修士論文

Tor秘匿サービスへの攻撃に対抗する偽装トラフィックの生成と運用

Dummy Hidden Service to Protect Tor Hidden Service

指導教員　　松浦幹太　教授

東京大学大学院 情報理工学系研究科 電子情報学専攻

48-156424　　竹之内 玲

平成29年2月3日提出

内容梗概

近年, インターネットを通じて個人情報を集めることが容易になっており, プライバシーエンハンシング技術の重要性が高まっている. 匿名通信システム Tor は, ユーザとサーバの繋がりを秘匿することで通信に匿名性を持たせる実システムである. Tor はサーバの IP アドレスを隠す,Tor 秘匿サービスと呼ばれるシステムも提供している. この Tor 秘匿サービスのプロトコルには 4 つの Tor ネットワーク上のサーキットが含まれるが, Tor クライアントと Entry Guard または, Tor 秘匿サービスと Entry Guard の間の通信パケットを観測し, 得られたデータに対して機械学習を用いてトラフィック分析することで, それぞれを区別出来るということが報告されている. これに対し我々は, 実際の Tor 通信ではない偽のトラフィックを生成し, Tor ネットワークに流すことで有効なデータセットの割合を減らし, 攻撃の推定精度を下げる手法を提案した. Tor 秘匿サービスが生成するトラフィックと紛らわしいトラフィックを我々は偽装トラフィックと呼び, 偽のトラフィックとして用いる.

本稿では我々が取り組んだ, 偽装トラフィックの生成方法やその運用に関する実験について述べる. 最初の実験では簡単なノイズパケットで生成した偽装トラフィックが分類精度に与える影響を実験し, 簡単なノイズパケットでは分類精度に効果的ではないこと, 学習データセットのトラフィックの多様さが分類器の性能に大きく関わることを明らかにした. 次に効果的な偽装トラフィックの生成方法の提案とその評価実験を行い, 秘匿サービスが生成するトラフィックの傾向と, 分類器の精度を有意に下げる偽装トラフィックの生成方法を明らかにした. 最後に, 生成した偽装トラフィックの運用に関する実験を行い, Tor クライアントが生成するトラフィックのデータ量の 40%程の量の偽装トラフィックが存在すれば分類精度を大きく下げることが可能であることと, 偽装トラフィックの生成元になるトラフィックデータの多様さは分類精度に大きい影響を与えないことを明らかにした.

目次

Chapter 1　序論

今日, インターネットにおいて個人の情報を集め, その情報に基づき提供されるサービスが増えている. このような情報はサービス側が個人に合わせたサービスを提供するのに役に立つが, 一方でメールアドレスや住所などといった情報は悪用されるおそれがあり, 敏感な情報を隠したい人が確かに隠すことが出来るプライバシーエンハンシングの需要が高まってきた. 通信の情報も敏感な情報であり, 通信内容を守る技術の一つに暗号化通信がある. 暗号化通信では通信の際にその内容を暗号化し, 傍受されても攻撃者はその内容を複号出来ずやり取りの内容が漏れないようになるという技術である. しかし暗号化通信においては, ユーザがどのウェブサイトを閲覧しているのかという情報を秘匿することについては考慮されていない. この接続先の情報については, 匿名化通信を用いることでこの情報を秘匿することが出来る.

1.1　匿名通信

実用的な匿名化通信システムの一つに Tor が存在する [9]. Tor は匿名化通信技術の一つである Onion Routing を実装しており, 多くのユーザが利用, さらには協力している. Tor が提供するシステムはユーザのブラウジングを匿名化するものの他に, Tor 秘匿サービス（Tor Hidden Service）と呼ばれるものがある. これは, サービスを提供しているサーバの IP アドレスを隠すというシステムである. 実際に用いられている Tor 秘匿サービスには人権運動組織や内部告発サイトのようなものがあり, 人々の役に立っていると言える. しかしその一方で, 麻薬売買サイトや誹謗中傷が多く書き込まれる掲示板などの悪質なサービスも存在する. このように Tor 秘匿サービスが注目されているなかで, サーバの IP アドレスを暴く様々な攻撃が提案されてきており, Tor 秘匿サービスの匿名性は科学的評価がまだ十分行われているとは言えず, その存在について社会的な議論が先行することは望ましくない.

Tor 通信の接続先を推定する攻撃で高い精度を挙げている指紋攻撃を, Tor 秘匿サービスの通信に適用し, Tor 秘匿サービスプロトコル上のどの段階の通信か推定することで注目しているノードが秘匿サービスか判定できる, Circuit Fingerprinting Attack（CFA）という攻撃を Kwon らが提案した [17]. 指紋攻撃については 2.2.2 項で, CFA については 3.1.3 項で詳しく述べる.

1.2　本研究の目的と貢献

目的　我々は, CFA に対する防御手法として, Dummmy Hidden Service（DHS）を提案する. DHS は秘匿サービスと紛らわしいトラフィックを能動的に生成するノードである. 今日の多くの指紋攻撃は機械学習に基づいているため, ノイズを加える事で精度が大きく下がると言われており [26], Tor ネットワークに DHS が生成したトラフィックが混ざることで, CFA にとって有効といえるデータの割合を下げることが可能だと考えられる. しかし, Tor の開発陣, Tor project が多くのダミーパケットを挿入することを避けていることからも, DHS は Tor ネットワークとノードに小さい負担で指紋攻撃の精度を下げる必要がある. そこで本稿では, まず秘匿サービスのトラフィックの傾向を分析し, CFA の精度を効果的に低下させる偽のトラフィックはどのようなものか検討する. そしてこの偽のトラフィックはどのように生成され, 運用するべきか実験によって明らかにすることで, 実際の環境における DHS の有効な扱い方を考察することを目的とする.

貢献　Tor 秘匿サービスの匿名性への攻撃のうち, 受動攻撃である Circuit Fingerprinting Attack(CFA) はラップトップ一台でも攻撃に成功する可能性のある, コストが低く精度も高い攻撃である. 本研究では, CFA の精度を下げる防御手法として, Tor 秘匿サービスが生成するトラフィックと紛らわしいトラフィックである偽装トラフィックを生成する Dummy Hidden Service(DHS) を提案し, コストが低く且つ効果的である偽装トラフィックの生成方法を明らかにした. その中で, CFA に用いる分類器に最近傍法や Suppor Vector Machine(SVM) を用いた場合のそれぞれの傾向や, 秘匿サービスが生成するトラフィックの特徴を分析し明らかにした. さらに, 偽装トラフィックの生成すべき量や, 生成元となるトラフィックの選び方について実験を行い, 運用方法について検討した.

1.3　本稿の構成

まず 2 章で, Tor と Tor 秘匿サービスの仕組みやその匿名性に対する攻撃について述べる. 次に 3 章で Tor 秘匿サービスへの受動攻撃として有力な Circuit Fingerprinting Attack を詳しく紹介する. 4 章で我々が提案する Dummy Hidden Service を説明し, 5 章では我々が Dummy Hidden Service を提案する際に行った実験とその結果について述べる. 最後に, 6 章で結論を述べる.

Chapter 2　Torと Tor秘匿サービス

2.1　匿名通信システム Tor

Onion Routing　Goldschlag らは匿名化通信技術の一つとして Onion Routing を提案した [11]. 通常, インターネット通信の際にはいくつかの中継ノードを経由してクライアントとサーバがやり取りする. Onion Routing はこの中継ノードに Relay と呼ばれるノードを含める. 図 2.1 に Onion Routing の概要を示す. まずクライアントは各 Relay と鍵を共有しておく. 通信の際には, クライアントは各 Relay と共有した鍵でメッセージを多重に暗号化しておく. 暗号文を受け取った Relay は自分の持っている鍵で複号するが, クライアントは暗号化の際に複号する Relay が次に渡すべきノードのアドレスを含めておく. すると Relay は隣の Relay がわかるのでこれを渡し, 受け取った Relay は同様に自分の鍵で複号を行い, 次の Relay に渡す. 最終的にサーバに平文が送られる. このようにすることで, クライアント以外の各ノードは自分の隣のノードのアドレスしか知ることが出来ず, クライアントとサーバの繋がりが秘匿される. Onion Routing では各ノードを完全に信用できなくとも単体ではクライアントとサーバの繋がりを暴くことは出来ない. このようにして, サーバを誰が閲覧しているのかが匿名化される.

Tor(The onion router)　Tor は Onion Routing を実装した, 非常に大規模な実システムである. 2017 年 1 月時点で 7000 個程の有志の Relay が存在し [4], 利用者数は 150 万人を超えている.

Tor を利用する際の概要を図 2.2 に示す. Tor Network に接続するにはまず, Tor クライアントは Directory Authority と呼ばれる Relay から, Consensus File をダウンロードする. これには利用可能な Relay の情報が含まれており, この中からクライアントは中継ノードを 3 つ選択する. 次にクライアントはそれぞれの Relay と鍵共有をした後, メッセージを多重に暗号化して通信を行う. この生成されたリンクは Tor circuit と呼ばれる. Tor circuit においてクライアントに一番近い Relay は Entry Guard と呼ばれる. Entry Guard は, クライアントが接続するサーバのアドレスはわからないもののユーザのアドレスを知っているため, 匿名化通信においては非常に重要なノードと言える. 悪意のある Relay が簡単に Entry Guard になることを防ぐために, このノードは Guard フラグを与えられた Relay のみが選ばれる. 一方, サーバに一番近い Relay を Exit Relay と呼び, 間の Relay は Middle Relay と呼ばれる. Exit Relay はサーバのアドレスを知っているため, 何らかの方法でユーザのアドレスを知ることが出来れば, 匿

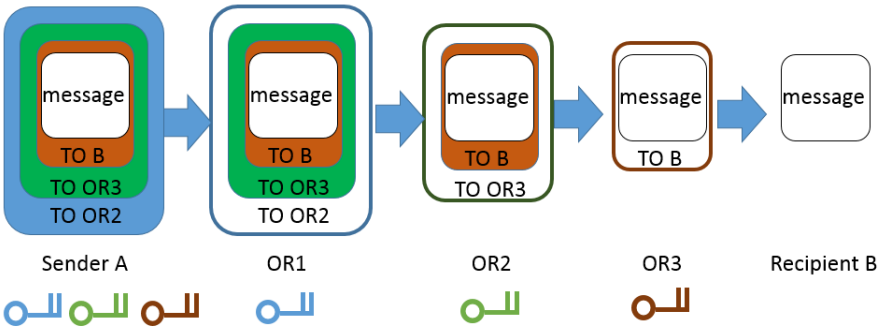

図 2.1: Overview of Onion routing

名性を破ることが出来る.

Tor project によれば Tor のデータ転送においては, パケットは 1 つが 512byte の固定長セルである. 固定長を用いない場合, 通信のトラフィック・パターンを観測することでどのようなウェブサイトを訪問しているのかを推定することが容易になるおそれがある. また, 実際には 512 の倍数のサイズのパケットが使われることがある.

2.2　Tor の匿名性に対する攻撃

本節では, Tor の匿名性を暴く既存の攻撃について説明する. Tor の匿名性を暴く上で重要なのは, Tor はユーザとユーザが閲覧しているウェブサイトの繋がりを秘匿することで匿名性を担保しているため, この繋がりを暴くことが匿名性を暴くことに繋がるということである. Wright らは匿名通信システムにおける攻撃の性能評価と防御手法について検討した [27].

2.2.1　複数の Relay を占拠して行う攻撃

一番単純な攻撃は, 攻撃者がクライアントとサーバの間の全ての中継ノードを占拠して行う, 結託攻撃である. 攻撃者はクライアントとサーバの繋がりを簡単に特定できる. また, Entry Guard と Exit Relay を占拠することで行う攻撃にタイミング攻撃がある. これは通信は短い時間の中で行われることに注目した攻撃であり, Entry Guard が通信中継したタイミングと Exit Relay が通信を中継したタイミングが似ているならば, Entry Guard の通信しているクライアント, そして Exit Relay が通信しているサーバの二つが通信を行っていると特定できる.

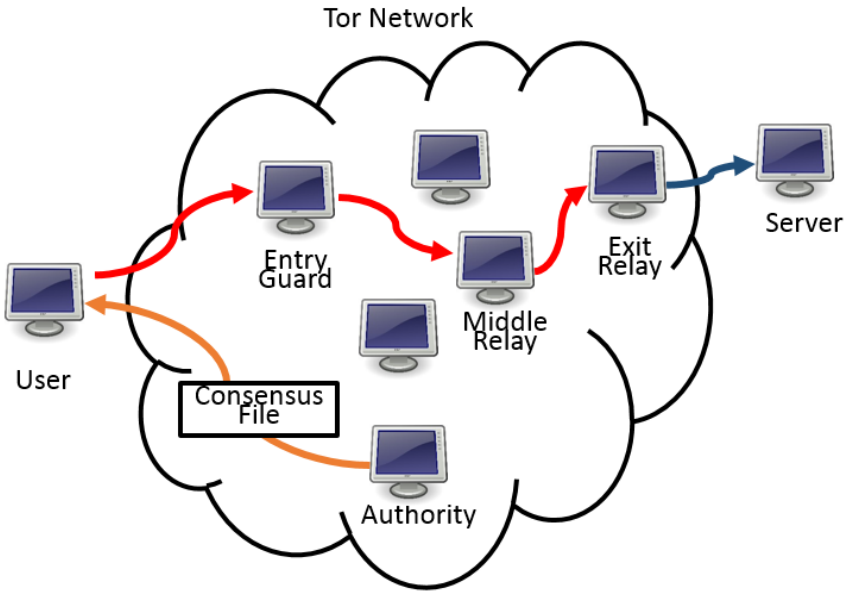

図 2.2: Overview of Tor

2.2.2　指紋攻撃

Entry Guard もしくは, Entry Guard とクライアントの通信を観測できるネットワーク管理者やインターネットサービスプロバイダが行うことが出来る強力な攻撃に, 指紋攻撃がある [21, 25, 24, 7, 23]. 指紋攻撃とは, ウェブサイトと通信した際に観測できるトラフィック・パターンを Website Fingerprint(WF) とし, これをウェブサイトを表す特徴として推定に用いる攻撃である. 指紋攻撃を用いることで, Tor が秘匿しているユーザーとウェブサイトの繋がりを攻撃することが出来る. 攻撃者は Tor ブラウザのバージョン等ユーザーと同様の環境を構築し, 自らウェブサイトを訪問しそのトラフィックを観測する. そして用いる攻撃手法に適した特徴量抽出を行い, これを用いて分類器に学習させておく. 改めてユーザーの Tor を用いたトラフィックを観測し, そのトラフィックから得られる特徴量を分類器にかけて訪問しているウェブサイトを推定する. なお, Tor パケットは 2.1 節で説明したとおり 512byte の倍数の長さであり, 単体のパケットを観測して得られる情報は長さと向き, 時刻だけである.

Juarez らによれば, 指紋攻撃には 6 つの仮定が存在する [16].

- Closed-world: 訪問されるウェブサイトはせいぜい k 個であるという仮定である. この仮定は非常に強く, k は実際のウェブサイトの数と比べ非常に低い値にされるからである. これに対し, open-world では被害者は分類器の学習データセットに含まれないウェブサイトも訪問するというシナリオであるが, Juarez らは評価用のデータセットに含まれないウェブサイトは訪問されない点を指摘している.

- Browsing behaviour: ユーザーは特定の行動をするという仮定である. 例えば,

ユーザーは同時には 1 つのタブのみで閲覧を行い, 1 つのウェブページから次のウェブページへ遷移するというシナリオが存在する. しかし, 現実のユーザーは同時に複数のタブを用いてブラウジングを行うことが多い. Tor を用いたブラウジングのデータは公開されていないが, 普通のブラウジングより遅いため, Tor においては同時に 1 つのタブで閲覧している, またはトラフィック解析においては区別可能であると Juarez らは言っている.

- Template websites: 全てのウェブサイトがテンプレートを用いて作られるという仮定である. Cai らは Hidden Markov Model を用いてウェブページの繋がりをモデル化し, 指紋攻撃に適用した [7].

- Page load parsing: 攻撃者はトラフィックを観測してユーザーのページ読み込みの始まりと終わりを認識できるとする仮定である. 2.1 節で説明した通り Tor のパケットは 512 バイトの固定長セルであり, 暗号化されているため, 攻撃者がトラフィック・パターンを観測しただけでページの区切りが分かるわけではなく, この推定は容易ではない [8].

- No background traffic: 攻撃者は, 他のアプリケーションによるトラフィックや同じ Tor circuit を流れるトラフィックといったノイズを全て区別出来, これを観測から取り除くことが出来るという仮定である. Tor Network に接続する手段は Tor ブラウザを使うだけでなく, 様々な方法が存在し, 複数のアプリケーションから Tor Network に接続することが可能である. 例えば, Talis は全てのインターネット通信を Tor Network を通して行う [1]. このような状況では, 注目したいトラフィックのみを選択することは困難である.

- Repicability: 攻撃者は被害者と同じ環境を構築できるという仮定である. 指紋攻撃において, 攻撃者は予め Tor Network を通してウェブサイトを訪問しトラフィックを解析しておくが, この環境が攻撃対象と同じでなければ正確なデータセットではなくなり, 推定精度が落ちる可能性がある. 環境としてはオペレーティングシステムやネットワーク接続, そして Tor ブラウザのバージョンといったものが挙げられる.

指紋攻撃の研究は, 分類器に用いる学習アルゴリズムや特徴量を変えて推定精度を上げるだけではなく, 攻撃者の仮定をより弱いものにし現実的なものに近づけることも重要である [26].

