端末の長期的な識別を目的とした内部ストレージ種別の推定

種岡 優幸[1] 高橋 和司[1] 安田 昂樹[1]
田邉 一寿[1] 細谷 竜平[1] 齋藤 孝道[2]

概要： HTTP ヘッダや JavaScript から採取可能な情報を組み合わせて端末を識別する Browser Fingerprinting という手法がある．ユーザエージェント文字列に代表される端末の識別に利用される情報のいくつかは時間経過に伴い変化することが先行研究により示された．よって，端末の長期的な識別を行う上で，時間経過による値の変化が生じにくいハードウエアに付随する情報は有用である．先行研究では，様々なハードウエア情報の取得手法を提案しているが，本論文では，ハードウエアに付随する新たな情報として HTML5 の Web Storage API を利用し，端末の内部ストレージ種別が HDD か SSD かを推定する手法を提案する．

Estimation of Internal Storage Type to Identify Devices for a Long Period

MASAYUKI TANEOKA[1] KAZUSHI TAKAHASHI[1] KOKI YASUDA[1]
KAZUHISA TANABE[1] RYOHEI HOSOYA[1] TAKAMICHI SAITO[2]

1. はじめに

端末から採取可能な複数の情報の組み合わせによって端末内のブラウザを識別する Browser Fingerprinting（以降，Fingerprinting という）と呼ばれる手法がある．この手法は，Web 広告事業者を中心に利用されている．Englehardt ら[2]の調査によると，Alexa の Top100 万サイトのうち Fingerprinting に関する技術を用いるサイトは 14,371（1.6%）であった．

しかし，識別に用いられるいくつかの情報は短期間で変化し，長期的な識別の精度を低下させることを示されている[1]．変化しやすい情報の例として，端末にインストールされているプラグインのリストやグローバル IP アドレスが挙げてられている．それらは JavaScript の実行および HTTP ヘッダから採取できる情報である．他方，画面解像度や端末のタッチ機能の有無のようにハードウエアに付随する情報は時間経過に伴う変化が少ないことが示されている．

本論文では，長期的な識別の精度向上を目的として，ハードウエアに付随する新たな情報の採取手法の一つとして，Web Storage API を用いたデータの読み込み速度からネットワーク経由で内部ストレージ種別を HDD か SSD のどちらであるかを推定する手法を提案する．提案手法の検証実験は，母集団の端末のうちいくつの端末で提案手法による推定が正確か（以降，実験 1 という），複数回のアクセスがあった端末のうち推定結果が変化しない端末はいくつか（以降，実験 2 という）の 2 つを行う．その結果，実験 1 において精度は 79.4%となり，実験 2 においては推定結果が変わりにくいことを示した．これにより，ネットワーク経由の内部ストレージの推定はハードウエアに付随する新たな情報として利用できる可能性があることを示す．

本論文の構成は次の通りである．2 節では，ハードウエアに付随する情報の先行研究および内部ストレージの一般的な速度の比較について説明する．3 節では，提案手法について説明する．サンプルの収集を行うために作成した Web ページ（以降，調査用 Web ページという）および 2 つの実験の詳細を述べる．4 節では，実験結果である時間計測による端末の識別と時間をおいた定期的なアクセスによる推定を示す．ネットワーク経由の内部ストレージ推定の正確さについて示す．5 節では，実験結果の考察を行う．6 節では本論文のまとめを行う．

2. ハードウエア特徴点

2.1 先行研究

Mowery と Shacham[3]は Canvas API や Web GL を用いて，文字列や画像を描画し，出力されたベクタ画像にピクセルレベルの差異が生じることを利用し，GPU，OS，ブラウザの組み合わせが特定できることを示した．

Mowery ら[4]は JavaScript ベンチマークソフトを用いて，OS，ブラウザ，CPU の識別を試み，45.3%の精度で CPU ア

1 明治大学大学院
Graduate School of Meiji University
2 明治大学
Meiji University

ーキテクチャを識別できることを示した．

Nakibly ら[5]は HTML5 API を用いて GPU のクロックレートとクロックスキューに基づいた新たな Fingerprinting を考案した．また HTML5 API からアクセスできる，GPU，カメラ，スピーカーやマイク，モーションセンサー，GPS およびバッテリーといったハードウエア情報を用いた Fingerprinting について紹介を行っている．

