Proof of Work を用いた検証可能なデバイスフィンガープリントの一提案

天笠智哉[1] 奥村紗名[1] 井坂佑介[1] 佐々木葵[1] 大木哲史[1] 西垣正勝[1]

概要： PC 内の様々な特徴の組み合わせを ID として用いることで，デバイスを識別するデバイスフィンガープリントという技術がある．デバイスフィンガープリントはブラウザや PC の識別に利用されており，ユーザトラッキング，リスクベース認証，reCAPTCHA，アクティベーションなどに活用されている．しかし，既存の一般的なデバイスフィンガープリントは PC 内の様々な静的属性・動的属性を取得して用いているに過ぎないので，改ざんは容易である．そこで本稿では，Proof of Work (PoW)を利用したデバイスフィンガープリントを提案する．PoW とは，「ある水準の作業をある一定の時間をかけて行ったことを明らかにするプロトコル」のことであり，PoW の実行時間をフィンガープリントの特徴量（識別子）とすることで，攻撃者が「自身の所持するデバイスの性能を超える特徴量」を提示することができない．また，PoW は「求解は困難だが，その検証は容易である」という一方向性を有するため，フィンガープリントの真正性の検証を行うことができる．提案方式の第一報として，CPU バウンドな PoW である HashCash とメモリバウンドな PoW である Equihash を用い，異なる CPU，RAM を搭載した PC の識別が可能であることを明らかにした．

A Study on Probable Device Fingerprints Using Proof of Work

TOMOYA AMAGASA[1] SANA OKUMURA[1] YUSUKE ISAKA[1]
AOI SASAKI[1] TETSUSHI OHKI[1] MASAKATSU NISHIGAKI[1]

1. はじめに

PC 内の様々な特徴の組み合わせを ID として用いることで，デバイスを識別するデバイスフィンガープリント[1]という技術がある．デバイスフィンガープリントはユーザの PC 内の多様化したデバイス構成･ソフトウェア構成の特徴量を用いるため，プライバシー感度（識別精度）の調整が可能（多数の特徴量を組み合わせるほどトレーザビリティの精度が高まる）であり，ユーザの能動的な操作なしに特徴量を取得することができるという性質がある．ブラウザや PC の識別に利用されており，ユーザトラッキング[1][2]，リスクベース認証[3]，reCAPTCHA [4][5]，アクティベーション[6]などに活用されている．

既存の一般的なデバイスフィンガープリントは PC 内の様々な静的属性・動的特性を取得して用いているに過ぎないので，改ざんは容易である[7]．しかし，認証・認可・課金などの用途に供されるフィンガープリントにおいては，その高信頼性が要求される．フィンガープリントの耐タンパ性を高めるためには，改ざんが困難な特徴量を用いてフィンガープリントを構成することが肝要である．また，フィンガープリントが改ざんされていないことを検証できることも，重要な要件である．

そこで本稿では，Proof of Work(PoW)を利用したデバイスフィンガープリントを提案する．PoW とは，コンセンサスアルゴリズムの技術要素であり，「ある水準の作業をある一定の時間をかけて行ったことを明らかにするプロトコル」のことである[8]．PoW の実行は，計算資源の消費を保証する．よって，PoW の実行時間を特徴量（識別子）とすることで，攻撃者が「自身の所持するデバイスの性能を超える特徴量」を提示することができないフィンガープリントが実現される．また，PoW は「求解は困難だが，その検証は容易である」という一方向性を有するため，フィンガープリントの真正性検証の要件も達成される．

提案方式の第一報として，CPU バウンドな PoW である HashCash [9]を利用したフィンガープリントと，メモリバウンドな PoW である Equihash [10] を利用したフィンガープリントを扱う．前者においては，HashCash の実行時間を特徴量とし，CPU の処理速度を測ることにより，CPU の識別が可能である．後者においては，Equihash の実行時間を特徴量とし，RAM の処理速度を測ることにより，RAM の識別が可能である．HashCash と Equihash を用いることにより，CPU と RAM が異なる PC の識別が可能であるか調査する．