Hermann らはナイーブベイズを用いた指紋攻撃の攻撃手法を提案した [14]. Pachenko らは Support Vector Machine を用い, 775 個の close world のデータセットにおいて 50% 以上の精度で推定した [21]. Wang らは, k 近傍法を用いて非常に高い精度で攻撃に成功した [24].

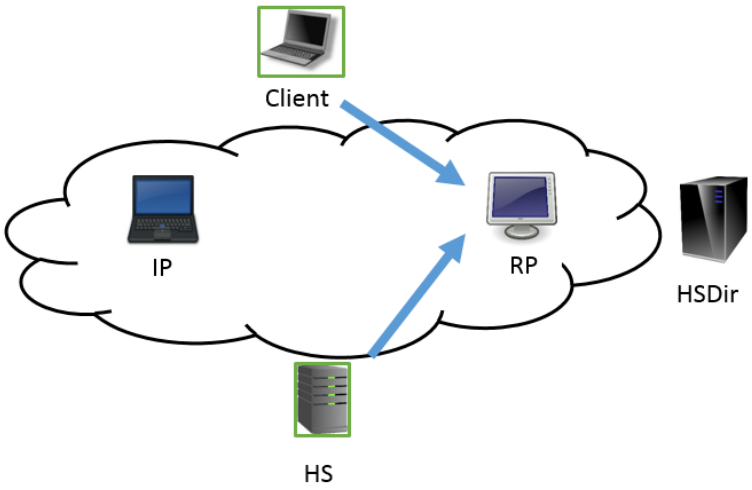

図 2.3: Overview of Hidden Service Protocol

2.3　Tor 秘匿サービス

Tor 秘匿サービス (Tor hidden service, HS) は Tor Network を用いて, サービスを提供しているサーバの IP アドレスを隠すシステムである. メインとなるノードは図 2.3 の database, introduction point(IP), rendezvous point(RP) である. 本節では Tor 秘匿サービスのプロトコルについて説明する. まず, HS は Tor Network 上の Relay から 3 つの IP を選び, 生成した公開鍵を送る. ここで注意すべきは, Tor 秘匿サービスのプロトコルにおいて, 全てのリンクが Tor Network 上で行われる点, つまり Tor circuit であるという点である. 次に, HS は HS descripter を生成する. HS descripter には, HS の公開鍵と IP の IP アドレスの情報が含まれる. この descripter と, decripter を秘密鍵で署名したものを HS directory(HSDir) と呼ばれるノードに送信する. この時, descripter に含まれる公開鍵から 16 桁英数字の onion アドレスが生成される. onion アドレスは HS を表す役割を担うが, 文字列と HS を紐付けるためにのみ用いられ, HS の IP アドレスの情報は含まれない. これで HS の登録が完了する. なおこの時, もし同じ公開鍵を違う HS が生成していたとしても, HS プロトコルでは同じサービスとして扱われ, 同じ onion アドレスが割り当てられる. そして後に登録された descripter が保存される.

次に, クライアントが HS に接続する前の準備について説明する. まずクライアントは接続したい秘匿サービスの onion アドレスを知る必要がある. 何らかの方法で onion アドレスを入手したクライアントは, HSDir に秘匿サービスの descripter を問い合わせる. 問い合わせを受けた HSDir は, onion アドレスが存在すればその descripter を返す. 登録していない, もしくは既に存在しない場合はエラーを返す. クライアントは descrioter を受け取り, IP の情報と HS の公開鍵を得る. そして IP に接続する前に Tor Network の Relay から RP を一つ選択する. RP は, クライアントと HS の中継を担う

重要なノードであり, 要求に対する返事としてワンタイムシークレットなクッキーを返す. クライアント側の準備は以上である.

実際に接続を開始するために, RP のアドレスと, ワンタイムシークレットを HS の公開鍵で暗号化した introduce message を IP に送信する. IP は以前の Tor circuit を用いて HS に introduce message を送信する. 秘匿サービスは自身の秘密鍵で受け取った introduce message を復号し, RP にワンタイムシークレットを送信する. RP はワンタイムシークレットを確認して, クライアントに正しくリンクが生成されたことを伝える. そして RP を中継して通信が開始される.

ここで注意すべきは, HS と RP の間の Tor circuit は HS に一番近い Relay が Entry Guard, RP に一番近い Relay が Exit Relay であり, Entry Guard は毎回同じセットの中から選ぶということである. そのような Tor circuit になっていない場合, タイミング攻撃と呼ばれる攻撃によって HS のアドレスを推定する手法が, Øverlier と Syverson によって提案された [20]. この攻撃については次章で説明する.

Chapter 3　関連研究

3.1　Tor 秘匿サービスの匿名性に対する攻撃

本節では, Tor 秘匿サービスの匿名性を暴く攻撃について説明する. Tor 秘匿サービス匿名性を暴く場合は一般の Tor を用いたブラウジングの際の匿名性を暴く場合と異なり, Onion アドレス, つまり秘匿サービスとその IP アドレスを紐付けることが目標である. ここで重要なことは, Tor circuit において秘匿サービスに一番近いノードは, 秘匿サービスの IP アドレスを知っているということである. つまりこのノードにとって, 通信しているエンドノードがある秘匿サービスであると特定出来れば匿名性を暴いたことになる. Tor 秘匿サービスの匿名性に関する研究は後述する論文の他にも, [6, 10] がある. Matic らは秘匿サービスの中から推定に使えるがサービス側が認識していない情報を探し, 推定に用いた [18].

3.1.1　秘匿サービスに対するタイミング攻撃

Øverlier と Syverson によって提案されたタイミング攻撃では, 2.3 節で述べたようにまだ秘匿サービス側から Tor circuit を引く際に秘匿サービスに一番近いノードを Entry Guard にしていなかった. 攻撃者は予め多くの Relay を占拠しておく. Øverlier の攻撃では, この Relay は Entry Guard である必要はないので, 攻撃者自身のもつ Relay でも多く用意することが可能である. 次に, 攻撃したい秘匿サービスに多くのクライアントから接続する. もし秘匿サービスと RP の circuit において, 秘匿サービスに一番近いノードが攻撃者の占拠した Relay ならば, クライアントからパケットを送信した際に短い時間の間に必ず攻撃者の Relay がこれを中継するので, 攻撃者はこれを観測すれば Relay が秘匿サービスに接続していると推定する事が出来る. その Relay は秘匿サービスのアドレスを知っているため, 秘匿サービスのアドレスが割り出されるという流れで攻撃が成功する.

また, 2014 年に実際に行われた攻撃ではØverlier らの仮定と異なり, Entry Guard を多く用意することでこのタイミング攻撃が行われた [2]. 2.1 節で述べた通り Entry Guard は, 信頼できて計算速度や通信速度が早いノードに与えられる Guard フラグをもつノードである. このノードになるためには多少なりともコストが掛かり, 多く用意するには非常に高いコストが掛かるが, コストさえ払えば攻撃が成功する可能性があることがこの攻撃で立証された. Tor の運用開発を行っている Tor Project はこの攻撃を受け, Guard フラグの管理方法を再検討した.

3.1.2 Relay early traffic confirmation Attack

Relay early traffic confirmation Attack は実際に行われ, Tor Project が記事として紹介した攻撃である [3]. この攻撃は秘匿サービスのアドレスに関する匿名性を暴くものではなく, ある Client が秘匿サービスを実際に利用するかを判別し, その Client の IP アドレスもわかるという攻撃である. 攻撃者は予め Entry Guard や HSDir をできるだけ占拠または用意しておく. HSDir は 2.3 節で説明したとおり, Client に問い合わせを受けた際に返事を送るがこの際に relay flag を付与する. しかしこの攻撃では relay early flag を付与する. ある Entry Guard が通信を中継する際, 本来使われないはずの relay early flag をエンドノードが使用していた場合, これは攻撃者の HSDir に接続した Client であり, 今秘匿サービスを利用しようとしていることがわかる. そして Entry Guard であるために IP アドレスもわかってしまうという攻撃である.

3.1.3 Circuit Fingerprinting Attack

本節では 2015 年に Kwon らが提案した Circuit Fingerprinting Attack(CFA) について紹介する [17]. この攻撃は我々の研究において特に重要視すべき攻撃である. 前節で述べたような能動攻撃とは異なり, CFA はパケット観測のみを行う受動攻撃であるため, 比較的少ないコストで成功する.

Kwon の攻撃は攻撃者は Entry Guard を一つ占拠することで行う, 二段階からなる攻撃である. 第一段階の攻撃が CFA である. Entry Guard が接続しているエンドノードが秘匿サービスであるのか, 通常の Tor クライアントであるのかを機械学習を用いたトラフィック分析で判別する. 秘匿サービスと判別した場合, 第二段階の攻撃を行う. 第二段階は Tor 秘匿サービスに対して 2.2.2 項で述べた指紋攻撃を行う. エンドノードのトラフィックを, 予め様々な秘匿サービスのトラフィックを学習しておいた分類器にかけ, 判別を行う. 本節では特に第一段階の CFA について説明する.

Tor Circuit のクラス　2.3 節で述べたように, Tor 秘匿サービスのプロトコルにおいてはいくつかの Tor circuit が出現する. Kwon らによれば, 図 3.1 の右に示した 4 つの circuit は Tor 秘匿サービスプロトコルに使われる circuit である. circuit のクラスの名前はそれぞれ, circuit のエンドノードの名前を並べたものである. これらと図 3.1 の左に示した Tor を用いた通常の Website 閲覧に使われる circuit は general というクラスである. CFA では, 対象の circuit を, トラフィック分析によってこれらの 5 つのクラスのどれかに分類を行う.

特徴量　Kwon らは CFA の分類器に用いる特徴量として三種類の特徴量を挙げている.

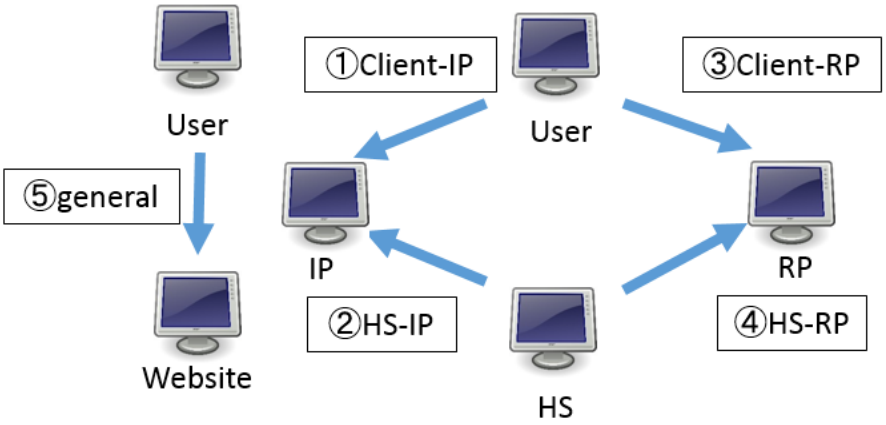

図 3.1: Circuits of Tor and Tor hidden service protocol

- 内向き外向きパケット: それぞれの circuit によって内向き, 外向きパケットの数に傾向があるが, ここでいう内向きとは 3.1 の Client または HS へ向かう向きが内向き, IP または RP へ向かう向きが外向きである. 3.1 で IP とつながりがある Client-IP と HS-IP の判別に特に有効である. Client-IP ではそれぞれの向きのパケットの数が同じになるという特徴がある. 彼らは今回は通信の最初の 50 個のパケットにおいて, それぞれの向きの数を計算し特徴とした. 外向きが 3 つ, 内向きが多くを占める場合は HS-IP らしさが強く, 外向きと内向きの数が同じならば Client-IP らしさが強い. HS-RP は外向きが内向きより多く, Client-RP は内向きが外向きより多い. これは実際の通信が RP を介した Circuit で行われることから, Web ブラウザとサーバの関係と似た性質になると考えられることにも沿っている.

- Duration of Activity(DoA): 注目している circuit で通信が行われている時間であり, Client-IP と HS-IP またはその他を区別するのに有効である. IP とのやりとりはプロトコル上長く行われるものではない一方で, 実際のやりとりである Client-RP, HS-RP, general の 3 つのについてはより長い時間のやりとりとなる. 図 3.2 に示したのは Kwon らが事前調査を行った結果であり, DoA の分布を示す. IP との通信の circuit の DoA は中央値が 1 秒ほどであったが, その他の circuit では中央値は 600 秒であった.

- サーキット構築シーケンス: 通信の最初パケットは circuit 構築のためのパケットであり, また Circuit の種類によってセルの数やプロセスが異なるため判別する特徴として利用できる. Kwon らは通信の最初の 10 個のセルの向きのシーケンスを特徴とした. Kwon らによれば, 内向きを”1”, 外向きを”-1”とすれば, ”-1 +1 -1 +1 -1 +1 +1 -1 +1 -1”というシーケンスは Client-RP に特徴的である.

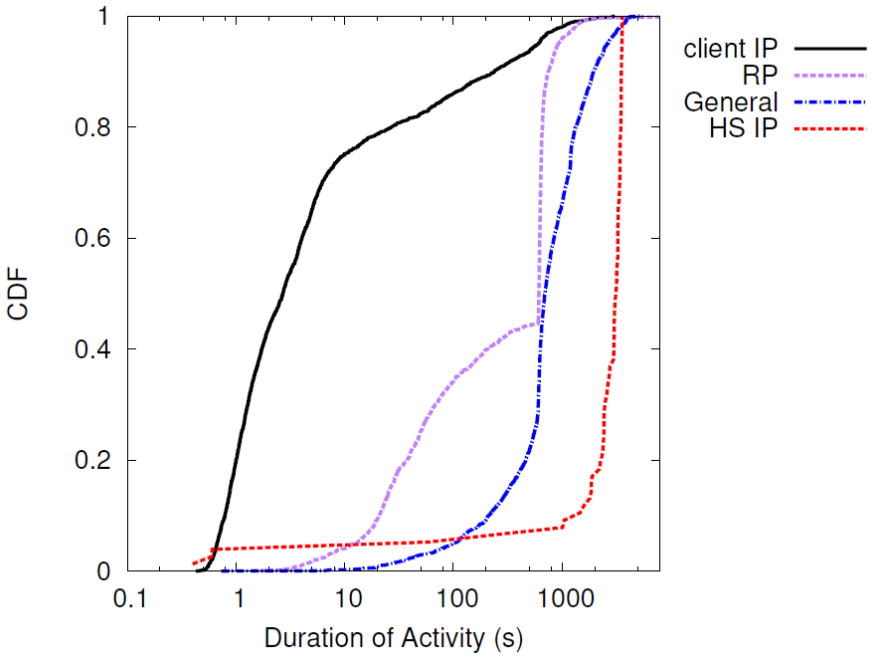

図 3.2: Distribution of the DoA of different Tor circuits from the hidden service- and the client-side. This figure is contained in [17].

攻撃者モデル　図 3.3 に示すように, 攻撃者は Entry Guard とエンドノードとの通信を観測することが出来るが, 実際の通信に影響を及ぼすことは出来ない. しかし攻撃者は予め, Tor Network に Tor クライアントを用いて自由にトラフィックを収集し, 分析することも出来る. このような攻撃者の具体的な立ち位置は, Entry Guard を占拠した攻撃者, もしくはインターネットサービスプロバイダやネットワーク管理者等である. また, 攻撃者はページの読み込み始めと読み込み終わりを正しく認識できるという仮定のもとデータは収集される. これは 2.2.2 項で説明した仮定のうち, Page load parsing にあたる.

実験環境とデータセット　Kwon らは 1000 個の秘匿サービスをたて, 実際の Tor 環境で接続し, パケットをキャプチャした. この際, 攻撃者はページのロードタイミングは正しく分割できるという仮定にあることに注意する. firefox と wget を Tor 環境で使用するようにしたものを, クライアントとした. firefox は 1 つ, wget は 4 つのクライアントを用意し, 1 回ずつの接続したため, 合計 5 回のインスタンスとなる. また秘匿サービスではない普通の Website との通信, つまり general クラスのデータも収集され, ネットワーク分析サイト Alexa のアクセスランキング Top1000 サイトに対して Tor 通信を行った. データ収集は 2015 年の 1 月から 3 月にかけて行われた.

