RouteDetector: 9軸センサ情報を用いた位置情報追跡攻撃

渡邉 卓弥 †　　秋山 満昭 ‡　　森 達哉 †

† 早稲田大学 基幹理工学研究科　　‡NTT セキュアプラットフォーム研究所
169-8555　東京都新宿区大久保 3-4-1　　180-8585　東京都武蔵野市緑町 3-9-11

あらまし 本研究は，スマートデバイスを携帯して鉄道で移動する人物の位置情報を追跡する新たなサイドチャネル攻撃の実現可能性を検証する．一般にモバイルアプリは GPS 等によって位置情報を取得するために，ユーザの許可を必要とする．一方，加速度，磁力，角速度等を計測するハードウェアセンサにアクセスする際は許可を必要としない．本研究の狙いは，ハードウェアセンサの情報のみを用いて人物の位置を推定することである．主な構成要素を以下に示す．まず，センサデータに対し教師あり機械学習を適用することで，標的の行動を歩行中，走行車両内，その他の 3 種に分類する．次に，時系列順に並んだ行動推定の結果から，列車の出発時刻と到着時刻のシーケンスを抽出する．最後に，得られた時刻シーケンスを鉄道の時刻表と路線図に照合し，標的の移動経路を特定する．上記のシステムを実装し，被験者実験によって集めたセンサデータと，172 の鉄道会社，9,090 の駅を対象にシミュレーションを行った結果，本攻撃による脅威が現実的であることを示した．

RouteDetector: Tracking your location with 9-axis sensors

Takuya Watanabe†　　Mitsuaki Akiyama‡　　Tatsuya Mori†

†Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555　　‡NTT Secure Platform Laboratories
3-9-11 Midoricho, Musashino-city, Tokyo 180-8585

Abstract We developed a novel, proof-of-concept side-channel attack framework, which identifies a route for a train trip by simply reading smart device sensors: an accelerometer, magnetometer, and gyroscope. All these sensors are commonly used by many apps without requiring any permissions. Our goal is to estimate the location of a user by reading sensors. The key technical components of the work can be summarized as follows. First, by applying a machine-learning technique to the data collected from sensors, RouteDetector detects the activity of a user, i.e.,“walking,”“in moving vehicle,”or“other.”Next, it extracts departure/arrival times of vehicles from the sequence of the detected human activities. Finally, by correlating the detected departure/arrival times of the vehicle with timetables/route maps collected from all the railway companies in the rider’s country, it identifies potential routes that can be used for a trip. We demonstrate that the strategy is feasible through field experiments and extensive simulation experiments using timetables and route maps for 9,090 railway stations of 172 railway companies.

1　はじめに

スマートフォンやスマートウォッチといった最新の端末は，実空間の情報を取得するためのハードウェアセンサを搭載している．加速度計や磁力計，ジャイロスコープ，気圧計や心拍計など，センサの種類と用途は多岐に渡る．ハードウェアセンサは，実空間と連動した革新的な体験をユーザに提供するが，一方で新たなサイドチャネル攻撃 [1, 2, 3, 4, 5] の危険性をもたらす．

本研究の究極の狙いは，GPS や WiFi，携帯基地局の情報を使った従来の位置情報サービスを用いることなく，センサ情報のみを用いてユーザの位置を特定する脅威の実証である．人物の移動経路は自由度が高く，居場所には無数の候補が存在するため，絶対的な位置を特定することは極めて困難である．そこで我々は，人間の移動様式に見られる時空間的な規則性 [6] に着目した．たとえば，ある人物が通勤や通学に用いる経路は限定される．また，公共交通機関は自然災害や鉄道事故等の問題が発生した場合を除き，定められた経路を規則的な時間に運行する．このような規則性を利用することで，人物の移動における自由度を削減する．

上述のアプローチに基づき，本研究ではセンサの情報から人物の移動経路を特定する新たな攻撃手法を開発し，その脅威の実現性を検証する．この手法を *RouteDetector* と命名する．RouteDetector は，加速度センサ，磁力センサ，ジャイロセンサの 3 種のハードウェアセンサを読み取ることで，標的となる人物が利用した列車の発着地点と時刻を推定する．これらのセンサは一般に 9 軸センサと呼ばれ，現在普及している多くのスマートデバイスに搭載されている．モバイルアプリは，特別な

パーミッションを宣言することなくこれらのセンサにアクセスすることができるため，ユーザは位置情報の漏洩を自覚することができない．RouteDetector の新規性は，単純なセンサデータの値を路線図や時刻表などの一般に公開されている外部の情報と組み合わせることで，ユーザのプライバシーに関連付ける点にある．

RouteDetector は以下に述べる 3 つの要素によって構成される．まず，スマートデバイスから収集したセンサデータに教師あり機械学習を適用することで，人物の状態を**歩行中**，**車両内**，**その他**の 3 種に分類する．次に，時系列順に並んだ人物状態の推定結果から，列車の出発時刻と到着時刻のシーケンスを抽出する．最後に，得られた時刻シーケンスを鉄道の時刻表と路線図に照合し，列車で移動する標的の経路を特定する．

本研究で得られた調査結果を以下に示す．

- 実際の鉄道路線で収集したセンサデータに RouteDetector を適用したところ，平均 6 秒未満の誤差で列車の出発時刻と到着時刻を検出した．
- 172 の鉄道会社が運営する 9,090 の駅を対象にシミュレーションを行った結果，標的が 6 駅以上を経由した場合に，RouteDetector が特定する候補経路数の平均値は約 1 程度に絞られる．

これらの結果は RouteDetector の脅威が現実的であることを示唆する．

本稿の構成は次の通りである． 2 章で RouteDetector が想定する脅威モデルを定義する． 3 章では RouteDetector を構成する要素を説明し，4 章で性能評価を行う．5 章でケーススタディを示し，6 章では本研究の制限事項と対策について考察する．7 章で関連研究を述べ，最後に 8 章で本研究のまとめを述べる．

