明治大学総合数理学部

2016 年度

卒 業 研 究

IP アドレスによる位置情報検索システムの開発と評価
（1）登録データの精度

学位請求者　先端メディアサイエンス学科

笹航太

目次

第 1 章

はじめに

1.1 研究背景

IP アドレスからその位置情報を提供する IP Geolocation サービスが広く用いられている．その代表である MaxMind 社は，米国のマサチューセッツ州ウォルサムを拠点としており，2002 年に設立され，IP 情報収集を強みとした企業で，IP 情報を提供するビジネスにおいて，業界の大手プロバイダーである．

1.2 問題点

MaxMind 社が提供するデータベースには，十分な IP アドレスが登録されてなく，しばしば大きな誤差が生じる．よって，IP アドレスが使用されている正確な位置がわからない．しかし，32 ビットの全ての IPv4 アドレスの位置を調べるには，大きな調査コストがかかる．

1.3 研究目的

そこで我々は，MaxMind 社の“Geo IP Tool”[1] よりもユーザの位置情報を正確に提供することを目的とする．

1.4 提案手法

本研究では，ユーザ登録型の IP アドレス登録システム “Prefecture maP Internet Protocol”[*1](以後 PPIP と呼ぶ) を開発し，ブラウザに割り当てられている IP アドレスと位置情報を登録させることで，登録にかかるコストを削除し，IP Geolocation サービスの精度を向上する．また，いくつかのデータについて，提案システムの登録データと既存システムの精度を比較する．

[*1] http://windy.mind.meiji.ac.jp/ ksa/senior/top.php

第 2 章

Prefecture maP Internet Protocol

2.1　システム概要

PPIP はユーザが位置情報や IP アドレスを登録でき，情報を検索することができる下記の二つの機能を持ったシステムである．

機能 1. Geolocation API を利用し，ユーザの現在地を取得し，PPIP データベースに登録する．
機能 2. PPIP データベースから登録されているデータを検索する．

PPIP システム開発担当
高橋　Google Maps JavaScript API[5],Geolocation API[6],TLS 公開鍵証明書の導入
笹　　Google Maps JavaScript API[5],Google Places API[7]
厚見　データベースの設計・作成 [4]，データのやりとり [4]

2.2　使用 API

2.2.1　Geolocation API

Geolocation API はデバイスの位置情報を携帯電話の基地局や Wi-Fi のアクセスポイント，GPS(Global Positioning System) などを利用し，緯度経度の値として取得する API である．GPS を搭載していないパソコンやタブレットなどでも Wi-Fi のアクセスポイントなどを利用して位置情報を取得する．セキュリティの観点より，位置情報を取得する場合は TLS 公開鍵証明書により認証されたサーバからの暗号化された通信でなければならない．

2.2.2　Google Maps JavaScript API

Google Maps JavaScript API は Google Inc. が提供している API である．一般のユーザは，web サイトに無料で Google Maps の機能を利用出来る．以下に，利用手順を示すなお，Syncer「Google Maps JavaScript API の使い方まとめ」を参考とした [5].

1. Google Maps API キーを取得する．

「Google Maps API ウェブ向け」https://developers.google.com/maps/web/?hl=ja へアクセスする．
キーを取得ボタンをクリックすると，「Google Maps JavaScript API をアクティベートする」という画面が出てくるので，続けるを押す．
プロジェクトを作成を選択して，同意して続行ボタンを押す．
「認証情報」の画面になるので，作成ボタンを押す．
API キーが出てくるので，キーを制限ボタンを押す．
HTTP リファラー（ウェブサイト）を選択し，GoogleMaps を利用する URL を入力して，保存ボタンを押す．
API キー設定完了．

2. ライブラリを読み込む．
   HTML のヘッダーにプログラム 2.1 のソースコードを組み込むことでライブラリを読み込む．

プログラム 2.1　GoogleMapApi

```
<script type="text/javascript" src="https://maps.googleapis.com
/maps/api/js?></script>
```

3. 地図を表示する．
   HTML で Map クラスを利用する．id 名は任意で決める（ここでは map-canvas）．プログラム 2.2 にソースコードを示す．

プログラム 2.2　HTML

```
<!-- 地図のキャンパス -->
<div class="map-embed">
<div id="map-canvasここに地図が表示されます"></div>
</div>
```

JavaScript では，HTML で指定した id 名を取得．最初に表示される地図の位置座標を（緯度，経度）の順に指定．zoom ではマップの縮尺を指定．地図のインスタンスを作成する．プログラム 2.3 にソースコードを示す．

プログラム 2.3　JavaScript

```
var canvas = document.getElementById('map-canvas');
var latlng = new google.maps.LatLng(35.792621, 139.806513);
var mapOptions = {
zoom: 15 , // 必須
center: latlng ,
};
var map = new google.maps.Map(canvas , mapOptions);
```

CSS は，地図のサイズを指定する．プログラム 2.4 にソースコードを示す．

プログラム 2.4　CSS

```
/* 地図のキャンパスを囲む要素 */
.map-embed
{
/* 地図を乗せる台の大きさ */
width: 500px ;
height: 300px ;
```

```
margin: 0px ;
padding: 0 0 0;

overflow: hidden ;

position: relative ;
top: 0px ;
left: 0px ;
}

div._embed-5625
{
padding: 0 0 56.25% ;
}

/* 地図のキャンパス */
.map-embed > div
{
position: absolute ;
top: 0 ;
left: 0 ;
/* 地図の大きさ */
width: 500px ;
height: 300px ;

margin: 0 ;
padding: 0 ;
}

/* 表示崩れを防ぐ */
.map-embed img
{
max-width: none ;
}

.js{
position: absolute;
top: 800px;
left: 150px;
}
```

2.2.3 Google Places API

Google Inc. が提供している API．Google Maps JavaScript API で設定した地図を使い，プレイス検索を行う．指定したキーワードで，施設や建物などを検索する．検索結果位置にマーカーを表示する．以下に，利用手順を示す．なお，Syncer「JavaScript で位置情報を取得する方法 (Geolocation API)」を参考とした [7].