実験は, データマイニングツール Weka[12] を用いて行われた. 分類器としては決定

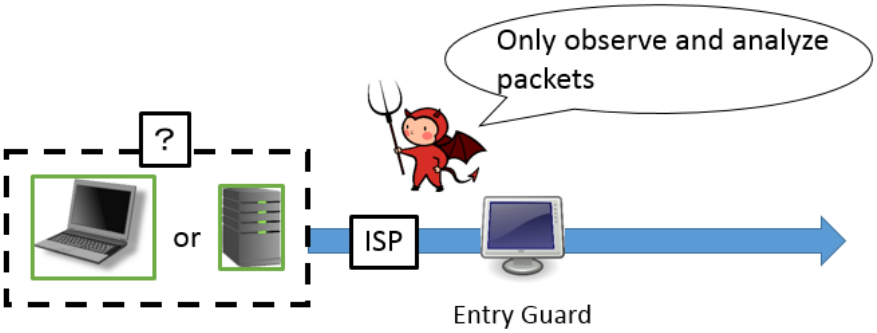

図 3.3: The adversary model in [17]

木が使われ, 決定木生成には C4.5, CART, k-NN が用いられ, k-NN は k=2 とされた. 彼らは収集したデータセットから特徴量を抽出し, また正解のクラスをラベル付けした. このラベル付けされたデータに対し CFA を行い, 交叉検定を 10 回行った. つまり, データを 10 個に分け, そのうち 1 個の集まりを評価用とし, 残りを学習用データとして実験を行うが, それぞれのデータが一度評価用として選ばれるよう合計 10 回実験を行ったということである. 評価方法としては下式 3.1, 3.2 に示す TPR, FPR を用いた. これらを, それぞれのクラスに対し算出する.

$$\text{TPR} = \frac{TP}{TP+FN} \tag{3.1}$$

$$\text{FPR} = \frac{FP}{FP+TN} \tag{3.2}$$

ここで, 注目しているクラスに対して,

- TP は注目しているクラスのラベルを持つデータを正しくそのクラスと分類すること
- FN は注目しているクラスのラベルを持つデータを誤って違うクラスに分類すること
- FP は注目しているクラスと異なるラベルを持つデータを誤って注目しているクラスと分類すること
- TN は注目しているクラスと異なるラベルを持つデータを注目クラスと違うクラスと分類すること

を意味しており, TPR は注目しているクラスのラベルを持つデータを正しく分類した割合, FPR は注目クラスと違うラベルを持つデータを誤って注目クラスと分類した割合である. 彼らはまず全てのデータセット (IP-Noise) に対し, Client-IP, HS-IP, その他

表 3.1: Dataset of evaluation in circuit fingerprinting attack[17]

Dataset	HS-IP	HS-RP	Client-IP	Client-RP	general
IP-Noise	76	954	200	4514	3862
RP-Noise	N/A	954	N/A	4514	3862

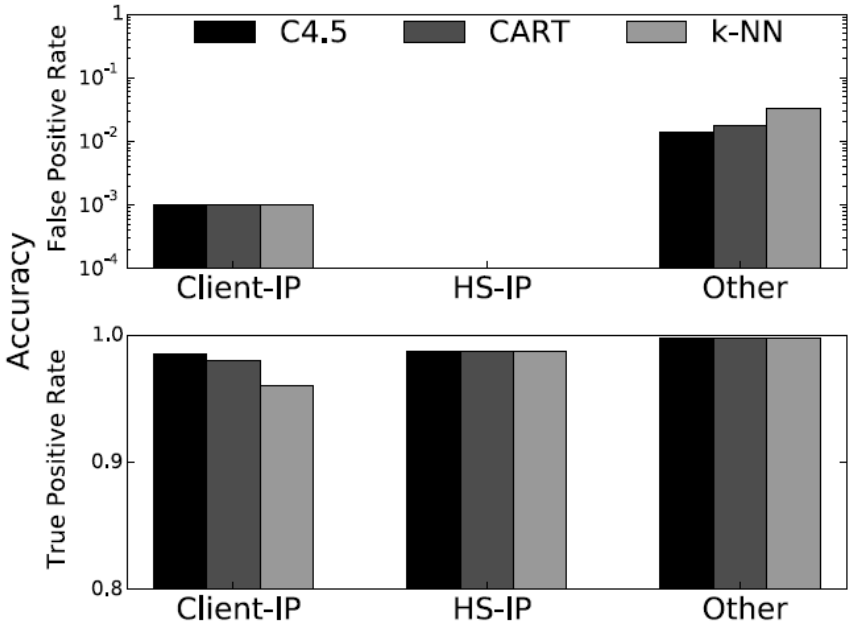

図 3.4: TPR and FPR of circuit classification with 3 -IP classes. This figure is contained in [17].

のクラスを判別する実験を行った. さらに, Client-RP, HS-RP, general のラベルを持つデータセット (RP-Noise) に対し, それらを判別する実験を別に行った. データセットのラベルの内訳は表 3.1 のとおりである.

結果と Kwon らによる議論　Client-IP, HS-IP, その他のクラスを判別する実験の結果は図 3.4 の通りである. TPR の縦軸は下端が 0.8 であり, FPR の縦軸は log スケールで表記されている. TPR はどのアルゴリズムを用いたときも 95%を超えているが, C4.5 を用いたときは若干 Client-IP の TPR が良い. FPR について, HS-IP に関してはどのアルゴリズムも 0%となった. Other を見ると C4.5 を用いたとき FPR が低く, 両方の評価からみて C4.5 アルゴリズムによる決定木が良い分類器と言える.

また, Client-RP, HS-RP, general を判別する実験の結果は図 3.5 の通りである. TPR の縦軸は下端が 0.8 であり, FPR の縦軸は log スケールで表記されている. TPR について, 先程とは異なり k-NN が HS-RP と general について良い精度となっている. 一方 FPR では HS-RP で k-NN が悪い精度を示しており, Client-RP はどの手法も 1%程の

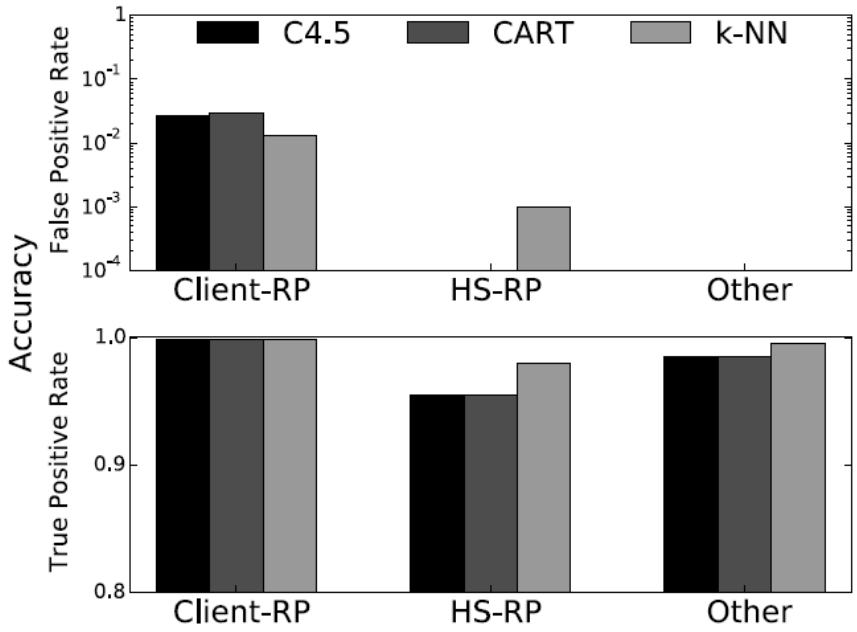

図 3.5: TPR and FPR of circuit classification with 3 -RP classes. This figure is contained in [17].

FPR となっている.

CFA に対する防御手法としては, ダミーパケットを混ぜることで推定精度を下げることが可能だと Kwon らは述べている. ダミーパケットは実際の通信には必要のないパケットであり, 観測者にとって有効なデータの割合を減らすことが可能である. Kwon らの手法においては IP を含む circuit はやり取りが少なく判別しやすいことと, パケットの数や並びに注目していることから, ダミーパケットは適した防御であるといえる. 実際 Kwon らが彼らのデータセットに対しダミーパケットを混ぜて再実験したところ, IP-Noise, RP-Noise のどちらのデータセットについても TPR が 15%低下した. ダミーパケットのデメリットとして, ダミーはノイズデータであるので, ダミーの割合が多ければ通信速度が下がりユーザビリティに影響が出てしまうという問題がある.

Chapter 4　Dummy Hidden Service

3.1.3 で説明した CFA はラップトップ一台程度の計算力で, 観測, 攻撃が可能であると Kwon らは述べている. 秘匿サービスではない普通の Tor ブラウジングの匿名性に対する指紋攻撃も, 今はラップトップで実行可能であり, 攻撃コストが低い攻撃として知られている. そこで我々は, CFA が機械学習に依る攻撃であることに注目し, 有効なデータセットの割合を減らす, Dummy Hidden Service(DHS) を提案する. DHS は秘匿サービスが生成するトラフィックと紛らわしいトラフィックを Tor Network に生成するノードである. この紛らわしいトラフィックを我々は偽装トラフィックと呼ぶ. DHS は多くの有志のユーザーが立ち上げることを想定しており, 特定の秘匿サービスを守るためのものではない. DHS によって, 全ての秘匿サービスが, CFA を行う攻撃者からは守られると考えられる.

4.1　Dummy Hidden Service による防御

本節では、 CFA を行う攻撃者から秘匿サービスを守る流れを説明する. まず, DHS の挙動や運用のされ方については攻撃者も知ることができる. これにより, 攻撃者は学習データセットに DHS が生成した偽装トラフィックのデータを, 正しいラベルとともに学習することが出来る. 偽装トラフィックは秘匿サービスの生成するトラフィックと紛らわしいので, 機械学習を用いても正しくは分類できないことがある. 偽装トラフィックを生成するノードを秘匿サービスと誤って分類することは攻撃者にとっては攻撃失敗である. また, 偽装トラフィックがデータセットに混ざった分類器は, 秘匿サービスの特徴を有効とする度合いが減り, 秘匿サービスのトラフィックデータを誤って, 他のノードが生成したトラフィックと分類する可能性がある.

4.2　Dummy Hidden Service が満たすべき要件と評価

DHS を提案するにあたり, DHS が満たすべき要件は以下の四個であると我々は考える.

- Effective: DHS のトラフィックは機械学習を用いた攻撃によって秘匿サービスと判定される必要がある.
- Low cost: DHS を運用することの Tor Network 上への負担は必要最低限である必要がある

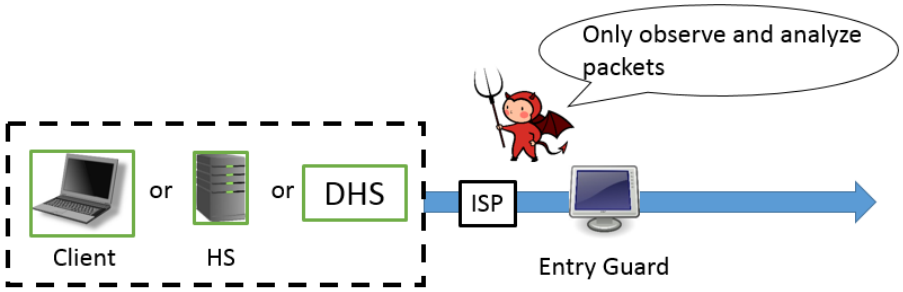

図 4.1: The adversary model in this reseach.

- Distribution: 多くの協力を得るために, DHS は簡単に導入出来る実装である必要がある.
- Indistinctive: ウェブサイト, または秘匿サービスの指紋攻撃によって特定のサーバーと推定されない必要がある.

この内重要なのは Effective と Low cost である. Effective である DHS でなければ CFA から秘匿サービスを守ることが出来ず, Low cost に関しては Tor project が特に重視している性質である. ここで, Effective の評価として 4.1 で説明した防御が成功したかどうかが挙げられる. 偽装トラフィックを秘匿サービス由来のデータと誤って分類することは False Positive にあたり, 偽装トラフィックがデータセットに混ざったことで秘匿サービスのトラフィックが他のクラスに分類されることは False Negative にあたる. そこで我々は, 秘匿サービスのラベルを持たない評価用データのうち, 誤って秘匿サービスに分類されたデータの割合を False Psotive Rate, 秘匿サービスのラベルを持つ評価用データのうち, 誤って他のクラスに分類されたデータの割合を False Negative Rate とし, これを評価に用いた. 実験ごとの詳しい評価に関しては 5 章のそれぞれの節で説明する.

4.3　偽装トラフィック

本章で偽装トラフィックを, 秘匿サービスが生成するトラフィックと紛らわしいトラフィックとしたが, 本節ではより詳しい説明をする. 図 4.1 に示す識別者は, エンドノードと Entry Guard の通信を観測することが出来る. エンドノードはページロードや, クライアントへの返事等である通信を行うが, この通信は同時に二つは行われず, また一

度の通信の始まりと終わりを識別者は正しく認識できる, 識別者は予めエンドノードが Tor クライアントの場合, Tor 秘匿サービスの場合, Dummy Hidden Service の場合のそれぞれのトラフィックを学習しておくことが出来, 新たに観測したトラフィックがどの種類のエンドノード由来であるのか識別する. このような識別者が, Dummy Hidden Service が生成するトラフィックを誤って Tor 秘匿サービスが生成するトラフィックと分類しやすいことを, 紛らわしいとしている.

偽装トラフィックを生成する実装手法として二つ提案する. 一つは Tor の環境を改変し, Middle Relay と DHS が通信を行う手法である. この手法は現在の Tor 環境のままでは出来ないが, 制約なくトラフィックを生成することが出来るという利点がある. もう一つの手法は, DHS と通信を行う Web サイトを用意し, Tor Circuit を介して通信を行う手法である. この手法は現在の Tor の使用を変更せずに行えるが, 特徴がよく現れる先頭パケットを自由には出来ず, 現在の Tor の仕様に従うという制約があるという欠点がある. 本稿では, 前者の自由にトラフィックを生成できるという仮定で検討, 実験を行う.

Chapter 5　実験と評価

4.2 節で説明した要件を満たすような偽装トラフィック, DHS はどのようなものであるのかを明らかにするため, 我々は三つの実験を行った. 5.1 節で実験の前提条件や攻撃者モデル等を解説する. 次に, 5.2 節では簡単な導入と, トラフィックや分類器の傾向調査のため, ダミーパケットを加えたトラフィックを用いて CFA の精度がどのように変化するか実験したことについて述べる. この実験結果はコンピュータセキュリティシンポジウム 2016 で報告した. 5.3 節では偽装トラフィックをどのように生成すれば CFA の精度をよく下げることが出来るのか実験した結果を説明する. この実験結果は暗号と情報セキュリティシンポジウム 2017 で報告した. 最後に, 5.4 で偽装トラフィックを実際に運用していく際の影響を, より現実に近い環境で実験した結果を述べる. この結果は, 第 22 回セキュリティ心理学とトラスト研究発表会で報告する予定である.

5.1　実験モデル

本節ではまず実験の前提条件を述べる. その後, 攻撃者モデル, 被害者モデルを細かく説明する. 今回の実験では, データは実際の Tor 環境のトラフィックを観測したものを使うが, 簡単のため, 実験モデルは実際の攻撃や防御とは異なる条件で行う.

長時間の観測について　まず, 今回の攻撃は一度のタイミングで行われる. 秘匿サービスの内容が更新されてトラフィックが変化したり, 秘匿サービス全体としての傾向が変化すること, そして Tor のバージョンが変わる前後で, 攻撃の精度が変化することと, 攻撃者が学習データを更新することについては考慮しない. さらに, 攻撃者は各エンドノードがトラフィックを生成する頻度を考慮しない. 世界中からアクセスされることが多い秘匿サービスと, 一日に数分のみ Tor を利用するクライアントではトラフィックの生成頻度が異なる. しかし今回はこの性質について考慮せず, 一つ一つのトラフィックから得られる特徴量に注目する. しかし, 秘匿サービスのトラフィック生成頻度はクラス分類に有効な特徴であるため, 今後の研究で Dummy Hidden Service がこの特徴を考慮した運用について検討する必要がある. 例えば, ブラウザでページを読み込む際のユーザーの行動をのモデルとしては Hernández-Campos らの研究がある [13].

Tor circuit　それぞれの実験の中では, 各 Tor circuit は対称であるとする. 評価用データ, 学習用データを収集する際になるべく同じ Tor circuit を用いる. また, 違う Tor circuit を用いて収集したデータについても, 抽出した特徴量の平均や分散に大き

な違いが見られないならば, Tor circuit の違いによる影響はないものとする. 今回収集したデータでは外れ値となるデータは多くは見られなかったが, 攻撃者は実際, 明らかに他のデータから外れた値をもつデータを学習データに入れないと考えられ, この仮定は妥当であると考えられる.

また, Kwon らの CFA では四種類の Tor 秘匿サービスプロトコル上の Tor circuit と, 一般 Website との通信における Tor circuit を考慮していた. しかし HS-IP の circuit 構築頻度は高くなく, また Kwon らも述べているとおりトラフィックが短いため, 今回の研究では HS-RP, Client-RP, general に注目した. 実際の攻撃における観測を考えると通信の開始と終了のタイミングは判別しやすいが, IP との Tor circuit の構築シーケンスの開始と終了を正しく判別することは難しい. この判別が可能な攻撃者ならば, IP との通信から得られる情報が有効だが, 実際の環境を考えると, DHS を提案するにあたって重視するべきなのは RP との通信であると考えられる.

攻撃者の能力の制約　攻撃者は 3.1 節で説明した能動的な攻撃は行えないとする. 我々は攻撃コストが低い CFA に対する防御としてのみ DHS が有効であると考え今回の実験を行ったが, 能動的な攻撃に対する根本的な防御手法はまだ提案されておらず, 検討する必要がある.

また, 攻撃者のもう一つの制約として, エンドノードと Entry Guard の通信トラフィック以外の情報を攻撃に使うことが出来ないとする. ここでいう通信トラフィックの情報とは, パケットの向きと時間のシーケンスを指す. 2.2.2 項で解説したとおり, 実際の Tor 環境では, Entry Guard とのエンドノードの通信はすべて暗号化され長さも 512byte の固定長であるため, トラフィックから得られる有用な情報としてはそれぞれのパケットの向きと時間である. エンドノードと Entry Guard の通信以外の情報を使うことが出来る攻撃者は例えば, 2.2 節で説明した結託攻撃のように, エンドノードが Tor ノードと Tor circuit を通して通信していることがわかれば, その Tor ノードは rendezvous point であり, エンドノードは秘匿サービスか秘匿サービスを利用しているクライアントとわかる. このような攻撃に関して, Tor ノード一つを占拠することはコストなしには出来ず, 特定の Tor circuit 上の全てのノードを占拠するには, 簡単に考えて Tor ノードの数のから選ばれる最低二つのノードも, Guard フラグをもつノードから選ばれる一つのノードも攻撃者が占拠しているということになる. この仮定は実現が困難であるため, 攻撃者は Entry Guard のみ占拠出来るとする研究が主流となっている.