2 脅威モデル

本研究の脅威モデルは，標的のスマートデバイスに，インターネットアクセスのパーミッションのみを要求する悪性アプリがインストールされていることを前提とする．アプリはセンサの値とその時刻を記録し続け，攻撃者のマシンに送信する．攻撃者は受け取ったデータを機械学習で分類し，標的の状態を**歩行中**，**車両内**，**その他**の 3 つのクラスに分類する．その後，時系列順に並んだ行動推定の結果を分析し，標的の移動経路を予想する．攻撃者は標的の使用する機種名を把握し，その機種に適した機械学習のモデルを用意する．例えば Android において，機種名は `Android.os.Build` クラスを呼び出すことで取得できる．また Android では *ACTIVITY_RECOGNITION* というパーミッションを宣言することで，ユーザの行動を推定する API を使用できるが，標的が攻撃を察知する可能性を排除するため，この機能を使用しない．脅威モデルは公共交通機関の利用者を対象としているため，公共交通機関が利用不可能な状況では，本攻撃は成立しない．さらに，RouteDetector による経路特定の結果を正確なものにするには，公共交通機関が時間通りに運行している必要がある．この問題については，本稿の 4 章にて論じる．その他の制限については本稿の 6 章にて検討する．

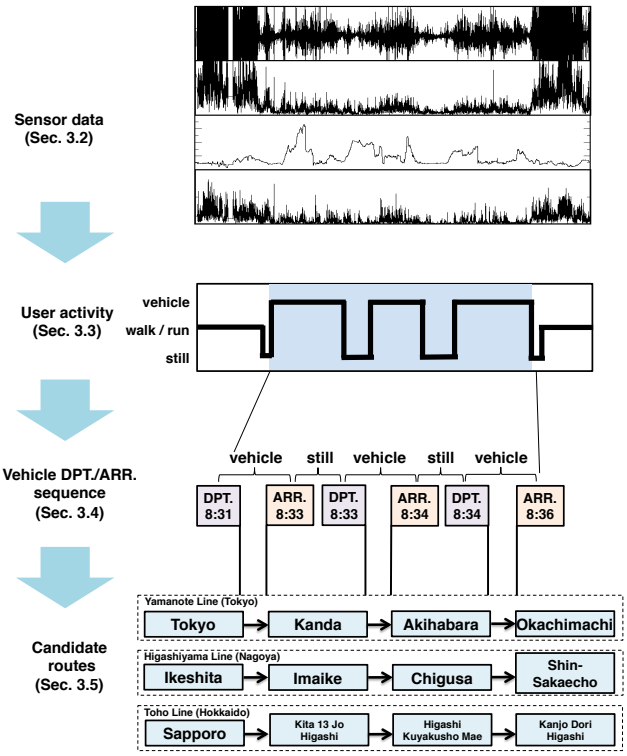

図 1: RouteDetector の概要

表 1: センサの概要

センサ	種別	単位	パーミッション
加速度計	物理センサ	m/s^2	不要
線形加速度	仮想センサ	m/s^2	不要
磁力計	物理センサ	μT	不要
ジャイロスコープ	物理センサ	rad/s	不要

3 RouteDetector

本章では，RouteDetector の全容を説明する．まず 3.1 節では，RouteDetector の概要を述べる．続いて 3.2 節で，使用したハードウェアセンサの情報を示す．その後，RouteDetector を構成する 3 つの要素について説明する．さらに 3.3 節では，行動推定について説明し， 3.4 節で発着時刻の抽出について説明する．最後に 3.5 節では，経路候補の列挙について説明する．

3.1 狙いと概要

本研究の狙いは，スマートデバイスのセンサ情報を読み取ることで，ユーザの利用する車両の移動経路を識別することである．ここで，車両が鉄道列車の場合，移動経路は経由駅の組として得られる．

RouteDetector の概要を図 1 に示す．最初にスマートデバイスに搭載されたセンサから値の読み取りを行う．ここでは，加速度，線形加速度，磁力，回転ベクトルの値を計測するセンサを対象とした．これらのセンサはいずれもアクセス許可を必要としない．データ収集の詳細は 3.2 節で説明する．続いて，収集したセンサデータを基に時刻ごとにユーザの行動を推定する．ユーザの行動を予測し，歩行中，走行中の車両内，その他（主に静止状態）の 3 クラスに分類する．分類においては，センサデータに前処理を施した後，教師あり機械学習を適用する．機械学習のアルゴリズムには，教師ありの多クラス分類において高い精度を発揮することで知ら

れるランダムフォレストを採用した．データの前処理及び機械学習の詳細は 3.3 節で説明する．さらに，時系列順に並んだ行動推定の結果から，列車の出発時刻と到着時刻のシーケンスを抽出する．行動状態が**その他**から**車両内**に切り替わる箇所が出発時刻を示し，**車両内**から**その他**に切り替わる箇所が到着時刻を示す．なお本手法は，ユーザの電車の乗り換えにも対応している．出発時刻と到着時刻のシーケンスの抽出方法については，3.4 節で説明する．最後に，得られた発着時刻のシーケンスから移動経路の候補を列挙する．候補の列挙のために，鉄道会社が提供している時刻表と路線図を用いて，標的が用いた出発駅，経由駅，到着駅を識別する．発着時刻から通過した駅を探索するために，幅優先探索による高速なアルゴリズムを実装した．経路候補の列挙についての詳細は，3.5 節で説明する．

3.2 ハードウェアセンサ

一般的なスマートデバイスに搭載されているセンサのうち，加速度センサ，線形加速度センサ，磁力センサ，ジャイロセンサの 4 つを採用した．加速度は端末の加速度，線形加速度は加速度から重力の影響を除いたもの，磁力は磁場のであり，ジャイロセンサは角速度を計測する．使用するセンサの概要を表 1 に示す．Android 端末のセンサは，物理センサと仮想センサの二種類に分かれる．物理センサは端末に搭載されたセンサの値を直接読み取るものであり，仮想センサは物理センサの値に計算処理を施すことで値を求める．加速度センサ，磁力センサ，ジャイロセンサは物理センサであり，線形加速度センサは仮想センサである．これら 4 つのセンサは市販される多くのスマートデバイスに搭載されており，高精度で行動推定を実現するために有用である．また，これらのセンサはアクセスの際にユーザの許可を必要としないため，悪意のあるアプリによって利用される危険性が高い．なお，照度センサや近接センサについても実験を行ったが，本研究で必要とする行動推定の精度向上には寄与しなかった．

