Computer Security Symposium 2019
21 - 24 October 2019

乱数性を用いたTLS通信の識別

神田 敦[1,2,a),b)]　橋本 正樹[1]

概要：現在，HTTPS をはじめとしてインターネット通信の暗号化が急速に普及してきている．しかし，暗号化通信技術の普及によりユーザ通信のセキュリティが向上する反面，通信から得られる情報が減ることによって正常な通信と攻撃者の通信の判別がつかなくなる懸念がある．SSL/TLS に関しては既存技術としてヘッダやハンドシェイクパラメータなど，表層的な情報を元にサーバ/クライアントを推定する TLS フィンガープリンティングがあるが，パラメータのランダム化やハンドシェイクの改変などにより検知回避をする攻撃が既に観測されている．本研究は偽装耐性のある通信識別技術の確立を目的として，暗号化通信の乱数性に着目し，乱数の持つ統計的特徴を用いて暗号化アルゴリズムや暗号ライブラリを推定する手法を提案する．HTTPS を対象として実施した実験では，TLS 通信の暗号化されたアプリケーションデータのみから鍵長を考慮しない暗号化アルゴリズム推定で 89.6%の精度を実現した．

キーワード：ネットワークセキュリティ，SSL/TLS，乱数検定，機械学習

Identifying TLS Communication Using Randomness

Atsushi KANDA[1,2,a),b)]　Masaki HASHIMOTO[1]

Abstract: The use of encryption in the Internet communication is rapidly spreading. While this trend will improve user's security, it'll make us difficult to distinguish between benign and malicious communication due to lack of information obtained from the communication itself. TLS Fingerprinting is one of the solutions. It uses surface information, like headers and handshake parameters, to identify Server/Client. However, attacks that try to evade such detections by randomizing parameters or modifying handshakes have already been observed. Our goal is to identify encrypted communication in a more robust manner against these kinds of evasions. We focused on the randomness of encrypted communication, and we propose a method that identifies encryption algorithm and/or encryption library by using statistical features of random number. Our experiment shows that by using only the encrypted application data from TLS communication, we can identify encryption algorithm (without considering the key length) with accuracy of 89.6%.

Keywords: Network Security, SSL/TLS, Randomness Test, Machine Learning

1. はじめに

現在，インターネットで急速に通信の暗号化が普及しつつある．Google 社によれば 2019 年 7 月現在，Google Chrome によるウェブアクセスの 70%以上が HTTPS によるもので，閲覧時間ベースでは 85%以上のウェブトラフィックがすでに HTTPS となっている [1]．また，DNS においてもプライバシー保護を目的として DNS over TLS(DoT) や DNS over DTLS[2]，DNS over HTTPS(DoH)[3] といった暗号化方式が標準化されている．

これら暗号化通信技術の普及によりユーザの通信の安全性が向上する反面，攻撃者が暗号化通信を悪用することでセキュリティが脅かされる懸念がある．すなわち，昨今の暗号化通信技術の普及が防御側にとって不利になる要素としては以下の 2 点が挙げられる．

[1] 情報セキュリティ大学院大学
Institute of Information Security
[2] NTT コミュニケーションズ株式会社
NTT Communications Corporation
[a)] mgs178501@iisec.ac.jp
[b)] atsushi.kanda@ntt.com

既存セキュリティソリューションが使えなくなる

ネットワーク上で稼働するセキュリティアプライアンスが従来の機能を果たせなくなる懸念がある．例えば，WAF(Web Applicaion Firewall) はウェブ通信の挙動や内容に応じて危険な通信を遮断，隔離するが，HTTPS 化するとコンテンツを見ることができなくなり，URL やコンテンツ内容にマッチさせたフィルタが効かなくなる可能性がある．ドメインベースのフィルタリングも DoT や DoH を使用した場合にはクエリの内容を見ることができなくなり，ドメイン名やクエリの内容に応じたフィルタリングができなくなる．

対応策として，SSL/TLS 通信を一度終端し，プロキシすることでセキュリティを維持するソリューションもあるが，暗号化・復号処理のためにマシンパワーが必要となり，大規模に展開するとコスト高になる傾向がある．また，データを復号する行為がそもそもユーザが知られたくないと考えている情報に触れる可能性もあり，プライバシーの観点からも安易に復号することは好ましくない．エンドポイント端末側でセキュリティの機能を実現する EDR（Endpoint Detection and Response）製品で暗号化される前の情報を活用することは解決策の 1 つである．しかし，全ての機器にそういったソリューションを展開するのは難しく，特に IoT(Internet of Things) に分類されるような機器は一般的にコンピューティングリソースの制約からその機能は限定的にならざるを得ない．

悪性通信と良性通信の判別がつかなくなる

通信が暗号化されると，可読領域が減るため，通信経路上で得られる情報が激減する．現状，長期間潜伏し活動するような高度な攻撃においては，通信をはっきりと悪性と断定できないことが多く，周辺情報とも紐付けながら総合的に対応を判断する必要があるが，通常のユーザの正常（良性）な通信が暗号化されるようになると，判断材料が限定され，悪性通信と良性通信の判別がより一層困難になる．

SSL/TLS 通信においても，ヘッダやハンドシェイクの情報等から表層的に得られる情報があり，そういった情報からサーバ/クライアントの情報を推定する技術として TLS フィンガープリンティングがある [4], [5], [6]．TLS フィンガープリンティングはヘッダやハンドシェイクの情報を端末を特徴づけるフィンガープリント情報としてデータベースに蓄積し，そのデータベースを参照することで端末を推定するものである．これらはサービス提供者がユーザを追跡する用途で使われることもあるし，マルウェア通信の識別に使われることもある．実際に Althouse らは JA3/JA3S と名付けたフィンガープリント情報を用いてマルウェア通信が識別できたとしている [6].