最後に, 今回用いる分類器はデータマイニングツール Weka[12] に実装された kNN と SVM を用いる. Weka に実装された kNN は Aha らの Instance-based learning algorithms[5] を用いており, SVM は Platt の Sequential Minimal Optimization アルゴリズムを用いている [22]. Kwon らの報告では決定木による分類を行い, 決定木の構築に三つのアルゴリズムを取り上げていた. しかし, 同様の特徴量を用いた kNN や SVM による攻撃

も精度が高くベースラインとして考慮に値すると考えられる. 近年注目されている分類器には Deep Neural Network[15] を用いたものがあり, 今後 Deep Learning を用いた指紋攻撃やそれに対する防御を検討する必要がある.

偽装トラフィック　今回の我々の研究では, 偽装トラフィックが CFA に与える影響の傾向を見る目的もあるため, 偽装トラフィックは自由に生成出来るという仮定で実験を行った. Tor Network にただトラフィックを流すだけならば自由に下りパケットを生成することは出来ず, また既存の Tor 秘匿サービスプロトコルに沿ったトラフィックではシーケンスの先頭に制約がつく. これらの制約は今回は考慮しない. 前者の解決方法として, DHS と通信を行う秘匿サービスまたはサーバを用意し, 目的の下りパケットを生成する手法が考えられるが, 後者の制約を満たしながら CFA の DHS の要件を満たす偽装トラフィックについては今後検討する必要がある.

5.1.1　攻撃者モデル

4.3 節で述べたモデルと同じで, 図 4.1 に示したように, Kwon らの実験や指紋攻撃と同様に, 攻撃者はエンドノードと Entry Guard の通信を観測できる攻撃者である. 攻撃者は予め, 自らが用意したクライアント, 秘匿サービス, DHS を使ってそのトラフィックを収集し, 学習することが出来る. トラフィックを観測した攻撃者は分類器を用いて, そのトラフィックが秘匿サービス, クライアント, DHS の三つのクラスの内どのクラスのノードによって生成されたトラフィックか分類する. ただし, 本研究ではクライアントクラスはさらに, 秘匿サービスとの通信であるのか, 一般の Website の閲覧であるのかも分類する. この分類はエンドノードの分類につながり, 秘匿サービスと分類されたノードに対して, 後に IP アドレスの情報と秘匿サービスに対する指紋攻撃の情報を組み合わせることで秘匿サービスの匿名性を暴くことが出来る. その為, トラフィックの分類のみで十分な意味があると言える.

5.1.2　被害者モデル

被害者は図 4.1 の左端のエンドノードであり, 攻撃者によって分類されるという被害を受ける. 被害者は秘匿サービス (HS), Tor クライアント (Client), DHS の三つのクラスのうち一つのクラスに属する. 被害者は通信を行う際に, 同時に二つの通信は行わない. この仮定は攻撃者がページロードの開始と終了を判別することを可能にし, 攻撃者に有利な仮定であると言える. また, それぞれの実験で, 全てのノードは同じ回数の通信を行う. さらに, Client クラスの被害者は秘匿サービスと通信を行う場合と, 秘匿サービスではない一般の Website に Tor Network を通じて通信する場合がある.

5.2　ダミーパケットがTor秘匿サービスの匿名性に与える影響

4.2 節で述べた Effective を満たすような偽装トラフィックはどのようなものであるのかを調べる研究として, 簡単な導入目的と, 傾向調査のため, 実際の Tor 秘匿サービスプロトコル上のトラフィックに対してノイズを加え, 機械学習の精度がどのように変化するか定性的な性質を調べた. 要件の Distribution を考えると, 一番簡単な DHS の実装では外向きパケットをランダムに選んだ Entry Guard に送信するだけのノードが考えられる. そのため, この研究で加えるノイズは外向きパケットのみに限定する.

5.2.1　実験内容と環境

この実験では, クライアントが秘匿サービスに接続した際のトラフィックと, 一般の Website に接続した際のトラフィックに対して, CFA の精度を下げることを期待する手法でダミーパケットを加える. そして, ダミーパケットを加えたデータと秘匿サービスが生成するトラフィックのデータを対象に学習, 分類を行い, ダミーパケットの加え方によって精度にどのような影響があらわれるのかを調べる.

特徴量　この実験の攻撃では最初の 50 個のパケットの向きに注目した三次元の特徴量を用いた. 最初の 50 個のパケットのうち, 内向きパケットの数, 外向きパケットの数, そして三つ目の特徴量は内向きパケットと外向きパケットの比である.

データセット　実際の Tor 環境のトラフィックを 2016 年 8 月に収集した.

Tor 秘匿サービスのトラフィック, 秘匿サービスに接続した際のクライアントのトラフィックとして Tor Network 上に 4 つの秘匿サービスをたて, Tor ブラウザから接続する. 秘匿サービスは同じ仮想マシン上に作った Web ページを登録し, 同じ仮想マシンで Tor ブラウザから接続した. このときのトラフィックを tcpdump でキャプチャする. Tor ブラウザと秘匿サービスでは異なる Tor circuit が用いられ, Entry Guard も異なるため分離は容易であり, また同じマシンから接続することが特殊なトラフィックを産む要因にはならない. ページのロード開始から終了までの全てのパケットをキャプチャし, それを一つのデータとした. その後実験前処理としてパケットの内, その通信の実際のデータの長さ, つまり tcpdump で得られたデータに表示される length を対象に, Tor 通信の基本単位である 512byte ごとにパケットを区切り, 分けられたそれぞれのパケットから時間と向きのみを取り出す. そのため容量の大きいパケットは, 同じ時間同じ向きの複数のパケットに分けられる. ここで, 512byte の長さを持たないパケットは削除される. こうして時間と向きの属性のみ持つパケットのシーケンス列を整形済みデータとして, 特徴量抽出に用いる.

4 個の秘匿サービスそれぞれに 40 回ずつ接続し, 秘匿サービスのトラフィックとクライアントのトラフィックをそれぞれ 160 個ずつ得た. また, 一般の Website との通信トラフィックを収集する際, 接続先としてアクセス解析サイト Alexa が公表するランキング上位のサイトから, google.com, youtube.com, facebook.com, yahoo.com, wikipedia.org, amazon.com を選択し, それぞれ 40 回ずつ接続してトラフィックデータを 240 個得た.

実験環境　仮想マシンはともに VMware 上で Ubuntu14.04LTS を用い, メモリは 3GB とした. Tor のバージョンは 0.2.7, Tor ブラウザのバージョンは 6.0.3 のものを用いた. 秘匿サービスのための Web ページは LAMP 環境を用意し, apache2.4 を用いた.

用意した秘匿サービス　今回は 4 つの Web ページを用意し, 秘匿サービスとして登録したが, その内容について詳しく説明する. 共通点として, 全ての Web ページは画像を一枚含む. 画像やテキストの中身は特徴量に影響を与えないと考えられるが, あまりにやりとりが短いトラフィックはこの研究の実験目的である傾向調査にそぐわないため, 下りの容量が多くなることを期待して画像を挿入した. もっともシンプルな Web ページとして, 一行のテキストと一枚の画像のみのページを一つ, スタイルシートを用い, 要素としてはテキストと一枚の画像のページを一つ用意した. また, オープンソースのブログテンプレートとして Wordpress また, Drupal を用いて一つずつ Web ページを用意した.

ダミーパケット　前述の通り, 今回は上りパケットのみをダミーパケットとして加える. この実験では以下に示す三つの手法でダミーパケットを加えた.

- ランダムノイズ - 各パケット (Random-Each): 1 パケットごとにランダムでパケットを挿入する. ここで, ランダムとは確率 p で生起する事象であり, p は実験ごとに自由に設定できるパラメータである. この手法では, ノイズは最大で 2 倍になる.

- ランダムノイズ - 1 クラスタ (Random-Cluster): 同じタイムスタンプの連続したパケットをひとまとまりとし, その後ろにノイズを挿入する. ひとまとまりのパケットのサイズを N とし, 乱数 r は $r = (0.0, 1.0]$ をとるとする. ノイズを加えたパケットのサイズ L は, 元のパケットのサイズ N の M 倍が最大であるという制約をつけると,

$$L = N * (M - 1) * r \tag{5.1}$$

  となる. 実験ごとに M を変化させて精度の変化を見る. この実験では r は 0.5 に固定されている.

- 擬態ノイズ (Morph): 一番最初のパケットはノイズにしないという制約のもと, 元のパケットに外向きパケットのみを加えて, 先頭の 4 パケットを内向き, 外向き, 内向き, 外向きになるように近づける. そして, 次のタイムスタンプがことなるパケットのうしろに外向きパケットを加える. 今回観測した秘匿サービスのトラフィックを見ると, 先頭の 4 パケットが内, 外, 内, 外のシーケンスである傾向が見られた. これに対しクライアント側のトラフィックは, 一番最初のパケットが外向きであることが多く, また加えるノイズは外向きに制限しているため, 完全に一致させることは難しい. 加え方としては, 例えば元のパケットが「外, 外, 内, 内」であれば, 3 パケット目と 4 パケット目の間に外向きパケットを加えると, 最初の 4 パケットは「外, 外, 内, 外」となり 3 パケットが一致する. このノイズに設定するパラメータとして, 後ろに加えるパケットのサイズを自由に設定できる.

評価手法　秘匿サービスが生成するトラフィックのクラス (HS) のデータ 160 個, 秘匿サービスにアクセスしたクライアントが生成するトラフィックのクラス (Client) のデータ 160 個, 一般 Web サイトと通信した際のクライアントのトラフィックのクラス (general)240 個について, データセットを学習用と評価用にわけ, データマイニングツール Weka に実装された k 近傍法で学習, 分類を行う. ただしデータのわけ方に結果が依存しないようにするために, 10 回の交叉検定を行った. k 近傍法は, k=1 とした最近傍法とし, 重みは距離の逆数を用いる. HS クラスのデータに対し, HS クラスと推定できなかったものを False Negative, HS クラスではないデータに対し HS クラスと推定された場合には False Positive とする. 前述したそれぞれのダミーパケットの加え方に対して, パラメータを動かしながら分類を行う. 評価としては下式 5.2, 5.3 で表した False Positive Rate(FPR), False Negative Rate(FNR) を用いる.

$$FPR = \frac{\#\text{ of False Positive}}{\#\text{ of Client} + \#\text{ of general}} \tag{5.2}$$

$$FNR = \frac{\#\text{ of False Negative}}{\#\text{ of HS}} \tag{5.3}$$

5.2.2　結果と考察

まず, 図 5.1 は Random-Each の結果である. 横軸はノイズの生起確率 p である. p が 0.4 に近づくまでは FNR, FPR ともに徐々に上昇しているが, FPR は $p = 0.5$ から下降し始める. 一方で, FNR については $p = 0.9$ で大きく下降するほかは上昇する. Random-Each は 1 パケットごとにランダムノイズを入れる機会があるため, 徐々に外向きパケットの割合が増えてくる. これにより秘匿サービスの中で外向きと内向きパケットの割合が近いものがノイズパケットと間違われるようになったが, HS クラスのトラフィックに似たトラフィックを作り出せず False Negative は上昇しなかったと考えられる.

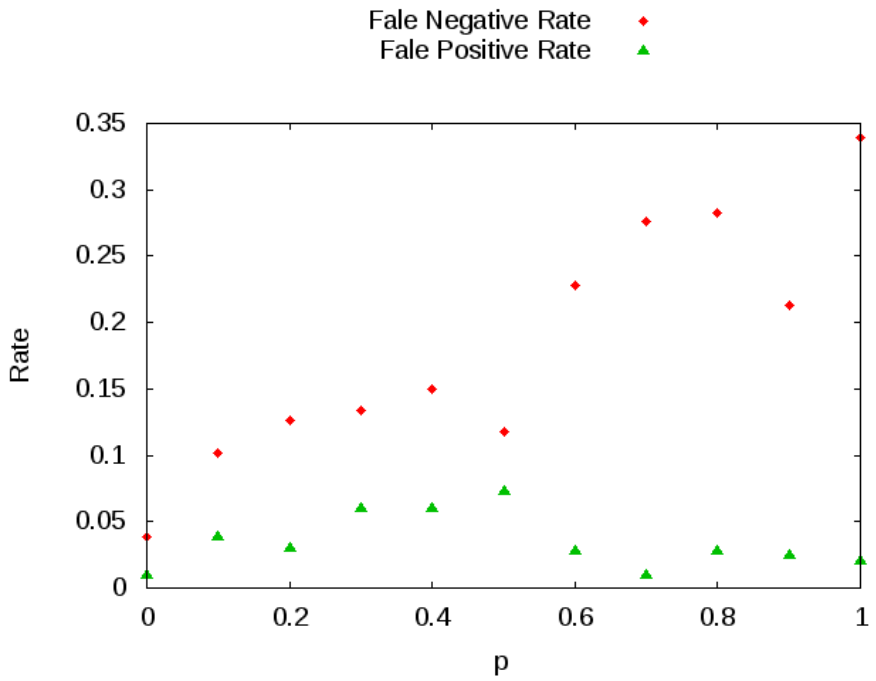

図 5.1: The result of Random-Each

一方, 図 5.2 は Random-Cluster の結果である. 横軸 M は, ノイズを入れたことでパケットが大きくなる上限の元のパケットと比べた割合である. $M = 2, 3$ では Random-Each と同様の傾向が見られるが, その後 $M = 5, 7$ 以外では FPR が上昇する. 一方で FNR に関しては $M = 6$ で大きく下降し, $M = 8$ 以降は減少していく. FPR が上昇した M に注目すると, その M では FNR が減少している. 今回はデータ取得に用いた秘匿サービス, ウェブサイトの種類が少なかったために, ノイズが加わった特定のウェブサイトのトラフィックが, ある秘匿サービスのトラフィックに近くなったと考えられる. そのため, 学習の結果, HS クラスと判定されるデータが増えた事が FPR, FNR の上下につながった.

最後に, Morph の結果について図 5.3 に示す. FNR についてはノイズパケット数が 30 個近くになるまでは上昇するが, その後減少し始め, ノイズパケット数が 45 個の時には FNR が 0 になっている. FPR に関しては, Random-Each, Random-Cluster と同様, 特定の秘匿サービスのトラフィックと近いトラフィックが生成されたかどうかで上下していると考えられる. FNR について, ノイズパケットが 30 個を超えると外向きが多すぎてしまい, 45 個では, 逆に秘匿サービスにはない特徴として扱われ, かえって HS クラスのトラフィックは正しく推定されるということになったと考えられる.

どの結果を見ても, FNR は 30%を超える程度であり, FPR に関しては 10%ほどしかでなかった. この原因として, 用いた秘匿サービス, ウェブサイトの種類が少なく, トラフィックも特徴的であり, かつ接続回数が 40 回と多かったため, トラフィックに対す

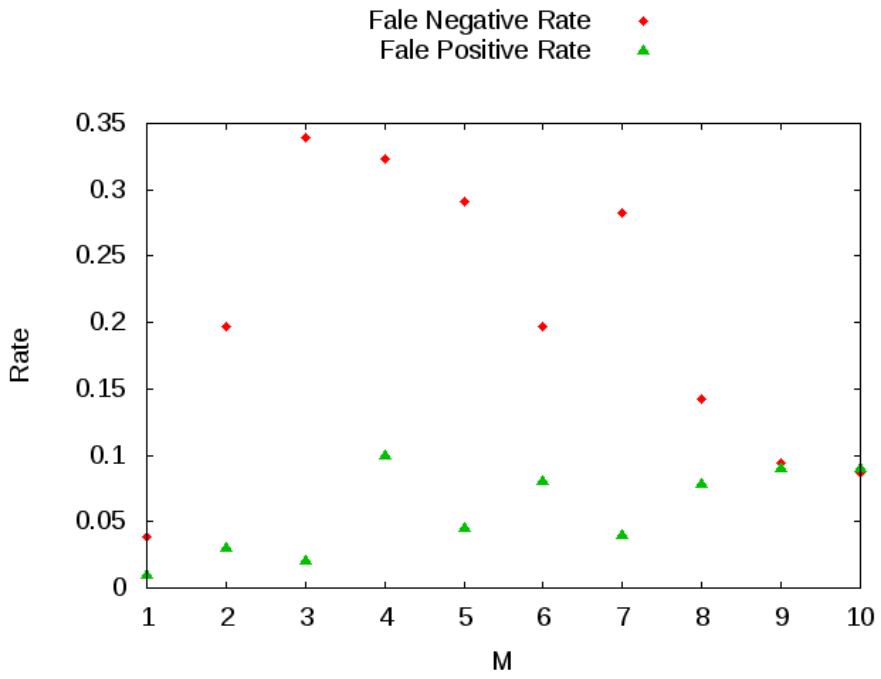

図 5.2: The result of Random-Cluster

る CFA 自体がウェブサイトの指紋攻撃と同じ働きをした. それによって FPR に関してある程度ノイズへの耐性が得られたと考えられる. また, 今回加えたノイズはクライアント側だけが外向きパケットのみを加えるであるが, もし HS クラスのトラフィックや内向きパケットのノイズを使用できるモデルならば, CFA の推定精度により大きな影響が現れると考えられる.

最後に, より工夫した攻撃者モデルに触れる. 今回の攻撃者は先頭 50 個の外向き, 内向きパケットの数にのみ注目していたが, 先頭 10 個のパケットの向きを特徴量に加え各手法のノイズを加えたデータに対し同様の実験を行った所, FPR は最大で 5.5%, FNR は最大で 1.0%となった. 各秘匿サービス, ウェブサイトのトラフィックを学習する十分なデータ量があったことでノイズ耐性が高くなっていることが原因として考えられるが, パケットの向きのシーケンスが特徴量として重要であることも確認できる. この後の実験においては, 先頭パケットの向きも考慮する.