我々は上記 4 種類のセンサデータの値と，取得した時刻を記録する Android アプリを開発した．アプリは 10Hz，すなわち 1 秒あたり 10 回の頻度でセンサの値を読み取る．この周波数は，少ない電力消費量で高い精度の行動推定を実現する．また，アプリにはユーザの行動を手動で記録するための機能を搭載した．これは行動推定の正解ラベルを作成するために使用する．

3.3 ユーザの行動推定

収集したセンサデータを解析し，ユーザの行動を**徒歩**，**車両内**，**その他**の 3 つのクラスに分類する．ここで**車両内**とは，移動中の車両を意味する．駅で停止中の列車の中でユーザが起立している場合，**その他**に分類する．まず，3.3.1 項でセンサデータに対し前処理を施す．次に 3.3.2 項で，前処理済みデータに対し教師あり機械学習を適用し，ユーザの行動を推定する．

3.3.1 データの前処理

収集した時系列センサデータの値に対して前処理を適用する．まず，端末の向きによる影響を除去するため，三次元の成分を持つセンサデータをノルムに変換する．すなわち，$a = \sqrt{a_x^2 + a_y^2 + a_z^2}$ を計算する．図 2 の (a) と (b) は，ノルム計算後のセンサデータの一例である．その後，時系列データを複数のブロックに分割する．1 つのブロックは N 個のサンプルを含む．すなわち，ブロック b_i はデータ $\mathbf{D}^{(i)}(a) = \{a_1^{(i)}, a_2^{(i)}, \ldots, a_N^{(i)}\}$ を持つ．ここでは経験的に $N = 20$ とした．センサデータは 10Hz の頻度で収集しているため，1 つのブロックは 2 秒間に相当する．さらに，各ブロックが示す人物の行動を分類するために，特徴ベクトルを定義する．1 つのブロックに含まれる 20 個のサンプルについて平均，標準偏差，最大値，最小値を計算し，各ブロックの特徴ベクトルとした．これらは代表値と呼ばれ，時系列データを要約した値である．最後に，算出されたそれぞれの値について平均と標準偏差を計算し，平均を引いて標準偏差で割ることで正規化する．結果として，1 つのブロックにつき $4 \times 4 = 16$ 次元の特徴ベクトルを得る．

3.3.2 機械学習の適用

前処理済みのデータを**歩行中**，**車両内**，**その他**の 3 つのクラスに分類する．教師あり機械学習のアルゴリズムとしてランダムフォレスト [7] を採用した．ランダムフォレストは多クラス分類において高い精度を発揮するという点で，本実験の目的に適している．我々は他にも，機械学習のアルゴリズムとして SVM を用いた実験を行った．精度に大きな差はなかったが，総じてランダムフォレストの結果が上回った．

3.4 出発時刻と到着時刻の抽出

前節の行動推定の結果を基に，出発時刻と到着時刻のシーケンスを抽出する．分類されたクラスのうち，**車両内**の開始と終了が列車の出発と到着に対応している．ここで，図 2 (c) に示す通り，行動推定の結果は誤分類によるノイズを含む．正しい発着時刻を検出するために，人物の行動の連続性に着目した．例えば，現実には多くの場合で**車両内**の状態は 1 分以上継続する．この特性を利用し，ノイズを除去するためのアルゴリズムとして，指数加重移動平均 (EWMA) を適用した．

A_n をブロック n における行動推定の結果とし，$\mathcal{W}$ を**歩行中**，$\mathcal{V}$ を**車両内**，O を**その他**とする．また，$W_n = \mathbf{1}_{\mathcal{W}}(A_n)$，$V_n = \mathbf{1}_{\mathcal{V}}(A_n)$，$O_n = \mathbf{1}_O(A_n)$ と定義する．ここで，$\mathbf{1}_Y(x)$ は，x が Y に含まれるならば 1 を与え，そうでなければ 0 を与える関数である．

はじめに，V_n について EWMA の値 $\overline{V_n}$ を計算する．

$$\overline{V_n} = \lambda V_n + (1 - \lambda)\overline{V_{n-1}},$$

$0 \leq \lambda \leq 1$ はデータに対する重みであり，0 に近いほど前ブロックまでの推定結果を重視し，1 に近いほど当該ブロックの推定結果を重視する．後に 4.3 節で述べる通り，λ の値は経験的に定めた．図 2 (d) に EWMA による平滑化の結果を示す．EWMA を適用した後，人物の行動を，次式にしたがって再度判定する．

$$\widehat{V_n} = \begin{cases} 1 & \text{if } \overline{V_n} \geq 0.5 \\ 0 & \text{if } \overline{V_n} < 0.5. \end{cases}$$

