

プロトコル構造を保持した特徴表現を用いた TLS パラメータの変異解析

神田 敦^{1,2,a),b)} 橋本 正樹^{3,c)} 大久保 隆夫^{1,d)}

概要：近年、通信の暗号化が広く普及したことにより、攻撃者も暗号化通信を利用するケースが増加している。その結果、従来のペイロード解析では通信の識別が困難となっている。このような状況下において、暗号化された通信を識別するための特徴を抽出、表現する技術の一つが JA4+や Mercury NPF を代表とする TLS フィンガープリンティングである。これらの技術は TLS ハンドシェイク中に送信されるパラメータ群をフィンガープリントに変換する。既存の生成アルゴリズムの多くは専門家の経験則に基づいて設計されており、パラメータ選定の妥当性について体系的な検証が十分とは言えない。本研究では、TLS パラメータの分布や変遷を明らかにするために、通信プロトコルの構造をコンパクトに保持しつつ、フィンガープリントとしても利用可能な表現形式 Compacted Protocol Representation (CPR) と、その構造的差異に着目した編集距離 Structural Edit Distance (SED) を設計した。CPR を TLS Client Hello メッセージに適用した Compacted TLS Client Hello (CTLS-CH) を導入し、10 年以上にわたる大規模 TLS 通信データセットを用いて、TLS パラメータの変異傾向を解析した。本調査により明らかとなった TLS パラメータ変異に基づいて、既存 TLS フィンガープリンティング技術の課題を明らかにし、提案手法の有効性を示した。

キーワード：SSL/TLS, TLS フィンガープリンティング

Mutation Analysis of TLS Parameters Using Structure-Preserving Feature Representations

ATSUSHI KANDA^{1,2,a),b)} MASAKI HASHIMOTO^{3,c)} TAKAO OKUBO^{1,d)}

Abstract: In recent years, the widespread adoption of encrypted communication has led to an increase in cases where attackers also utilize encrypted channels. Consequently, traditional payload-based analysis has become less effective for traffic identification. One of the key techniques for extracting and representing features to identify encrypted communication under such circumstances is TLS fingerprinting, exemplified by methods such as JA4+ and Mercury NPF. These techniques convert parameters exchanged during the TLS handshake into unique fingerprints. However, most existing fingerprint generation algorithms are designed based on expert heuristics, and the validity of parameter selection has not been systematically verified. In this study, we propose a novel representation format called Compacted Protocol Representation (CPR), which compactly preserves the structural characteristics of communication protocols and can also be used as a fingerprint. Additionally, we introduce Structural Edit Distance (SED), a metric that focuses on structural differences between CPRs. We apply CPR to TLS Client Hello messages to create Compacted TLS Client Hello (CTLS-CH) and analyze long-term trends in TLS parameter variations using a large-scale dataset spanning over a decade. Based on the observed changes in TLS parameters, we identify limitations in existing TLS fingerprinting techniques and demonstrate the effectiveness of our proposed approach.

Keywords: SSL/TLS, TLS Fingerprinting

1. はじめに

近年, Transport Layer Security (TLS) [1] の普及により, 悪性な活動を識別・検知することが困難になってきている。プライバシーやセキュリティ意識の高まりを受けて, TLS はインターネットで最も広く使われているプロトコルの一つとなった一方で, 攻撃者も TLS を活用して攻撃のステルス性を高めるようになった。2023年10月から2024年9月の間に Zscaler が遮断した攻撃通信のうち, TLS を用いていたものは 321 億件にのぼり, これは脅威全体の 87.2% を占めていたという [2]。通信が暗号化されると, ペイロード情報を前提とする技術は適用できず, 防御側は限られた情報から悪性/良性の判断をしなければならなくなる。

こういった背景から暗号化された通信を識別するための特徴を抽出し, 効果的に表現する技術の一つとして用いられているのが TLS フィンガープリンティングである。TLS フィンガープリンティングでは, TLS 通信の特徴を固有の識別情報 (フィンガープリント) として表現し, 既知フィンガープリントとの照合を通じて通信主体を識別する。

TLS フィンガープリンティングは特徴情報の収集方法により 2 つに大別される。能動的に通信を発生させて特徴情報を収集する手法がアクティブフィンガープリンティング, 受動的に通信を観測して特徴情報を収集する手法がパッシブフィンガープリンティングと呼ばれる。アクティブ TLS フィンガープリンティングは近年学術界で研究が活発化しているものの, 収集の容易さと情報収集に伴うリスクの低さから現在でも実運用の場面で広く用いられているのはパッシブ TLS フィンガープリンティングである。パッシブ TLS フィンガープリンティングでは, 主に TLS ハンドシェイクの Client Hello/Server Hello の可読領域からパラメータ情報を抽出し, 特徴とする。代表的なパッシブ TLS フィンガープリンティング技術としては JA3/JA3S[3] やその後継の JA4+[4], Mercury Network Protocol Fingerprinting (NPF)[5] などがある。