Bojinov ら[6]はスピーカーフォンまたはマイクの周波数応答を利用する Fingerprinting と端末の加速度センサーのエラーに付随する Fingerprinting の 2 種類を提案した．ユーザエージェント文字列と組み合わせることで，53%の精度で 1,900 の端末を識別できることを示した．

後藤ら[7]は，CPU のコア数，GPU レンダリングの有無，メディアデバイスの有無など複数のハードウエア情報を採取できることを示した．

桐生ら[8][9]は単一スレッドでの演算結果の差によって CPU の拡張命令である Streaming SIMD Extensions 2 （SSE2）への対応の有無を特定することを示した．また，Web Worker API を用いてブラウザ上でマルチスレッド処理を実行し，スレッド数ごとの演算速度の差を測定することで CPU コア数と Hyper Threading Technology の搭載を推定できることを示した．

安田ら[10]は演算速度の差異を機械学習にかけることで，CPU 命令の AES-NI と Turbo Boost への対応の有無を特定できることを示した．さらに，JavaScript ベンチマークソフトの結果と CPU 機能の識別を組み合わせることで 75.8%の精度で CPU アーキテクチャを識別できることを示した[11]．

2.2 内部ストレージの比較

デスクトップやラップトップに代表されるコンピュータ端末に搭載される内部ストレージには大きく HDD と SSD の 2 種類が挙げられる． HDD と SSD のデータアクセス速度を比較したとき，一般的に SSD の方が高速である．Amazon の売り上げランキングから最もよく売れている，Western Digital Corporation 社の HDD[12]，および，Micron Technology 社の SSD[13]のデータシートから引用したアクセス速度の比較を図 1 に示す．ここで，HDD はホストからドライブへのデータ転送レート，SSD はシーケンシャルリードにおけるデータ転送レートをアクセス速度である．

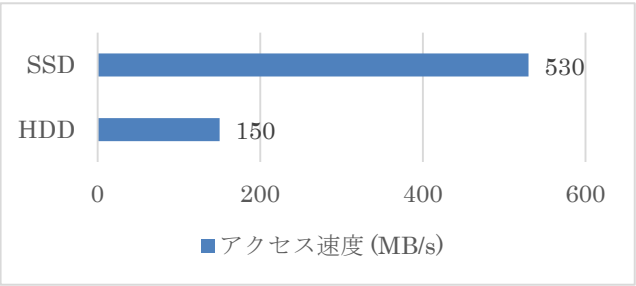

図 1　HDD と SSD のアクセス速度の比較

図 1 から内部ストレージ種別によってアクセス速度に大きな差があることがわかる．よって，ブラウザからこれらのデバイスを直接利用するのであるならば，アクセス速度の差から搭載されている内部ストレージを推定できる可能性が高いといえる．

3. 提案手法

3.1 関連知識

3.1.1 Web Storage API

Web Storage API は HTML5 API の 1 つであり，ブラウザ上でデータの読み書きを行う．データは key-value 形式で管理される．Web Storage API には localStorage と sessionStorage の 2 種類が存在する．これらの違いはデータを保存する期間であり，localStorage は永続的に保存するのに対して，sessionStorage はセッションが切断されるまで保存する．

本研究ではそれぞれ localStorage, sessionStorage のいずれも利用する．なお，localStorage，sessionStorage ともに，提供されているメソッドが等しいので，以降，読み書き処理の解説は localStorage の使用例のみを示す．データの書き込みは localStorage.setItem（[引数 1]，[引数 2]）を用いる．[引数 1]には書き込むデータに対応する key を設定し，[引数 2]にはデータを設定する．データの読み込みは localStorage.getItem（[引数 1]）により行うことができる．[引数 1]には書き込み時に設定した対応する key を設定する．

Web Storage API はモダンブラウザでサポートされている．各ブラウザにおけるサポート状況を表 1 に示す．

表 1　Web Storage API のサポート状況

ブラウザ	サポートされている バージョン
Chrome	5 以降
Firefox	3.5 以降
Safari	4 以降
Internet Explorer	8 以降

3.1.2 High Resolution Time API

High Resolution Time API はブラウザにおいて高精度に時刻計測を行うことのできる API で，performance.now（）によってアクセスすることができる．performance.now（）は呼び出した際に協定世界時からの経過時刻を 5 マイクロ秒単位の値として返す．