2. 既存研究

2.1 デバイスフィンガープリント

デバイスフィンガープリントとは，ソフトウェア構成情報とハードウェア構成情報の組み合わせを ID として用い

1 静岡大学
Shizuoka University

表 1：デバイスフィンガープリントの代表的な例

静的情報	ブラウザ	Accept ヘッダ，userAgent，プラグインの種類
	ハードウェア	CPU，GPU，画面解像度，リフレッシュレート
	OS	OS の種類，言語，タイムゾーン
動的情報	演算速度	ベンチマークスコア
	演算結果	Canvas API の描画結果

ることで,個々人のデバイスを識別する技術のことである．デバイスフィンガープリントの特徴量の代表的な例[11]を表 1 に示す．

静的なデバイスフィンガープリントに関しては，Eckersley は，Flash や Java が実行可能なブラウザにおいては，HTTP ヘッダと JavaScript・Flash・Java から得られるデバイス情報より，ユーザを 94.2%の精度で一意識別可能であることを示している [1]．北條らは，クライアントからの HTTP リクエスト時の HTTP ヘッダに対し，深層学習を用いてユーザの識別を行った結果，F1 値（適合率と再現率の調和平均）が 0.99 以上という精度を得られることを示した [12]．

動的なフィンガープリントに関しては，Mowery らはブラウザに搭載されている JavaScript エンジンの性能を測定する SunSpider や自作のベンチマークテストを実行することでデバイスを識別する手法を提案し，79.8%の精度でブラウザとそのバージョンを検出できることを示している [13]．Nakibly らは，ウェブブラウザ上で 3D グラフィックをレンダリングする JavaScript API である WebGL を用いて，複雑な 3D グラフィックをレンダリングし，フレーム数を測定することで GPU を識別する手法を提案している [14]．齋藤らは暗号化関数を実行し実行時間を比較することで，Advanced Encryption Standard New Instructions (AES-NI)と Intel Turbo Boost Technology (Turbo Boost)と呼ばれる CPU 拡張機能の有無を推測する手法を提案している[15]．

2.2 デバイスフィンガープリントの応用例

2.2.1 ユーザトラッキング

現在，Web サイトの多くでは何らかのユーザトラッキングが行われている．Durey らは，サインアップページ，サインインページ，E コマースサイトにおける決済情報の入力を求めるページ，E コマースサイトにおける買い物かごページの 4 つのタイプのセキュアな Web ページを含む Web サイトにおいて，446 件の Web サイトの 1,485 ページから 169 個のフィンガープリントスクリプトを発見している [16]．AI-Fannah らは，The Majestic Million[17]掲載のウェブサイト 10,000 件をクロールした結果，6,876 件(68.7%)のウェブサイトが何らかのフィンガープリント属性を取得し，そのうち 84.5%はサードパーティ製のフィンガープリンティングツールを利用していることを明らかにした[2]．

2.2.2 リスクベース認証

リスクベース認証とは，アクセスログなどからユーザの振る舞いのアノマリ分析を行い，リスク値に応じてユーザ認証の強度を変更する認証方式である [18]．アクセスログとしては，ユーザの位置情報，利用デバイス，タイムゾーン，IP アドレスなどが用いられる[19]．シングルサインオン(SSO)を実現する OpenAM というオープンソースソフトウェア（OSS）において実装されているリスクベース認証の機能では，実際にブラウザフィンガープリントが用いられている [3]．

2.2.3 reCAPTCHA v3

reCAPTCHA v3 とは，Google 社が提供するボット検知システムであり，スパム攻撃対策として利用されている[20]．reCAPTCHA v3 では, Canvas API による描画結果, userAgent, 画面解像度，プラグインの種類などの情報を利用してボットか否かを検知している[4][5]．

2.2.4 アクティベーション

インストールされている OS やソフトウェアが正規品であるか否かを確認するアクティベーションにおいて，ライセンス情報をデバイスフィンガープリントと関連付けて管理する方法が採用されている．Microsoft 社の Windows OS では，ライセンス情報がデバイスのハードウェア情報と関連付けられており[6]，マザーボードの交換など，ハードウェア情報に大幅な変更が行われた場合には，OS のライセンスが一時的に無効化される．