既存のパッシブ TLS フィンガープリントの生成アルゴリズムはその多くが専門家の経験則に基づいて設計されており, パラメータ選定やデータ加工方法の妥当性について体系的な検証が十分に為されたとは言えない。実際に, 既存研究では同じブラウザアプリケーションからコンテキストによって異なるフィンガープリントが生成される事例 [6]

や逆に同一フィンガープリントが複数のアプリケーションに紐づく事例 [7] が報告されている。

本研究では, 将来 TLS フィンガープリンティングに求められる課題を明らかにするために, TLS Client Hello のパラメータ分布や変遷, その変異量を定量的に解析し, TLS Client Hello におけるパラメータ変異の実態を明らかにすることを目的とする。変異解析を効率的に進めるために, 通信プロトコルのパラメータ構造をコンパクトに保持した表現形式 Compacted Protocol Representation (CPR) と, その構造的差異に着目した編集距離 Structural Edit Distance (SED) を設計した。CPR を TLS Client Hello に適用した Compacted TLS Client Hello (CTLS-CH) を用いて, 10 年以上に渡る悪性/良性混在の大規模 TLS 通信データセットを解析し, TLS パラメータ, 特に TLS1.2/TLS1.3 についてのパラメータ変異を明らかにした。本研究における主な貢献は以下の通りである。

- 通信プロトコルのパラメータ解析に適した新たな表現形式 CPR, パラメータ差分の精緻な算出に適した距離概念 SED を設計した
- TLS パラメータ変異を踏まえて適切なクラスタ粒度を明らかにした
- 持続性の観点から TLS パラメータクラスタの特性を明らかにした
- TLS1.2, TLS1.3 について, クラスタサイズやクラスタ数の観点からその特性の違いについて明らかにするとともに, クラスタ内の詳細なパラメータ差異について明らかにした

本論文ではまず第 2 章で関連技術・研究について述べ, 第 3 章で提案手法である CPR, SED の設計についてまとめる。第 4 章では調査対象となるデータセットについて説明し, 第 5 章で解析内容と考察, 今後の課題を整理したのちに第 6 章で全体を総括する。

2. 関連技術・研究

2.1 パッシブ TLS フィンガープリンティング

パッシング TLS フィンガープリンティングは, 自らは通信をせずに流れる TLS トランザクションから受動的にデータを収集し, フィンガープリントを生成する技術である。通信が発生するタイミングでセンサーが稼働している必要があるという制約条件がある一方で, 原理上はサーバ/クライアントいずれについてもフィンガープリントを生成できる。

Qualys 社 SSL Labs の Ivan Ristic が開発した sslhaf[8] は HTTP クライアント推定を目的としており, SSL/TLS のバージョン情報と暗号スイート情報, HTTP UserAgent 情報を特徴として利用していた。[9] は受動的なフィンガープリンティングツールである p0f に SSL/TLS のフィンガープリンティング機能を追加したもので, OS/アプリ

¹ 情報セキュリティ大学院大学
Graduate School of Information Security, Institute of Information Security

² NTT ドコモビジネス株式会社
NTT DOCOMO BUSINESS, Inc.

³ 香川大学
Kagawa University
a) dgs198101@iisec.ac.jp
b) atsushi.kanda@ntt.com
c) hashimoto.masaki@kagawa-u.ac.jp
d) okubo@iisec.ac.jp

ケーション推定を目的として、SSL/TLS のバージョン情報と暗号スイート情報の他に TLS の拡張パラメータやいくつかのヒューリスティックなフラグ情報を特徴情報に利用した。FingerprinTLS[10] も同様にアプリケーション推定を目的として開発されたフィンガープリンティングツールだが、圧縮方式や楕円曲線暗号のパラメータも特徴として収集している。Salesforce 社の Althouse らが開発した JA3/JA3S[3] はマルウェアや攻撃者環境の識別を目的としており、使用する情報は FingerprinTLS とほぼ同じであるが、VirulTotal[11] や Shodan[12]、Wireshark[13]、Suricata[14] など主要なツール、サービスに採用され、広く使われるようになった。例えば Abuse.ch が公開している SSLBL ではマルウェア通信の JA3 値が公開されている [15]。なお、現在 JA3 の開発は停止しており、後継のフィンガープリントスイートである JA4+[4] に移行している。Frolov らは検閲回避可能性を論じるために TLS フィンガープリンティング手法を提案した [6]。フィンガープリントには TLS のバージョン情報と暗号スイート情報の他にいくつかの拡張パラメータを選定してその拡張フィールド値も利用している。Cisco 社の Anderson らが開発した Joy[16], [17] は TLS や DNS, HTTP のクライアントプロセス推定を目的として開発された。Joy におけるフィンガープリント表現は Network Protocol Fingerprint(NPF) と名付けられており、他のフィンガープリント手法と異なり、生成時に Generate Random Extensions And Sustain Extensibility (GREASE)[18] の値も正規化して取り込んでいる点が特徴である。Joy はその後さらに取り込む拡張パラメータを増やし、その開発は Mercury[5], [19] に引き継がれた。