1. ライブラリを追加する．
   2.1.1 で取得した Google Maps API を使用．「自分の API キー」の後に，「&libraries=places」を追加する．
2. 検索機能を追加する．

JavaScript で，request に中心座標，検索半径を指定．query は入力されたテキストをキーワードとする．検索結果の位置座標にマーカーを表示する．プログラム 2.5 にソースコードを示す．

プログラム 2.5　Place 検索

```
var request = {
location: pyrmont,
radius: '500',
query: document.getElementById("address").value
};

service = new google.maps.places.PlacesService(map);
service.textSearch(request, callback);検索結果の場所にマーカーを立てる処理
/**/
function callback(results, status) {
map.setCenter(results[Math.floor(results.length/2)].geometry.location);
if (status == google.maps.places.PlacesServiceStatus.OK) {
for (var i = 0; i < results.length; i++) {
var place = results[i];
createMarker(results[i]);
  }
 }
}

function createMarker(place) {
var Marker = new google.maps.Marker({
position: new google.maps.LatLng(place.geometry.location.lat(),place.geometry.location.lng()),
map: map,
title :place.name
 });
}
```

2.3　登録システム

ユーザは PPIP の登録画面にアクセスすると自身が使用している IP アドレスと位置情報を得ることができる．この 2 つに加えて都道府県名・位置情報の詳細・匿名の情報提供者名を入力して登録する．また，電波状況が悪い場合や位置情報の取得に時間がかかる場合，ユーザが位置情報取得を許可しなかった場合でも検索フォームにより自分のいる位置を検索して登録することや，GoogleMap を直接タップもしくはクリックして登録することができる．登録システム画面を図 2.1 に表す．

図 2.1　登録システム画面

登録システムでは IP アドレスと現在地の緯度経度に加え，都道府県名，現在地情報，匿名の名前を入力して登録する．また，位置情報の取得に失敗した場合や位置情報の取得を不許可にした場合に検索フォームを用意し，現在地周辺の地名を検索し，GoogleMap 上の現在地を指定してもらうことで緯度経度の情報を取得できるようにしている．図 2.2 に現在地検索システムを示す．

現在地検索では検索したい地名を検索フォームに入力し，現在地検索ボタンを押すと図 2.2 の GoogleMap 上で検索した地名に移動し，該当する場所にピンが立つ．このピンをクリック，もしくはタブレット機ならタップすることにより緯度，経度，都道府県，現在地名を自動的に入力し，登録をスムーズに行うことができる．

登録内容例を表 2.1 に，データベース設計を表 2.2 に示す．実際にデータベースに格納されるのは登録内容に加えて id，時刻である．これら二つは登録時に自動的にデータベースに格納される．

登録システムのフロー図を図 2.3 に表す．

PPIP 登録システムで行っている各処理について記述する．トップ画面の地図を表示するシステムでは

図 2.2　現在地検索システム

Google Map API を利用して地図を表示している．次に現在地利用を行わなかった場合，Google place API を使ってキーワードを検索すると検索した場所を表示し，現在地をクリックすると，緯度，経度，都道府県，所在地名を一部例外を除いて自動で入力されるようにしている．例外は Google 側でポイに入っているワードの設定が異なる場合に発生する．手動でニックネームを入力し，送信ボタンを押す．

次に現在地検索の利用を行った場合，緯度，経度が自動で入力され，Google Map が現在地を表示しているので，都道府県名，所在地名（現在地に最も近い目印，例　駅名など），ニックネームを入力し，送信ボタンを押す．

送信ボタンを押すとデータベースに登録される．

表 2.1　登録データ例

id	IP アドレス	緯度	経度	都道府県名	所在地名
1	133.26.34.22	35.706962	139.659547	東京都	明治大学
2	116.82.xxx.xxx	36.060xxxx	139.509xxxx	埼玉県	xx 駅
3	180.43.xxx.xxx	35.4222xxxx	139.463xxxx	神奈川県	店名 xx 店舗

表 2.2　データベース構造

	id	IP アドレス	緯度	経度	都道府県名	所在地名	時刻	名前
データ型	int(11)	varchar(15)	double	double	varchar(6)	varchar(20)	timestamp	varchar(20)
変数名	id	ipaddress	lat	lng	prefec	name	timestamp	nick

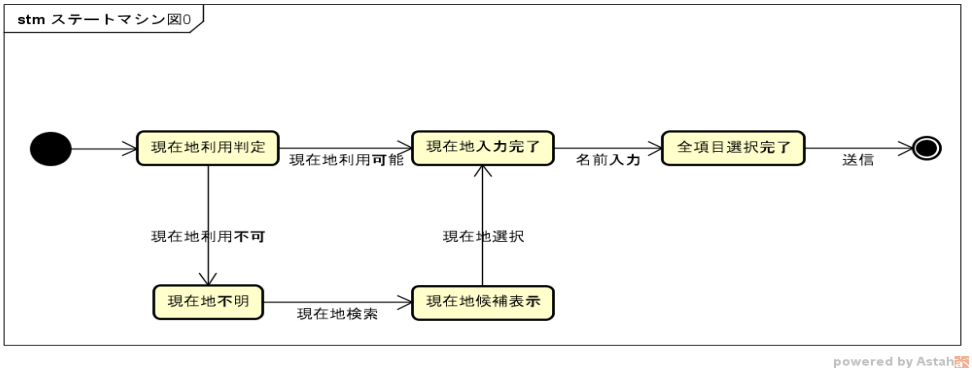

図 2.3　登録システムフロー図

2.4　検索システム

IP アドレスからその位置情報を提示する．IP アドレスが登録済みの場合は位置情報を提示する．未登録の場合は，MaxMind が提供するデータベースの情報を参照して提示する．検索システム画面を図 2.4 に示す．