High Resolution Time API のサポート状況を表 2 に示す．

表 2　High Resolution Time API のサポート状況

ブラウザ	サポートされている バージョン

Chrome	20.0 以降
Firefox	15.0 以降
Safari	8.0 以降
Internet Explorer	10.0 以降

3.2 提案の概要

2.2 節では，端末の内部ストレージへのアクセス速度が一般に HDD よりも SSD の方が高速であることを述べた．

本研究では, JavaScript によるブラウザへのデータの書き込みや読み込みを行った際にも内部デバイスに応じた速度差が生じると仮定し実験を行った．実験に際し，Web Storage API を用いてブラウザ上でデータの書き込みと読み込みの操作を行い，その実行時間を High Resolution Time API で計測する調査用 Web ページを作成した．

ストレージへのアクセス速度の指標として書き込み速度と読み込み速度が存在するが，提案手法では読み込み処理についてのみ計測を行う．理由は次の通りである．書き込み処理を行う localStorage.setItem（）は関数が呼び出された後，実際の書き込み処理を非同期に実行する．これにより，処理のタイミングはブラウザに依存し，実際の処理にかかる時間を測定するのが困難である．一方，読み込み処理を行う localStorage.getItem（）は関数が呼び出された後，同期的に処理を行うので, 時間計測に適していると言える．

3.3 調査用 Web ページ

以下の（A）～（C）に示す手順により Web Storage API を用いて保存されたデータのアクセス時間を計測する Web ページを作成した．

（A）データの保存

localStorage.setItem（）により文字（ここでの例は'a'）を 1024*1024 個連結した 1MB のデータをブラウザに書き込む．これを（B）で読み込む．次に示すのは，その JavaScript コードである．

```
var data_1mb = new Array（1024 * 1024 + 1）.join（"a"）;
localStorage.setItem（"1mb", data_1mb）;
```

読み込み時，データがブラウザにキャッシュされることを防ぐために, 続けて, ランダムなデータを 3 回書き込む．次に示すのはランダムなデータを生成する JavaScript コードである．

```
function random_data（）{
 var data_length = 1024 * 100;
 var charset = "abcdefghijklmnopqrstuvwxyz";
 var charset_size = charset.length;
 var data = "";
 for（var i = 0; i < data_length; i++）{
  data += charset[Math.floor（Math.random（）*charset_size）];
 }
 return data;
}
```

次に示すのは，前述したランダムな文字列を生成する関数 random_data を用いてランダムなデータを 3 回書き込む JavaScript コードである．これを読み込み直前に都度行う．

```
var random = random_data ()
localStorage.setItem ("random1", random);
localStorage.setItem ("random2", random);
localStorage.setItem ("random3", random);
```

（B）データの読み込み

localStorage.getItem（）により，手順（A）で書き込んだデータを読み込む．この際，localStorage.getItem（）の前後で 3.2.2 節において示した performance.now（）を呼び出し，その差を読み込み時間として計測した．以下の例では変数"diff"が計測時間となっている．次に示すのは，その JavaScript コードである．

```
var start = performance.now（）;
localStorage.getItem（"1mb"）;
var diff = performance.now（）- start;
```

（C）計測時間の平均化

安定した計測時間得るために手順（A）および（B）の計測を都度 50 回行い, その平均値を内部ストレージの推定に利用する計測時間とした．読み込みデータはブラウザにキャッシュされないと仮定する．

3.4 識別精度の算出

ここでは，3.3 節で示した調査用 Web ページから収集したデータによる内部ストレージの推定を行い，その識別精度を算出する方法を示す．

調査用 Web ページに同じ端末から複数回アクセスがあった場合，最初のアクセスから取得した情報のみを分析用データとする．

データベースに保存するデータのフォーマットを以下に示す．"内部ストレージ種別"としてとりうる値は"HDD"もしくは"SSD"の 2 値となっている．

{“内部ストレージ種別”, “読み込み時間”}

3.4.1 時間計測による内部ストレージの識別

本論文で示す実験の全てにおいて，検証方法は LOOCV（Leave-One-Out Cross Validation）を採用した．

内部ストレージの推定は，読み込み時間の閾値を定め，計測した読み込み時間が閾値未満のサンプルを SSD，閾値以上のサンプルを HDD とする．

ここで，内部ストレージ種別を 2 分割するような読み込み時間を閾値とする（式（1）および式（2））．AVE（）は読み込み時間の平均値，MED（）は読み込み時間の中央値を表す．例えば，AVE（HDD）は HDD の読み込み時間の平均値を表す．