J4 のように現在もフィンガープリント単体で活用されるユースケースがある一方で、TLS フィンガープリンティングには異なるアプリケーションが同じフィンガープリントになる「衝突」の問題も指摘されている。Anderson らはマルウェアのものとされる 68 個の JA3 フィンガープリントが、Internet Explorer や Python, Java といった良性なプロセスにも紐づいていることを指摘している [7]。TLS フィンガープリントの欠点を補いつつ活用する方法として、Anderson らの研究 [7] や MVDet[20] のように TLS フィンガープリントに加えて他のコンテキスト情報を組み合わせて通信を識別する研究も登場している。

2.2 TLS パラメータの類似性

Frolov らは、TLS の正常利用時における TLS パラメータの変異をフィンガープリント化前の中間表現の段階でのレーベンシュタイン距離を使ってクラスタリングし、考察している [6]。また、彼らは調査の過程において、同一デバイスの Chrome から少なくとも 4 種類のフィンガープリントが生成されることを報告している。Anderson らは未

知のエンドポイントへの対応を目的として、完全一致するフィンガープリントが存在しなかった際にレーベンシュタイン距離を使って近傍探索し、次点となるフィンガープリントを紐づける手法を提案している [7]。いずれの研究においても、主たる目的はあくまでもフィンガープリントの完全一致判定であり、補助的に類似性を用いるに留まっている。

また、既存のフィンガープリンティング手法はフィンガープリント間のパラメータを比較するのに適した形式とは言えない。JA3/JA4 は暗号学的ハッシュによる変換処理をしているため、元のパラメータを復元することが困難である。NPF はパラメータの復元が可能なフォーマット設計であるが、データ長などのメタデータを含めたまま拡張パラメータのフィールド値を取り込んでおり、1 つのパラメータ変化がフィンガープリントの複数箇所に波及してしまう。

3. 提案手法

3.1 Compacted Protocol Representation (CPR)

Compacted Protocol Representation (CPR) は通信プロトコルのパラメータを構造的に分析、利用するためのデータ表現形式である。キーとなるアイディアは通信プロトコルのデータ構造をそのまま Concise Binary Object Representation (CBOR) [21] で表現するところにある。CBOR は JSON のバイナリ版とも言えるデータ表現形式で、RFC 8949 として標準化されており、「極めて小さいコードサイズで処理を実装でき」、「小さなメッセージサイズで表現でき」、「バージョンネゴシエーションをせずとも拡張性を備えている」ことを目標として設計された。その柔軟性、拡張性からさまざまな派生仕様が策定されており、軽量なセキュリティトークンである CBOR Web Token (CWT)[22] や軽量な DNS パケットキャプチャフォーマットである C-DNS[23] など、さまざまな用途に CBOR が使われている。

CPR は類似技術である Mercury NPF[5] と同じく、さまざまな通信プロトコルに対応可能な汎用性をコンセプトとしながらも次の点で NPF とは異なる特徴を持つ。

- 標準化技術を用いることによる実装の容易さ
- 可読性要件を分離することによる軽量化
- JSON と同等の表現能力

標準化技術を用いることによる実装の容易さ

CPR は、そのベースとなる CBOR 自体が RFC 標準化された技術であり、JavaScript をはじめとして PHP, Python, Ruby, C などさまざまなプログラミング言語ですべてライブラリが実装され公開されているため、それらを活用することで簡単に導入できる。一方で NPF は括弧で区切られた 16 進数表記を用いた独自の表現形式で定義されており、その仕様は公開されてはいるもののリファレンス実装

は提案者である Cisco 社が公開しているツールしかなく、導入障壁は高い。

また、CBOR は JSON との互換性を意識して設計されており、JSON との相互変換が可能であるため、CPR データは JSON データとしても扱うことができる。つまり JSON フォーマットで元となるデータを構築したのちに CPR にエンコードしたり、CPR をそのデータ構造を保持したまま JSON フォーマットに変換することで JSON データとして分析したりすることが可能であり、データの加工・分析が容易である。

さらに、CPR は入出力時のデータ検証も実装しやすい。CBOR のデータ構造を定義するための記述言語である Concise Data Definition Language (CDDL)[24] が標準化されており、CDDL に基づくデータ検証のライブラリ実装も存在するため、堅牢なアプリケーションの開発にかかるコストを下げられる。

可読性要件を分離することによる軽量化

CPR は省サイズを志向したバイナリ表現形式である CBOR を採用することでデータサイズの観点で情報を効率的に表現できる。一方で NPF はそれ自体をフィンガープリントとして配布し、利用することを想定しており、バイナリデータを 16 進数表記し、全て可読文字列のみで表現する設計となっている。そのため、同じ量の情報を表現した際に、元々の情報量が極端に少ない場合を除いて CPR の方が NPF よりもデータサイズを抑えられる。