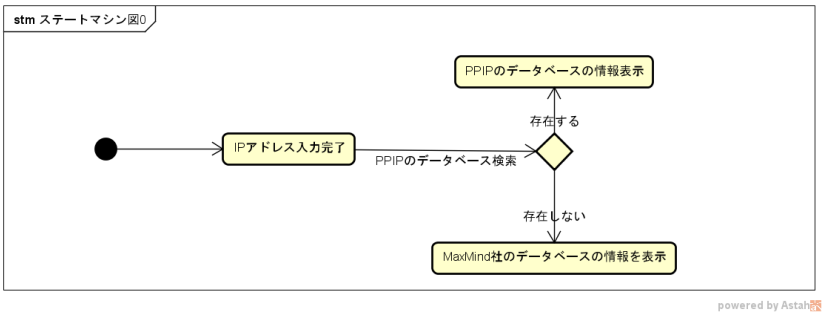

図 2.4　検索システム図

PPIP では現在，検索フォームに IP アドレスを入力し，検索ボタンを押すと PPIP データベースにアクセ

スし，IP アドレスを検索し，情報を取得します．IP アドレスの情報を取得すると，その IP アドレスの情報を画面に表示し，GoogleMap にその情報をマッピングする．その図を 2.5 に示す．

図 2.5　検索結果

検索システムのフロー図を図 2.6 に示す．

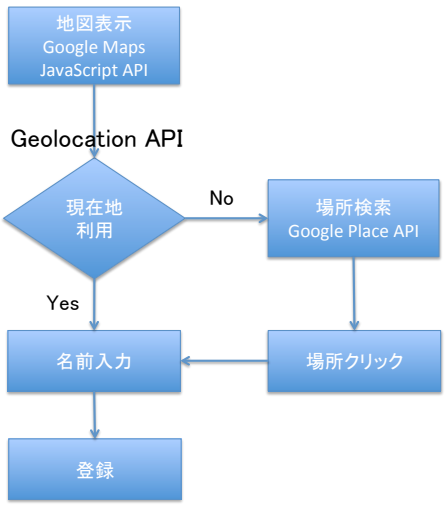

図 2.6　検索システムフロー図

第 3 章

PPIP のデータ収集実験

3.1　実験背景

スマートフォンやノートパソコンの普及により，外でインターネットを利用する機会が増えている．それに伴い，フリー Wi-Fi が利用できる店舗が増えてきいるため，悪意のあるサイバー攻撃に利用される危険性も増加してきている．特に，SSDP（Simple Service Discovery Protocol）を利用したサイバー攻撃の危険性が報告されている．よってフリー Wi-Fi に流れる SSDP パケットを調査する．

3.2　通信プロトコル

3.2.1　UPnP

UPnP(Universal Plug and Play) は 1999 年に Microsoft 社が発表したプロトコルである．それまで機器間の通信をするためには，各々複雑な設定をする必要があり，容易ではなかった．しかし，UPnP は機器をネットワークにつなぐだけで，他の機器と通信することを可能とした [9].

3.2.2　SSDP

SSDP（Simple Service Discovery Protocol）は UPnP で，他の機器との相互確認に用いられる通信プロトコルである．他の機器に呼びかける M-SEARCH と呼びかけに答える NOTIFY があり，M-SEARCH はマルチキャストで特定の複数の相手にパケットを送信する．M-SEARCH を受け取った機器はユニキャストである NOTIFY を返信することで通信が可能であることを知らせる．[10]

3.2.3　UPnP の脆弱性

UPnP には SSDP を利用した脆弱性がある．

「SSDP リフレクター攻撃」
攻撃者は発信元を偽装して M-SEARCH を送る．送られた機器は偽装された発信元に NOTIFY を送る．情報量は M-SEARCH ＜ NOTIFY なので，大量の攻撃パケットがさらに増大されて攻撃対象機器やネットワークを襲う．

フリー Wi-Fi で誰もが UPnP を使えると利用されたり攻撃される可能性があるため危険である．よって，フリー Wi-Fi を流れる SSDP パケットを調査した．

3.3　フリー Wi-Fi に流れる SSDP パケットの調査

3.3.1　パケット収集方法

SSDP パケットを収集するために，ネットワークに流れるパケットをリアルタイムで取得できるソフト「Wireshark」を使用する．フリー Wi-Fi に接続して，Wireshark で 30 分間パケットを収集する．

3.3.2　収集結果

以下の施設でフリー Wi-Fi でパケットを収集した．

表 3.1　実験場所

ローソン　あざみ野駅前店	
ファミリーマート　中川駅前店	ファミリーマート　鷺沼駅前店
セブンイレブン　あざみ野駅前店語	セブンイレブン　センター北駅前店
マクドナルド　センター北モザイクモール店	マクドナルド　センター北あいたい店
コートダジュール　センター北プレミア店	コートダジュール　センター南駅前店
タリーズ　あざみ野駅前店	タリーズ　センター北よつばこ店
ドトール　あざみ野駅前店	ドトール　溝の口駅前店
スターバックス　センター北モザイクモール店	スターバックス　センター北みなも店
横浜市立山家図書館	横浜市立都筑図書館（wifi square）
ブックオフ (freespot）　あざみ野駅前店	
ドン・キホーテ　センター北プレミア店	