$$\text{閾値 I} = \{AVE\,(HDD) + AVE\,(SSD)\}\,/\,2 \quad (1)$$

$$\text{閾値 II} = \{MED\,(HDD) + MED\,(SSD)\}\,/\,2 \quad (2)$$

識別精度は，以下の式（3）と定める．

$$\text{識別精度} = \frac{\text{正しく識別した数}}{\text{サンプル数}} \quad (3)$$

加えて，収集したサンプルのうち読み込み時間が他のサンプルと比較した際に大きく外れた値をとるサンプルを外した（以降，”除外処理を施した”という）分析も行う．読み込み時間が他のサンプルと比較した際に大きく外れた値を除いたサンプルにおいて，それぞれ式 (1) および式 (2) によりそれぞれ閾値を求め，識別を行う．この際，除外するデータは（内部ストレージの読み込み時間の平均値）±（3×標準偏差）の範囲外にあるものとする．これは一般に（平均値±3×標準偏差）の範囲に母集団の 99%以上が属すことに基づき，除外範囲をこのように定めた．この処理を HDD および SSD のサンプルそれぞれに施す．

3.4.2 時間をおいた定期的なアクセスによる推定

1 日以上の間隔を空けて調査用 Web ページへの 5 回以上の定期的なアクセスを行ったサンプルのみを対象とし，時間経過による識別精度の変化を調査する．

閾値は，テストデータ以外のサンプルから前述した式 (1) および式 (2) により計算する．ただし，複数回のアクセスがあった端末においては最初にアクセスしたサンプルのみを利用して閾値を定める．この閾値により，複数回アクセスのうち最初にアクセスしたサンプルの内部ストレージ種別を推定する．

正しく推定できた場合，この端末からアクセスがあった 5 つのサンプルを本節での分析対象とする．そして，2 回目のアクセス，3 回目のアクセス，…，5 回目のアクセスを同じ閾値で識別を行う．

最初のアクセスが正しく判定された端末のうち，5 回のアクセスとも正しく判定された端末の割合を長期的な識別の精度とする．

4. 実験

4.1 概要

本研究では内部にストレージをもつ端末としてデスクトップ PC およびノート PC を調査対象の端末とし，Android や iPhone といったスマートフォンは調査対象外とした．実験に用いたブラウザは Chrome と Firefox であり，3.3 節で示した調査用 Web ページへのアクセスを被験者に依頼することで，読み込み時間の測定を行った．実験期間は 2016 年 11 月 22 日から 2017 年 5 月 8 日である．

4.2 時間計測による内部ストレージの識別（実験 1）

有効なサンプルとして，Chrome から収集したサンプル数は 87 で，そのうち内部ストレージが HDD であったものの数が 49，SSD であったものの数が 38 であった．収集したサンプルと読み込み時間の関係を図 2 および図 3 に示す．なお，図 2 は localStorage を，図 3 は sessionStorage の結果を示す．