データ流通を考えた際に可搬性の観点で CPR がバイナリデータであることがネックとなることも考えられるが、その場合は Base64 などでエンコードしてやることで可読文字列として流通させることも可能である。

JSON と同等の表現能力

先にも述べた通り、CPR は JSON フォーマットと互換性があるため、JSON で表現可能な配列やキーバリューを組み合わせた複雑なデータ構造を表現でき、データスキーマがなくともデコードできる。NPF は括弧で閉じられた文字列を組み合わせることで木構造データを表現しているが、そのままではキーバリューは表現できない。木構造のリーフに格納されたデータにキーバリューの意味合いを持たせることは出来なくはないが、現状のデータスキーマではそれには対応していない。

3.2 Compacted TLS Client Hello (CTLS-CH)

本研究において、クライアント TLS フィンガープリントの主要な情報源である Client Hello メッセージを CPR で表現した Compacted TLS Client Hello (CTLS-CH) を用いてデータの分析を行なった。CTLS-CH の初期的実装として今回は既存研究である NPF の tls/2 と同じパラメータデータを収集し、同じ前処理を行なった上で CPR として表現した。CTLS-CH のフォーマットは次の通りである

(CDDL 定義は A.1 を参照)。

```
[ legacy_version,  
  normalized_cipher_suites,  
  sorted_normalized_extensions ]
```

CTLS-CH は 3 つの要素の配列からなる。これらは Random フィールドや Session ID フィールドといったセッション毎に値の変わるフィールドを除いて Client Hello メッセージに登場するフィールドの順序がそのまま保持されている。

legacy_version は TLS バージョンフィールドの値である。TLS1.2 以前はクライアントがサポートしている最大のバージョン番号を表すパラメータであったが、TLS1.3 では 0x0303 (TLS1.2 のバージョン番号) で固定とされることになった。

normalized_cipher_suites は、GREASE[18] の値を正規化した暗号スイートのリストである。GREASE の値は GREASE 専用に予約された値リストの中から毎回ランダムに設定されるため、そのままではパラメータの値が揺らいでしまう。そのため、GREASE の値を 0x0a0a に置換する正規化処理を施す。

sorted_normalized_extensions は、GREASE 値を正規化し、拡張パラメータのタイプ値でソートしたキーバリューのペアである。順序をソートする理由は extension permutation による影響の抑止のためである。2022 年に Chrome は潜在的な脆弱性への対策として Client Hello 拡張パラメータの順序を無作為に並び替える extension permutation を実装し [25]、Mozilla NSS もこれに追従した [26]。extension permutation によるフィンガープリントのばらつきを防ぐため、JA4 や NPF tls/1, NPF tls/2 といった比較的新しい方式では拡張パラメータの順序をソートした上でフィンガープリントを生成している。なお、各拡張パラメータフィールドの値の中にはセッション毎に値が変わるものがあるため、NPF と同様に特定の拡張パラメータに限りそのフィールド値を取り込み、それ以外の拡張パラメータについては null 値をセットする（詳細は A.2 を参照）。NPF と異なる点として、CTLS-CH では取り込むフィールド値はバイナリ形式のまま格納し、リストの形態をとるフィールド値については配列化して取り込む。これにより、フィールド値の変化をより精緻に分析することが可能となる。

3.3 Structural Edit Distance (SED)

TLS パラメータの構造的差分に着目した距離の概念として Structural Edit Distance (SED) を導入する。既存研究では、レーベンシュタイン距離を用いて二つのフィンガープリント間の距離を定義したもののがあるが [6], [7]、その具体的な計算方法について明らかにされておらず不明瞭さが残っている。本研究ではその不明瞭さを解消し、手法の再現性を高めるために SED を定義する。

SED は「CBOR データ A から CBOR データ B に変換する際の編集操作 (add/remove/replace/move) の回数」と定義される。ここで言う編集操作とは具体的には JSON データにおける JSON Patch[27] の編集操作に相当するものである。JSON Patch では変換のためのパッチが編集操作のシーケンスとして表現されるため、この考え方を CBOR に転用し、その操作回数を距離として考える。「移動 (move)」操作を許容していることからレーベンシュタイン距離ではなく、Cormode らが提案した編集距離 [28] を構造的データに拡張した考え方につい。

既存研究では暗号スイートの増減や拡張パラメータの増減のみに着目していたが、CTLS-CH の SED 値を測ることでリスト形式の拡張パラメータフィールド値の細かい差異まで捉えることができるようになる。例えば、二つの CTLS-CH について拡張パラメータの padding の有無のみが差異であった場合、その SED 値は 1 となる。同様に拡張パラメータの supported_version で指定されるバージョン一覧で TLS1.1(0x0302) と TLS1.0(0x0301) の有無のみが差異であった場合、その SED 値は 2 となるが、NPF での差異がどう評価されるかは明確には示されていない。