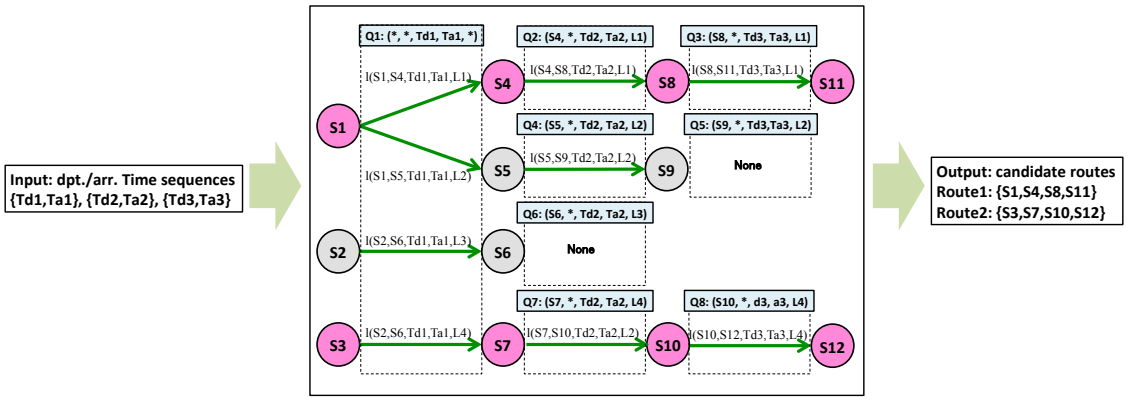

図 3: 経路候補検出のアルゴリズム

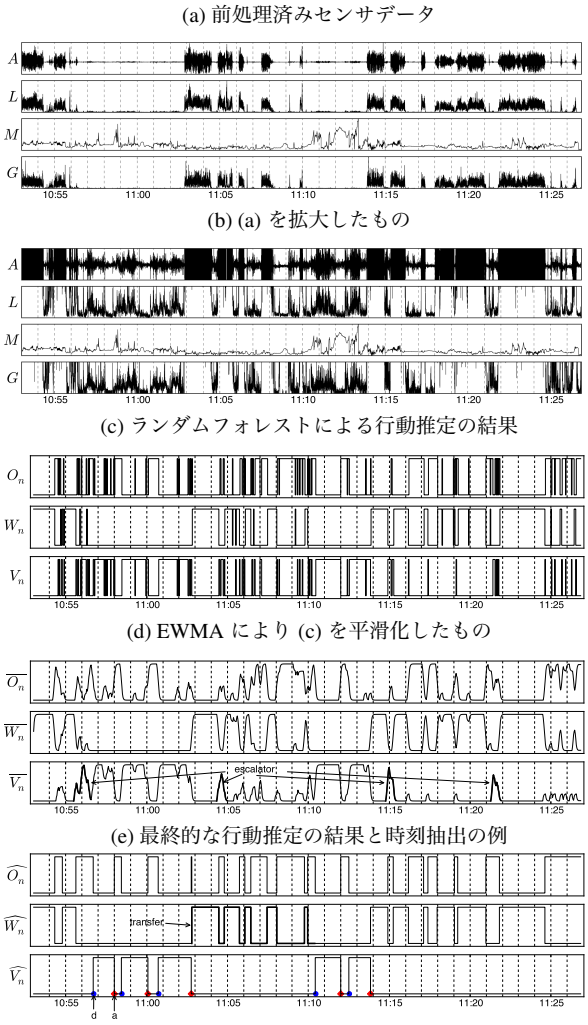

図 2: センサデータから行動を推定し，時刻抽出するまでの流れ．(a)，(b) の A は加速度，L は線形加速度，M は磁力，G はジャイロスコープを示す．(a)，(b) の Y 軸はセンサの値の大きさであり，(c)，(d)，(e) の Y 軸は行動推定の判定である．(e) で $\widehat{V_n}$ における円形の点は出発時，菱型の点は到着時を示す．

続いて，$\widehat{V_n}$ を用いて，列車の出発時刻と到着時刻のシーケンスを抽出する．アルゴリズムを以下に示す．

Algorithm 1 出発時刻と到着時刻の抽出アルゴリズム

```
1:  D =false                                              ▷ Initial state
2:  for all n = 1, 2, ... do
3:      if V̂_n = 0 AND V̂_{n+1} = 1 then
4:          T_d = t_{n+1}                                 ▷ t_n is time at block n.
5:          D =true                                       ▷ A vehicle has been departured.
6:      if V̂_n = 1 AND V̂_{n+1} = 0 AND D = 1 then
7:          T_a = t_{n+1}
8:          D =false
9:          if T_a − T_d > τ then
10:             return T_a, T_d
```

上記アルゴリズムは $\{n; \widehat{V_n} = 0\}$ となる場合，すなわち**歩行中**や**その他**に対しても，W_n 及び O_n について同様に適用できる．W_n と O_n を観測することで，乗り換えの有無を判定することができる．例えば，**車両内** (3 分間)，**歩行中** (2 分間)，**その他** (4 分間)，**車両内** (5 分間)，という一連の推定結果が得られた場合，標的は乗り換えを行ったと判断できる．最終的な行動推定の結果と，発着時刻の抽出を図 2 (e) に示す．このケースでは，まず出発駅で列車に乗り込み，2 駅を経由する．次に駅に到着し，$\widehat{W_n}$ が示す乗り換えの歩行を行う．さらに乗り換えた電車で 1 駅を経由し，目的地の駅に到着した．

今回ランダムフォレストによって構築した分類器は，エスカレーター上にいる人物を**車両内**と分類することがある．この誤分類を，アルゴリズムの 9 行目に示したヒューリスティックによって除去する．ここでは**車両内**の継続時間が τ より長い場合のみ，発着時刻を抽出する．経験的に $\tau = 60$ と定めた．例えば，，図 2 (d) ではエスカレータを**車両内**と見なす誤分類が発生しており，(e) ではヒューリスティックによって除去されている．

3.5 経路候補の列挙

最後に，抽出された発着時刻を用いて，経路候補を列挙する．経路候補を列挙する手順を以下に述べる．はじめに，鉄道会社の提供する路線図からノード（駅）とリンク（駅間を走る各列車）を定義し，1 つのグラフを作成した．次に，時刻表を用いて，リンクに出発時刻と到着時刻の情報を付与した．以下，これを鉄道グラフと呼ぶ．鉄道グラフにおける 1 つのリンクは $l(A, B, T_d, T_a, L)$ と表現できる．A は出発駅，B は到着駅，T_d は出発時刻，T_d は到着時刻，L は路線を表す．ここで，全鉄道グラフの情報は膨大なものになるが，探索の際には事前

表 2: 本実験で使用した端末

機種名 (略称)	種別	OS
HTC J Butterfly (HTC)	スマートフォン	Android 4.1.1
Nexus 7 (Nexus)	タブレット	Android 4.4.4

