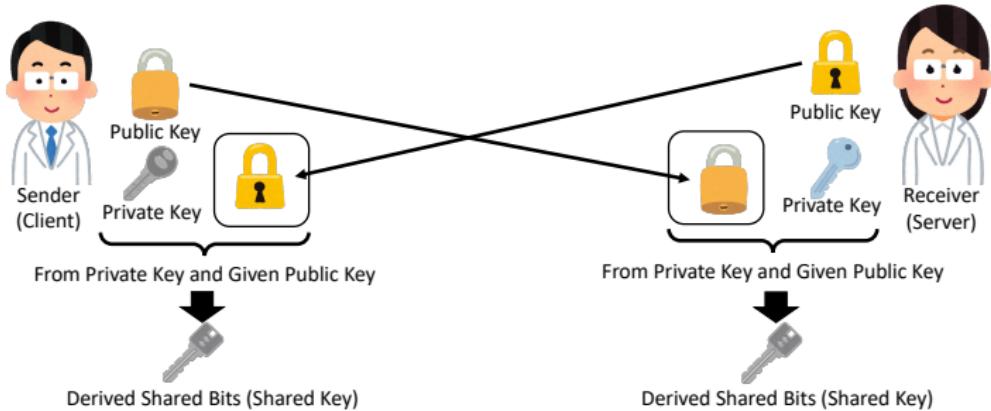
Key Encapsulation Mechanism (KEM) の手順

<sup>26</sup>Diffie-Hellman 鍵交換は、送受信者間で「共通鍵を暗号化して送って共有」するのではなく、「同じ共通鍵を独立に作る(鍵交換)」手法。一方、RSA では直接暗号化できる。

## 現行の TLS v1.3 の規格 [31]

公開鍵暗号方式による鍵交換として、(EC)DHE<sup>27</sup> 方式のみを規定<sup>28</sup>

⇒ **自身の秘密鍵と相手の公開鍵から Shared Bits = 共通鍵を導出**



TLS で従来用いられる (EC) Diffie-Hellman 鍵交換 ((EC)DH KX) の手順

<sup>27</sup>(Elliptic Curve) Diffie-Hellman Ephemeral: 毎回鍵ペアを生成・使い捨てる Ephemeral 方式。Perfect Forward Secrecy (PFS) を有する。

<sup>28</sup>固定のサーバ RSA 鍵による鍵の共有は、「Bleichenbacher 攻撃に脆弱」という理由から、TLS 1.3 では削除された。おそらく、「RSA 鍵の毎回の生成は低速」もある。RSA 署名のついた証明書は利用できることに注意。

鍵共有をするという意味では、やってることは KEM も鍵交換も同じ。  
紛らわしいので、以降の言葉の定義<sup>29</sup>

- 鍵交換 (Key Exchange; KX): (EC)DH 鍵交換方式
  - 公開鍵 Public Key
  - 秘密鍵 Private Key
- 鍵カプセル化 (Key Encapsulation Mechanism; KEM):  
ランダム共通鍵を暗号化して送付する方式
  - カプセル化鍵 Encapsulation Key: 暗号化に用いる公開鍵
  - カプセル化解除鍵 Decapsulation Key: 復号に用いる秘密鍵

---

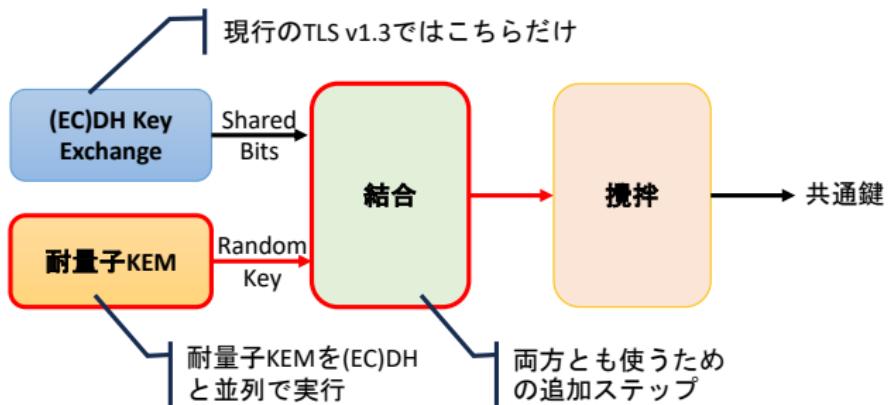
<sup>29</sup>FIPS 203 ML-KEM [27] および IETF Draft [35] に準拠