4. データセット

本研究では、調査対象のデータセットとして Malware Traffic Analysis (MTA) [29] を使用した。MTA は Palo Alto Networks Unit 42 のアナリスト Brad Duncan が運営するサービスで、マルウェアトラフィックに関するデータが公開・共有されている。マルウェア TLS 通信に関する既存研究でもたびたび利用され、よく知られたデータセットである [30], [31], [32], [33]。Malware Traffic Analysis はそのサービスの性質上、マルウェア検体の収集しやすさやアナリスト本人の嗜好によるデータセットの偏りや網羅性の欠如が考えられるが、一方で、トレンドに合わせてタイムリーに解析されたデータが 10 年以上の長期に渡り蓄積された稀有なオープンデータであるとも言える。なお、MTA データセットはサンドボックス環境でマルウェアに感染させるなどして収集されたものであり、収集された pcap には感染には直接関係のない良性な通信も含まれている。本研究では、TLS の使われ方を経年で追跡し、その TLS パラメータの変異・変遷を明らかにすることを目的としている。将来的に悪性通信識別を一つのユースケースとして考えているため、正常通信と悪性通信が混在し、長期間に渡ってデータが公開されている MTA を採用した。

データの収集対象期間は MTA がサービス開始した 2013 年 6 月から 2024 年 12 月までのおよそ 11.5 年間で、TLS Client Hello が含まれる 1,436 の pcap ファイルから 155,047 の TLS ClientHello を収集した。全期間を通して収集できたユニークな CTLS-CH は 371 種類であった。その内訳は表 1 の通りである。

表 1 ユニーク CTLS-CH 数 (MTA データセット)

Table 1 Unique count of CTLS-CH (MTA dataset)

SSL/TLS version	CTLS-CH count
SSL(v3.0)	2
TLS1.0	40
TLS1.1	2
TLS1.2	202
TLS1.3 (draft)	6
TLS1.3	119

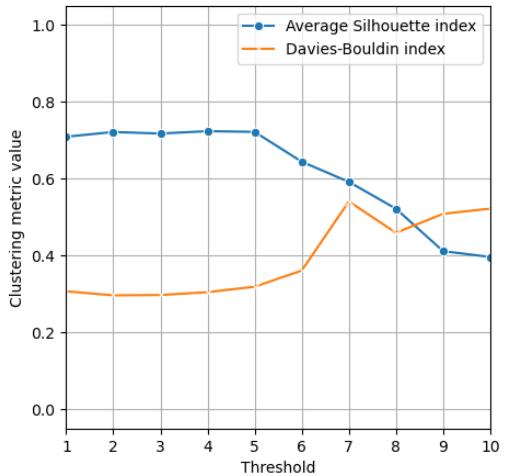


図 1 クラスタリング性能指標

Fig. 1 Clustering metrics

5. TLS Client Hello のパラメータ変異解析

5.1 編集距離に基づくクラスタリング

先行研究 [6] にならい、TLS パラメータの編集距離 (SED) に基づいて TLS パラメータ群をクラスタリングする際の最適なクラスタリングしきい値を考える。

クラスタの均質性の観点からは、異なる TLS バージョンが同じクラスタに含まれないことが望ましい。データセット中で 2021 年以降の TLS 通信の 99% 以上が TLS1.2 あるいは TLS1.3 であった点、今回収集できた CTLS-CH の 86.5% が TLS1.2 あるいは TLS1.3 であった点を踏まえて、TLS1.2 と TLS1.3 の CTLS-CH が同一クラスタに入りはじめる最短距離を調べたところ、その SED 値は 6 であった。このことからクラスタリングしきい値は 5 以下であることが望ましいと考えられる。

次にクラスタリング性能の観点から最適なしきい値を考える。よいクラスタリングはクラスタ内部がなるべく密集していて、かつ他のクラスタとは離れている。そこで代表的なクラスタ性能指標であるシルエット指数 (Silhouette index) とデイビス・ボールディン指数 (Davies-Bouldin index) を算出した (図 1)。シルエット指数は値が高いほど性能がよく、デイビス・ボールディン指数は値が低いほど性能がよい。いずれの指標もしきい値 6 を超えると性能