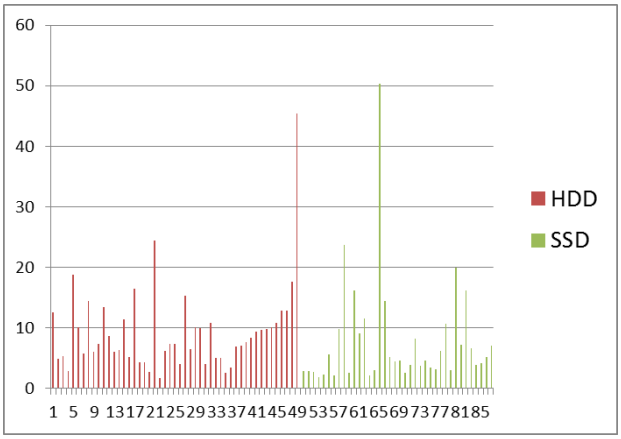

図 2　Chrome における収集サンプルと読み込み時間の関係（localStorage）

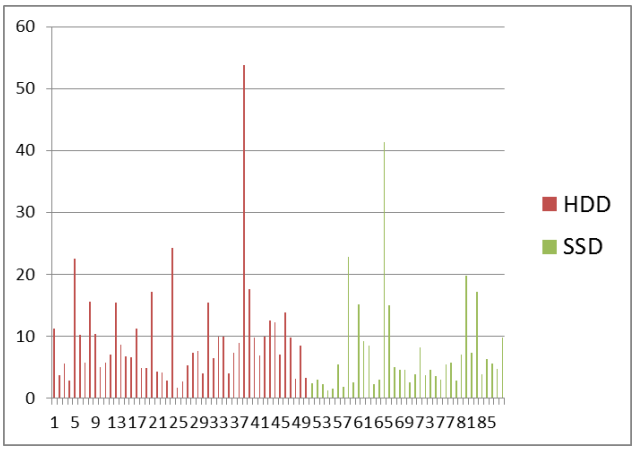

図 3 Chrome における収集サンプルと

読み込み時間の関係（sessionStorage）

図 2 および図 3 の縦軸は読み込み時間（ms）であり，横軸はアクセスを行った端末を表している．赤の棒グラフが HDD のサンプルを表し，緑の棒グラフが SSD のサンプルを表している．図 2 および図 3 より，2.2 節で示した HDD と SSD 間でアクセス速度に数倍の違いがあるが，提案手法ではそのような結果にはならなかった．

また，localStorage と sessionStorage との読み込み時間を比較すると，localStorage と sessionStorage との読み込み時間に違いがなく，同様の結果が得られていることがわかる．

localStorage，sessionStorage のどちらのサンプルにおいても，3.4.1 節で示したような，他のサンプルと比較した際に読み込み時間が大きく外れていたサンプルが HDD と SSD でそれぞれ一つずつあった．

localStorage において，読み込み時間の平均値を用いた手法（以降，手法 I という）による識別精度は 57.5%，除外処理を施した際の識別精度は 58.8%であった．また，読み込み時間の中央値を用いた手法（以降，手法 II という）による識別精度は 65.5%，除外処理を施した際の識別精度は 65.9%であった．

以上の識別結果を表 3 に示す．

表 3　Chrome における識別（localStorage）

	サンプル数	正しく判定された数	識別精度
手法 I（除外なし）	87	50	57.5%
手法 I（除外あり）	85	50	58.8%
手法 II（除外なし）	87	57	65.5%
手法 II（除外あり）	85	56	65.9%

sessionStorage において手法 I による識別精度は 59.1%，除外処理を施した際の識別精度は 59.3%であった．また，手法 II による識別精度は 67.8%，除外処理を施した際の識別精度は 68.2%であった．

以上の識別結果を表 4 に示す．

表 4 Chrome における識別（sessionStorage）

	サンプル数	正しく判定された数	識別精度
手法 I（除外なし）	87	52	59.8%
手法 I（除外あり）	85	51	60.0%
手法 II（除外なし）	87	59	67.8%
手法 II（除外あり）	85	58	68.2%

Firefox から収集したサンプル数は 34 で，そのうち内部ストレージが HDD であったものの数が 18，SSD であったものの数が 16 であった．収集したサンプルと読み込み時間の関係を図 4 および図 5 に示す．