2.3 Proof of Work

Proof of Work(PoW)とは，コンセンサスアルゴリズムを構成する技術要素であり,「ある水準の作業をある一定の時間をかけて行ったことを明らかにするプロトコル」である．PoW の重要な要件の 1 つが一方向性である[21]．一方向性とは,「求解は困難だが，その検証は容易である」という性質である．本稿では HashCash と Equihash の 2 つの PoW を扱う．

2.3.1 Hashcash

HashCash は，「SHA256 を用いてハッシュ化すると先頭のDビットが 0 となる入力値$Nonce$を探す」という方式の PoW である．迷惑メールやサービス不能攻撃（DoS）を防止するタールピットとして[9]，あるいは，ビットコインをはじめとした暗号資産のマイニングアルゴリズムとして採用されている[22]．

暗号学的ハッシュ関数である SHA256 の性質から，PoW を完遂するには出力値の先頭Dビットが 0 となるような入力値$Nonce$を総当たりで探す必要がある．この総当たりで条件を満たす入力値$Nonce$を見つけることの難易度はDの大きさによって決まる．「出力値の先頭Dビットが 0 となる入力値$Nonce$を探す」という操作は，「出力値が2^{256-D}以下となる入力値を探す」という操作と同等である．したがって，D を大きくするほど条件を満たす出力値がハッシュ関数から出力されにくくなる．また，ハッシュ関数は，入力値に対して出力値が一意に定まる．したがって，同じ入力値$Nonce$が使われるリプレイ攻撃を防ぐために，入力値$Nonce$に$Salt$と呼ばれるランダムなビット列を連結して探索を行わせる必要がある．

HashCash は，証明者は条件を満たす$Nonce$を見つけるために，平均して2^D回のハッシュ関数の実行が必要であるのに対し，検証者は証明者が見つけた$Nonce$を入力として，ハッシュ関数を一度実行するのみで検証が完了するため，一方向性を満たしている．

2.3.2 Equihash

Equihash は Biryukov らによって提案された PoW の派生技術である[10]. HashCash は CPU バウンドな PoW であり，ASIC（特定用途向け集積回路）を用いることで通常の CPU よりも高速に実行できることが知られている[23]．そのため，HashCash を採用する暗号資産では ASIC を所持するユーザと通常の CPU を用いるユーザで格差が生じていた. これに対し，Equihash はメモリバウンドな PoW であり，その求解に要する時間においてメモリの転送速度が支配的となる．このように Equihash は，CPU の性能差に大きく左右されることのない PoW を実現している[10].

Equihash は，以下の式を満たす$Nonce$及び$x_1, x_2, \dots, x_{2^k}$を探し出すことを要求する．ここで，$H(x)$はxを入力値として得られるハッシュ関数Hの出力結果である．$Salt$の役割は HashCash と同様にリプレイ攻撃を防ぐことである．

$$H(Salt||Nonce||x_1) \oplus H(Salt||Nonce||x_2) \oplus \cdots \oplus H(Salt||Nonce||x_{2^k}) = 0 \quad (1)$$

式(1)の求解は以下に示す Wagner のアルゴリズム[24]が最も効率的であることが知られており，Equihash はこのアルゴリズムを採用している．ここでは$X_i = H(Salt||Nonce||i)$であり，$2^{\frac{n}{k+1}+1} \gg 2^k$である．