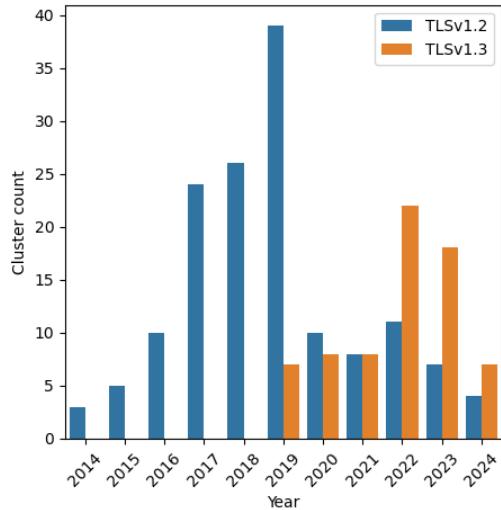


図 2 クラスタ数の推移
Fig. 2 Number of cluster

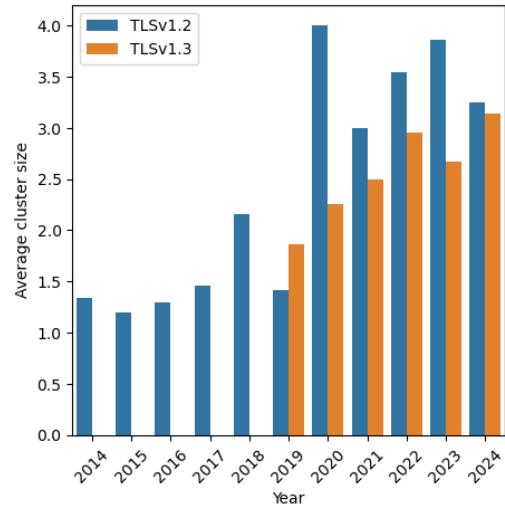


図 3 平均クラスタサイズの推移
Fig. 3 Average cluster size

が劣化することが明らかとなった。しきい値 5 以下の範囲では軽微な差ではあったが、シルエット指数、ディビス・ボールディン指数の最良値はそれぞれしきい値 4 のときに 0.724、しきい値 2 のときに 0.296 であった。さらにデータを TLS1.3 が本格的に使われ始めた 2020 年以降に限定した場合、どちらの指標もしきい値 2 のときに最良値が得られた。

以上を総合して、本データセットにおいて最適なクラスタリングの粒度はしきい値 2 と考えられる。

5.2 クラスタの経年変化

クラスタが経年でどのように変化するかを分析する。なお、5.1 節の結果に基づき、クラスタリングしきい値は 2 を採用した。

まずクラスタの持続性について分析する。先行研究 [6] や JA4[4] では、ソフトウェアアップデートなどの要因によって次々と新しい TLS フィンガープリントが観測されると報告している。そこで本データセットにおいてそれぞれの TLS パラメータクラスタの寿命を算出した。ここでクラスタの寿命とはそのクラスタが観測された最も古い年と最も新しい年の差分である。その結果、138 個のクラスタのうち 88 個 (63.8%) の寿命は 0 年、すなわち特定の年のみで観測されたものであることがわかった。クラスタ単位で見ても TLS パラメータはその移り変わりが早いことが見て取れる。

続いて TLS1.2 と TLS1.3 のクラスタの性質が経年でどのように変化しているかを調査した。TLS1.2 と TLS1.3 を比較すると、2022 年以降、TLS1.3 の方がクラスタ数が多く、平均クラスタサイズは小さい。TLS1.2, TLS1.3 とも 2022 年から 2024 年にかけてクラスタ数が減少しているが、この期間は TLS パケット数自体も経年で減少しており、そ

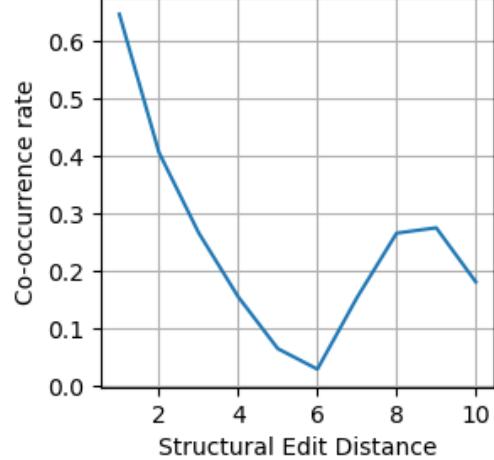


図 4 変異ペアの共起率
Fig. 4 Co-occurrence rate

の影響の可能性も考えられるため、さらなる調査が必要である。一方で、TLS パケットが減少している中でも 2024 年の TLS1.3 の平均クラスタサイズは過去最大値を記録しており、今後 TLS1.3 クラスタ内の変異バリエーションが増える可能性も考えられる。

5.3 パラメータ変異の共起性

TLS パラメータ変異が時間的に同じ時期に観測されているかを分析する。ある TLS パラメータ変異ペアが同じ一連の通信の中で現れる場合、同一通信主体が複数の TLS フィンガープリントを発生させている、すなわち TLS パラメータが揺らいでいる可能性がある。そこで、編集距離 (SED) に応じて変異の共起率を調べた (図 4)。ここで共起率は当該編集距離にある変異ペア全体のうちで同じ pcap ファイル内で変異が観測された変異ペアの数の比率である。

TLS1.2 と TLS1.3 が同じクラスタに混在しはじめる SED

表 2 SED 値 2 以内のパラメータ差分

Table 2 Parameter differences within SED 2

TLS parameter difference	TLS1.2	TLS1.3
status_request	✓	✓
application_layer_protocol_negotiation	✓	✓
server_name	✓	✓
token_binding	✓	
暗号スイート（リスト値）	✓	✓
padding	✓	✓
application_layer_protocol_negotiation （リスト値）	✓	✓
session_ticket	✓	✓
signature_algorithms（リスト値）	✓	✓
pre_shared_key		✓
encrypted_client_hello		✓
application_settings		✓
supported_groups（リスト値）		✓
supported_versions（リスト値）		✓
ec_point_formats		✓
delegated_credentials		✓
psk_key_exchange_modes		✓
post_handshake_auth		✓
early_data		✓
cookie		✓