5.3　Tor 秘匿サービスへの攻撃に対抗する偽装トラフィック生成

前節で明らかにした傾向と結果を考慮し, Effective な偽装トラフィックを生成する手法を提案し, その影響を実験で評価した結果について本節で説明する.

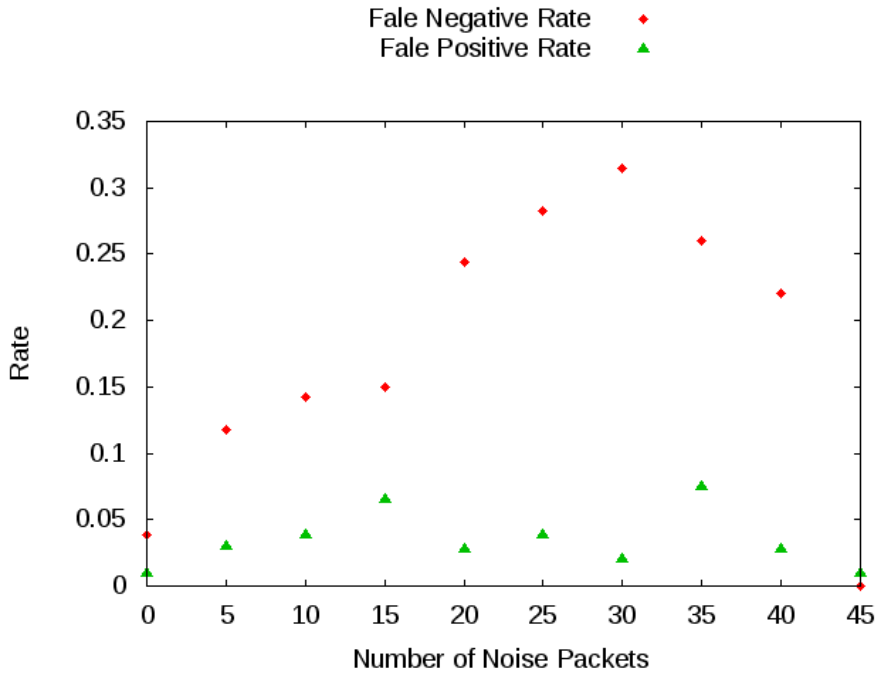

図 5.3: The result of Morph

5.3.1　秘匿サービスのトラフィック分析

偽装トラフィックが効果的に機械学習の精度を下げるためには, 実際の秘匿サービスの種類やその割合を考慮する必要がある. Moore らは秘匿サービスの利用形態について, Deep なクローリングを行うことで調査した [19]. 秘匿サービスの実態をデータセットとして再現するのは困難であるため, 5.2 節の実験よりも多様なウェブサイトをセットアップすることに取り組んだ. 我々は 2016 年 12 月に, 44 個の秘匿サービスを用意し, それらが生成するトラフィックを実際の Tor 環境で観測した. 同時に参考のために, この秘匿サービスに接続したクライアントのトラフィックについても観測した. このデータセットから, 3.1.3 項で説明した CFA に用いた三種類の特徴量を抽出し, その傾向について調べた.

まず, 先頭 50 パケットに含まれる内向きパケットの数を図 5.4, 外向きパケットの数を図 5.5, また, 通信時間である DoA を図 5.6 に示す. それぞれのデータの平均と標準偏差は表 5.1 の通りである. クライアントのトラフィックは, 内向き, 外向きパケットは合計が 50 になるようになっており, 標準偏差が一致する. それに対し秘匿サービスのトラフィックについて, 数パケットで通信が終わってしまうデータが存在したため, 合計が 50 未満であり, 外向きパケットの標準偏差が高くなっている. DoA については秘匿サービスのトラフィック, クライアントのトラフィックで大きな差は見られない.

次に, 先頭のトラフィックについて説明する. 図 5.7 に先頭 4 パケットの分析結果を示した通り, 前節でも触れたが, 秘匿サービスのトラフィックの先頭の 4 パケットの向

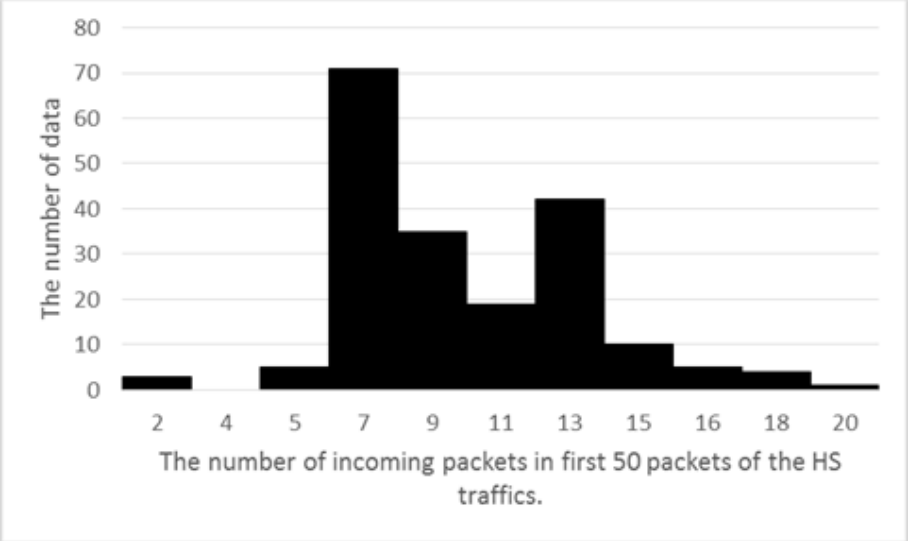

図 5.4: The number of incoming packet in top 50 packets.

表 5.1: The average and standard deviation value of features from traffics in Tor hidden service protocol.

	Hidden service		Client	
feature	average	s.d.	average	s.d.
# of incoming packets	9.051	3.037	37.714	3.276
# of outgoing packets	39.128	6.95	12.286	3.276
DoA	19.268	16.476	19.549	16.65

きは内, 外, 内, 外であることが多い. ここで, トラフィックのデータの数は 195 個であることに注意する. また, 5 番目以降のパケットの向きについて, 内向き, 外向きの数を図 5.8 に示す. 先頭から離れるにつれ外向きパケットの数が増えるが, プロトコル上のパケット交換が終了し, 実際の通信が始まるトラフィックが増えていくのだと考えられる. また, 秘匿サービスのトラフィックは外向きが多く, クライアントのトラフィックは内向きが多い. これは, 秘匿サービスが Web ページらしさをもち, クライアントが Web ブラウザらしさをもつことからも想定される.

5.3.2　偽装トラフィックの生成

前述した秘匿サービスのトラフィックの傾向から, 偽装トラフィックの生成手法として以下の四つを取り上げる. ただし, 偽装トラフィックの生成はクライアントのトラフィックから生成するものとする. ゼロから生成しないのは, クライアントのトラフィックのみ簡単に得られるような種類の秘匿サービスからも, その種類の秘匿サービスのトラフィックに近い傾向を持つ偽装トラフィックを得ることが出来ることを期待しているか

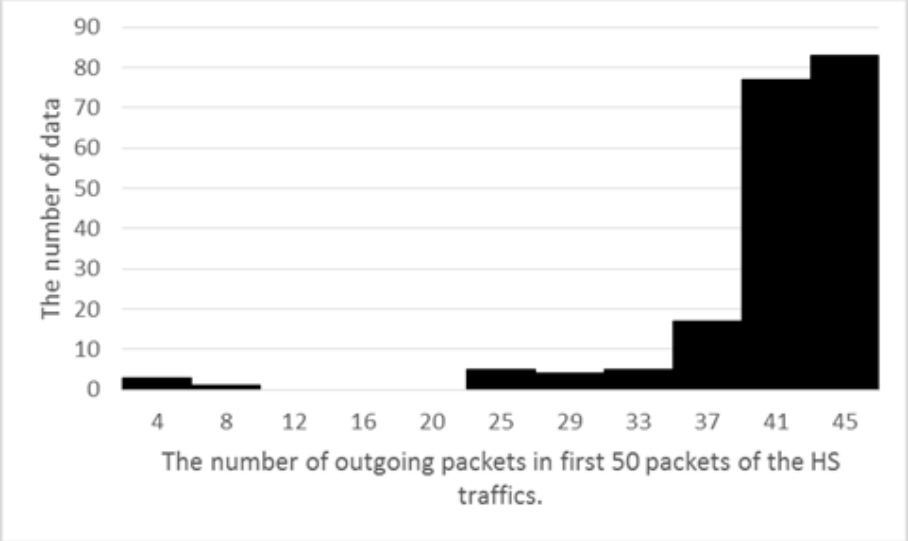

図 5.5: The number of outgoing packet in top 50 packets.

らである.

- Simple Reverse: 図 5.9 のように, 全てのパケットの向きを入れ替える. クライアントと秘匿サービスは 1 対 1 の通信を行っているわけではなく, ページをロードする中で外部サイトや API 呼び出しがあるが, ある程度の効果はあると考えられる. 簡単な手法であり, 比較のため用いる.

- Reverse without 2nd3rd: 図 5.10 のように, 先頭 4 パケットのうち特徴が強く, 分類器の重みが強い第 2 第 3 パケットはそれぞれ外向き, 内向きとしその他のパケットについては向きを入れ替える. これは 2 番目と 3 番目のパケットの向きを重視した手法である.

- PNPN: 図 5.11 のように, 先頭 4 パケットを内向き, 外向き, 内向き, 外向きに固定し, それ以降のパケットについては一定の確率で外向きとする. ただし外向きとされなかった場合は, パケットの向きを入れ替える処理を行うため, 最終的にパケットの向きが外向きになることはありえる. これは, クライアントのトラフィックから多様な偽装トラフィックを生成することを期待するためである. PNPN は先頭 4 パケットの向きを重視した手法である.

- Rev-NPN: 図 5.12 のように, 先頭 4 パケットのうち, 比較的重みが小さい 1 番目のパケットについてはパケットの向きを入れ替える. 2 番目から 4 番目については, それぞれ予め決めた確率でパケットの向きを外向き, 内向き, 外向きとする. また, そうしなかった場合にはそれぞれ内向き, 外向き, 内向きとする. つまり 2 番目から 4 番目のパケットは元のクライアント側のトラフィックに関わらず, 確

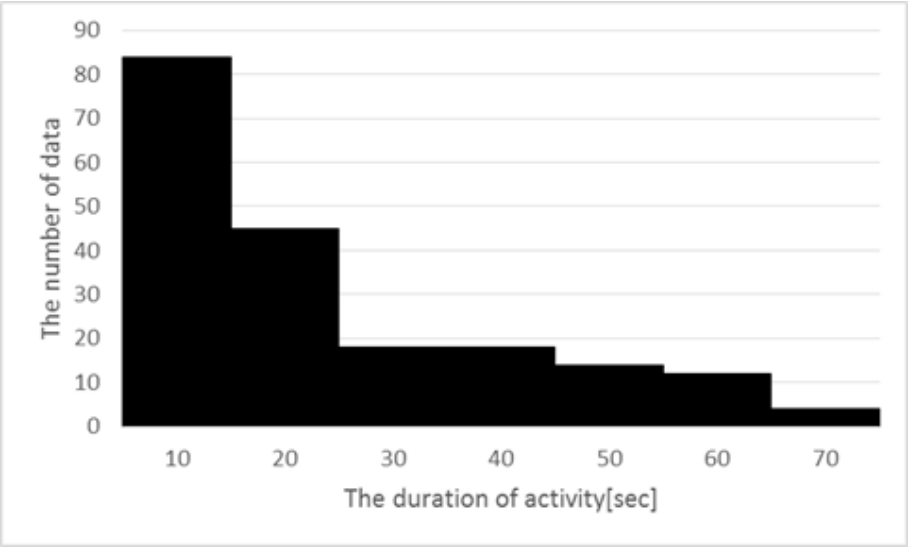

図 5.6: The duration of activity of HS traffics.

率によって向きが決まる. この確率についてはヒューリスティックに求まると考えられる. それ以降のパケットは PNPN 同様, 一定確率で外向きとするかまたはパケットの向きを入れ替える処理を行う. Rev-NPN の先頭パケットの生成手法が PNPN と異なるのは, 全ての秘匿サービスのトラフィックの先頭パケットが, 内向き, 外向き, 内向き, 外向きの順になっていたわけではないことを考慮した手法である.

5.3.3　実験内容と環境

この実験では, クライアントが秘匿サービスに接続した際のトラフィックをもとに, 5.3.2 項で示した手法で偽装トラフィックを生成する. そして, 偽装トラフィックと秘匿サービスが生成するトラフィックのデータを対象に学習, 分類を行い, 偽装トラフィックの生成手法によって分類精度にどのような影響があらわれるのかを調べる.

特徴量　この実験の攻撃では CFA と同じ三種類の特徴量を用いた. 特徴量は最初の 50 個のパケットについての内向きパケットの数, 外向きパケットの数をそれぞれ一次元, 通信時間を表す DoA が一次元, そして最初 10 個のパケットの向きの十次元の十三次元となっている. ただし, 向きについては内向きを 1, 外向きを-1, 通信が終わってしまってデータが存在しないインデックスのパケットの向きを 0 とした.

データセット　実際の Tor 環境のトラフィックを 2016 年 12 月に収集した.

Tor 秘匿サービスのトラフィック, 秘匿サービスに接続した際のクライアントのトラフィックとして Tor Network 上に 44 つの秘匿サービスをたて, Tor ブラウザから接

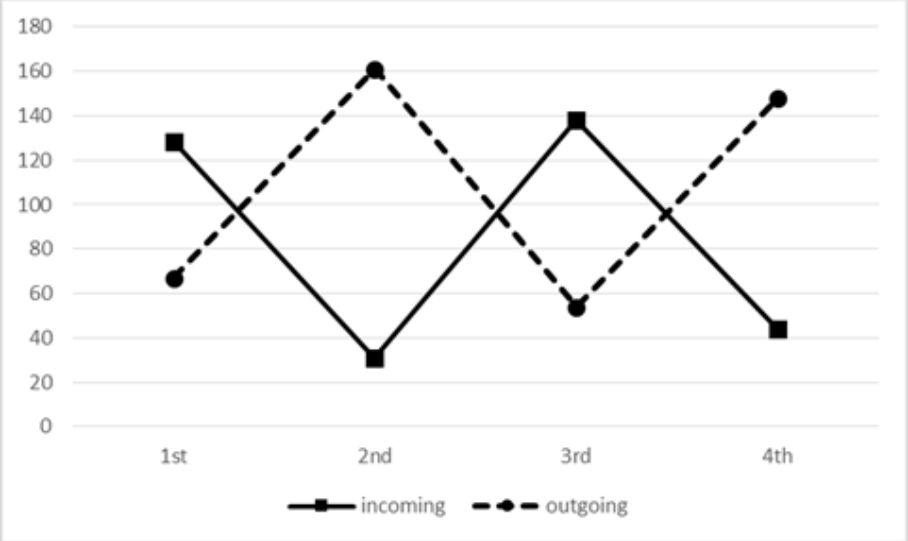

図 5.7: The number of incoming and outgoing packet in top four packets.

続する．5.2 節の実験と同様, 秘匿サービスは同じ仮想マシン上に作った Web ページを登録し, 同じ仮想マシンで Tor ブラウザから接続した．このときのトラフィックを tcpdump でキャプチャする．Tor ブラウザと秘匿サービスでは異なる Tor circuit が用いられ, Entry Guard も異なるため分離は容易であり, また同じマシンから接続することが特殊なトラフィックを産む要因にはならない．データ形式も 5.2 節と同様であり, 本節では簡単に述べる．ページのロード開始から終了までの全てのパケットをキャプチャし, 一つのデータとした．Tor 通信の基本単位である 512byte ごとにパケットを区切り, 分けられたそれぞれのパケットから時間と向きのみを取り出す．ここで, 512byte の長さを持たないパケットは削除される．

44 個の秘匿サービスそれぞれに 5 回ずつ接続し, 秘匿サービスのトラフィックとクライアントのトラフィックをそれぞれ 220 個ずつ得た．このうち, まったく秘匿サービスのトラフィックのデータが存在しなかった 25 個を省き, 秘匿サービスのトラフィックは 195 個とした．このデータは 5.3.1 項で説明したトラフィック分析にも用いた．なお, 秘匿サービスのトラフィックのうちデータの数が少ないものや存在しなかったトラフィックについて, 同時に行われたクライアントのトラフィックは, 他のトラフィックと大きな差は見られないため, Tor ノードがキャッシュのような挙動を示したのだと考えられる．

用意した秘匿サービス　5.2 節で述べた実験で明らかにしたように, 秘匿サービスのトラフィックに関する実験では多様な Web ページを扱う必要がある．そこで本節の実験では, 4 種類の Web ページとして, Web ページテンプレート Bootstrap を用いたページ, Wiki, ブログテンプレート Wordpress のページ, テキストと画像のみのページを用

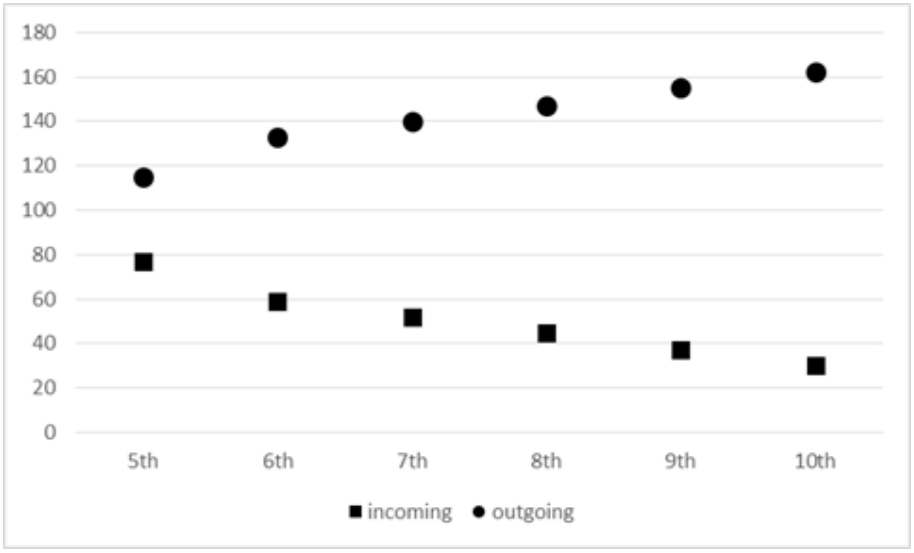

図 5.8: The number of incoming and outgoing packet in fifth packet and the followings.