# 利用・標準化が進むハイブリッド耐量子鍵交換

ざっくり言うと：ハイブリッド耐量子 (Hybrid Post-Quantum; PQ) 鍵交換

- (EC)DH KX (e.g., X25519, secp256r1)
- 耐量子 (PQ) KEM (i.e., ML-KEM)

を両方・独立に実行、両者の結果を結合・攪拌して1つの共通鍵を生成。



なぜ2つを同時に使うのか？ 耐量子KEMに移行すればいいのでは？

回答: 暫くの間一方のみでは安全性が担保できない恐れがあるから。

	Pros	Cons
(EC)DH KX	長い安全性検証の歴史	量子コンピュータに対して危険化が懸念
PQ KEM	量子コンピュータで効率的な解法がない	浅い安全性検証の歴史 (未知の脆弱性の可能性 <sup>30)</sup>

⇒ ハイブリッド = (EC)DH KX と PQ KEM が安全性を相互に補完

- PQ KEM (i.e., ML-KEM) を効率的に破る手法が発見されても、(EC)DH KX で安全性を担保。
- 量子コンピュータが急速に発展して (EC)DH KX が危険化しても、PQ KEM で安全性を担保。

---

<sup>30</sup>既存のコンピュータに対する、アルゴリズム的な脆弱性。

例えば以下の Hybrid PQ KX が IETF ドラフト提案されている<sup>31</sup>

- Post-quantum hybrid ECDHE-MLKEM Key Agreement for TLSv1.3 [25]
- X-Wing: general-purpose hybrid post-quantum KEM [9]
- Post-Quantum and Post-Quantum/Traditional Hybrid Algorithms for HPKE [6]

特に、TLS v1.3 の ECDHE-MLKEM は、その前バージョンを含めてインターネット上での実装・利用が進んでいる。<sup>32</sup> 例えば、

- ブラウザ: Firefox 123 以上、Chrome 124 以上
- CDN: Cloudflare • Google Cloud 等<sup>33</sup>
- TLS ライブライアリ: BoringSSL, Rustls 0.23.22+, OpenSSL 3.5.0+

---

<sup>31</sup> <https://wiki.ietf.org/group/sec/PQCAgility>

<sup>32</sup> 前バージョン: X25519Kyber768Draft00。 <https://pq.cloudflareresearch.com> を参照。

<sup>33</sup> 対応ブラウザであれば <https://www.google.com> へ Hybrid PQ KX で接続可能

<https://www.google.com>

Secure origins

<https://waa-pa.clients6.go...>

<chrome-extension://oboon...>

<chrome-extension://jffboc...>

<https://ogads-pa.clients6....>

<https://play.google.com>

<https://ogs.google.com>

<https://www.gstatic.com>

<https://ssl.gstatic.com>

<https://lh3.googleusercontent...>

<https://fonts.gstatic.com>

This page is secure (valid HTTPS).



Certificate - **valid and trusted**

The connection to this site is using a valid, trusted server certificate issued by WR2.

[View certificate](#)



Connection - **secure connection settings**

The connection to this site is encrypted and authenticated using QUIC, X25519MLKEM768, and AES\_128\_GCM.



Resources - **all served securely**

All resources on this page are served securely.