しかし，現行の TLS フィンガープリンティングを防御ソリューションに活用する際の課題は表層的な情報を扱っているがゆえに，偽装が容易であるという点である．Akamai 社によれば 2018 年 9 月頃から検知回避のために TLS のパラメータネゴシエーションをランダムにする通信が観測され始め，2018 年 10 月から 2019 年 2 月の間に約 20%も増加したという [7]．また検知回避のために SSL を改変して利用しているマルウェアも既に報告されている．具体な実例として，BADCALL[8] は，TLS のハンドシェイクをしているように見せかけて独自実装の方式で暗号化データをやりとりする挙動をする．

今後 TLS フィンガープリンティングが普及すればするほど，攻撃者も検知を逃れるべく対策してくることは容易に想像できる．そのため，将来的には偽装耐性のある通信識別技術が求められてくると考えられる．

本研究は，偽装耐性のある暗号化通信識別技術の確立を目的とし，暗号化されたデータ自体が持つ特徴の活用を模索するものである．通信の識別技術には TLS フィンガープリンティングのように表層的な情報を特徴量するものやパケットサイズや通信間隔などのトラフィック特性を特徴量にする技術などがあるが，本研究はそれらの代替になるものではなく，それら技術を補完するものとして位置付ける．また，暗号化されたデータを扱うが，プライバシー保護の観点からデータを復号化することは目的としない．

本稿においてはその第一歩として，正常な通信でも多く使われ，攻撃者も使うことの多い SSL/TLS について，暗号化されたアプリケーションデータ部から特徴を抽出し，TLS のバージョンや暗号スイートの識別を行った結果について報告する．

提案手法の貢献を以下に示す．

- TLS の暗号スイート推定に, 特に Frequency (Monobit) Test と Runs Test の特徴が有用であること示した．
- Frequency (Monobit) Test と Runs Test の特徴を用いた識別手法を提案し，48.3%の精度で暗号スイートの推定が可能であることを示した（TLS バージョン推定であれば 100%，暗号化アルゴリズム推定であれば 89.6%まで精度向上）．

本稿の構成は以下の通りである．まず第 2 章で本研究の研究背景について述べ，第 3 章で特徴量の選定にあたり実施した調査実験について説明する．第 4 章では選定した特徴量の評価実験について報告し，第 5 章で実験結果について考察するとともに今後の課題と得られた知見について整理する．最後に，第 6 で関連研究について触れ，第 7 章にて本稿を総括する．

2. 研究背景

2.1 暗号技術と乱数性

暗号化されたデータは究極的には第三者から見たときに乱数にしか見えないことが理想である．本研究では乱数が持つ性質を「乱数性 (Randomness)」と称することとする．乱数性の例としては，等出現性（値が等しい頻度で出現する性質）や無規則性（値の出現が過去の値と独立しており法則性がない性質）などが挙げられる．乱数性は暗号技術の性能を計る 1 つの指標であり，乱数性の高いデータを生成する暗号技術はよい暗号技術と言える．事実，技術標準に関する研究機関である米国の国立標準技術研究所（NIST: National Institute of Standards and Technology）では DES に代わる新たな暗号 AES を公募した際に暗号の性能評価に後述の NIST SP800-22 で定義されている検定手法を用いた [9].

2.2 乱数検定

乱数検定とは，生成されるデータの乱数性から乱数生成器の合格・不合格を判定することである．代表的な乱数検定の手法としては NIST SP800-22[10] や Dieharder[11], TestU01[12] などが挙げられる．これらの検定では複数の統計的検定を組み合わせて総合的に乱数性を判定する．

2.3 NIST SP800-22

NIST SP（Special Publication）は，NIST が公開している文書である．その中でも SP800 シリーズはセキュリティに関する文書で SP800-22 は乱数の検定手法を規定しており，検定手法を実装したツールも同時に公開している．

NIST SP800-22 ではビット列を対象として全部で 15 種類の統計的検定手法を定義している．各検定では理想的な真の乱数を仮定したときにそのビット列が生成される確率である p 値を算出し，有意水準（1%）を上回った場合に乱数性があると判断する．ただし，個々の検定が着目している乱数の性質は異るため，複数の検定を組み合わせ，なおかつ 1000 以上の標本に対して検定を実施して総合的に判断することを推奨している．

紙面の都合上，以降，本研究にて候補として採用した 7 つの検定手法について概要のみ説明する．

2.3.1 Frequency (Monobit) Test

Frequency (Monobit) Test は対象ビット列における 0 と 1 の出現頻度を元に乱数性を評価する．

2.3.2 Frequency Test within a Block

Frequency Test within a Block は対象ビット列を長さ M ビットごとの N 個のブロックに分割し，各ブロックの 1 の出現頻度を元に乱数性を評価する．