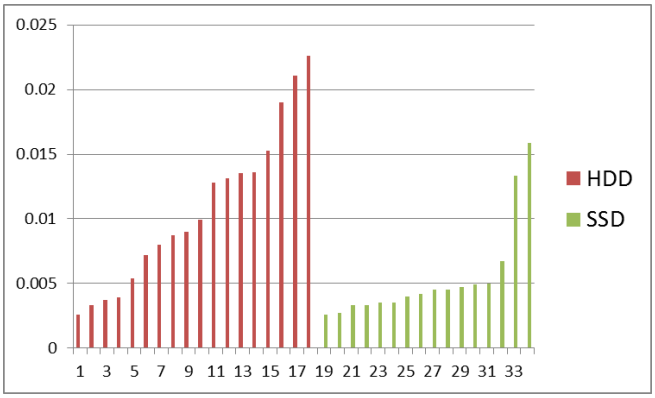

図 4　Firefox における収集サンプルと読み込み時間の関係（localStorage）

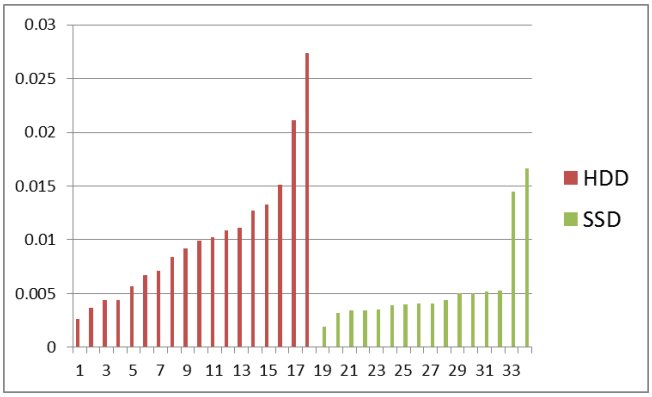

図 5　Firefox における収集サンプルと読み込み時間の関係（sessionStorage）

図 4 および図 5 は図 2 と同様に縦軸が読み込み時間，横軸がアクセスを行った端末である．縦軸の読み込み時間の値が図 2 の Chrome のものと比較して明らかに速いことがわかる．また，収集したサンプル数は Chrome に対し半分以下である．収集サンプルの中に除外サンプルにあたるものはなかった．

localStorage において手法 I による識別精度は 73.5%であり，手法 II による識別精度は 79.4%だった．

以上の識別結果を表 5 に示す．

表 5　Firefox における識別（localStorage）

	サンプ	正しく判定さ	識別精度

	ル数	れた数	
手法 I	34	25	73.5%
手法 II	34	27	79.4%

sessionStorage において手法 I による識別精度は 73.5%であり，手法 II による識別精度は 76.5%だった．

以上の識別結果を表 6 に示す．

表 6　Firefox における識別（sessionStorage）

	サンプル数	正しく判定された数	識別精度
手法 I	34	25	73.5%
手法 II	34	26	76.5%

4.3 時間をおいた定期的なアクセスによる推定（実験 2）

実験 2 は Chrome のみの調査となる．同一端末から 5 回アクセスを行った端末数は 15 だった．

localStorage において初回のアクセスで内部ストレージ種別を正しく推定できたサンプル数は手法 I において 8 であり，このうち 5 回のアクセスすべてが正しく判定されたサンプル数は 5 だった．同様に手法 I のサンプルに対し除外処理を施した際の分析サンプル数は 7 であり，このうち 5 回のアクセスすべてが正しく判定されたサンプル数は 6 だった．手法 II の分析サンプル数は 9 であり，このうち 5 回のアクセスすべてが正しく判定されたサンプル数は 8 だった．手法 II のサンプルに対し除外処理を施した際の分析サンプル数は 9 であり，このうち 5 回のアクセスすべてが正しく判定されたサンプル数は 8 だった．以上の識別結果を表 7 に示す．

表 7　実験 2 による識別（localStorage）

	サンプル数	初回アクセス正判定	5 回すべて正判定（識別率）
手法 I（除外なし）	15	8	5 (62.5%)
手法 I（除外あり）	15	7	6 (85.7%)
手法 II（除外なし）	15	9	8 (88.9%)
手法 II（除外あり）	15	9	8 (88.9%)

sessionStorage において初回のアクセスで内部ストレージ種別を正しく推定できたサンプル数は手法 I において 9 であり，このうち 5 回のアクセスすべてが正しく判定されたサンプル数は 7 だった．同様に手法 I のサンプルに対し除外処理を施した際の分析サンプル数は 8 であり，このうち 5 回のアクセスすべてが正しく判定されたサンプル数は 7 だった．手法 II の分析サンプル数は 8 であり，このうち 5 回のアクセスすべてが正しく判定されたサンプル数は 8 だった．手法 II のサンプルに対し除外処理を施した際の分析サンプル数は 8 であり，このうち 5 回のアクセスすべてが正しく判定されたサンプル数は 8 だった．

以上の識別結果を表 8 に示す．

表 8　実験 2 による識別（sessionStorage）

	サンプル数	初回アクセス正判定	5 回すべて正判定（識別率）
手法 I（除外なし）	15	9	7 (77.8%)
手法 I（除外あり）	15	8	7 (87.5%)
手法 II（除外なし）	15	8	8 (100%)
手法 II（除外あり）	15	8	8 (100%)