値 6 よりも近い区間においては、距離が近づけば近づくほど共起率は上昇する。SED 値 6 以上の区間においても共起率が 0.4 を超えることはなく、特に SED 値 2 以下の区間に同一通信主体によるパラメータ変異（揺らぎ）が存在していることが示唆される。

次に同一クラスタ内での具体的なパラメータ差分に着目する。特に近年の TLS1.2, TLS1.3 に関する変異について詳細に解析するために 2020 年以降の TLS1.2, TLS1.3 に限定して、差分となっているパラメータをリストアップした（表 2）。表中で「リスト値」と表記されたパラメータは、「そのパラメータ自体はどちらにも存在するが、フィールド値のリストに差分があったもの」である。差分のあったパラメータはその多くが仕様上、同一通信主体でもコンテキストによって変動しうるものであった。例えば status_request は OSCP Stapling で利用される拡張パラメータで、同一のアプリケーション、同一対地であっても全てのセッションで付与されるものではない。これはマルウェアについても同様で、MTA データセットにおいても IcedID や Cobalt Strike について同一感染事象の中で status_request に有無によるパラメータの揺らぎが確認されている。

変異パラメータのバリエーションを見ると、TLS1.2 よりも TLS1.3 の方がバリエーションが豊富である。pre_shared_key や early_data など TLS1.3 から実装された拡張パラメータも含まれているほか、encrypted_client_hello や delegated_credentials など近年標準化された拡張パラメータや現在標準化作業中の拡張パラメータも含まれてい

る。潜在的には今後 TLS1.3 はさらに変異のバリエーションが増えることが考えられる。

5.4 TLS フィンガープリンティングの今後の課題

既存の TLS フィンガープリンティングではパラメータ変異に対してそれぞれ異なるフィンガープリント値が生成されるため、フィンガープリントを通信識別に活用するようなユースケースでは、通信コンテキストのちょっとした変化に追従できず見逃しのリスクが高まる危険性がある。Mercury[5] では最近傍のフィンガープリントを検索する機構が組み込まれているが、あくまでも完全一致するフィンガープリントが存在しなかった場合の次善的な挙動であるため、完全一致するフィンガープリントが存在する場合の見逃しリスクは残る。

6. おわりに

本論文では、10 年以上に渡る悪性・良性通信データから TLS Client Hello を解析し、パラメータ変異の実態を明らかにした。詳細な解析を実現するために CBOR ベースの表現形式 CPR と構造的な編集距離 SED を導入し、それらを用いて TLS Client Hello パラメータセットをクラスタリングし、クラスタが短命であること、特に距離 2 以内の範囲において共起するパラメータ変異（揺らぎ）が存在することを明らかにした。

今回明らかとなったパラメータ揺らぎの特性を踏まえて揺らぎ耐性のある TLS フィンガープリンティング手法を開発することが今後の課題である。

参考文献

- [1] Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446 (2018).
- [2] ThreatLabz: ThreatLabz 2024 Encrypted Attacks Report, Technical report, Zscaler, Inc. (2024).
- [3] John Althouse, J. A. and Atkins, J.: JA3 - A method for profiling SSL/TLS Clients, Salesforce, Inc. (online), available from <https://github.com/salesforce/ja3> (accessed 2025-08-20).
- [4] Althouse, J.: JA4+ network fingerprinting, FoxIO (online), available from <https://github.com/FoxIO-LLC/ja4> (accessed 2025-08-20).
- [5] Cisco: Mercury, , available from <https://github.com/cisco/mercury> (accessed 2025-08-20).
- [6] Frolov, S. and Wustrow, E.: The use of TLS in Censorship Circumvention., NDSS (2019).
- [7] Anderson, B. and McGrew, D.: Accurate TLS Fingerprinting using Destination Context and Knowledge Bases (2020).
- [8] Qualys SSL Labs: HTTP Client Fingerprinting Using SSL Handshake Analysis, , available from <https://www.ssllabs.com/projects/client-fingerprinting/> (accessed 2025-08-20).
- [9] Marek: SSL fingerprinting for p0f, , available from <https://idea.popcount.org/2012-06-17-ssl-fingerprinting-for-p0f/> (accessed 2025-08-20).