に全ての鉄道グラフを構築する必要はない．本手法では，リンクのデータベースのみを事前に用意し，探索アルゴリズムを適用する際に動的にグラフを生成する．

経路候補を探索するアルゴリズムを図 3 に示す．まず，出発時刻，到着時刻のシーケンスである Ta j、Td j(j = 1,2,3) を入力として与える．入力に対し，$l(*,*,T_d1,T_a1,*)$ を満たすリンクを抽出する．図中ではこれらのリンクを Q1(クエリ 1) とした．この例では，$(S1,S4,T_d1,T_a1,L1)$, $(S1,S5,T_d1,T_a1,L2)$，$(S2,S6,T_d1,T_a1,L3)$, $(S3,S7,T_d1,T_a1,L4)$ の 4 つのリンクが得られた．こうして得られたリンクについて，ノード (駅) を介して隣接関係にあり，なおかつ入力された発着時刻を満たすリンクを，再帰的に探索していく．例えば，$(S1,S4,T_d1,T_a1,L1)$ に続くリンクとして，$l(S4,*,T_d2,T_a2,L1)$ を満たすリンクを探索すると，駅 $S4$ を時刻 T_d2 に出発し，駅 $S8$ に時刻 T_a2 に到着する路線 $L1$ のリンクが該当した．これを図中の Q2 に示す．探索の結果，該当するリンクが存在しなかった場合，探索を中断し，その経路を候補から除外する．図中の Q5，Q6 はその例である．以上の処理によって，入力された発着時刻のシーケンスを満たす経路が列挙される．今回の例では，経路 $\{S1,S4,S8,S11\}$ と経路 $\{S3,S7,S10,S12\}$ が候補となる．

最後に，もし複数のルートが列挙された場合に，より標的が利用した可能性が高い経路を選択するため，経路ごとにスコアを計算する．このスコアは，ある経路の利用者の多寡を擬似的に計算したものである．ある経路に含まれる各リンクについて，同じ出発駅と到着駅の組を持つすべてのリンクをデータベースから抽出し，リンクの数を数える．得られた数の合計をその経路のスコアとする．スコアの値が高ければ高いほど，その経路の利用者数が多いと考えられる．したがって，標的はスコアが上位の経路を利用した可能性が高い．

4 性能評価

本章では，RouteDetector の性能評価を行った結果を示す．はじめに，本研究で用いたデータセットについて述べる．次に，ユーザの行動推定の精度を評価する．その後，出発時刻と到着時刻の抽出について精度を評価する．最後に，経路候補の列挙の有効性を示す．

4.1 データセット

RouteDetector の実装と評価のために，本研究では 2 種のデータを収集した．1 つ目は実際にスマートフォンで記録したセンサデータである．このデータは，出発時刻と到着時刻の抽出とその評価のために使用する．2 つ目は日本全国の鉄道の路線図と時刻表である．このデータは，経路候補の列挙を行う際，時刻データ付き鉄道グラフの生成に使用する．

表 3: 収集したセンサデータの統計

データ名	端末	所持状態	駅数	路線数	ブロック数
HTC_H	HTC	H	57	5	12,007
HTC_B	HTC	B	29	1	2,561
Nexus_H	Nexus	H	29	1	2,543
Nexus_B	Nexus	B	54	5	8,576

表 4: 路線図と時刻表の統計

鉄道会社の数	路線数	駅数	リンク数
172	597	9,090	2,277,397

4.1.1 センサデータ

実験で使用したスマートデバイスを表 2 に示す．異なるハードウェアに搭載されたセンサは，同じ入力に対して異なる値を出力する．これはセンサチップの違いや，端末の形状の違いに由来する．したがって，機械学習を適用する際に機種ごとのモデルを用意する必要がある．端末間の差異に関する詳細は 6 章で論じる．

収集したセンサデータの統計を表 3 に示す．センサデータは，2 つの鉄道会社，7 つの路線の列車に実際に乗って計測した．路線の内訳は，JR 東日本が運営する山手線、中央線、京浜東北線、埼京線の 4 つの路線と，東京メトロが運営する副都心線、丸ノ内線、南北線の 3 つの地下鉄路線である．表に示した通り，端末の所持状態は，手に持った状態 (H) と鞄の中に入れた状態 (B) で別々に記録した．端末を入れた鞄は，膝の上や網棚の上に置くことを想定している．2 章で述べたように，攻撃者は標的の使用する機種を容易に識別でき，またスマートフォンを手で握っているか鞄に入れているかを，光センサや近接センサから識別できる．

4.1.2 鉄道路線図と時刻表

前項でセンサデータを収集した場所は一定の範囲に限定されるが，鉄道グラフは日本のほぼ全ての在来線，新幹線，モノレールを網羅している．よって RouteDetector は，日本で鉄道を利用する全てのユーザに対して適用できる．収集した路線図と時刻表に関する統計を表 4 に示す．リンクは，$l(A,B,T_d,T_a,L)$ として 3.5 項で定義したものである．ここで，あらかじめ標的の行動範囲が限定されている場合，標的の移動経路をより一意に特定しやすくなる．例えば標的が京都府で生活していることがわかっているならば，京都府の路線図と時刻表のみを用いて鉄道グラフを生成すればよいため，候補数を大幅に削減できる．

4.2 ユーザの行動推定

行動推定の精度を評価するため用いたデータの統計を表 5 に示す．ランダムフォレストのパラメータは，経験的に $n = 50$，$m = 4$ とした．ここで，n は決定木の数，m は決定木の作成に使用する特徴量の数である．分類精度の評価に際して，10 分割のクロスバリデーションを，乱数のシードを変更して 10 回実施した．また，3 つのクラスのうち，**車両内**を検出する精度を評価した．出発時刻と到着時刻のシーケンスを抽出する上では，**車両内**か否かが最も重要であるためである．正解が**車両**

表 5: 行動推定に用いたラベル付きデータの統計

データ	車両内	歩行中	その他
HTC_H	609	1,327	510
HTC_B	691	1,360	510
Nexus_H	686	1,352	505
Nexus_B	602	1,304	505