Chrome の flags において "Use ML-KEM in TLS 1.3" を Enabled にした状態で、google.com へアクセス

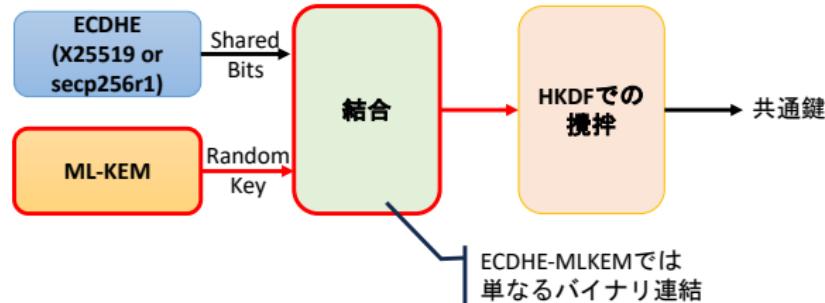
# Hybrid (PQ) KX を用いた TLS v1.3 での鍵交換

## Hybrid Key Exchange in TLS v1.3

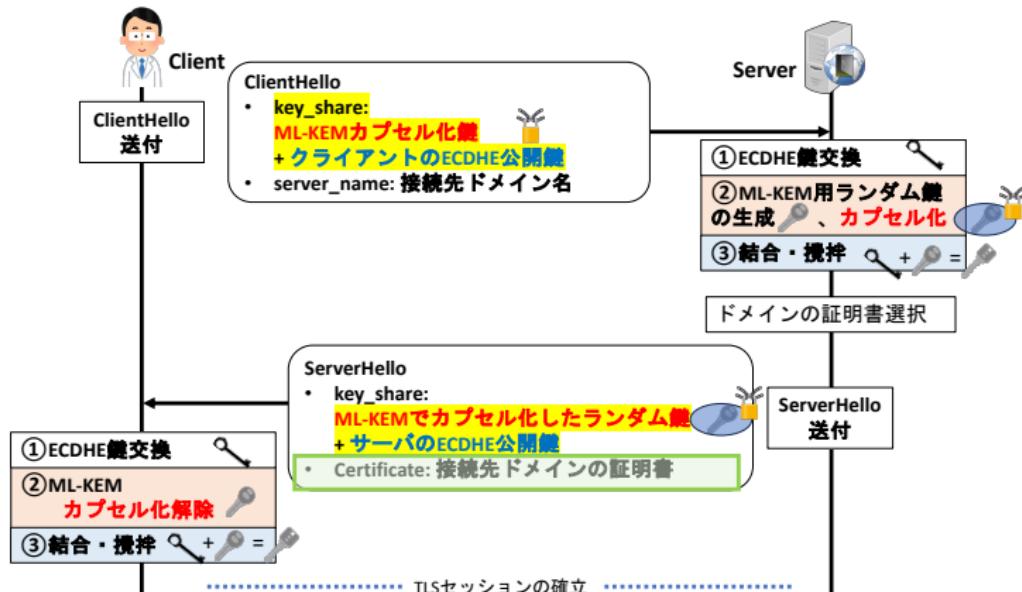
IETF RFC Draft [35] で策定中のフレームワーク。

- (EC)DHE など「Traditional」な鍵交換
- PQ KEM を含む新しい「Next-generation」な手法

の両方を同時に実行することで、**片方が破られてももう片方で TLS 接続の秘匿性を担保し続ける**ことが目的。このフレームワーク内で設定される具体的なパラメタ (ECDHE-MLKEM) が同時に策定中 [25]。現在、このドラフト提案中のパラメタで、広く展開されつつある。



- クライアントは TLS ClientHello で、KX 公開鍵と KEM カプセル化鍵を両方送信。
- サーバ側は、従来の KX 実行とともに、ランダム鍵を生成・そのランダム鍵を PQ KEM によりカプセル化。