2.3.3 Runs Test

Runs Test は対象ビット列中の連（同じ値が連続してつながったかたまり）の個数を元に乱数性を評価する．長さ k の連とは，同じ値が k 個連続していて，かつその前後が異なる値をとる状態を指す．なお，NIST SP800-22 にて規定されている Runs Test では，事前に頻度検定を行い，頻度検定を合格した場合にのみ連検定を実施する．

2.3.4 Test for the Longest Run of Ones in a Block

Test for the Longest Run of Ones in a Block は対象ビット列を長さ M ビットごとの N 個のブロックに分割し，各ブロックにおける連の最長値（最長連）を元に乱数性を評価する．

2.3.5 Serial Test

Serial Test は対象ビット列について，全ての m ビットパターンの出現頻度を元に乱数性を評価する．

2.3.6 Approximate Entropy Test

Approximate Entropy Test は対象ビット列について，全ての m ビットパターンの出現頻度を元に乱数性を評価する．m ビットパターンの出現頻度をベースにする点は Serial Test と同じだが，Approximate Entropy Test ではエントロピー（平均情報量）の近似値を用いる点が異なる．

2.3.7 Cumulative Sums (Cusum) Test

Cumulative Sums (Cusum) Test は対象ビット列のうち 0 を -1 に置換した数値列に対して，前方あるいは後方から順次加算していった際の最大絶対値を元に乱数性を評価する．

3. 手法検討

3.1 実験設計

TLS 暗号スイート識別に適用可能な特徴量を検討するために，TLS 通信のトラフィックデータを収集し，調査実験を行った．以下，実験の設計について述べる．

3.1.1 データセット

実験は最も普及しているアプリケーションの 1 つである HTTPS を対象に，今後も利用が見込まれる TLS v1.2 と TLS v1.3 に限定した．実験では，Docker を用いて同一端末内で HTTPS サーバと HTTPS クライアント，およびトラフィックキャプチャを稼働させることでデータを収集した．いずれも x86(64bit) の Alpine Linux をベースとしており，SSL/TLS ライブラリによる違いの検証も視野に入れて，HTTPS クライアントについては SSL/TLS ライブラリの異なる 2 種類のクライアントを作成した．それぞれの諸元については**表 1** の通り．

表 1 の構成を用いて，HTTPS アクセスが成立した計 13 種類の暗号スイートについて，それぞれの HTTPS クライアントで各 1000 回アクセスしてトラフィックデータ（pcap ファイル）を収集した．各クライアントで収集したデータセットの暗号スイートは**表 2** の通り．

3.1.2 特徴量設計

識別に用いる特徴量の候補は，NIST SP800-22 をベース

表 1 調査実験構成
Table 1 Investigation experiment environment.

	OS	SSL/TLS ライブラリ	Web サーバ/ クライアント
サーバ	Alpine Linux (3.9)	Openssl (1.1.1b)	Nginx (1.17.0)
クライアント A	Alpine Linux (3.9)	Openssl (1.1.1b)	Curl (7.64.0)
クライアント B	Alpine Linux (3.9)	Wolfssl (4.0.0)	Curl (7.65.1-DEV)

() 内はバージョン

表 2 データセット
Table 2 Dataset.

TLS Ver.	暗号スイート 鍵交換	認証	暗号化（鍵長）	MAC	クライアント A	B
1.2	DH	RSA	AES(128)	SHA256	✓	✓
1.2	DH	RSA	AES(256)	SHA256	✓	✓
1.2	DH	RSA	AESGCM(128)	AEAD		✓
1.2	DH	RSA	AESGCM(256)	AEAD	✓	✓
1.2	DH	RSA	CHACHA20/ POLY1305(256)	AEAD	✓	✓
1.2	ECDH	RSA	AES(128)	SHA256	✓	✓
1.2	ECDH	RSA	AES(256)	SHA384	✓	✓
1.2	ECDH	RSA	AESGCM(128)	AEAD	✓	✓
1.2	ECDH	RSA	AESGCM(256)	AEAD	✓	✓
1.2	ECDH	RSA	CHACHA20/ POLY1305(256)	AEAD	✓	✓
1.3	–	–	AESGCM(128)	AEAD	✓	✓
1.3	–	–	AESGCM(256)	AEAD	✓	✓
1.3	–	–	CHACHA20/ POLY1305(256)	AEAD	✓	✓

として，各検定の p 値や検定の途中で使われる統計量，その統計量をビット長に応じて割合に変換したものを選定した．なお，NIST SP800-22 の各検定には入力ビット長に関して推奨値があり，検定によっては通信セッションのデータサイズでは推奨値を満たせないケースが多くなることが予想されたため，ビット長の推奨値が短い検定 7 種類に限定した．

収集したデータセット（pcap ファイル）は TLS のアプリケーションデータ部のみを抽出し，セッション単位で結合したバイナリデータを生成した．本実験においては，原則データサイズに上限値を設定せずにセッション終了までの全手のデータを使用した．ただし，7 種類の検定のうち，Test for the Longest Run of Ones in a Block についてはビット長に応じて 3 段階にパラメータセットが変わるため，パラメータセットを揃えて評価するために，全てのデータが最小のクラスセットに収まるように，6272 ビットを超えるデータについては，先頭 6271 ビットのみを使用した．