表 6: **車両内**の分類性能，それぞれ平均/標準偏差と記す．

データ	精度	FNR	FPR
HTC_H	0.941/0.011	0.042/0.022	0.078/0.013
HTC_B	0.965/0.009	0.024/0.012	0.047/0.014
Nexus_H	0.943/0.013	0.041/0.014	0.074/0.021
Nexus_B	0.969/0.009	0.023/0.012	0.041/0.016

内のブロックを**歩行中**や**その他**と分類した場合，False Negative とする．反対に，正解が**歩行中**や**その他**のブロックを**車両内**と分類した場合，False Positive とする．

表 6 に示す通り，全てのケースで高い分類精度となった．また，手に持った場合と鞄の中に入れた場合では，鞄の中のほうがより高い精度となった．原因として，手で持った方が端末の状態が不安定であり，加速度や角速度のノイズが増加するためと考えられる．

4.3 出発時刻と到着時刻シーケンスの抽出

続いて，ユーザの行動推定の結果から，出発時刻と到着時刻のシーケンスを抽出するアルゴリズムの性能を評価する．収集したセンサデータのうち，30 駅分の発着時刻を評価に用いた．30 個のデータを 20 個の訓練データと 10 個の評価データに分割する．乱数のシードを変えて，3 分割のクロスバリデーションを 10 回行った．まず訓練データを用いて，EWMA における λ の値を最適化した．λ の値は，本手法で抽出した発着時刻（推定時刻）と正解データの発着時刻（観測時刻）の誤差が最も小さくなるようになるよう定めた．この時の推定時刻と正解時刻の誤差を表 7 に示す．表の通り，推定時刻と観測時刻の誤差は極めて小さい結果となった．最大でも 3-11 秒程度であり，平均すると 2-6 秒程度となる．また本手法は，列車の停車と発車を 1 つも見逃すことなく検出した．このように，RouteDetector は列車の発着時刻を極めて正確に検出する．

正解データの時刻は手動で記録した観測時刻であり，必ずしも時刻表の時刻 (定刻) と一致しない．観測時刻と定刻の誤差について分布を図 4 に示す．全体として観測時刻と定刻の誤差は小さい．観測したうち，約 85%の列車は定刻から 60 秒以内の遅れで出発した．また，約 75%の列車は定刻の前後 30 秒以内の時刻に到着した．

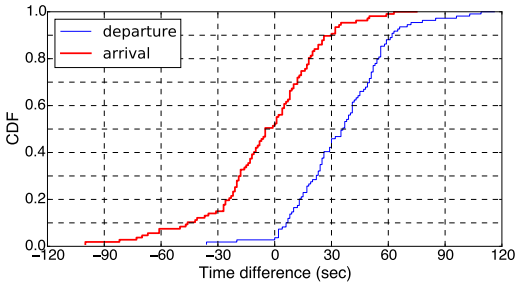

図 4: 観測時刻と定刻の差の分布

表 7: 抽出された時刻と観測された時刻の誤差

出発時刻の誤差				
データ	最大値 (秒)	最小値 (秒)	平均 (秒)	標準偏差
HTC_H	1.97	3.54	2.79	0.46
HTC_B	2.04	3.06	2.53	0.23
Nexus_H	2.33	7.94	4.60	1.84
Nexus_B	1.55	2.76	2.17	0.24
到着時刻の誤差				
データ	最大値 (秒)	最小値 (秒)	平均 (秒)	標準偏差
HTC_H	2.52	6.75	4.13	1.18
HTC_B	1.71	4.63	3.21	0.77
Nexus_H	3.07	10.78	6.03	2.22
Nexus_B	2.22	5.16	3.43	0.80

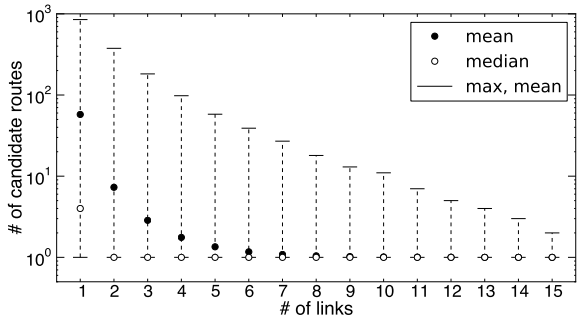

図 5: リンクの数と経路候補の数の関係

4.4 経路候補の列挙

最後に，抽出された発着時刻から経路候補を列挙する手法の評価を行う．まず，表 4 に示したデータより作成した鉄道グラフに対して，考えうる全ての経路を走査し，発着時刻のシーケンス群を得る．ここで組み合わせ爆発を回避するため，リンクの数は 15 個まで，乗り換えの回数は 2 回までとした．その後，各時刻シーケンスに対していくつの経路が候補として列挙されるかを数える．以上のシミュレーションによって，入力されるリンクの数に対し，列挙される経路候補の数を計算する．シミュレーションの結果を図 5 に示す．例えば，標的が 6 リンク，すなわち 6 駅分の列車移動を行うとき，移動経路を平均で 1 に近い候補数に絞り込むことができる．また，たった 1 つの発着時刻 T_d, T_a を用いた時でも，約 50%の割合で 4 つ以下の候補に絞られることがわかる．経由する駅の数が増加すればするほど，経路を一意に特定できる可能性は高まる．

続いて，本アルゴリズムの計算時間を評価する．前述の通り，シミュレーションでは鉄道グラフを用いて，リンク数 15 以下，乗り換え数 2 回以下のすべての場合について探索を行った．結果として，6,404,455,757 個の経路が列挙された．C++で実装した本アルゴリズムを市販の PC 上で動作させたところ，全ての経路に対して 74 分以内に探索が完了した．平均すると 1 経路につき 7.1 マイクロ秒となる．よって，経路候補の列挙を行う本アルゴリズムは，日本全土を対象とした大規模な鉄道グラフに対して高速に動作することが示された，