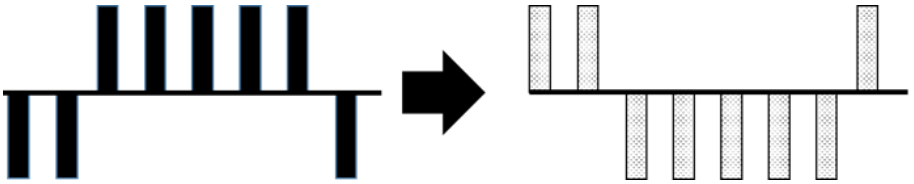

図 5.9: The example of the generation of a dummy traffic by Simple Reverse.

意した．Wiki, Wordpress ブログ, テキストと画像のページのコンテンツは, 著作権が切れた日本語文章と著作権フリーの画像を用いた．Bootstrap を用いたページとして, クリエイティブ・コモンズもしくはオープンソースのテンプレートを 10 個用意し秘匿サービスとした．Wiki として, オープンソースである pukiwiki そして mediawiki を用い, pukiwiki に関しては 5 個, mediawiki は 4 個用意した．Wordpress ブログは 10 個用意し, テキストと画像のみのページは 16 個用意した．ただし, この実験ではページの種類とその数については十分な考慮を行えなかった．今後一般 Web ページそして秘匿サービスの種類や作成方法の傾向調査を行い, データセットはこの傾向を考慮したものにする必要がある．

実験環境　仮想マシンはともに VMware 上で Ubuntu14.04LTS を用い, メモリは 3GB とした．Tor のバージョンは 0.2.7, Tor ブラウザのバージョンは 6.0.3 のものを用いた．秘匿サービスのための Web ページは LAMP 環境を用意し, apache2.4 を用いた．Rev-NPN の 2 番目から 4 番目のパケットの向きは, 2 番目が 85%の確率で外向き, 3 番目が 80%の確率で内向き, 4 番目が 75%の確率で外向きとした．

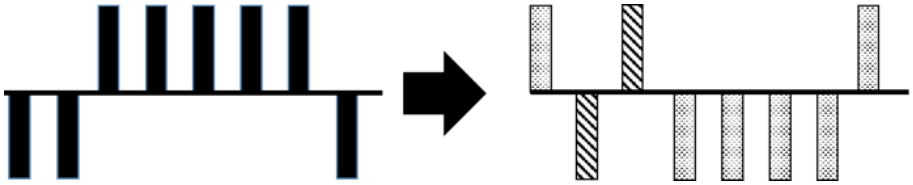

図 5.10: The example of the generation of a dummy traffic by Reverse without 2nd 3rd.

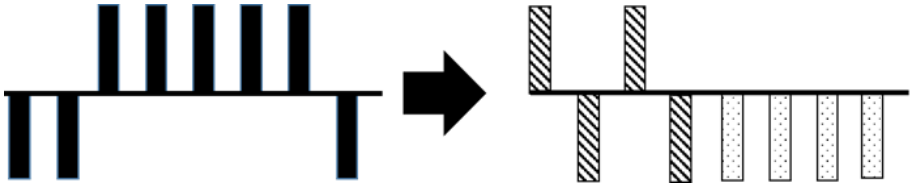

図 5.11: The example of the generation of a dummy traffic by PNPN.

評価手法　秘匿サービスが生成するトラフィックのクラス (HS) のデータ 195 個, 偽装トラフィックのクラス (dummy) のデータ 220 個に対し, データセットを学習用と評価用にわけ, データマイニングツール Weka に実装された k 近傍法そして SVM を用いた学習, 分類を行う. 偽装トラフィックは前述した四種類の生成方法の他に, 比較として 5.2 節で説明したダミーパケットを加えたトラフィックも用いた. ただしデータのわけ方に結果が依存しないようにするために, 10 回の交叉検定を行った. k 近傍法は, k=1 とした最近傍法とし, 重みは距離の逆数を用いる. HS クラスのデータに対し, HS クラスと推定できなかったものを False Negative, dummy クラスデータに対し HS クラスと推定された場合には False Positive とする. 前述したそれぞれのダミーパケットの加え方に対して, パラメータを動かしながら分類を行う. 評価としては下式 5.4, 5.5 で表した False Positive Rate(FPR), False Negative Rate(FNR) を用いる.

$$FPR = \frac{\#\text{ of False Positive}}{\#\text{ of dummy}} \tag{5.4}$$

$$FNR = \frac{\#\text{ of False Negative}}{\#\text{ of HS}} \tag{5.5}$$

5.3.4　結果と考察

まず, Simple Reverse と Reverse without 2nd3rd, そして 5.2 節で説明した手法によって生成した偽装トラフィックを分類した結果について説明する. 図 5.13, 5.14 に FPR, FNR に結果を示す. 左の図が k-NN による分析結果, 右の図が SVM による分析結果である. k-NN で分類した場合はどの手法もさほど有効とはいえず, FPR, FNR ともに最高 10%程度である. SVM では FNR が高くなる傾向があり, Simple Reverse と Random Cluster では FPR も 15%を超えるほどである. 一方で, Random-Each, Morph の分類

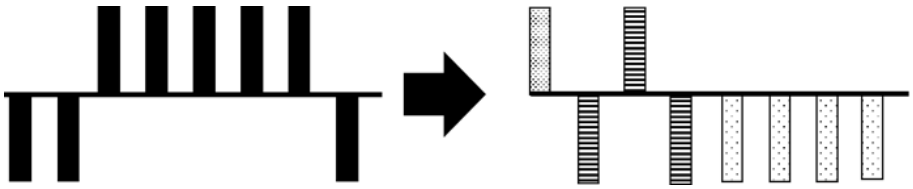

図 5.12: The example of the generation of a dummy traffic by Reverse NPN.

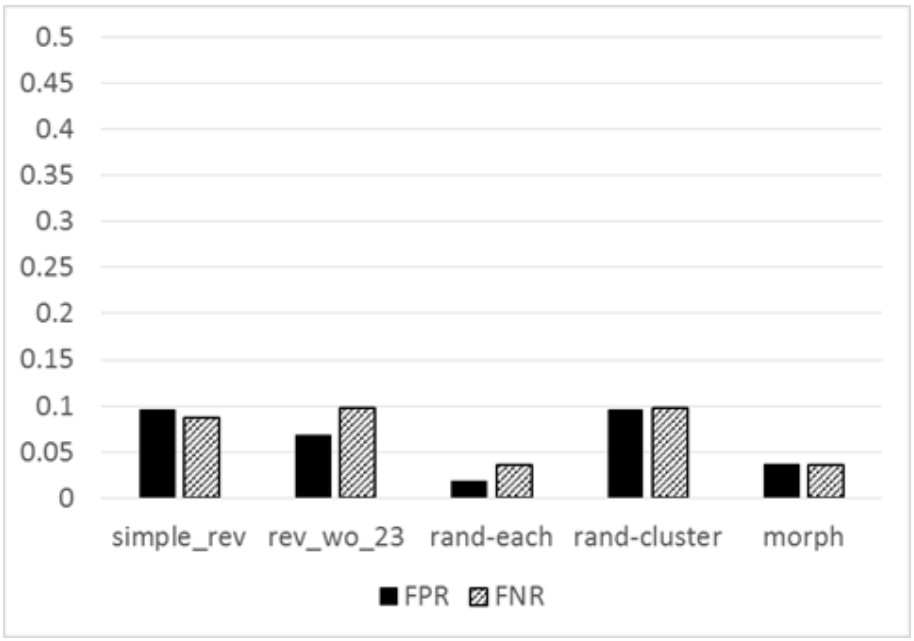

図 5.13: FPR and FNR of classification with knn.

精度が良い. k-NN のほうが全体的に見て分類精度が良い傾向があるが, これはこの実験のデータセットでもまだ多様なトラフィックが得られたとは言えず, 先頭パケットの向きの特徴量を考えると最近傍法では同じクラスのトラフィックが近い距離にあることが多いからであると考えられる. よって, 秘匿サービスがより多様になれば, k-NN の分類精度は悪くなる可能性があり, SVM による分類結果と同様の結果に近づくと考えられる.

次に, PNPN によって生成した偽装トラフィックと秘匿サービスのトラフィックを分類結果について, 図 5.15, 5.16 に示す. 10 回の実験を行った平均の値が示されており, エラーバーは標準偏差を表す. 横軸は, 外向きパケットを挿入する確率 p である. まず k-NN の分類結果を見ると, FPR は 15%前後, FNR は 15%を超える程度である. $p = 0.3$ のとき, 少しだけ FNR が高くなっている. 一方 SVM の分類結果を見ると, $p = 0.3$ で FPR は極大, FNR は極小となっており, FNR が上昇すると FPR が下降する傾向があると言える. これは秘匿サービスを秘匿サービスと判定する精度が下がっていく一方で, 偽装トラフィックを偽装トラフィックを判定する精度が上がっていることを示す.

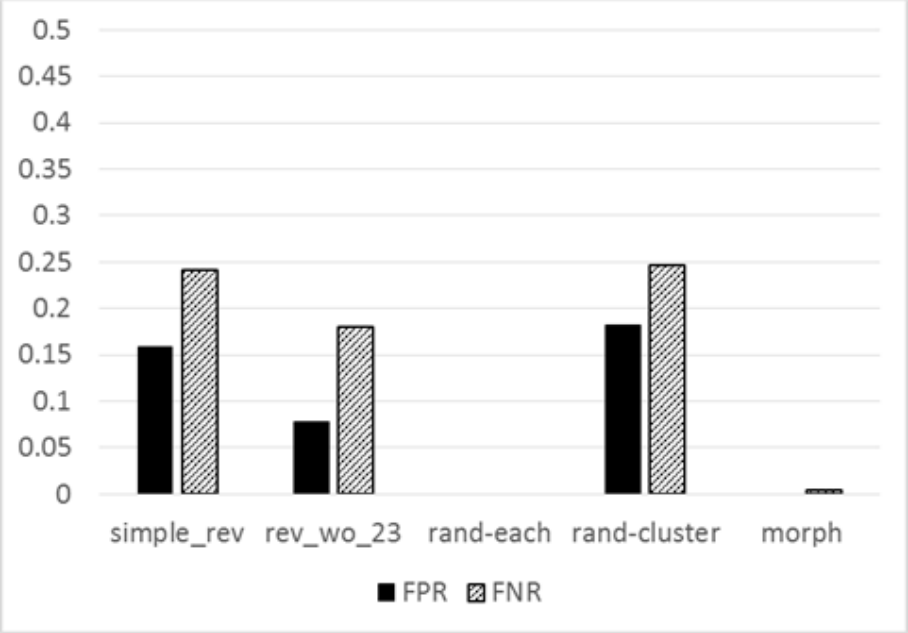

図 5.14: FPR and FNR of classification with svm.

これらの結果について, kNN に対してはこれまでに示した手法と同様, 先頭パケットの向きが強い特徴として扱われていると考えられる. $p = 0.3$ のときに FNR が高くなる理由として, このときの偽装トラフィックの内向きパケット, 外向きパケットの数の傾向は秘匿サービスのトラフィックと近くなっていたことが挙げられる. この時は内向きパケット, 外向きパケットの数が有効な特徴ではないため, 秘匿サービスらしい特徴が現れない. また, SVM の FPR について, PNPN では先頭の 4 パケットの向きを固定しているため, 先頭の 4 パケットの向きが内, 外, 内, 外であることは偽装トラフィックらしさとして学習される. よって偽装トラフィックを偽装トラフィックと分類する精度が $p = 0.3$ で高くなったのだと考えられる. 一方で, 先頭 4 パケットの向きが内, 外, 内, 外になっている秘匿サービスのトラフィックは多く, これらが偽装トラフィックと分類されたことで FPR が高くなったのだと考えられる.

最後に, Rev-NPN によって生成した偽装トラフィックと秘匿サービスのトラフィックを分類結果について, 図 5.17, 5.18 に示す. 10 回の実験を行った平均の値が示されており, エラーバーは標準偏差を表す. PNPN と同様, 横軸は, 外向きパケットを挿入する確率 p である. kNN による分類結果を見ると, 他の手法と比べ FPR が高くなりがちであるが, FNR は PNPN とさほど変わらない結果となっている. FNR, FPR 共に $p = 0.3$ で極大になる傾向がある. SVM による分類結果を見ると, $p = 0.2$ で FPR, FNR が極大になっており, 35%程である. 未だに kNN に対して有効な手法とは言えず, 多様な秘匿サービスに対して実験した際に精度がどのように変化するのか見る必要がある. 一方 SVM では, 全体的に精度が悪くなっており有効な手法であると言える. 先

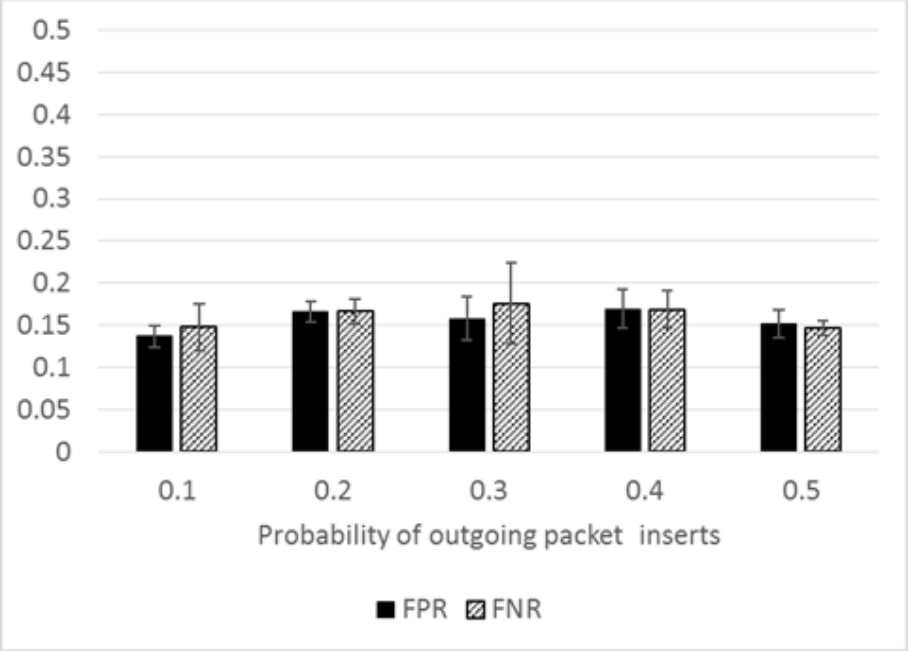

図 5.15: FPR and FNR of classification with knn. The dummy traffic is made by PNPN.

頭 4 パケットの向きが偽装トラフィックらしさとしても秘匿サービスらしさとしても特徴が弱くなり, また内向きパケットの数, 外向きパケットの数に関してもそれぞれのクラスで近い特徴があり, このような結果になっていると考えられる. さらに効果的な偽装トラフィックを生成する手法は, 先頭 10 パケットの向きのうち 5 番目以降のパケットについて傾向を詳しく分析し, 考慮したものであると考えられる.

この実験では, Rev-NPN による偽装トラフィックの生成によって, 秘匿サービスと偽装トラフィックの分類において, FPR, FNR を 35%まで上げることが出来た. この実験の課題点として, Rev-NPN でパケットの向きを決める確率についてはヒューリスティックなものになっている点である. また分類に用いる特徴量として先頭 10 パケットの向きを用いているが, シーケンスが十分に考慮されているとは言えない. これらの点については今後考慮していく必要がある. この後の実験では, 偽装トラフィックを生成するにあたり, 生成元になるトラフィックの選択や生成したトラフィックの運用について検討していく.