IETF ドラフト ECDHE-MLKEM [25] における、TLS ハンドシェイクの流れ

# TLSにおけるHybrid PQ KX/ECDHE-MLKEMの懸念点

もちろん、現行プロトコルを更新する実装の問題は存在。

更に、**ClientHello/ServerHello の巨大化**というデメリットが、プロトコル自身に存在。<sup>34</sup>



ServerHello はまだしも<sup>35</sup>、ClientHello のパケットが下位レイヤでフラグメントされる可能性。



中間のスイッチでは予期しない動作を引き起こす可能性や、分割・結合によるパフォーマンスロスも無視できない可能性(が、解はない)。

---

<sup>34</sup>ML-KEM-768 の公開鍵は 1184 バイト、カプセル化共有鍵は 1088 バイト。一方で、X25519 公開鍵は 32 バイト……実に 30 倍以上。

<sup>35</sup>証明書を含む場合は元々デカい

# TLS v1.3 の耐量子鍵交換のまとめ

- 量子コンピュータの発展速度や HNDL 攻撃の脅威を考慮し、**耐量子公開鍵暗号の開発・標準化が進行中。**
- TLS v1.3 におけるハイブリッド耐量子鍵交換の標準化・実運用への展開が進んでいる [25, 35]。 (ECDHE-MLKEM)<sup>36</sup>**
- パフォーマンスへの影響・ネットワーク機器の誤作動誘発の可能性も存在。

(おまけ) Out-of-the-Box で TLS v1.3 Hybrid PQ KX を利用可能な Rust 製高速 HTTP(S) リバースプロキシをリリース<sup>37</sup>

---

<sup>36</sup> ODoH/ECH の HPKE においても、ハイブリッド耐量子鍵交換の標準化が進行中 [6]。

<sup>37</sup> [rpxy: https://github.com/junkurihara/rust-rpxy](https://github.com/junkurihara/rust-rpxy),  
<https://rpxy.io/>,  
<https://developers.cloudflare.com/ssl/post-quantum-cryptography/pqc-support/>

# まとめ

# まとめ

- インターネットにおけるユーザプライバシは危機的な状況
- このため、DNS と TLS の進化・展開が急速に進行中
  - DNS: Oblivious DoH, DoH, DoT, DoQ
  - TLS: Encrypted ClientHello (ECH), Hybrid Post-Quantum Key Exchange (PQ KX)
- 上記の技術は、**大規模な事業者にインターネットアクセスのプライバシを集中させる懸念も存在**
- これらの技術が普及することで、ユーザのプライバシがどのように影響を受けるか、引き続き注視する必要

# 参考文献 I

- [1] <https://github.com/DNSCrypt/dnscrypt-proxy>.
- [2] <https://github.com/jedisct1/encrypted-dns-server>.
- [3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "DNS security introduction and requirements," RFC4033, Mar. 2005. [Online]. Available: <https://tools.ietf.org/html/rfc4033>
- [4] ——, "Protocol modifications for the DNS security extensions," RFC4035, Mar. 2005. [Online]. Available: <https://tools.ietf.org/html/rfc4035>
- [5] ——, "Resource records for the DNS security extensions," RFC4034, Mar. 2005. [Online]. Available: <https://tools.ietf.org/html/rfc4034>
- [6] R. Barnes, "Post-quantum and post-quantum/traditional hybrid algorithms for hpke," IETF RFC Draft 0, Apr. 2025. [Online]. Available: <https://www.ietf.org/archive/id/draft-barnes-hpke-pq-00.html>
- [7] R. Barnes, K. Bhargavan, B. Lipp, and C. Wood, "Hybrid public key encryption," RFC 9180, Feb. 2022. [Online]. Available: <https://tools.ietf.org/html/rfc9180>
- [8] M. Boucadair, T. Reddy.K, D. Wing, N. Cook, and T. Jensen, "Discovery of named resolvers," IETF RFC 9463, Nov. 2023. [Online]. Available: <https://tools.ietf.org/html/rfc9463>
- [9] D. Connolly, P. Schwabe, and B. Westerbaan, "X-Wing: general-purpose hybrid post-quantum KEM," IETF RFC Draft 7, May 2025. [Online]. Available: <https://www.ietf.org/archive/id/draft-connolly-cfrg-xwing-kem-07.html>
- [10] C. Contavalli, W. van der Gaast, D. Lawrence, and W. Kumari, "Client subnet in DNS queries," RFC7871, May 2016. [Online]. Available: <https://tools.ietf.org/html/rfc7871>
- [11] J. Damas, M. Graff, and P. Vixie, "Extension mechanisms for DNS (EDNS(0)), " RFC6891, Apr. 2013. [Online]. Available: <https://tools.ietf.org/html/rfc6891>
- [12] C. Deccio and J. Davis, "DNS privacy in practice and preparation," in *Proc. ACM CoNEXT 2019*, Orlando, FL, USA, 2019, pp. 138—143.
- [13] F. Denis, "Anonymized dnscrypt specification," <https://github.com/DNSCrypt/dnscrypt-protocol/blob/master/ANONYMIZED-DNSCRYPT.txt>, Jun. 2020, commit ID: 78547018.