商業施設のフリー Wi-Fi は暗号化されていなかったが，横浜市立山内図書館に，暗号化されているフリー Wi-Fi が存在した．

図 3.1 から，SSDP パケットが使われていることがわかる．しかし，192.168.0.120(自分) が M-SEARCH を送っても NOTIFY は返信さず，同じ IP アドレス同士で通信を繰り返していた．これより，あらかじめ設定してある IP アドレスからの通信しか受け取らない設定になっていることがわかる．

18 箇所では SSDP を利用しておらず，横浜市立山内図書館で使われていた SSDP は外部からの通信を受け付けない設定になっていた．よって，本調査のフリー Wi-Fi はすべて安全である．

この結果から，フリー Wi-Fi の通信の安全性に問題はないと仮定し，有用性について研究を行い，PPIP の開発を行った．

3.4　PPIP 実験方法

PPIP の実用可能性を確かめる目的で，2016 年 11 月 15 日 ～ 2016 年 12 月 11 日の間，実証実験を行った．SNS により集めた被験者は，自宅や，施設内で利用できる Wi-Fi を利用して，IP アドレスと位置情報の登録

No.	Time	Source	Destination	Protocol	Length	Info
26…	1458.143…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1459.149…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1460.162…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1577.134…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1578.148…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1579.152…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1580.165…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1697.138…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1698.150…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1699.164…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1700.176…	192.168.0.120	239.255.255.250	SSDP	216	M-SEARCH * HTTP/1.1
26…	1740.588…	192.168.0.106	239.255.255.250	SSDP	557	NOTIFY * HTTP/1.1
26…	1740.593…	fe80::6198:309c:e…	ff02::c	SSDP	585	NOTIFY * HTTP/1.1
26…	1743.660…	192.168.0.106	239.255.255.250	SSDP	557	NOTIFY * HTTP/1.1
26…	1744.671…	fe80::6198:309c:e…	ff02::c	SSDP	585	NOTIFY * HTTP/1.1
26…	1747.891…	192.168.0.106	239.255.255.250	SSDP	478	NOTIFY * HTTP/1.1
26…	1747.895…	fe80::6198:309c:e…	ff02::c	SSDP	506	NOTIFY * HTTP/1.1
26…	1748.779…	192.168.0.106	239.255.255.250	SSDP	487	NOTIFY * HTTP/1.1
26…	1748.784…	fe80::6198:309c:e…	ff02::c	SSDP	515	NOTIFY * HTTP/1.1

図 3.1　キャプチャー例（一部）

をする．実験環境を表 3.2 に示す．

表 3.2　実験環境

OS	Windows 8.1
メモリ	8GB
言語	PHP, JavaScript

3.5　実験協力者

47 名の被験者の協力で，147 件のデータが集まった．分類を表 3.3 に示す．

表 3.3　県ごとのデータ数

東京都	神奈川県	埼玉県	千葉県	その他
77 件	32 件	23 件	4 件	11 件

第 4 章

実験結果評価

4.1　登録データの精度評価方法

PPIP に登録された所在地名を Google map[11] で検索し，該当場所の住所を Geocoding[12] で検索．ここで得た緯度経度を，真値とする．

次に，“keisan”[13] を用いて，PPIP の誤差である，PPIP に登録された緯度経度と真値の直線距離の差を求める．そして Geo IP Tool の誤差である，Geo IP Tool に IP アドレスを入力して得られる緯度経度と真値の直線距離の差を求める．計算方法は，地球を赤道半径 r=6378.137km を半径とする球体として，球面三角法の公式から，以下の式で行う．

地点 A(経度 $x1$, 緯度 $y1$)，地点 B(経度 $x2$, 緯度 $y2$)
AB 間の距離 $d = rcos^{-1}(siny1 * siny2 + cosy1 * cosy2 * cosX)$
$X = x2 - x1$

本評価では PPIP の登録データ 147 件を調査する．Geo IP Tool では十分な IP アドレス数が登録されていないため，緯度経度の精度は低いと考えられる．また，PPIP は，被験者が現在地情報を利用するため，緯度経度は高い精度を示すと考える．

4.2　登録データの精度評価

既存システム Geo IP Tool と PPIP の登録位置の精度を表 4.1 に示す．PPIP は Geo IP Tool に比べ，平均誤差が少なく，データのばらつきも少ない．

表 4.1　精度評価

	差の平均	標準偏差
Geo IP Tool	29.12km	83.78
PPIP	1.27km	4.16

4.2.1　被験者数

本研究では短期間の実験だったので，被験者数，登録データ共に十分ではなかった．今後登録データを増やしていくためには，周期的に実験協力を求め，協力者に情報を拡散させ，多くの人に PPIP を認知させること

が重要である．

4.3　考察

PPIP は Geo IP Tool より高い精度を示したが，平均で 1.27km の誤差が生じてしまった．その原因は次の 2 つと考える．(1) 自宅などの公開されたくない位置情報を登録する場合は，所在地名に最寄り駅を登録することを指示していたので，その場所から大きく外れた位置が登録されることで，誤差が生じたと考えられる．(1) が原因の誤差は 2 件存在し，平均 0.7km である．(2)wi-fi の届く範囲が広いため，図 4.1 のように，中心の座標から離れた位置にいても，wi-fi に接続出来たことが原因と考えられる．(2) が原因の誤差は 145 件存在し，平均 1.28km である．よって誤差の主な原因は (2) である．