選択した検定と定義した特徴量および今回のデータセットを用いて算出した各特徴量の最小値・平均値・最大値・標準偏差値は**表 3** の通り．

以下，各特徴量について説明する．全ての検定に共通して「p 値」は NIST SP800-22 で定義された検定の p 値を表し，n はデータのビット長を表す．

Monobit Test ベースの特徴量としては，p 値の他に，ビット列中の 0 の総数と 1 の総数をそれぞれ S_{0n}・S_{1n} とした時の S_{0n}/n・S_{1n}/n・$S_{0n}/\sqrt{n}$・$S_{1n}/\sqrt{n}$ を用いた．また，S_{obs} は検定でも使用している統計量で下記のように定義される．

$$S_{obs} = \frac{|S_{0n} - S_{1n}|}{\sqrt{n}}$$

データをブロック単位で分割する Frequency Test within a Block と Test for the Longest Run of Ones in a Block をベースにした特徴量としては，p 値の他に検定で使用しているカイ二乗値 $\chi^2(obs)$ を採用した．

Runs Test ベースの特徴量は p 値の他に連の個数を $V_n(obs)$ とした時の $V_n(obs)/\sqrt{n}$ を使用した．

Serial Test はパターンサイズ m を 4 ビットに固定した．検定には 2 種類の p 値を用いるためそれぞれを p 値 1・p 値 2 として利用するとともに，検定の途中で算出する ψ^2_m・ψ^2_{m-1}・ψ^2_{m-2}・$\nabla\psi^2_m$・$\nabla^2\psi^2_m$ を使用した．紙面の都合上，詳細は割愛するが，定義は NIST SP800-22 と同じである．

Approximate Entropy Test ベースの特徴量はパターンサイズ m を 3 ビットに固定した上で，p 値の他に $\varphi^{(m)}$・$\varphi^{(m+1)}$ とカイ二乗値 χ^2 と近似エントロピー値 $ApEn(m)$ を用いた．紙面の都合上，こちらも定義の詳細は NIST SP800-22 を参照されたい．

Cumulative Sums (Cusum) Test は加算の方向によって $mode = 0$（前方から加算）と $mode = 1$（後方から加算）が存在し，それぞれについて p 値を算出する．これらの p 値に加えて，加算時の最大絶対値を z とした時の $z/\sqrt{n}$ を使用する．

3.2 調査実験

選定した特徴量を用いて有効な特徴量（の組み合わせ）を検証した．分類のアルゴリズムには特徴量の判断ロジックの説明が容易な決定木およびランダムフォレストを採用し，10 分割交差検証により TLS バージョンと暗号スイートの組み合わせを推定する．最終的に精度の平均値によりその特徴量（の組み合わせ）を評価する．

決定木やランダムフォレストではジニ係数の減少量を元に，分類において各特徴量の寄与する度合い（重要度）を算出することができるため，重要度の観点からも各特徴量を評価する．

実験は下記の 4 つの観点から評価した．

- 1 特徴量の分類性能
- 2 特徴量の組み合わせの分類性能
- 全特徴量を用いた時の重要度
- Frequency (Monobit) Test と Runs Test の特徴量の組み合わせの分類性能

3.2.1 1 特徴量の分類性能

選定した特徴量の中から 1 特徴量のみを使用した場合，Frequency (Monobit) Test と Runs Test の特徴量が上位 10 位までを占めた（**表 4**）．

*1 mode=0
*2 mode=1

表 3 定義した特徴量とその統計値
Table 3 Defined Features and their statistics

NIST SP800-22 検定	特徴量	最小	平均	最大	標準偏差
Frequency (Monobit) Test	p 値	0.000	0.496	1.000	0.288
	S_{0n}/n	0.479	0.500	0.522	0.005
	S_{1n}/n	0.478	0.500	0.521	0.005
	$S_{0n}/\sqrt{n}$	42.15	50.37	69.79	9.823
	$S_{1n}/\sqrt{n}$	42.15	50.37	69.79	9.823
	S_{obs}	0.000	0.803	3.938	0.601
Frequency Test within a Block	p 値	0.000	0.499	1.000	0.287
	$\chi^2(obs)$	53.21	99.05	164.6	13.94
Runs Test	p 値	0.000	0.500	1.000	0.289
	$V_n(obs)/\sqrt{n}$	42.05	50.38	69.92	9.837
Test for the Longest Run of Ones in a Block	p 値	0.000	0.497	0.993	0.284
	$\chi^2(obs)$	0.084	2.990	31.45	2.385
Serial Test	p 値 1	0.000	0.498	1.000	0.288
($m = 4$)	p 値 2	0.000	0.498	1.000	0.289
	ψ^2_m	0.586	15.04	91.97	8.647
	ψ^2_{m-1}	0.033	7.025	61.76	5.495
	ψ^2_{m-2}	0.000	3.020	36.89	3.017
	$\nabla\psi^2_m$	0.260	8.023	41.12	4.019
	$\nabla^2\psi^2_m$	0.015	4.019	24.36	2.839
Approximate Entropy Test	p 値	0.000	0.498	1.000	0.288
($m = 3$)	$\varphi^{(m)}$	-4.382	-4.382	-4.379	0.000
	$\varphi^{(m+1)}$	-5.075	-5.074	-5.070	0.000
	χ^2	0.2594	8.028	41.842	4.021
	$ApEn(m)$	0.6910	0.6927	0.693	0.000
Cumulative Sums (Cusum) Test	p 値 *1	0.000	0.500	1.000	0.289
	p 値 *2	0.000	0.501	1.000	0.289
	$z/\sqrt{n}$*1	0.331	1.253	4.373	0.510
	$z/\sqrt{n}$*2	0.345	1.252	3.983	0.511

表 4 1 特徴量の分類性能（Top 10）
Table 4 Performance of single feature (Top 10).