5 ケーススタディ

本章では，フィールドワークを通して RouteDetector の実現可能性を示す．スマートデバイスから収集した

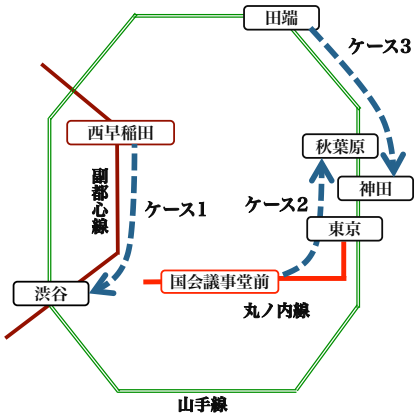

図 6: ケーススタディに用いた路線の概略

表 8: ケース 1 における推定時刻，観測時刻，定刻

状態	推定時刻	観測時刻	定刻
歩行とその他		–	
出発	10:56	10:56	10:56
到着	10:58	10:58	10:58
出発	10:58	10:58	10:58
到着	11:00	11:00	11:00
出発	11:00	11:00	11:00
到着	11:03	11:03	11:03
歩行とその他		–	
出発	11:10	11:10	11:10
到着	11:12	11:12	11:12
出発	11:12	11:12	11:12
到着	11:14	11:14	11:14
歩行とその他		–	

センサデータを使用して，列車移動の経路を特定する．ケーススタディとして，3 つの典型的な例を挙げる．まず，利用した路線の概略を図 6 に示す．

ケース 1　図 2 に示したのは，1 つ目のケースで収集したデータに RouteDetector を適用した結果であった．このケースでは山手線と丸ノ内線の 2 つの路線を利用した．図 2 (e) にて出発時刻と到着時刻のシーケンスを抽出した．表 8 に示す通り，全ての時刻を正しく検出することに成功した．ここで，推定時刻はセンサデータを基に抽出された時刻であり，観察時刻は手動でラベル付けした停発車の時刻であり，定刻は時刻表に記された本来の運行ダイヤである．次に，推定時刻のシーケンスを用いて，経路を探索する．結果として，列挙された 2 つの経路を表 9 に示す．2 つの候補のうちスコアが高い方は，実際に利用した経路であった．よって RouteDetector は，センターデータから列車移動の経路を特定することに成功した．

表 9: ケース 1 で列挙された 2 つの経路候補

No.	正解	経路 1	経路 2
1	国会議事堂前	国会議事堂前	江戸川橋
2	霞ヶ関	霞ヶ関	護国寺
3	銀座	銀座	東池袋
4	東京	東京	池袋
乗り換え			
4	東京	東京	池袋
5	神田	神田	要町
6	秋葉原	秋葉原	千川
スコア	–	2,664	2,277

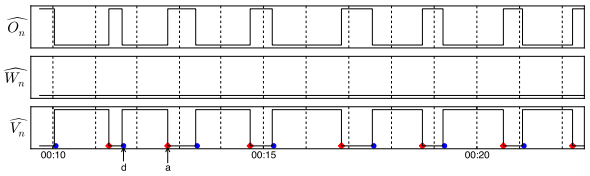

図 7: ケース 2 の行動推定の結果

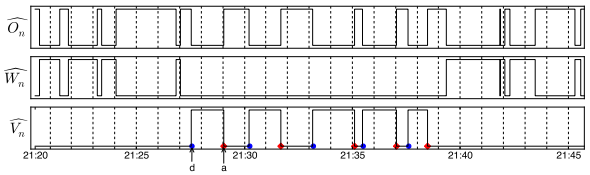

図 8: ケース 3 の行動推定の結果

ケース 2　ケース 2 は山手線で測定を行った．この例では，田端駅を出発し，乗り換えなしで 6 駅を経由して神田駅に到着する．発着時刻を抽出する様子を図 7 に示す．ケース 2 も時刻は正しく検出された．推定時刻シーケンスを用いて探索した結果，経路が一意に特定された．この経路は，実際に利用した経路であった．

ケース 3　ケース 3 は副都心線にて測定した．この例では，西早稲田駅を出発し，乗り換えなしで 3 駅を経由して渋谷駅に到着する．表 10 に示す通り，ケース 3 でも発着時刻は正しく抽出された．一方で，観測時刻が定刻とわずかに異なっていた．これは，測定の際に列車の遅延が発生したためである．次の章で，列車運行の遅延による問題について論じる．探索の結果，推定時刻シーケンスに該当する経路は存在しなかった．本ケースは，RouteDetector の失敗例である．

6　考察

本章では，RouteDetector を適用する上での制限事項について考察を行う．また，RouteDetector がもたらす新たな脅威への対策方法について論じる．

6.1　制限事項

端末間の差異　本研究の脅威モデルは，標的のハードウェアを事前に把握する必要があった．センサが記録する値は，搭載されているセンサの種類や，端末の形状などに依存する．そのため，ある端末のデータを用いて訓練したランダムフォレスのモデルが，別の端末

表 10: ケース 3 における推定時刻，観測時刻，定刻

状態	推定時刻	観測時刻	定刻
歩行とその他		–	
出発	21:27	21:27	21:26
到着	21:29	21:29	21:28
出発	21:30	21:30	21:28
到着	21:32	21:32	21:32
出発	21:33	21:33	21:32
到着	21:35	21:35	21:35
出発	21:35	21:35	21:35
到着	21:37	21:37	21:37
出発	21:37	21:37	21:37
到着	21:39	21:39	21:39
歩行とその他		–	

に対して正しく機能しない．本研究では，この問題を端末ごとの学習モデルを用意することで解決した．他のアプローチとして，端末間の差異を吸収するデータ処理が考えられる．この方法は今後の課題とする．

車両の種別　本研究では，鉄道列車による移動を標的とした．一定の地点を経由する公共交通機関として，他には航空機やバス等が挙げられる．RouteDetector は，時刻表に沿って運行する他の交通機関に対しても適用可能であると考えられる．一方で，一般道路を走行する交通車両に対してはうまく動作しないことが懸念される．交通信号や交通渋滞の影響で停発車が不定期的であり，時刻表通りに運行することが困難なためである．