5.4　偽装トラフィックの影響評価と運用

本節ではこれまでに明らかにした秘匿サービスのトラフィックの傾向や, 偽装トラフィックの生成方法から実際の DHS を考えるにあたり, 偽装トラフィックを生成する元のトラフィックの選択方法や, 偽装トラフィックを学習データセットに占める割合がどの必

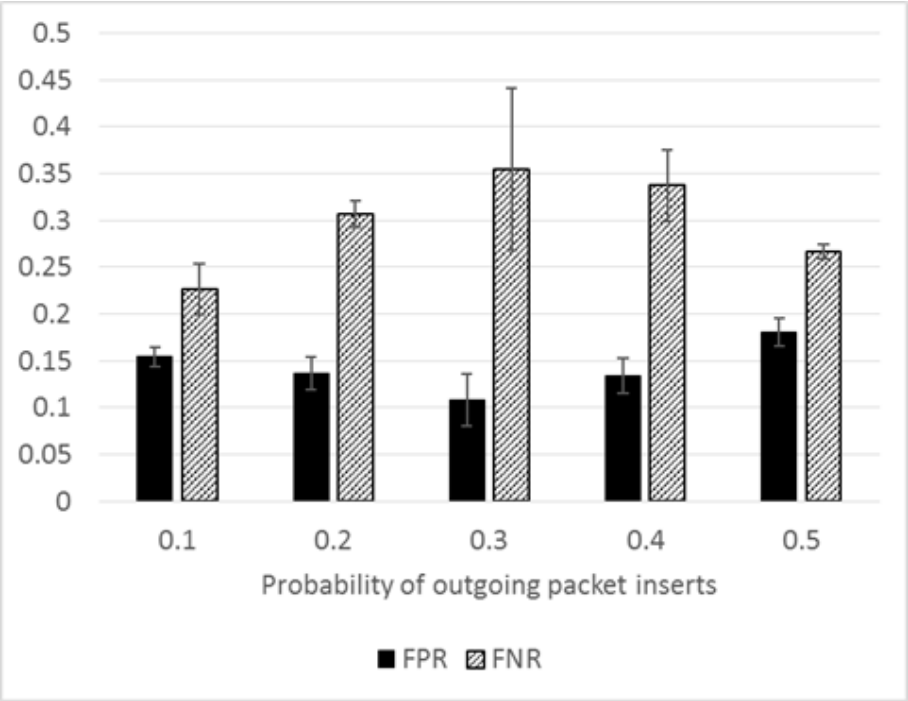

図 5.16: FPR and FNR of classification with svm. The dummy traffic is made by PNPN.

要か, より実際の Tor 環境に近いデータセットを用いて検討していく.

5.4.1　実験内容と環境

この実験では, クライアントが秘匿サービスに接続した際のトラフィックをもとに, 5.3.2 項で示した手法で偽装トラフィックを生成する. そして, 一般の Tor 通信のトラフィック (general), クライアントの通信のトラフィック (Client), 偽装トラフィックと秘匿サービスが生成するトラフィック (HS) のデータを対象に学習, 分類を行う. 一つ目の実験では, 偽装トラフィックを生成する量が分類精度に与える影響について調べる. 偽装トラフィックは, Client, general クラスから生成するが, これらは学習, 評価データセットにも用いられる. Client, general クラスデータの個数に対する, 偽装トラフィックのデータ個数の割合をパラメータとして分類精度の変化を調べる. 二つ目の実験では, 偽装トラフィックを生成する元のトラフィックの選択方法について実験する. 偽装トラフィックを生成する元の Client, general クラスのデータを全体のうち何割かに制限し, その割合をパラメータとして分類精度の変化を調査する.

特徴量　この実験の攻撃では CFA と同じ三種類の特徴量を用いた. 特徴量は最初の 50 個のパケットについての内向きパケットの数, 外向きパケットの数をそれぞれ一次元, 通信時間を表す DoA が一次元, そして最初 10 個のパケットの向きの十次元の十三次元

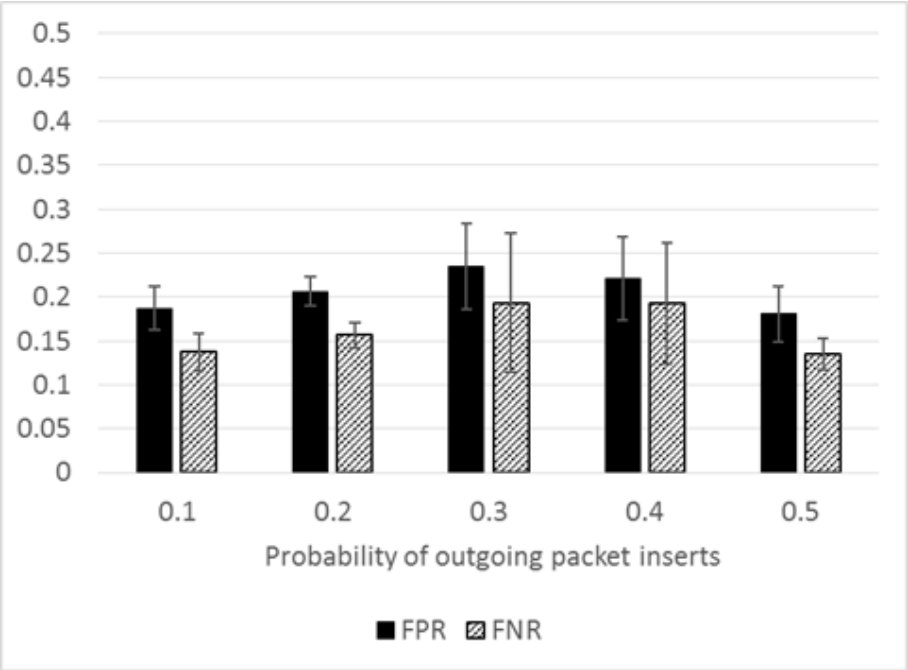

図 5.17: FPR and FNR of classification with knn. The dummy traffic is made by NPN.

となっている. ただし, 向きについては内向きを 1, 外向きを-1, 通信が終わってしまってデータが存在しないインデックスのパケットの向きを 0 とした.

データセット　5.3 節で用意した 220 個の Client トラフィック, 195 個の HS トラフィックは今回も用いる. また, 2017 年 1 月のアクセス分析サイト Alexa のアクセスランキング 150 から, リダイレクトされうるサイトを省いた 101 のサイトに対し Tor Circuit を介して接続し, このトラフィックをキャプチャした general トラフィックを 505 個のうち, データが得られなかった 3 個のトラフィックを省いて, 502 個のトラフィックを用いた. そして偽装トラフィックを Client, general クラスのトラフィックから生成する. 手法は 5.3 節の Rev-NPN をとり, パラメータは 2 番目が外向きになる確率が 85%, 3 番目が内向きになる確率が 80%, 4 番目が外向きになる確率が 75%とし, 5 番目以降のパケットの向きは 30%の確率で外向き, 70%の確率で元パケットの向きを反転させるとした. データ形式は 5.2 節と同様であり, 本節では簡単に述べる. ページのロード開始から終了までの全てのパケットをキャプチャし, 一つのデータとした. Tor 通信の基本単位である 512byte ごとにパケットを区切り, 分けられたそれぞれのパケットから時間と向きのみを取り出す. ここで, 512byte の長さを持たないパケットは削除される.

今回の実験ではまず, 偽装トラフィックを生成する個数をこの Client, general 全体の何割かに制限する. その割合をパラメータとし, 偽装トラフィックの生成すべき割合を明らかにする. 今回は 10%から 100%までの割合で 10%刻みに偽装トラフィックを混

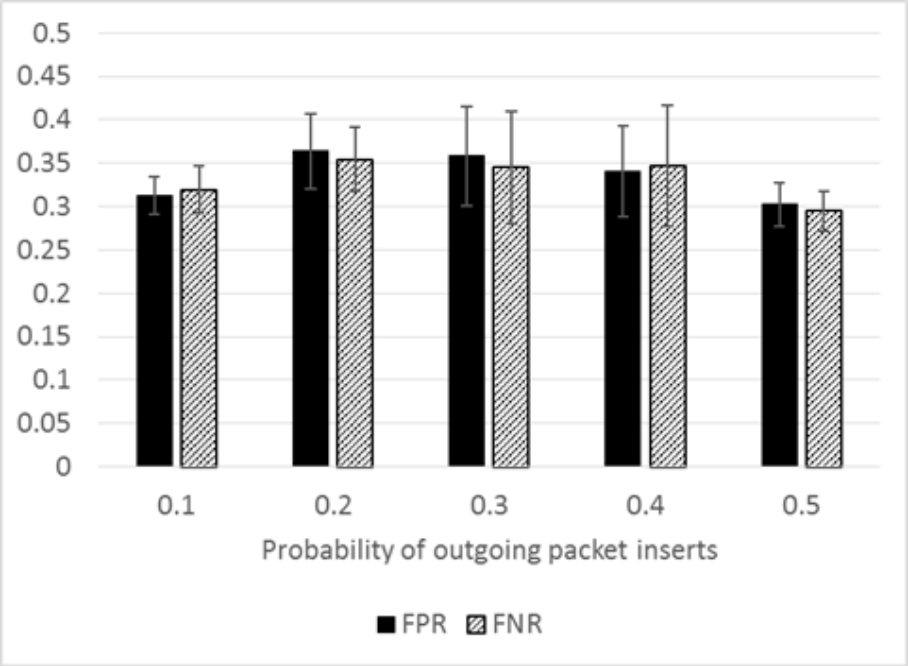

図 5.18: FPR and FNR of classification with svm. The dummy traffic is made by NPN.

ぜ, 分類実験を行った. この実験では, 偽装トラフィックの生成元になるデータは, 全ての Client, general クラスの中から一様に等しい確率で一つのデータを選択する. そして一度選ばれたデータはもう偽装トラフィックの生成元として選ばれることがない, という条件をつける. これは, 全てのデータから一回ずつ偽装トラフィックを生成し, その中から実験に使うデータセットをランダムに選択することに相当する. 次に, 偽装トラフィックの生成元となる Client, general クラスのデータの個数を制限する. その割合をパラメータとし, 偽装トラフィックの生成元となるトラフィックの多様さが分類精度に与える影響について調べる. 今回生成元となるトラフィックの個数を全体の 1%, 2%. 5%, 10%, 20%, 50%, 100% とし, 実験を行った. この実験では, Client, general クラスから先に述べた割合の個数のデータを生成元候補として選択する. そして後述する我々が定めた, 偽装トラフィックの生成すべき割合である, Client, general のデータ全体の個数の 40%の回数だけ生成元候補から偽装トラフィックを生成する. 毎回, 候補全ての中から一様に等しい確率で一つのデータが生成元として選択される. この実験では同じデータが何度も偽装トラフィックの生成元として選択される可能性がある.

評価手法　データセットを学習用と評価用にわけ, データマイニングツール Weka に実装された k 近傍法そして SVM を用いた学習, 分類を行う. ただしデータのわけ方に結果が依存しないようにするために, 10 回の交叉検定を行った. k 近傍法は, k=1 とした最近傍法とし, 重みは距離の逆数を用いる. 評価指標として, HS トラフィックの分類精度

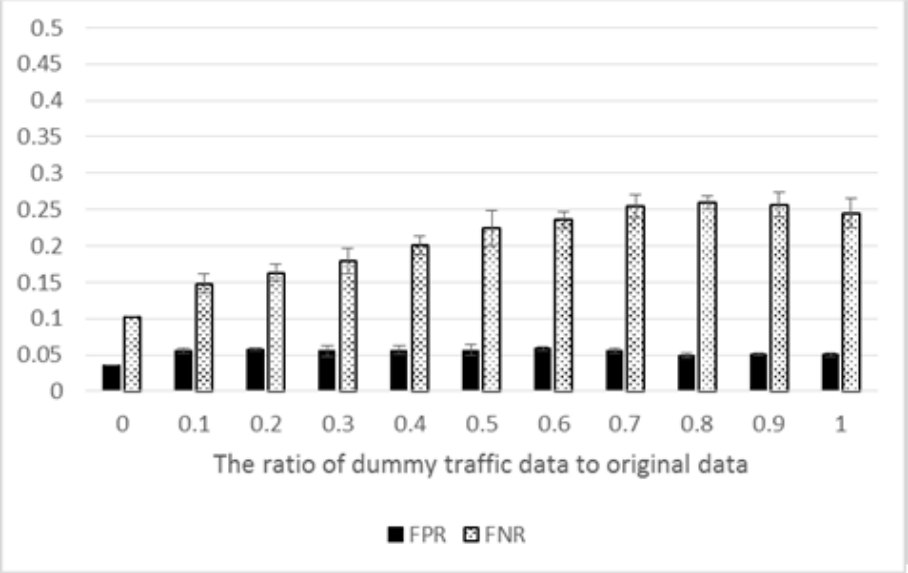

図 5.19: FPR and FNR of classification with knn. The ratio of dummy traffic varies.

に注目し, 下式5.4, 5.5 で表した False Positive Rate(FPR), False Negative Rate(FNR) を用いる.

$$FPR = \frac{\#\text{ of False Positive}}{\#\text{ of False Positive} + \#\text{ of True Negative}} \tag{5.6}$$

$$FNR = \frac{\#\text{ of False Negative}}{\#\text{ of False Negative} + \#\text{ of True Positive}} \tag{5.7}$$

ここで,

- HS トラフィックを HS クラスと正しく分類した場合を True Positive
- HS トラフィックを HS クラスではないクラスに分類した場合を False Negative
- HS クラスでないトラフィックを HS クラスと分類した場合を False Positive
- HS クラスでないトラフィックを HS 以外のクラスに分類した場合を True Negative

である.

5.4.2　結果と考察

偽装トラフィックの生成割合に関する実験　図 5.19, 5.20 に混ぜる偽装トラフィックの量を Client, general クラスのトラフィックデータの 10%から 100%までの割合で 10%ずつ変化させた際の, FPR, FNR を示す. 5 回の実験を行った平均の値が示されており, エラーバーは標準偏差を表す.

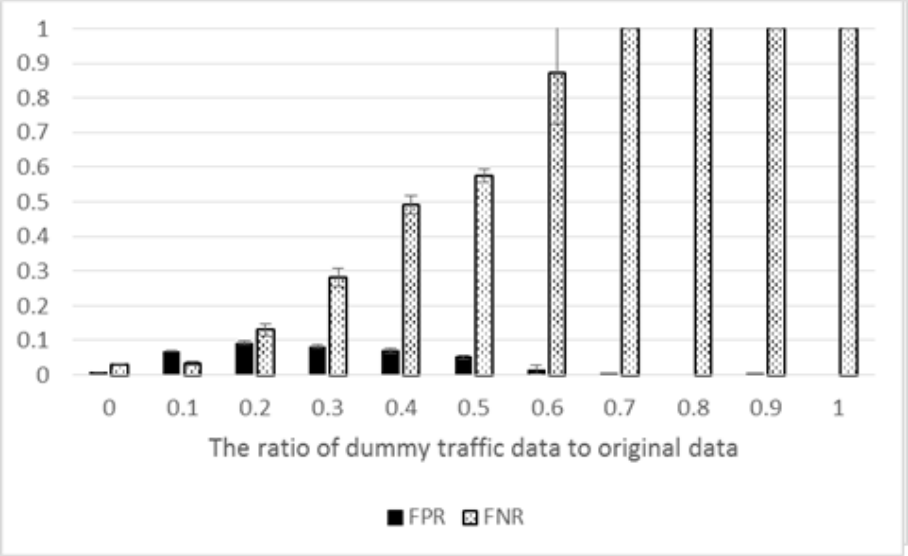

図 5.20: FPR and FNR of classification with svm. The ratio of dummy traffic varies.

図 5.19 は kNN による分類結果であり, 割合が 60%になるまでは徐々に FNR が上昇するが, それ以降は FNR が 25%程で安定する. 一方, FPR は 5%程を保っている. 最近傍法にとって重要なのは先頭 10 パケットの向きであることを 5.3 節で明らかにしたが, 偽装トラフィックを増やしても秘匿サービス同士で近い先頭 10 パケットのシーケンスを持つものは増えず, 今回の実験では FPR が上昇せず, 秘匿サービスの多様さによって上昇するのだと考えられる.

一方, 右図 5.20 は SVM による分類結果を示しているが, 偽装トラフィックを増やすにつれ FNR が上昇するが, FPR が低下していくことがわかる. 特に, Client, general クラスのトラフィックの 70%の量以上に偽装トラフィックを混ぜると, HS クラスのトラフィックを HS クラスと分類することができなくなり, 一方偽装トラフィックを偽装トラフィックと分類することができなくなる. 偽装トラフィックは HS クラスのトラフィック特徴をよく再現しているため, 偽装トラフィックの数が増えることで確かに秘匿サービスが偽装トラフィックと分類されることが多くなり, 逆に秘匿サービスが偽装トラフィックと分類されることが少なくなっていることがわかる. ここで, 実際には HS クラスのトラフィックを HS クラスと分類できない割合である FNR は 50%が最高である. これは, HS クラスであるかそうでないかの二つをランダムで決めるという手法が存在するからである. 偽装トラフィックの割合が 50%を超える際には, 攻撃者は HS クラスでないと分類されたトラフィックを HS クラスのトラフィックと分類してしまえば, ランダムな推定よりかえってより当たりやすくなる. そのため, FNR が 50%に達する, 40%のときに効率よく防御出来ていると言える.

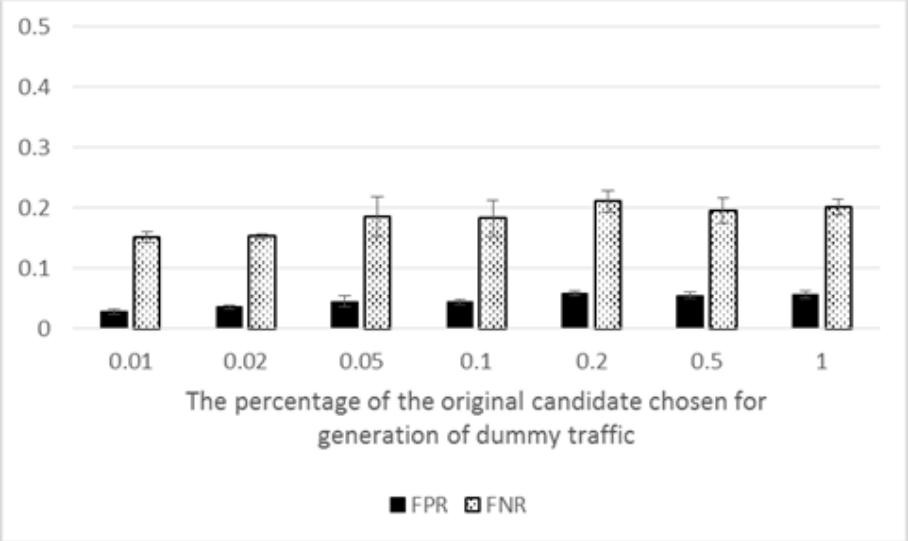

図 5.21: FPR and FNR of classification with knn. The percentage of the original traffic used to generate dummy traffic varies.