特徴量	アルゴリズム	精度 [%]
S_{obs}	DecisionTree	47.15
p 値 (Monobit)	DecisionTree	47.14
$S_{1n}/\sqrt{n}$	DecisionTree	47.02
$S_{0n}/\sqrt{n}$	DecisionTree	47.02
S_{obs}	RandomForest	46.83
p 値 (Monobit)	RandomForest	46.82
$V_n(obs)/\sqrt{n}$	DecisionTree	46.77
$S_{1n}/\sqrt{n}$	RandomForest	46.77
$S_{0n}/\sqrt{n}$	RandomForest	46.77
$V_n(obs)/\sqrt{n}$	RandomForest	46.56

表 5 2 特徴量の分類性能（Top 10）
Table 5 Performance of feature pair (Top 10).

特徴量 1	特徴量 2	アルゴリズム	精度 [%]
S_{1n}/n	$S_{1n}/\sqrt{n}$	DecisionTree	48.10
S_{1n}/n	$S_{0n}/\sqrt{n}$	DecisionTree	48.10
$S_{1n}/\sqrt{n}$	S_{0n}/n	DecisionTree	48.10
S_{1n}/n	$S_{0n}/\sqrt{n}$	RandomForest	48.08
S_{1n}/n	$S_{1n}/\sqrt{n}$	RandomForest	48.08
S_{0n}/n	$S_{0n}/\sqrt{n}$	DecisionTree	48.07
$S_{1n}/\sqrt{n}$	S_{0n}/n	RandomForest	48.05
S_{0n}/n	$S_{0n}/\sqrt{n}$	RandomForest	48.03
p 値 (Monobit)	$S_{0n}/\sqrt{n}$	DecisionTree	48.02
p 値 (Monobit)	$S_{1n}/\sqrt{n}$	DecisionTree	48.02

3.2.2 2 特徴量の組み合わせの分類性能

選定した特徴量の中から特徴量 2 つを組み合わせた場合，上位 10 位までが Frequency (Monobit) Test の特徴量のみで占められる結果となった（**表 5**）．

3.2.3 全特徴量を用いた時の重要度

選定した特徴量の全てを使用した場合の各特徴量の重要度を算出した結果が**図 1**，図 1 である．重要度の観点からは，決定木では Runs Test の p 値と $\frac{V_n(obs)}{\sqrt{n}}$，Frequency (Monobit) Test の $\frac{S_{0n}}{\sqrt{n}}$ に強く依存しており，ランダムフォレストでは Frequency (Monobit) Test の $\frac{S_{0n}}{\sqrt{n}}$ と $\frac{S_{1n}}{\sqrt{n}}$，Runs Test の $\frac{V_n(obs)}{\sqrt{n}}$ に強く依存している．

3.2.4 Frequency (Monobit) Test と Runs Test の特徴量の組み合わせの分類性能

ここまでの実験において Frequency (Monobit) Test と

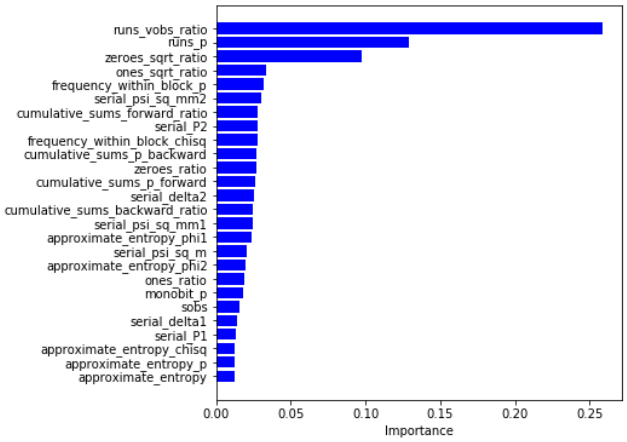

図 1 特徴量の重要度（決定木）

Fig. 1 Importance of each feature (Decision Tree).

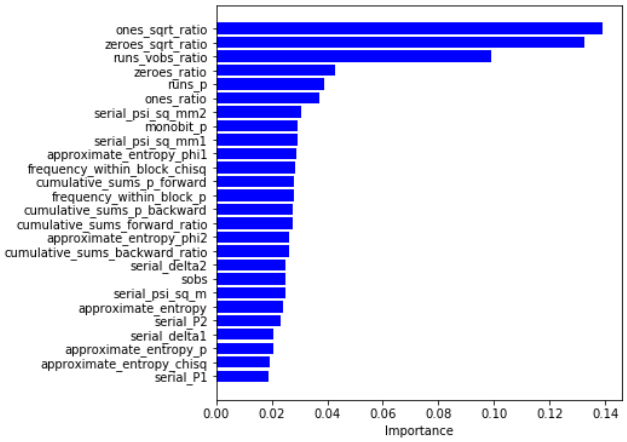

図 2 特徴量の重要度（ランダムフォレスト）

Fig. 2 Importance of each feature (Random Forest).