また 3.4 節の行動推定において，エスカレータによる移動を**車両内**と分類することがあった．そこで**車両内**が 1 分以上継続する場合のみ発着時刻を検出するヒューリスティックを適用したが，この方法では長いエスカレータを列車と見なすおそれがある．Hemminki ら [8] は加速度センサから列車・バスなど移動手段の種類を検出する手法を提案した．車両の種別まで分類することは，エスカレータの影響を除去したり，多様な交通手段に RouteDetector を適用するための一助となる．

運行状況による影響　RouteDetector の成否は，列車運行の正確さに依存する．多くの列車が遅延する環境ではうまく動作しないため，標的がなるべく時刻通りに運行する地域に住んでいる必要がある．一方で，交通システムの利便性を向上させる最新の技術が，経路特定の実現性をより高める．例えば米国には，列車をリアルタイムに追跡し，何分後に列車が到着するかを利用者に通知するサービスが存在する [9]．この情報は API として一般に提供されているため，経路探索のために利用することができる．あるいは自動運転技術の普及に伴い，列車運行時刻が正確になる可能性もある．

また，標的が通勤や通学などで同じ経路を頻繁に利用する場合，一度の推定が失敗しても複数回推定を繰り返し，統計的な方法で推定を行うことで特定精度の向上が見込まれる．

6.2　対策方法

本節では，RouteDetector による攻撃を防ぐための対策について論じる．Michalevsky らはジャイロセンサを読み取ることで音声を認識する *Gyrophone* [3] を考案した．彼らは GyroPhone への対策方法としてセンサの生データにローパスフィルタを適用する方法を講じた．この方法は，高周波なセンサへのアクセスを必要としない大多数のアプリに対し機能を損なうことなくフィルタリングできる．さらに彼らは高周波なアクセスをパーミッションによって制限すべきと主張した．生データに対するアクセス制御は，RouteDetector への対策としても有用である．このとき，センサデータへのアクセスを制限する理由を開発者やユーザが認識していなければ，根本的な解決にはならない．モバイルアプリのプラットフォームは，センサデータから位置情報や音声を識別される危険性があることを通知すべきである．

7　関連研究

GPS などの既存のサービスを用いることなく，人物の位置を特定することを目指した研究を述べる．Gu ら [10] は，電波，音響，光などの情報を解析し建物内の物体や人物の位置を特定する *IPS* を考案した．また，Sapiezynski ら [11] は，スマートフォンが受信した WiFi シグナルの強さを測定することで移動経路を追跡できることを示した．同様に，Michalevsky ら [5] による *PowerSpy* ではスマートフォンの電力消費量を計測することで移動経路を推測する．Hua ら [12] は，磁場などの環境情報を学習し，人物が滞在した駅を特定する手法を示した．

前述の研究と比較して，RouteDetector は経路ごとの訓練データが不要であるという点で優位性がある．訓練データを必要とする手法では，事前に特定したい全ての場所に赴き，センサの値や電力消費量などを計測しなければならない．一方で，行動推定は普遍的に適用できるため，時刻表や API によって列車の発着時刻を取得できる場所ならば RouteDetector の適用範囲となる．

8　まとめ

本研究は，*RouteDetector* と名付けた新たなサイドチャネル攻撃の実証に成功した．RouteDetector は，人間の移動様式にみられる時空間的な規則性に着目し，鉄道を用いて移動する人物の経路を特定する．教師あり機械学習による行動推定によって，列車の出発時刻と到着時刻を平均 6 秒未満の誤差で抽出した．また，172 の鉄道会社が運営する 9,090 駅の路線図と時刻表を用いたシミュレーションの結果，6 駅以上経由する経路の候補数は，平均で 1 近くに絞り込まれることを示した．以上の結果は，RouteDetector による経路特定の脅威が現実的であることを定量的に示したといえる．

参考文献

[1] E. Owusu, J. Han, S. Das, A. Perrig, and J. Zhang, “ACCessory: Password Inference using Accelerometers on Smartphones,” in *in Proc. of HotMobile*, 2012.

[2] S. Dey, N. Roy, W. Xu, R. R. Choudhury, and S. Nelakuditi, “AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable,” in *in Proc. of NDSS*, 2014.

[3] Y. Michalevsky, D. Boneh, and G. Nakibly, “Gyrophone: Recognizing Speech from Gyroscope Signals,” in *in Proc. of USENIX Security*, 2014.

[4] A. Das, N. Borisov, and M. Caesar, “Do You Hear What I Hear?: Fingerprinting Smart Devices Through Embedded Acoustic Components,” in *in Proc. of CCS*, 2014.

[5] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, “Powerspy: Location tracking using mobile device power analysis,” in *in Proc. of USENIX Security*, USENIX Association, 2015.

[6] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, “Understanding individual human mobility patterns,” *Nature*, vol. 453, pp. 779–782, June 2008.

[7] L. Breiman, “Random forests,” *Machine learning*, vol. 45, pp. 5–32, 2001.

[8] S. Hemminki, P. Nurmi, and S. Tarkoma, “Accelerometer-based transportation mode detection on smartphones,” in *in Proc. of ACM Embedded Networked Sensor Systems*, ACM, 2013.

[9] “Metro -rider tools- real time arrivals.” `http://www.wmata.com/rider_tools/pids/real_time_arrivals.cfm/`.

[10] Y. Gu, A. Lo, and I. Niemegeers, “A Survey of Indoor Positioning Systems for Wireless Personal Networks,” in *IEEE Communications Surveys & Tutorials*, 2009.

[11] P. Sapiezynski, A. Stopczynski, R. Gatej, and S. Lehmann, “Tracking human mobility using wifi signals,” *CoRR*, vol. abs/1505.06311, 2015.

[12] J. Hua, Z. Shen, and S. Zhong, “We can track you if you take the metro: Tracking metro riders using accelerometers on smartphones,” *CoRR*, vol. abs/1505.05958, 2015.