5. 考察

4.2 節より，提案手法では中央値を利用し定めた閾値での識別の方が，平均値を利用し定めた閾値での識別に比べて精度が高くなった．このことから，提案手法による読み込み時間は HDD や SSD 同士の中でもある程度の振れ幅があると考えられる．しかし，Chrome で 68.2%，Firefox で 79.4%の識別の精度があることから，同一のストレージ種別の中で振れ幅があったとしても HDD と SSD の間には差異が生じるということが推測できる．

また，localStorage と sessionStorage による識別の精度の大きな違いは確認できなかった．

Chrome と Firefox での読み込み時間に大きな差異が生じており，Firefox の方が 3 桁ほど速いことから 3.3 節の (A) で示したキャッシュ対策が働いていないことが考えられる．

4.3 節により提案手法による内部ストレージ種別の推定結果は時間経過による変化の影響を受けにくいことがわかる．このことから，端末の長期的な識別を行う目的で利用できる可能性が高いと考えられる．

6. まとめ

本論文では，ブラウザを通じて端末の内部ストレージ種別が HDD もしくは SSD なのかをサーバサイドで推定する手法を提案した．その結果，内部ストレージ種別の識別精度は Chrome で 68.2%, Firefox で 79.4%であった. localStorage, sessionStorage を利用した識別の精度に関しては，二者間での差異はなかった．

サンプル数が少ないものの，提案手法での内部ストレージ種別の推定は一定期間での判定結果に変化が生じにくいことから，長期的な識別に利用できる可能性があることがわかった．

今後の課題として，サンプル数を増やした上で，評価を取ることを行いたい．

参考文献

[1] 磯侑斗，桐生直輝，塚本耕司，高須航，山田智隆，武居直樹，齋藤孝道，“Web Browser Fingerprint を採取する Web サイトの構築と採取データの分析”，コンピュータセキュリティシンポジウム 2014 , 2014.

[2] https://webtransparency.cs.princeton.edu/webcensus/

[3] K. Mowery, H. Shacham, Pixel perfect: fingerprinting canvas in HTML5, in Proc. of Web 2.0 Security and Privacy （W2SP）, 2012.

[4] K. Mowery, D. Bogenreif, S. Yilek, H. Shacham, “Fingerprinting information in JavaScript implementations”, Proc. of Web 2.0 Workshop on Security and Privacy （W2SP）, 2011.

[5] G Nakibly, G Shelef, S Yudilevich, “Hardware Fingerprinting Using HTML5”, Cornell University Library, 2015.

[6] H. Bojinov, Y. Michalevsky, G. Nakibly, D. Boneh, Mobile Device Identification via Sensor Fingerprinting, arXiv:1408.1416 [cs.CR], 2014.

[7] 後藤浩行，齋藤孝道，”Web 行動追跡のためのハードウエア特徴点の抽出”，暗号と情報セキュリティシンポジウム，2013

[8] 桐生直輝，磯侑斗，金子洋平，齋藤孝道，“Web Workers を用いた演算処理性能の差による CPU コア数の推定”，コンピュータセキュリティシンポジウム 2014，2014

[9] 桐生直輝，後藤浩行，齋藤孝道，“CPU 拡張命令の対応の有無による CPU アーキテクチャの推測”，第 75 回情報処理学会全国大会，2013

[10]安田昂樹，高須航，山田智隆，武居直樹，西倉裕太，石川貴之，細井理央，高橋和司，齋藤孝道，"Web Browser Fingerprinting 技術を用いた CPU 拡張機能の推定法の提案と実装", 暗号と情報セキュリティシンポジウム，2016

[11]齋藤孝道，安田昂樹，石川貴之，細井理央，高橋和司，細谷竜平，田邉一寿，種岡優幸，"ブラウザにおけるサイドチャネルを用いた CPU 推定", 暗号と情報セキュリティシンポジウム，2017

[12] WD Blue PC Hard Drive Series Distribution Specification Sheet, https://www.wdc.com/content/dam/wdc/website/downloadable_assets/eng/spec_data_sheet/2879-771436.pdf

[13] Crucial® MX300 Solid State Drive, http://www.crucial.com/wcsstore/CrucialSAS/pdf/product-flyer/crucial-mx300-ssd-productflyer-a4-en.pdf