表 4.2　精度評価

	差の平均	標準偏差
Geo IP Tool	29.12km	83.78
PPIP	1.27km	4.16

図 4.1　Wi-Fi の範囲

第 5 章

終わりに

5.1　まとめ

IP アドレスからユーザの位置情報を登録できるシステムを開発し，既存のシステムの Geo IP Tool と比較評価をした．Geo IP Tool では，平均誤差が 29.12km，PPIP では 1.27km となり，PPIP の方が高い精度を示した．

本研究では，IP アドレスはユニークな場所で使われていると仮定したが，ISP によっては同じ IP アドレス広域に割り当てている場合がある．株式会社ワイヤ・アンド・ワイヤレスは異なる店舗に同一帯の IP アドレスを振り分けているため，時間の経過により，1 つの IP アドレスに対して，ユニークな緯度経度ではなくなってしまう．1 つの IP アドレスに対して，緯度経度の重複を認めるなど，複数の候補をどう扱うかが今後の課題となる．

謝辞

本論文を執筆するにあたり，多くの方からご指導を賜りました．

特に，多大なるご指導を承りました，明治大学総合数理学部先端メディアサイエンス学科，菊池浩明教授に深く感謝を申し上げます．

また，実験データの収集に協力していただいた，菊池研究室の皆さまに感謝の意を表するとともに，謝辞とさせていただきます．

参考文献

[1] Geo IP Tool（https://geoiptool.com/ 2016 年 10 月年参照）

[2] Geo IP Tool（https://geoiptool.com/ 2016 年 10 月年参照）

[3] PPIP(http://windy.mind.meiji.ac.jp/ ksa/senior/top.php 2016 年 10 月年参照)

[4] PHP(http://php.net/manual/ja/langref.php 2016 年 10 月参照)

[5] Syncer (https://syncer.jp/google-maps-javascript-api-matome 2016 年 10 月年参照)

[6] Geolocation API(http://www.htmq.com/geolocation/ 2016 年 10 月年参照)

[7] VINTAGE (http://www.vintage.ne.jp/blog/2015/04/395 2016 年 10 月年参照)

[8] Syncer (https://syncer.jp/how-to-use-geolocation-api 2016 年 10 月年参照)

[9] UPnP とは？ その仕組みの原点に立ち返る (http://www.itmedia.co.jp/broadband/0210/18/honda4.html 2016 年 8 月参照)

[10] SSDP とは、を説明してみる http://d.hatena.ne.jp/ozakira/20141130 2016 年 8 月参照)

[11] Google map(https://www.google.co.jp/maps 2016 年 10 月年参照)

[12] Geocoding(http://www.geocoding.jp/ 2016 年 10 月年参照)

[13] keisan(http://keisan.casio.jp/exec/system/1257670779 2016 年 10 月年参照)

[14] SCSI,“次元圧縮によるダークネットトラフィックデータの可視化”（2016 年 4 月参照）

[15] パケットキャプチャ入門-第 3 版-LAN アナライザ Wireshark 活用術-竹下恵（2016 年 7 月参照）

[16] パケットキャプチャ無線 LAN 編-Wireshar による解析-竹下恵（2016 年 7 月参照）

付録 A

あみだくじを用いた対話的なブラウザ履歴漏洩の研究

A.1　研究背景

近年, 利用者が気付かないうちに自分のブラウザ履歴を第三者に知らせてしまうブラウザ履歴盗聴攻撃の危険性が高まっている. Weinberg らは, チェスを模した Captcha を提案している [1]. 不正ボットを防止するための Captcha を偽装し, チェスの駒をユーザにクリックさせることにより, ブラウザの履歴を盗聴する. 全ての駒には URL が一意に設定されており, すでに訪問したサイトの駒は背景と同色で表示されるため, ユーザは気づかないままクリックした URL が既訪問であることを取得されてしまう. しかし, Weinberg らの提案した Captcha では 1 クリックで 1 履歴を盗聴することしかできない. そこで, 本稿では, 1 クリックで複数のブラウザ履歴を同時に盗聴することが出来る, あみだくじを用いたブラウザ履歴盗聴システムを提案する. 本研究は履歴盗聴サイトのリスクについて報告する.

A.2　ブラウザ履歴盗聴の方法

A.2.1　chessboard 方式 [1]

Weinberg らの提案する chessboard[1] では, HTML の A タグと CSS の visit タグを使用する. 未訪問 URL を指定した A タグは背景と同色に, 既訪問 URL は背景と違う色に CSS で設定することにより, 対象者の履歴に応じて表示結果が変化することを利用する.

図 A.1 に示す chessboard の原理図において, A に設定した URL は既訪問である. 背景は茶色なので, 駒の色は背景と異色の黒になっている. B も同様である. 一方, C に設定した URL は未訪問で, 駒は背景と同色の茶色になっているため, 存在しないように見える. Captcha として利用者に駒をクリックさせることにより, その対象者が A, B のサイトを訪問したことがサーバにわかる.