- [10] Brotherston, L.: TLS fingerprinting, , available from <<https://github.com/LeeBrotherston/tls-fingerprinting>> (accessed 2025-08-20).
- [11] Google: VirusTotal, , available from <<https://www.virustotal.com/>> (accessed 2025-08-20).
- [12] Shodan: Shodan, , available from <<https://www.shodan.io/>> (accessed 2025-08-20).
- [13] Wireshark Foundation: Wireshark, , available from <<https://www.wireshark.org/>> (accessed 2025-08-20).
- [14] Open Information Security Foundation: Suricata, , available from <<https://suricata.io/>> (accessed 2025-08-20).
- [15] SSLBL: JA3 Fingerprints, , available from <<https://sslbl.abuse.ch/ja3-fingerprints/>> (accessed 2025-08-20).
- [16] Anderson, B.: The Generation and Use of TLS Fingerprints, *FloCon* (2019).
- [17] Cisco: Joy, , available from <<https://github.com/cisco/joy>> (accessed 2025-03-04).
- [18] Benjamin, D.: Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility, RFC 8701 (2020).
- [19] McGrew, D.: Bayes at 10+ Gbps: Identifying Malicious and Vulnerable Processes from Passive Traffic Fingerprinting, *FloCon* (2020).
- [20] Cui, S., Han, X., Dong, C., Li, Y., Liu, S., Lu, Z. and Liu, Y.: MVDet: Encrypted malware traffic detection via multi-view analysis, *Journal of Computer Security*, Vol. 32, No. 6, pp. 533–555 (2024).
- [21] Bormann, C. and Hoffman, P. E.: Concise Binary Object Representation (CBOR), RFC 8949 (2020).
- [22] Jones, M. B., Wahlstrom, E., Erdtman, S. and Tschofenig, H.: CBOR Web Token (CWT), RFC 8392 (2018).
- [23] Dickinson, J., Hague, J., Dickinson, S., Manderson, T. and Bond, J.: Compacted-DNS (C-DNS): A Format for DNS Packet Capture, RFC 8618 (2019).
- [24] Birkholz, H., Vigano, C. and Bormann, C.: Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures, RFC 8610 (2019).
- [25] Benjamin, D. and Adrian, D.: Feature: TLS ClientHello extension permutation, , available from <<https://chromestatus.com/feature/5124606246518784>> (accessed 2025-08-20).
- [26] Schwarz, L. and Jackson, D.: Add Option for Randomizing TLS Client Hello Extension Order, , available from <https://bugzilla.mozilla.org/show_bug.cgi?id=1789436> (accessed 2025-08-20).
- [27] Bryan, P. C. and Nottingham, M.: JavaScript Object Notation (JSON) Patch, RFC 6902 (2013).
- [28] Cormode, G. and Muthukrishnan, S.: The string edit distance matching problem with moves, *ACM Trans. Algorithms*, Vol. 3, No. 1 (online), DOI: 10.1145/1186810.1186812 (2007).
- [29] Duncan, B.: Malware Traffic Analysis, , available from <<https://www.malware-traffic-analysis.net/>> (accessed 2025-03-04).
- [30] Moussaileb, R., Cuppens, N., Lanet, J.-L. and Le Bouder, H.: Ransomware network traffic analysis for pre-encryption alert, *Foundations and Practice of Security: 12th International Symposium, FPS 2019, Toulouse, France, November 5–7, 2019, Revised Selected Papers 12*, Springer, pp. 20–38 (2020).
- [31] Almousa, M., Osawere, J. and Anwar, M.: Identification of ransomware families by analyzing network traffic using machine learning techniques, *2021 Third international conference on transdisciplinary AI (TransAI)*, IEEE, pp. 19–24 (2021).
- [32] Guo, J., Sang, Y., Chang, P., Xu, X. and Zhang, Y.: MGEL: a robust malware encrypted traffic detection method based on ensemble learning with multi-grained features, *International Conference on Computational Science*, Springer, pp. 195–208 (2021).
- [33] Barradas, D., Novo, C., Portela, B., Romeiro, S. and Santos, N.: Extending C2 Traffic Detection Methodologies: From TLS 1.2 to TLS 1.3-enabled Malware, *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses, RAID '24*, Association for Computing Machinery, p. 181–196 (2024).

付 錄

A.1 CTLS-CH の CDDL 定義

```
compacted_tls_client_hello = [
    legacy_version: uint .size 2,
    normalized_cipher_suites:
        [+ cipher_suite: uint .size 2],
    sorted_normalized_extensions:
        {* extension_type => extension_data }
]
extension_type = uint .size 2
extension_data = bstr / [+ bstr] / null
```

A.2 フィールド値を取り込む拡張パラメータ

表 A.1 フィールド値を取り込む TLS 拡張パラメータ

Table A.1 TLS Extension with field data

Extension type	value	Extension name
0x0001		max_fragment_length
0x0005		status_request
0x0007		client_authz
0x0008		server_authz
0x0009		cert_type
0x000a		supported_groups
0x000b		ec_point_formats
0x000d		signature_algorithms
0x000f		heartbeat
0x0010		application_layer_protocol_negotiation
0x0011		status_request_v2
0x0018		token_binding
0x001b		compress_certificate
0x001c		record_size_limit
0x002b		supported_versions
0x002d		psk_key_exchange_modes
0x0032		signature_algorithms_cert
0x5500		channel_id