# 参考文献 II

- [14] ——, "Anonymized dns," <https://github.com/DNSCrypt/dnscrypt-proxy/wiki/Anonymized-DNS>, Jan. 2021, commit ID: 9e384ee.
- [15] ——, "The DNSCrypt protocol," IETF RFC Draft 6, Apr. 2025. [Online]. Available: <https://www.ietf.org/archive/id/draft-denis-ddrive-dnscrypt-06.html>
- [16] DNSCrypt, <https://www.dnscrypt.org>.
- [17] DNSCurve, <https://dnscurve.org>.
- [18] S. Farrell and H. Tschofenig, "Pervasive monitoring is an attack," RFC7258, May 2014. [Online]. Available: <https://tools.ietf.org/html/rfc7258>
- [19] P. Hoffman and P. McManus, "DNS queries over HTTPS (DoH)," RFC8484, Oct. 2018. [Online]. Available: <https://tools.ietf.org/html/rfc8484>
- [20] Z. Hu, L. Zhu, J. Heidemann, A. Mankin, D. Wessels, and P. Hoffman, "Specification for DNS over transport layer security (TLS)," RFC7858, May 2016. [Online]. Available: <https://tools.ietf.org/html/rfc7858>
- [21] C. Huitema, S. Dickinson, and A. Mankin, "DNS over dedicated QUIC connections," RFC9250, May 2022. [Online]. Available: <https://tools.ietf.org/html/rfc9250>
- [22] J. Iyengar and M. Thomson, "Quic: A udp-based multiplexed and secure transport," IETF RFC 9000, May 2021. [Online]. Available: <https://tools.ietf.org/html/rfc9000>
- [23] E. Kinnear, P. McManus, T. Pauly, and C. A. Wood, "Oblivious DNS over HTTPS," RFC9230, Jun. 2022. [Online]. Available: <https://datatracker.ietf.org/doc/rfc9230/>
- [24] J. Kurihara, T. Tanaka, and T. Kubo, " $\mu$ ODNS: A distributed approach to DNS anonymization with collusion resistance," *Computer Networks*, vol. 237, p. 110078, Dec. 2023.
- [25] K. Kwiatkowski, P. Kampanakis, B. E. Westerbaan, and D. Stebila, "Post-quantum hybrid ecdhe-mlkem key agreement for tlsv1.3," IETF RFC Draft 3, Dec. 2024. [Online]. Available: <https://www.ietf.org/archive/id/draft-kwiatkowski-tls-ecdhe-mlkem-03.html>
- [26] National Institute of Standards and Technology, "Module-lattice-based digital signature standard," Aug. 2024. [Online]. Available: <https://csrc.nist.gov/pubs/fips/204/final>