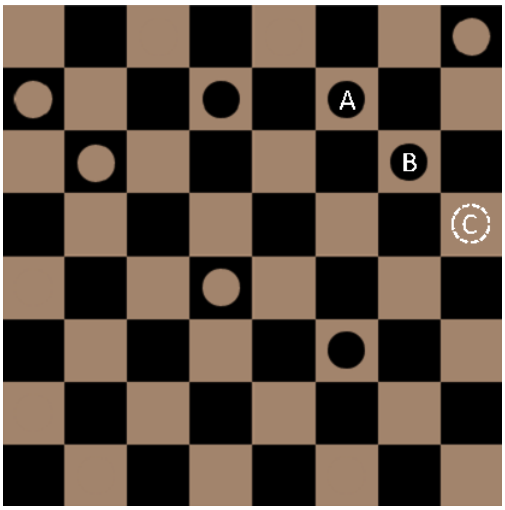

図 A.1　chessboard 方式の原理図

A.2.2　提案方式

あみだくじの横線を複数の URL に対応した A タグとすることで, 指定した URL が未訪問か既訪問かによって, たどり着くゴールが変化するシステムを提案する. ゴールをクリックさせることで履歴を盗聴する. このシステムの利点はあみだくじの組み方により, 1 クリックで同時に複数の履歴を盗聴することが可能という点である. しかし同時に盗聴できる履歴が増えるにあたり, あみだくじの本数が増え, 複雑さが増すため, その効果を明らかにするために 3 節の実験を行った. 図 A.2 の 4 本のあみだくじ方式の実行画面を使用してシステムの説明を行なう.

まず, 一意の URL A, B に対応する複数の枝を設定する. の場合, A, B 両方の URL を未訪問のため, 図 A.2 の N がゴールとなる. も同様である. ここで, g を 1 クリックで同時に取得できる URL 履歴盗聴数と定義する. よって図 A.2 のは g=0,, は g=1, は g=2 である. また g=0 の場合, あみだくじとして不自然である. よって実験用ではどのサイトの閲覧履歴がなくとも表示されるダミーの横線を配置した. チェスボード方式およびあみだくじ方式は,HTML と CSS, PHP, SQL を用いてウェブブラウザ上での動作を確認した.

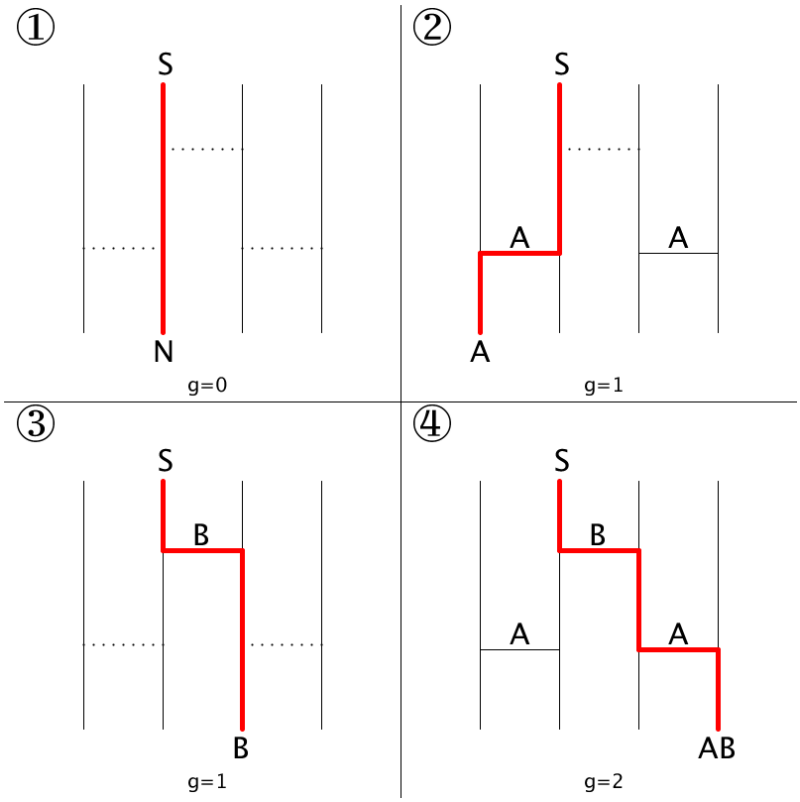

図 A.2　あみだくじ方式

A.3　実験

A.3.1　概要

chessboard 方式では g=1 の 時間を計測すること, あみだくじ方式では g=0〜3 の時間を計測することで, 単位時間当たりの履歴盗聴数を明らかにすることを実験目的とする. 実験は 20 代から 50 代の男女 20 名に対して, 2015 年 10 月 1 日から 1 か月の間に行った.

A.3.2　実験方法

実験 1　あみだくじの方式の比較

Processing を用いてランダムなあみだくじを表示する.1 回目は g が 0〜2, 2 回目は g が 0〜3 の提案方式で, それぞれのあみだくじを解きゴールをクリックするまでの応答時間を計測する.（これ以降, g が [0,2] からランダムに選ばれることを, g=[0,2] と表記する.）

実験 2　chessboard 方式とあみだくじ方式の比較

8 × 8 マスのチェス盤にランダムに茶色と黒の駒を 5 個ずつ配置し, 計 10 個表示して, そのすべてをクリックするのにかかる時間を計測する.

A.3.3　実験 1 の結果

実験 1 の結果を表 A.1 および図 A.3 に示す. g=[0,2] と [0,3] の平均応答時間を公平に比較するため, g=6 にそろえて, それぞれの平均応答時間を g=[0,2] は 3 倍, g=[0,3] は 2 倍する. その結果, g=[0,3] の方法がより平均応答時間が短くなった. よって, g=3 を chessboard 方式と比較する. あみだくじの同時履歴盗聴数 g について線形回帰したところ,

TA(g)=2306.9+503.9g

であった. g が小さい場合, 応答時間の分散は小さいが, g が大きくなるにつれて増加している.　この原因は, あみだくじを解く個人差にあると考える. また図 A.3 の「amida」の傾きは 503.9,「chess」の傾きは 1334.3 となった. 従って, 3<g のとき, 「amida」の方が「chess」よりも経過時間が短くなると推測する.