表 6 Frequency (Monobit) Test と Runs Test の特徴量組み合わせ

Table 6 Combinations of Frequency (Monobit) Test and Runs Test.

	Frequency (Monobit)						Runs	
	$\frac{S_{0n}}{n}$	$\frac{S_{1n}}{n}$	$\frac{S_{0n}}{\sqrt{n}}$	$\frac{S_{1n}}{\sqrt{n}}$	S_{obs}	p 値	$\frac{V_n(obs)}{\sqrt{n}}$	p 値
1			✓	✓	✓	✓	✓	
2	✓	✓	✓	✓		✓		
3			✓	✓			✓	✓
4	✓	✓	✓	✓	✓	✓	✓	✓

Runs Test に関する特徴量が分類に対する寄与が大きいことから，Frequency (Monobit) Test と Runs Test の特徴量の最適な組み合わせを求めるために，**表 6** の 4 パターンについて検証を行った．

パターン 1 は 1 特徴量での分類性能で上位 10 位に含まれる特徴量である．パターン 2 は 2 特徴量の組み合わせでの分類性能で上位 10 位に含まれる特徴量である．パターン 3 は特徴量の重要度で特に値の高かった 4 項目である．パターン 4 は Frequency (Monobit) Test と Runs Test について本実験で定義した全ての特徴量である．

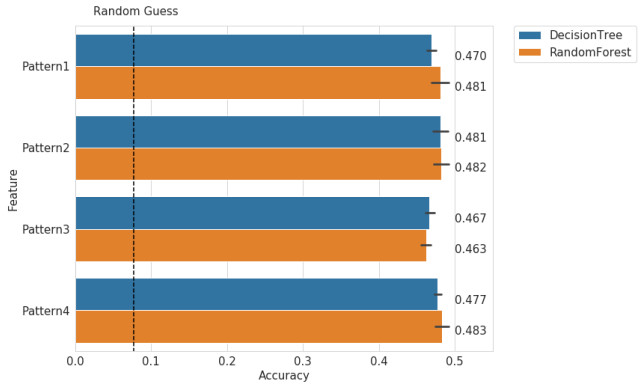

図 3 Frequency (Monobit) Test と Runs Test の特徴量組み合わせによる分類性能

Fig. 3 Performance using combinations of Frequency (Monobit) Test and Runs Test features.

これらのパターンについて決定木およびランダムフォレストで分類を行い，10 分割交差検証を実施した結果が**図 3** である．

いずれもランダムに推定させた場合（7.70%）に比べて高い精度を実現しているが，その中でも決定木はパターン 2 が，ランダムフォレストはパターン 4 が最も高い精度を実現した．以上により，パターン 2 およびパターン 4 が最も有効な特徴量の組み合わせと考えられる．

4. 評価実験

第 3 章の検討結果を踏まえて，Frequency (Monobit) Test と Runs Test をベースとした特徴量の組み合わせ（パターン 2・4）について，暗号化技術の推定性能の観点から分類粒度による性能の違いについて評価実験を行った．本実験では，第 3.1.1 項のデータセットを用いて，分類粒度の異なる下記 4 パターンについて検証を行った．識別のアルゴリズムには決定木およびランダムフォレストを用い，10 分割交差検証で精度の平均値で評価した．

TLS バージョン＋暗号スイート推定

TLS バージョンと暗号スイートを推定する．

暗号スイート推定

TLS バージョンは考慮せず暗号スイートを推定する．

暗号化アルゴリズム推定（鍵長あり）

TLS バージョンや認証・鍵交換アルゴリズムは考慮せず暗号化アルゴリズムを鍵長も含めて推定する．

暗号化アルゴリズム推定（鍵長なし）

TLS バージョンや認証・鍵交換アルゴリズムは考慮せず暗号化アルゴリズムを推定する．鍵長も考慮しない．

TLS バージョン推定

暗号スイートは考慮せず TLS バージョンを推定する．

「TLS バージョン＋暗号スイート推定」，「暗号スイート推定」，「暗号化アルゴリズム推定（鍵長あり）」，「暗号化アルゴリズム推定（鍵長なし）」，「TLS バージョン推定」の

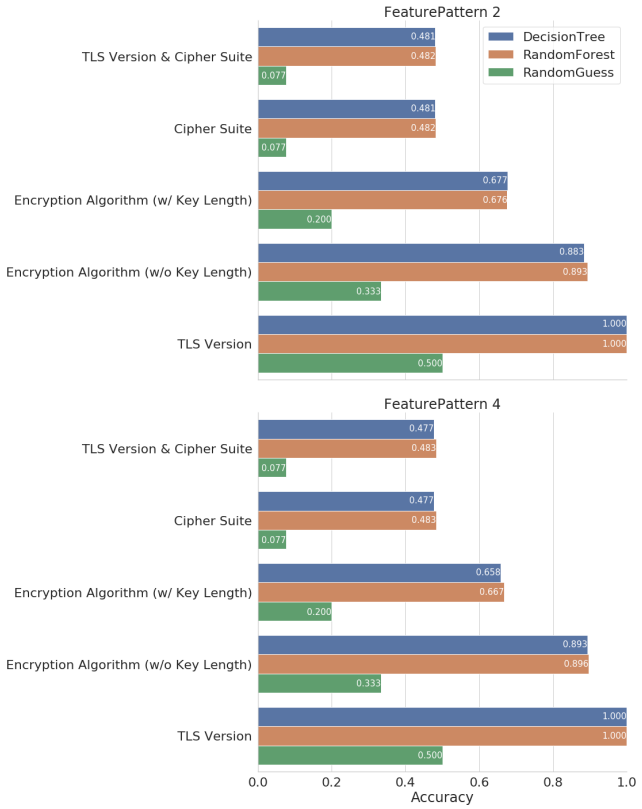

図 4　分類粒度による推定精度の違い

Fig. 4　Difference in performance by classification granularity.