# 参考文献 III

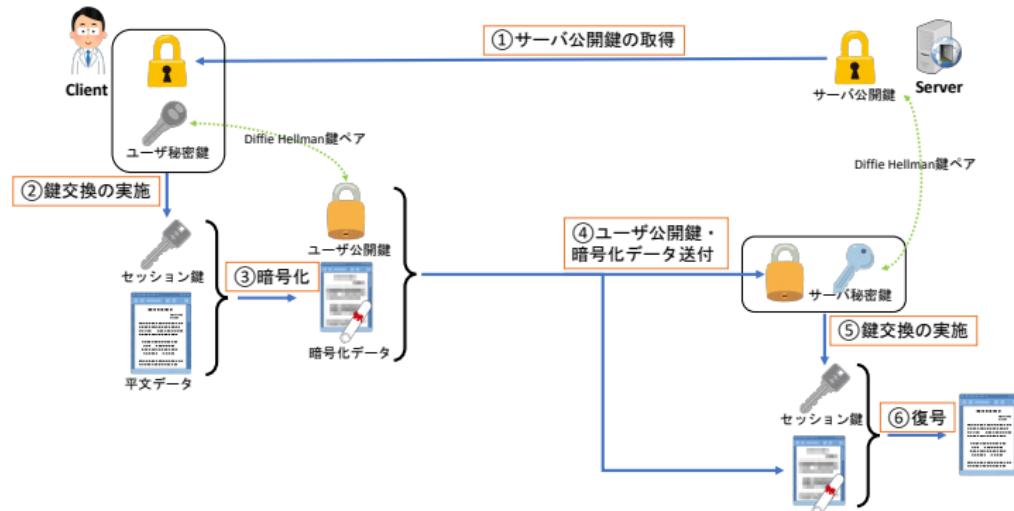
- [27] ——, "Module-lattice-based key-encapsulation mechanism standard," Aug. 2024. [Online]. Available: <https://csrc.nist.gov/pubs/fips/203/final>
- [28] ——, "Stateless hash-based digital signature standard," Aug. 2024. [Online]. Available: <https://csrc.nist.gov/pubs/fips/205/final>
- [29] ——, "What is post-quantum cryptography?" Jun. 2025, accessed: 2025-06-20. [Online]. Available: <https://www.nist.gov/cybersecurity/what-post-quantum-cryptography>
- [30] T. Pauly, E. Kinnear, C. A. Wood, P. McManus, and T. Jensen, "Discovery of designated resolvers," IETF RFC 9462, Nov. 2023. [Online]. Available: <https://tools.ietf.org/html/rfc9462>
- [31] E. Rescorla, "The transport layer security (tls) protocol version 1.3," IETF RFC 8446, Aug. 2018.
- [32] E. Rescorla, K. Oku, N. Sullivan, and C. A. Wood, "TLS Encrypted ClientHello," IETF RFC Draft 25, Jun. 2025. [Online]. Available: <https://www.ietf.org/archive/id/draft-ietf-tls-esni-25.html>
- [33] B. M. Schwartz, M. Bishop, and E. Nygren, "Service binding and parameter specification via the DNS (SVCB and HTTPS records)," IETF RFC 9460, Nov. 2023. [Online]. Available: <https://tools.ietf.org/html/rfc9460>
- [34] P. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. FOCS*, 1994, pp. 124–134.
- [35] D. Stebila, S. Fluhrer, and S. Gueron, "Hybrid key exchange for TLS 1.3," IETF RFC Draft 13, Jun. 2025. [Online]. Available: <https://www.ietf.org/archive/id/draft-ietf-tls-hybrid-design-13.html>
- [36] D. Team, "DEFO: Developing ECH for OpenSSL," accessed: 2025-06-24. [Online]. Available: <https://defo.ie>
- [37] Tor Project, <https://www.torproject.org>.
- [38] N. Weaver, C. Kreibich, and V. Paxson, "Redirecting DNS for ads and profit." in *Proc. USENIX FOCI 2011*, 2011.