表 A.1　あみだの本数 n についての処理時間

n	4	8
g	[0,2]	[0,3]
1 クリックの平均達成時間 [ms]	2805.9	3273.7
標準偏差	966.5	1449.6

A.3.4　実験 2 の結果

実験 2 の結果を表 A.2 および図 A.4 に示す. 平均応答時間は「amida」が 3.27 秒,「chess」が 1.33 秒である. ただし, あみだくじ方式と chessboard 方式を公平に比較するために, 同時履歴盗聴数 g=3 にそろえて考える. chessboard 方式は g に比例して時間がかかるので,TC(g)=1.33g と予測する. この TC(g) を図 A.3 の上に重ねてプロットする. 図 A.4 に, 両方式のクリックあたりの応答時間の分布を示す. 以上より, あみだくじ方式の方が 1 履歴を取得する際にかかる時間の平均が短いと結論づける.

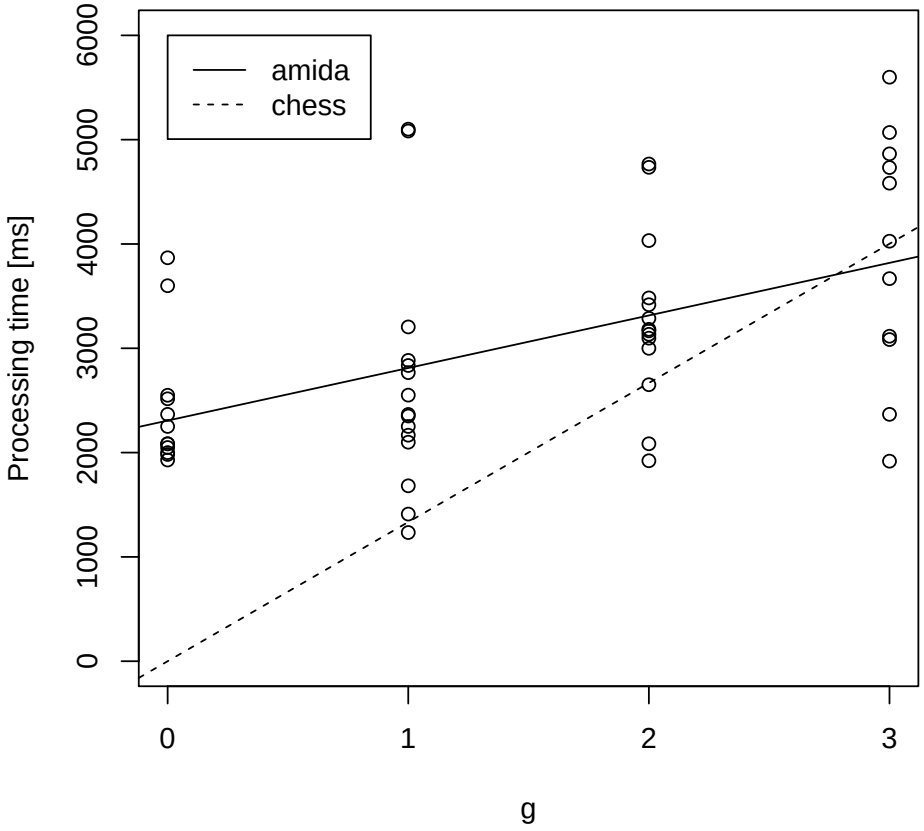

図 A.3　取得履歴数 g ごとのあみだくじ方式の処理時間

表 A.2　chessboard 方式とあみだくじ方式の応答時間

方式	g	1 クリックの平均応答時間 [ms]	標準偏差
chessboard	1	1334.3	503.87
あみだくじ	0〜3	3273.7	1449.6

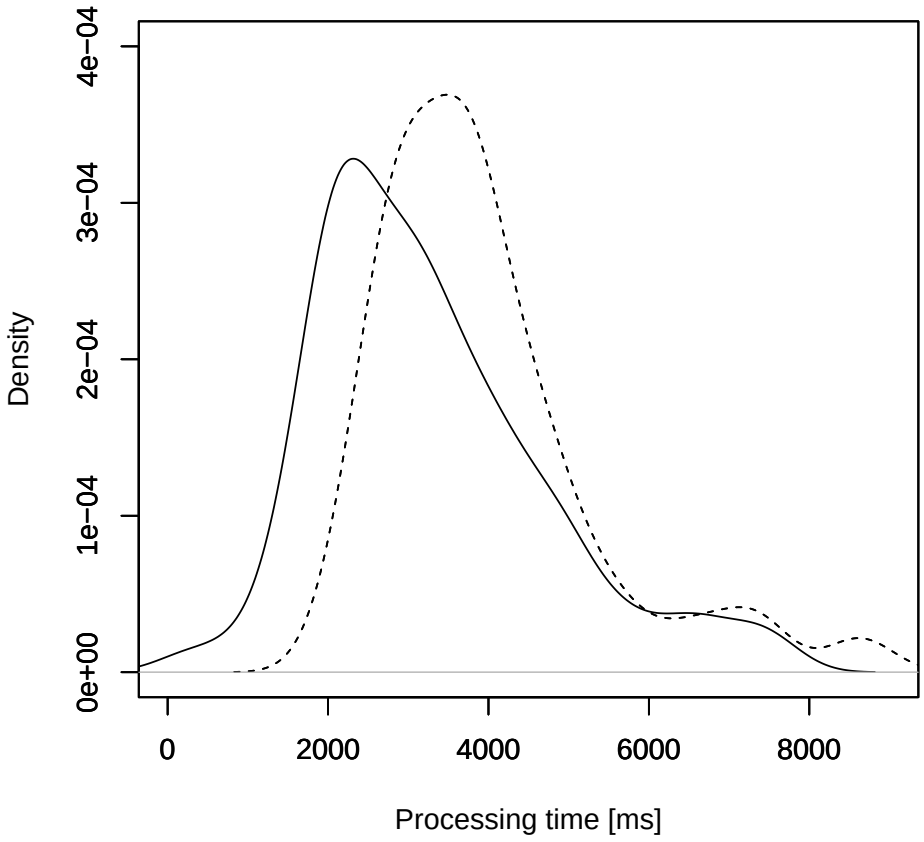

図 A.4　chessboard 方式とあみだくじの応答時間分布

A.4　まとめ