順にラベル数が減り，分類の粒度が粗くなる．実験の結果は図 4 の通り．

全体的に決定木よりもランダムフォレストの方が若干精度が向上した．各分類で最も精度がよかった時の値は，「TLS バージョン＋暗号スイート推定」および「暗号スイート推定」で 48.3%，「暗号化アルゴリズム推定（鍵長あり）」で 66.7%，「暗号化アルゴリズム推定（鍵長なし）」で 89.6%，「TLS バージョン推定」で 100%であった．

5.　考察

第 4 章の結果について考察する．

「TLS バージョン＋暗号スイート推定」および「暗号スイート推定」の精度が変わらなかったのは，「TLS バージョン推定」が 100%できていたことによるものと推察される．

いずれの分類粒度の場合もランダムに推定させた場合よりも高い精度を実現しており，特に「TLS バージョン＋暗号スイート推定」および「暗号スイート推定」については，ランダム推定のおよそ 6.2 倍の精度であった．

図 5 は「TLS バージョン＋暗号スイート推定」を特徴量パターン 2 とランダムフォレストの組み合わせで実行した際の混同行列の一例である．TLS v1.3 については，AES256-GCM-SHA384 が精度 100%だった一方で，AES128-GCM-SHA256 と CHACHA20-POLY1305 の間では無視できないレベルの誤判定が生じていた．TLS v1.2 については，同じ暗号化アルゴリズム（AES，AESGCM，CHACHA20-POLY1305）間での誤判定が目立った．このことから本提案手法で暗号化アルゴリズム自体を識別する特徴が抽出できていると見ることができる．なお，特徴量パターン 2・4 と決定木・ランダムフォレストのいずれの組み合わせについてもこの傾向が見られた．

6.　関連研究

重本らは暗号化を用いる P2P 通信を識別するために初期パケットの乱数性に着目し，乱数性の特徴量として NIST SP800-22 のうち，Frequency (Monobit) Test，Cumulative Sums (Cusum) Test，Runs Test の p 値を用いた [13]．また，重本らは通信セッションの安全性の指標として，TCP/UDP や IP のペイロードに NIST SP800-22 を適用して，その検定結果を用いて暗号化レベルを判定する手法も提案している [14]．特徴量として p 値のみに着目している点と P2P/非 P2P の識別や暗号化レベル算出を目的にした点が本研究とは異なる．

Sengupta らはモバイルアプリの通信識別を目的に NIST SP800－ 22 を用いて特徴量の検討を行った [15]．Sengupta らは 11 種類の暗号化アルゴリズムに対して実験を行い，Discrete Fourier Transform (Spectral) Test の p 値が最も識別に適しているとの検証結果を元に，パケットデータの周波数変換をベースとした特徴量を提案している．パケットデータの周波数特性に着目した点と NIST SP800－ 22 に由来しない独自の手法を設計・提案している点で異なる．

Kwon らは圧縮アルゴリズム Lempel-Ziv-77（LZ77）で圧縮されたデータのパラメータ推定に NIST SP800－ 22 の Frequency (Monobit) Test，Runs Test と同様の統計量を使用している．乱数検定由来の特徴量を圧縮パラメータ推定に利用した点で本研究とは異なる．

7.　終わりに

本研究では，乱数検定の標準の 1 つである NIST SP800-22 で使われる統計的特徴量を用いた TLS の暗号スイート推定を行い，TLS の暗号スイート推定に，特に Frequency (Monobit) Test と Runs Test の特徴が有用であること示した．また，Frequency (Monobit) Test と Runs Test の特徴を用いた識別手法を提案し，48.3%の精度で暗号スイートの推定が可能であることを示した．TLS バージョン推定であれば 100%，暗号化アルゴリズム推定であれば 89.6%まで精度が向上した．本成果は暗号化されたデータから本来得られることが想定されていない情報を抽出できる可能性を示したものである．

今後は実通信環境における識別性能の検証を行い，提案手法の汎化性能を評価するとともに，特徴量計算に用いるデータサイズや計算量の観点から実通信への適用可能性を検証する予定である．