# Appendix

# Appendix: Hybrid Public Key Encryption (HPKE)

HPKE: IETF RFC9180 [7] で標準化 (2022 年 2 月)

- 公開鍵暗号を用いた、鍵交換とメッセージ暗号化のためのフレームワーク。
- TLS v1.3 の鍵交換方式 (DHKEM) をベースにした、軽量な鍵交換方式。  
⇒ TLS v1.3 同様に、耐量子鍵交換方式への拡張が提案中 [6]。
- ODoH や ECH など、様々なプロトコルで利用。



HPKE の基本フロー

## Appendix: DNSCrypt

DNSCrypt [16]: OSS としてコミュニティ開発。<sup>38</sup>

- DNS メッセージを暗号化によりラップ、UDP/TCP パケット単位で送受信。汎用的セキュアチャネルは構築しない。
- クライアント用暗号化プロキシ (dnscrypt-proxy [1]) および DNSCrypt 対応リゾルバ (フォワーダ) の実装 [2] が OSS として公開。
- OpenDNS, Yandex, Adguard, Quad9 などが DNSCrypt 対応の公開リゾルバとして存在。
- ODoH 同様の匿名化手法が実装済<sup>39</sup>
- コミュニティベースで普及したのち、2025 年現在、IETF 標準化活動中 [15]

---

<sup>38</sup><https://github.com/dnscrypt>

<sup>39</sup>Anonymized DNSCrypt [13]

# Appendix: DoHoT (DNS over HTTPS over Tor)

## DoHoT

- Tor [37] ネットワーク上を通じて DoH を行う。
- DoH でクエリを暗号化した上で、経路上の通信も Tor で暗号化される = DNSだけのためとしては Overkill なことも。
- 信頼できるノードを経由しているのかどうか不明。3 ノードを経由するため、UX を阻害する深刻な遅延も発生。
- 地域によってレスポンスが変化する DNS レコードについては、Exit ノードの適切な設定が必要<sup>40</sup>

DoT や DNSCrypt over TCP も Tor 経由で動作する。<sup>41</sup>

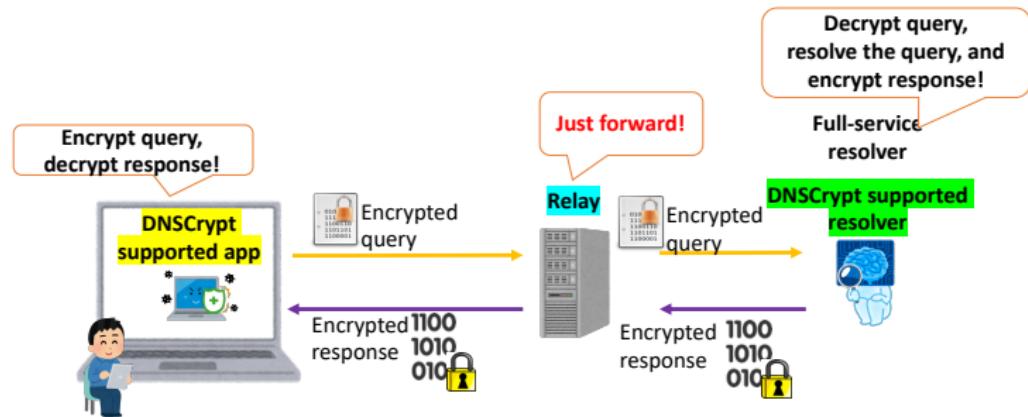
<sup>40</sup> 「StrictExitNodes 1」および「ExitNodes {node nickname}」を torrc に指定

<sup>41</sup> Tor は TCP のみをサポートするので UDP 不可。

# Appendix: Anonymized DNSCrypt

## Anonymized DNSCrypt [14, 13]

- 2019 年に DNSCrypt の拡張仕様として実装。
- Public relay を通じて、DNSCrypt のクエリ・レスポンスを送受信することでユーザ匿名性を担保。
- ODoH との違い: HTTPS ではなく生の UDP/TCP を利用。
- やっぱり relay が宛先と共に謀したら匿名性が崩れる。