偽装トラフィックの生成元に関する実験　図 5.21, 5.22 に, 生成元となる Client, general クラスの候補の割合を変化させた際の分類結果の FPR, FNR を示す. 5 回の実験を行った平均の値が示されており, エラーバーは標準偏差を表す.

左図 5.21 は kNN による分類結果を示す. 生成元候補の割合が全体の 1%, 2%のあたりでは FPR が 3%程, FNR が 15%程であり, 割合が 5%以上のところでは, FPR が 5%程, FNR は 20%程である. 前述の通り, kNN では先頭 10 パケットの向きの特徴が重みとして強く, 候補があまりにも少ない状態では今回用いた生成手法では先頭 10 パケットの多様さを作り出せず, kNN が誤ることが少なかったのだと考えられる. 一方, 割合が 5%以上では FPR, FNR に有意な差はみられない. 5.3 節の Rev-NPN で生成した偽装トラフィックは, 生成元候補が増えても先頭 10 パケットの向きの多様さがそこまで増えないと言える.

一方, 右図 5.22 は SVM による分類結果を示す. 全体を通して, FPR は 5%程である. 一方 FNR は上下するものの決まった傾向はないが, 生成元候補が少ないときに分散が大きくなっている. SVM にとって重要な特徴は, 先頭 4 パケットの向きと先頭 50 パケットの中の内向き, 外向きパケットの数であるが, 候補元の選び方によって Rev-NPN がこれらを秘匿サービスらしく生成できなかったと言える. よって, SVM での分類結果に対しては, Rev-NPN は候補元の選び方が偏ると分類精度が低くならないのだと考えられる.

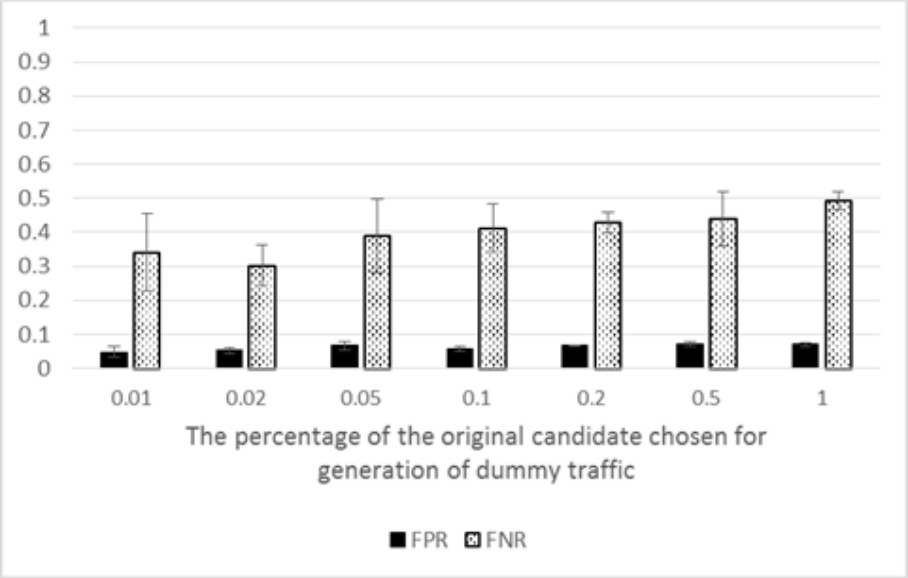

図 5.22: FPR and FNR of classification with svm. The percentage of the original traffic used to generate dummy traffic varies.

5.4.3　運用に関する考察

偽装トラフィックの生成候補　偽装トラフィックの生成候補元について, 5.4.2 項の結果から, 全体の 5%ほどで Rev-NPN で生成した偽装トラフィックの多様さが限度に達するが, 選び方によっては偏った偽装トラフィックになるといえる. また CFA の特徴量の多様さにはそもそも限度が存在し, 実際の Tor 環境の規模では 5%より少ない割合で十分な多様さが産まれると考えられる. よって, 候補元として, Client, general のトラフィックを公開情報としておきこれを DHS に用いてもらえばよい. また攻撃者この候補が知られても本節の実験の仮定の範囲内であり, 分類精度に与える影響は変わらないと言える. 偽装トラフィックを各 DHS ノードが作るということは, 自由に生成して良いということであるため, 攻撃者が各ノードを占拠し, 偽装トラフィックを生成させず秘匿サービスのトラフィックを流すかもしくは DHS を停止させることが考えられるが, DHS は Tor のエンドノードであり, また多数存在するため, 攻撃者の労力は非常に高いと言える.

偽装トラフィックの生成割合　次に偽装トラフィックをどの程度 Tor Network に流すかについて考える. ここでは. Client, general クラスのデータに対する偽装トラフィックのデータの割合を Rate とし, これを動かす問題として考える.

Rate を固定する場合は, 本節の実験の仮定と同様である. この時は, 攻撃者はあらかじめ本節と同じ実験を行っておき, 分類器の Rate ごとの精度を調べておく. Rate が固定されているということは Rate は公開情報であるため, 攻撃者は分類器の精度を知

ることが出来, エンドノードの予測に役立たせることは出来る.

Rate が時間によってランダムに変動する場合, こちらも攻撃者はどう動くか, どう動いたのかを知ることが出来る. Tor の通信が多い時間について考えると, HS クラスのトラフィックの先頭 10 パケットの向きが多様になり, kNN によって偽装トラフィックを誤って HS クラスと分類することが増えると考えられる. よって, この時間では SVM による分類が有効であると言える. Tor 通信が多い時間に偽装トラフィックが少ないと, その時間に学習データセットを作った攻撃者は SVM によって高い精度で攻撃が成功する. 一方で, Tor の通信が多い時間に偽装トラフィックを多くすると, その分 Tor Network に対する負荷が大きくなってしまう. また, Tor 通信が少ない時間に偽装トラフィックが多いと, SVM の分類精度を大きく下げることが出来る. kNN は HS クラスのトラフィックの多様さが少ないため, False Positive が少なくなると考えられるが, FNR には一定の効果がある. この時間に偽装トラフィックが少ないと, SVM も kNN も高い精度で攻撃が成功する一方で, Tor Network には余力があると言えるため, 不適切な状況である. よって, ランダムな遷移では攻撃者にとって有利な状況を産んでしまう.

各 DHS がランダム, または各ノードが自由に偽装トラフィックを生成する場合, Rate は攻撃者にもわからなくなる. つまり, 攻撃者は学習データセットにどの程度偽装トラフィックを混ぜればいいかわからない. 実際の偽装トラフィックの Rate 通りならば本節で述べた結果と同じになるが, 実際の偽装トラフィックの Rate が攻撃者の想定より少ないと, 攻撃者の分類器は本来秘匿サービスのトラフィックが持つの特徴を偽装トラフィックの特徴と学習してしまっているので, と考えられ, 秘匿サービスを謝って偽装トラフィックと判定する False Negative が増えると考えられる. 実際の偽装トラフィックの Rate が攻撃者の想定より多いと, 分類器は偽装トラフィックの特徴を十分に学習できず, 偽装トラフィックは秘匿サービスに近い特徴を持つので, 偽装トラフィックを誤って秘匿サービスとして判定する False Positive が増えるよって, 前後で FNR の変動が大きい Rate を基準としてユーザに提示しておき, 後に各ノードが各々の設定で DHS を利用することが有効な手段であると考えられる. なお, 本研究では図 5.20 から分かる通り, $Rate = 0.3$ が適していた.

Chapter 6　結論

本研究で我々は, 匿名通信システム Tor と, Tor を利用したサーバの匿名性を担保する技術である Tor 秘匿サービスを説明し, Tor 秘匿サービスの匿名性を暴く攻撃に対する防御手法に取り組んだ. Tor や Tor 秘匿サービスの匿名性は科学的に十分な検証がされておらず, 社会的な是非に関する議論が先行すべきではないという観点から, まず Tor 秘匿サービスに対する攻撃を説明し, 特に低いコストで攻撃に成功する Kwon らの Circuit Fingerprinting Attack(CFA) について詳しく述べた. CFA に対して我々は, Tor 秘匿サービスが生成するトラフィックと紛らわしいトラフィックである, 偽装トラフィックを生成するノードとして, Dummy Hidden Service(DHS) を提案し, DHS の満たすべき要件と本研究での実験の仮定について検討した. DHS は低コストでありながら CFA の分類精度を有意に下げ, またユーザに導入してもらいやすい実装が可能であり, ウェブページに対する指紋攻撃を DHS に対して行っても DHS であると判定することが難しいトラフィックを生成することが出来るということが要件として挙げられた. そして本研究の仮定では, 時間的要素を存在しないものとしており, これによって攻撃者はページのロード開始と終了が判別できることと, 全てのトラフィックが独立していること, そして各ノードのトラフィック発生頻度や時間は同一であることが仮定される. 後者の二つについては防御者有利と言える仮定であり, 今後より現実に近い仮定で研究を行う必要性があることを示した.

本研究の実験では, まず我々は, 外向きのノイズパケットのみを用いた簡単な偽装トラフィック生成が CFA の分類器の精度に与える影響について調べる実験を行い, 外向きのノイズパケットのみを加える場合や, そしてデータセットのデータが小規模である場合に分類器の精度を有意には下げられないことを明らかにし, これ以降の実験ではトラフィックを収集する際の Web ページ次に我々は, 秘匿サービスのトラフィックの分析を行い, その結果から偽装トラフィックの生成手法の提案とその評価実験を行った. 秘匿サービスのトラフィックは先頭 4 パケットの向きが内向き, 外向き, 内向き, 外向きになる傾向があることと, 先頭 50 パケットの中で外向きであるパケットの数は平均 9 個程であることを明らかにした. 偽装トラフィックは Tor クライアントのトラフィックから生成するが, 秘匿サービスのトラフィックの傾向から, 1 番目のパケットの向きは元のパケットの 1 番目の向きを反転させ, 2 番目から 4 番目のパケットの向きは確率で決め, それ以降のパケットの向きは外向きにするか元のパケットの向きを反転させることで生成した偽装トラフィックが効果的であると考え実験したところ, SVM を用いた分類器の精度を有意に下げることが出来た. また, この実験によって, k 近傍法は先頭 10 パケットの向きを特徴として重みを強くすることが分かった. 最後に, 偽装ト

ラフィックをどのトラフィックから生成し, どの程度生成するかを検討するための実験を行った. その結果, Tor クライアントのトラフィックの 40%の量の偽装トラフィックが適切であることと, 生成元となるトラフィックの候補の数は, 分類精度に大きい影響を与えないが, 偏った特徴をもった候補だけでは, 分類精度を有意に下げることが出来ないことが分かった. さらに, この実験の結果から DHS を運用する際偽装トラフィックをどのように生成するべきか検討を行った. DHS ノードごとに偽装トラフィックを生成する量を決めることが出来るようにし, また偽装トラフィックの生成元となる Tor クライアントのトラフィックは公開情報にしておくことが適切であると結論付けた.

謝辞

本研究を遂行するにあたり, 日頃から常にご指導をいただきました松浦幹太先生に感謝致します. 松浦先生のご指導で研究の進め方や, 結果のまとめ方を身につけることが出来ました. さらに, 研究への取り組み方や日頃から研究に持つべき姿勢, そして実験結果に対して一歩踏み込む姿勢をご指導いただき, 自分の研究をまとめることが出来ました.

また, 的確な助言をくださり, 内容について議論していただいた松浦研究室打ち合わせ参加者の皆様に心から感謝申し上げます.

参考文献

[1] Tails - privacy for anyone anywhere. https://tails.boum.org/.

[2] Thoughts and concerns about operation onymous. https://blog.torproject.org/blog/thoughts-and-concerns-about-operation-onymous.

[3] Tor security advisory: "relay early" traffic confirmation attack. https://blog.torproject.org/blog/tor-security-advisory-relay-early-traffic-confirmation-attack.

[4] Welcome to tor metrics. https://metrics.torproject.org/.

[5] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, pages 37–66, 1991.

[6] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *Proc. of the 2013 IEEE Symposium on Security and Privacy*, pages 80–94, 2013.

[7] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching from a distance: website fingerprinting attacks and defenses. In *Proc. of the 2012 ACM SIGSAC Conference on Computer and Communications Security*, pages 605–616, 2012.

[8] S. E. Coull, M. P. Collins, C. V. Wright, F. Monrose, and M. K. Reiter. On web browsing privacy in anonymized NetFlows. In *Proc. of 16th USENIX Security Symposium on USENIX Security Symposium*, pages 1–14, 2007.

[9] Roger Dingledine, Nick Mathewson, and Paul Syverson. "tor: the second-generation onion router". In *Proc. of the 13th USENIX Security Symposium*, 2004.

[10] Juan A. Elices, Fernando Perez-Gonzalez, and Carmela Troncoso. Fingerprinting Tor's hidden service log files using a timing channel. In *Proc. of the 2011 IEEE International Workshop on Information Forensics and Security*, pages 1–6, 2011.

[11] David M. Goldschlag, Michael G. Reed, and Paul F. Syverson. Hiding Routing Information. In *Proc. of the First International Workshop on Information Hiding*, 1996.

[12] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, pages 10–18, 2009.

[13] Felix Hernández-Campos, Kevin Jeffay, and F. Donelson Smith. Tracking the Evolution of Web Traffic: 1995-2003. In *Proc. of the 11th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 16–25, 2003.

[14] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naive-bayes classifier. In *Proc. of the 2009 ACM Workshop on Cloud Computing Security*, pages 31–42, 2009.

[15] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, pages 1527–1554, 2006.

[16] Marc Juarez, Sadia Afroz, Gunes Acar, and Claudia Diaz. A Critical Evaluation of Website Fingerprinting Attacks. In *Proc. of the 2014 ACM Conference on Computer and communications security*, pages 263–274, 2014.

[17] Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services. In *Proc. of the 24th USENIX Security Symposium*, 2015.

[18] Srdjan Matic, Platon Kotzias, and Juan Caballero. CARONTE: Detecting Location Leaks for Deanonymizing Tor Hidden Services. In *Proc. of the 2015 ACM SIGSAC Conference on Computer and Communications Security*, pages 1455–1466, 2015.

[19] Daniel Moore and Thomas Rid. Cryptopolitik and the Darknet. *Survival: Global Politics and Strategy*, pages 7–38, 2016.

[20] Lasse Øverlier and Paul Syverson. Locating Hidden Servers. In *Proc. of the 2006 IEEE Symposium on Security and Privacy*, pages 100–114, 2006.

[21] Andriy Panchenko, Lukas Niesse, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proc. of the 10th ACM workshop on Privacy in the Electronic Society*, pages 103–114, 2011.

[22] John C. Platt. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In Bernhar Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[23] Yi Shi and Kanta Matsuura. Fingerprinting attack on the tor anonymity system. In *Proc. of the 11th Information and Communications Security*, pages 425–438, 2009.

[24] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In *Proc. of the 23rd USENIX Security Symposium*, pages 143–157, 2014.

[25] Tao Wang and Ian Goldberg. Improved website fingerprinting on Tor. In *Proc. of the 12th ACM Workshop on Privacy in the Electronic Society*, pages 201–212, 2013.

[26] Tao Wang and Ian Goldberg. On Realistically Attacking Tor with Website Fingerprinting. Technical report, University of Wataloo, 2015.

[27] Matthew Wright, Micah Adler, Brian N. Levine, and Clay Shields. Defending Anonymous communications Against Passive Logging Attacks. In *Proc. of the 2003 IEEE Symposium on Security and Privacy*, pages 28–41, 2003.

発表文献

国際会議ポスター発表 (査読無し)

i Akira Takenouchi, Kanta Matsuura. "Using a dummy node to protect Tor Hidden Service from the threat of a link classifier", International Workshop on Security 2016. Tokyo, Japan, 9 月, 2016 年.

国内会議

ii 竹之内玲, 松浦幹太. "ダミーパケット挿入が Tor 秘匿サービスの匿名性に与える影響について", 2016 年 コンピュータセキュリティシンポジウム 2016（CSS2016）予稿集 CD-ROM, 1A3. 秋田, 10 月, 2016 年.

国内会議

iii 竹之内玲, 松浦幹太. "Tor 秘匿サービスへの攻撃に対抗する偽装トラフィック生成", 2017 年 暗号と情報セキュリティシンポジウム (SCIS2017) 予稿集 USB メモリ, 4D1. 沖縄, 1 月, 2017 年.

国内会議 発表予定

iv 竹之内玲, 松浦幹太. "学習データに加えられた偽装トラフィックが Tor 秘匿サービスへの攻撃に与える影響について", 2017 年 第 22 回セキュリティ心理学とトラスト研究発表会. 長崎, 3 月, 2017 年.

国際ワークショップ口頭発表 (査読無し) 発表予定

v Akira Takenouchi, Kanta Matsuura. "Dummy Hidden Service to Protect Tor Hidden Service", Indo-Japanese Workshop on Cryptographic Techniques for Cybersecurity, Indian Institute of Technology Roorkee. Roorkee, India, 2 月, 2017 年.