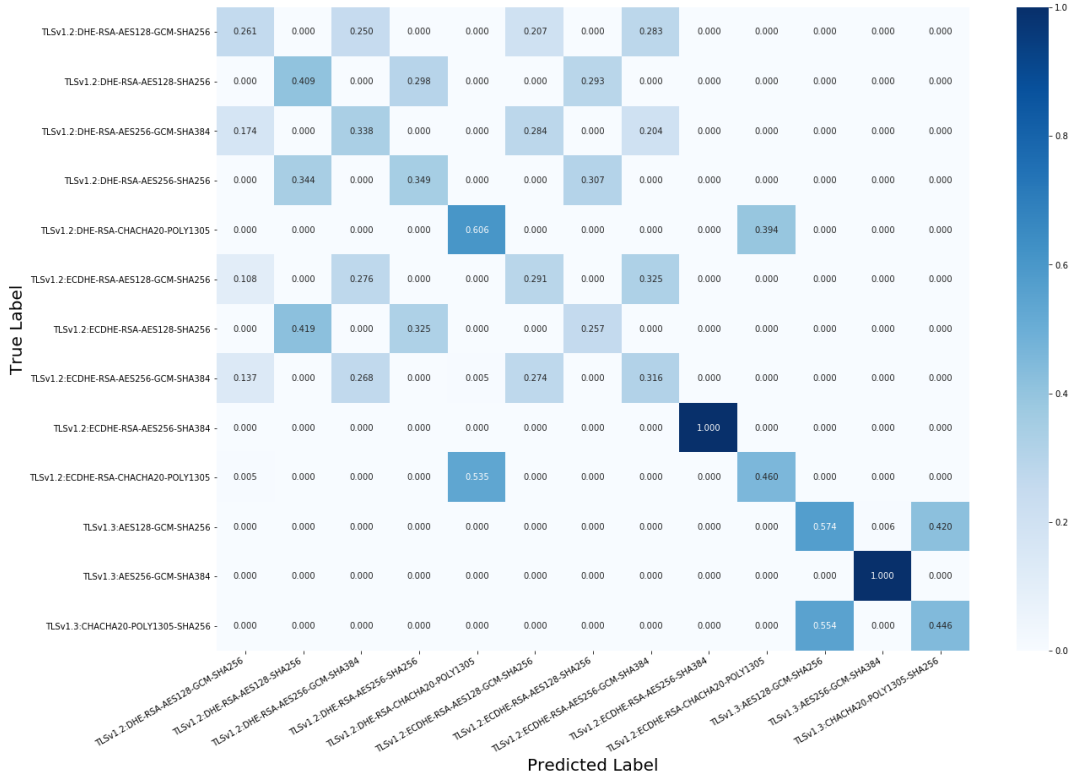

図 5 TLS バージョン+暗号スイート推定の混同行列 (特徴量パターン 4，ランダムフォレスト)

Fig. 5 Confusion matrix of "TLS version + Cipher suite" prediction (Feature pattern 4, Random Forest).

参考文献

[1] Google: 透明性レポート：ウェブ上での HTTPS 暗号化，Google（オンライン），入手先 ⟨https://transparencyreport.google.com/https/⟩（参照 2019-07-20）.

[2] Sara Dickinson, D. K. G. and Reddy, K. T.: Usage Profiles for DNS over TLS and DNS over DTLS, RFC 8310 (2018).

[3] Hoffman, P. E. and McManus, P.: DNS Queries over HTTPS (DoH), RFC 8484 (2018).

[4] Frolov, S. and Wustrow, E.: The use of TLS in Censorship Circumvention, *Proc. Network and Distributed System Security Symposium* (NDSS 2019), The Internet Society (2019).

[5] Anderson, B.: The Generation and Use of TLS Fingerprints, *Proc. FloCon 2019*, CERT (2019).

[6] Althouse, J.: TLS Fingerprinting with JA3 and JA3S, Salesforce (online), available from ⟨https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967⟩ (accessed 2019-07-20).

[7] Akamai Threat Research Team: Bots Tampering with TLS to Avoid Detection, Akamai Technologies (online), available from ⟨https://blogs.akamai.com/sitr/2019/05/bots-tampering-with-tls-to-avoid-detection.html⟩ (accessed 2019-07-20).

[8] US-CERT: MAR-10135536-G, Malware Analysis Report (MAR) 10135536-G, US-CERT (2018).

[9] Soto, J.: Randomness Testing of the Advanced Encryption Standard Candidate Algorithms, NIST Interagency/Internal Report (NISTIR) 6390, NIST (1999).

[10] Lawrence Bassham, Andrew Rukhin, J. S. e. a.: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Special Publication (SP) 800-22 Revision 1a (2010).

[11] Robert G. Brown, D. E. and Bauer, D.: Dieharder: A Random Number Test Suite, Duke University (online), available from ⟨https://webhome.phy.duke.edu/~rgb/General/dieharder.php⟩ (accessed 2019-08-14).

[12] L'Ecuyer, P. and Simard, R.: TestU01: A C Library for Empirical Testing of Random Number Generators, *ACM Transactions on Mathematical Software*, Vol. 33, No. 4, pp. 22:1–22:40 (2007).

[13] 重本倫宏，仲小路博史，寺田真敏：初期パケットの乱数性に着目した P2P 通信検知方式，研究報告コンピュータセキュリティ（CSEC），Vol. 2010, No. 38, pp. 1–6 (2010).

[14] 重本倫宏，鬼頭哲郎，仲小路博史ほか：データの乱数性に着目した暗号化レベル判定方法の提案，電子情報通信学会技術研究報告. SITE, 技術と社会・倫理， Vol. 111, No. 124, pp. 63–69 (2011).

[15] Sengupta Satadal, Ganguly Niloy, D. P. and Chakraborty, S.: Exploiting Diversity in Android TLS Implementations for Mobile App Traffic Classification, *Proc. The World Wide Web Conference* (WWW '19), ACM, pp. 1657–1668 (2019).