DNSCrypt supported resolver is 'target' in ODoH

# Appendix: (O)DoH/DoT のブートストラップ

## (O)DoH/DoT のブートストラップ

DoH/DoT を利用するためには、まず信頼できる (O)DoH (URL) /DoT (ドメイン名) のサーバを見つける必要

- 現状、ほとんどの場合はブラウザや OS の設定で、事前に信頼できる **DoH/DoT サーバを指定**  
⇒ 初回接続時、そのサーバの IP アドレスは通常の DNS over port 53 (Do53) で名前解決 (一発目は平文。どうしようもない。)
- デフォルト DNS と同様に「(O)DoH/DoT を自動的に設定する仕組み」が徐々に展開されつつある
  - ① **DDR (Discovery of Designated Resolvers) [30]:**  
⇒ DHCP/IPv6 RA で事前に与えられた DNS リゾルバに対し、(O)DoH/DoT サーバの情報を問合せ (via SVCB レコード [33])。
  - ② **DNR (Discovery of Network-designated Resolvers) [8]:**  
⇒ DHCP/IPv6 RA で直接 DoH/DoT サーバの情報を配布

# Appendix: NIST PQC 標準化の補足

## NIST FIPS の元となった Round 3 Finalists

- SLH-DSA: SPHINCS<sup>+</sup> (ハッシュ関数ベース)
- ML-DSA: CRYSTALS-DILITHIUM (格子問題ベース)
- ML-KEM: CRYSTALS-KYBER (格子問題ベース)
- FN-DSA: FALCON (高速フーリエ変換ベース、FIPS 206 予定)

## NIST PQC 標準化 Round 4

Round 2 の Finalist は、特に有望で先に標準化すべきと判断されたアルゴリズムと、それ以外の 2 種類に分類された。このため、Round 3 では前者を中心に評価が行われたようである。

Round 3 終了後、2022 年 7 月より Round 4 が開始され、残った Round 2 Finalist のうち 4 つ<sup>42</sup> を再度選定し、評価を開始している。

<sup>42</sup>開始までに破られなかつたもの

## Appendix: ML-KEM の種類・鍵長・パフォーマンス

$k \in \{2, 3, 4\}$  に対し、ML-KEM- $256k$  を定義。鍵長は全てバイト単位。

ML-KEM- $256k$  は AES- $64k$  と同等のセキュリティ強度と見做される。(e.g., AES-192 = ML-KEM-768)

Table: ML-KEM における各データサイズ

	カプセル化鍵	カプセル化解除鍵	暗号文	共有鍵
ML-KEM-1024	1,568	3,168	1,568	32
ML-KEM-768	1,184	2,400	1,088	32
ML-KEM-512	800	1,632	768	32

データサイズは非常に大きいが X25519 (ECDH の一種) よりも高速な処理を実現

Table: Cloudflare による X25519 と ML-KEM の比較<sup>43</sup>

	データサイズ (Bytes)		処理回数/秒	
	Client→Sever	Server → Client	Client	Server
ML-KEM-768	1,184 (カプセル化鍵)	1,088 (暗号文)	31,000 (鍵生成・復号)	70,000 (暗号化)
ML-KEM-512	800 (カプセル化鍵)	768 (暗号文)	50,000 (鍵生成・復号)	100,000 (暗号化)
X25519	32 (公開鍵)	32 (公開鍵)	17,000 (鍵生成・共有)	17,000 (鍵生成・共有)

<sup>43</sup><https://blog.cloudflare.com/post-quantum-for-all/>, 計測環境は不明

## ML-KEM の推奨鍵長や処理速度

既存の公開鍵暗号同様に、鍵長が長くなれば処理速度が低下する。  
NIST は ML-KEM-768 (カプセル化鍵長 1184 bytes) を、セキュリティ・パフォーマンスの観点から推奨している

(*Section 8, FIPS 203*)

*NIST recommends using ML-KEM-768 as the default parameter set, as it provides a large security margin at a reasonable performance cost.*