本研究では, あみだくじを用いた対話的なブラウザ履歴盗聴システムの提案を行った. また, 評価実験によりブラウザ履歴盗聴システムとの効率を評価した. あみだくじを用いたシステムの効率が予想以上に低かった理由として「応答時間の長さ」が挙げられる. chessboard 方式は視覚で捕え, 反射的に応答できるのに対して, あみだくじ方式は線をなぞらなければならないためより時間を要する. 今後の課題として, あみだくじの本数を増やした場合の効率, 従来の対話的なブラウザ履歴盗聴システムの単位時間あたりの履歴盗聴数を調べることが挙げられる.

A.5　備考

A.5.1　取得履歴数を増やす

縦線の数を n，履歴数を g とすると，

$$n = 2^g$$

となり n は指数関数的に増えてしまうが，理論上は無限に履歴数を増やすことが出来る．しかし，あみだくじの規模が大きくなってしまうので，実際に使用することは困難だと思われる．以下に n=4 のあみだくじを作る例を基に，その手法を示す．

1. n=n-1(本手法では 3) の同一のあみだくじを 2 つ並べる (図 A.5 参照).

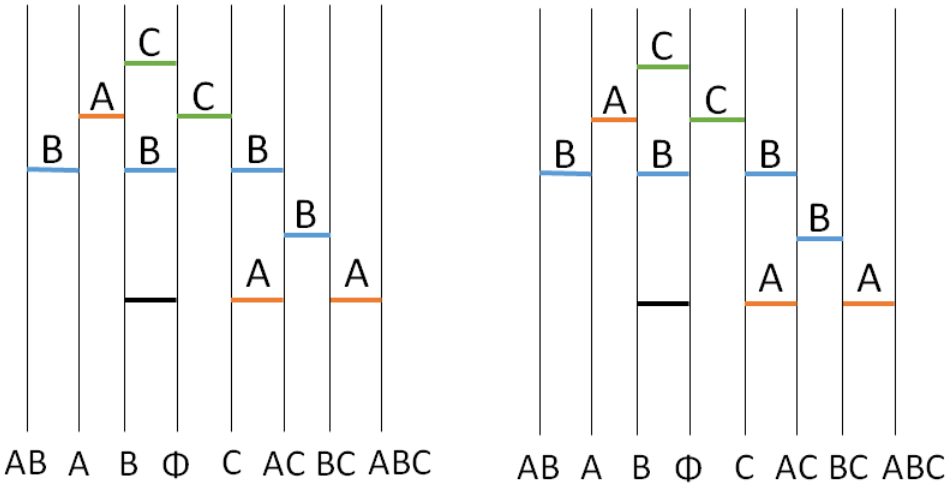

図 A.5　n=3

2. 2 つのあみだを，追加する URL によって変化する横線を用いて，上部でつなげる (図 A.6 参照).

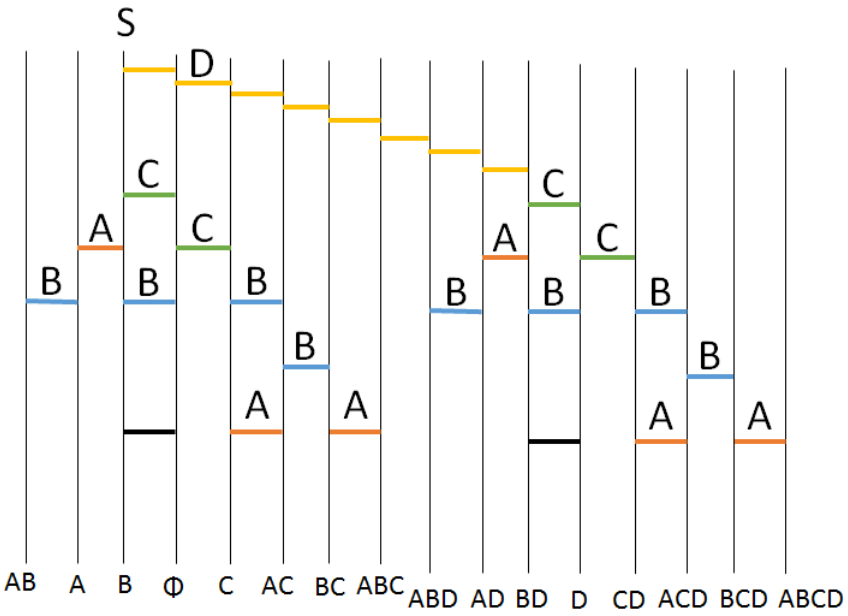

図 A.6　n=4

参考文献

[1] Zachary Weinberg, Eric Y. Chen, Pavithra R. Jayaraman and Collin Jackson, “I Still Know What You Visited Last Summer”,2011 IEEE Symposium on Security and Privacy, pp. 147-161, 2011.

[2] 笹航太，清水雄太，菊池浩明，情報処理学会第 78 回全国大会，6V-04，Vol. 3, pp. 557-558， 2016.