1. $0 \le i \le 2^{\frac{n}{k+1}+1}$に対してハッシュ値$X_i$を求め，$X_i$と添え字$i$のタプル$(X_i, i)$をテーブルに格納する．
2. テーブルをX_iでソートする．
3. X_iとX_j（$i \neq j$）の第 1～$\frac{n}{k+1}$ビットが衝突するタプル$(X_i, i), (X_j, j)$のペアをすべて取り出す．各ペアにおいて，$X_{i,j} = X_i \oplus X_j$を計算し，タプル$(X_{i,j}, (i, j))$をテーブルに再格納する．$X_{i,j}$は第 1～$\frac{n}{k+1}$ビットまでが 0 になっている．
4. X_iとX_j（$i \neq j$）の第$\frac{n}{k+1}+1$～$\frac{2n}{k+1}$ビットが衝突するタプル$(X_i, i), (X_j, j)$のペアをすべて取り出す．各ペアにおいて，$X_{i,j} = X_i \oplus X_j$を計算し，タプル$(X_{i,j}, (i, j))$をテーブルに再格納する．$X_{i,j}$は第 1～$\frac{2n}{k+1}$ビットまでが 0 になっている．
5. 衝突するビットの位置をずらしながら，$X_{i,j}$の第 1～$\frac{(k-1)n}{k+1}$ビットまでが 0 になるまで手順 4 を繰り返す．
6. X_iとX_j（$i \neq j$）の第$\frac{(k-1)n}{k+1}+1$～nビットが衝突するように複数のタプル(X_i, i)を取り出す．このタプルに含まれる添え字の集合が式(1)の解の候補となる．
7. 解の候補のうち，要素（添え字）に重複がないものが解となる．

Equihash は，証明者は求解に$k2^{\frac{n}{(k+1)}+2}$回の計算が必要であるのに対し，検証者は2^k回の計算で検証が完了するため，$n > k^2$の条件の下に一方向性を満たしている．

2.4 Proof of Work の応用例

2.4.1 ブロックチェーン

ブロックチェーンは主に暗号資産で利用されている分散台帳技術である[22]. ブロックチェーンでは，トランザクションと呼ばれる取引データ，タイムスタンプ，前ブロックのハッシュ値などの情報から構成されるブロックが鎖状に連なり，データが保持される．

新たにブロックを連結する際には，PoW を利用したコンセンサスアルゴリズムが課される．前述のとおり取引データが含まれるブロックには，前ブロックのハッシュ値が含まれている．したがって，ブロックチェーンのデータを改ざんするには，改ざん対象のブロックより後ろにつながるブロックで行われてきたコンセンサスアルゴリズムの演算をすべてやり直す必要があるため，現実的に改ざんが不可能であるとされている．

2.4.2 コンタクトトレーシング

福田らは，ユーザによるデータの偽造およびプライバシーを考慮したブロックチェーンを用いて，コンタクトトレーシング手法を提案している[25]. 先行研究[26]ではユーザによるデータの偽造を考慮していなかったが，この手法では，チャレンジアンドレスポンスとスマートコントラクトによる検証を組み合わせることで，ユーザによるデータの偽造を検知可能にしている．普段は位置情報を暗号化し，ユーザのプライバシーを保証するが，必要とあらばユーザの承認を得ることで詳細な行動履歴の追跡（オプトイン型のコンタクトトレーシング）が可能となっている．

2.4.3 ソフトウェア著作権保護システム

高木らは，ブロックチェーンを用いてクラウド上でのソフトウェア著作権保護システムを構築し，悪意あるユーザによるライセンス情報の流出や不正なログインのリスクを軽減する手法を提案している[27]. 先行研究[28]では，デジタル著作権管理(Digital Rights Management: DRM)に関する情報を一元管理し，ユーザにサービスを提供する DRM プロキシと呼ばれるモデルが提案されていた．しかし，このモデルでは DRM に関する通信とユーザ・ライセンス情報が DRM プロキシに集中する形になっており，プロキシが攻撃されてしまうと，アカウント情報・認証ツールの改ざんや流出などリスクが非常に大きいという問題があった．これに対し，この提案手法では，ユーザ・ライセンス情報をブロックチェーンで分散させることでリスクを低減させている．

2.4.4 IoT 機器のセキュアブート

伊藤らは，PoW の派生技術である Proof of Space を用いて，IoT 機器ソフトウェアに対する改ざん検知システムの提案を行っている[29]. デバイス内で稼働するソフトウェアの真正性をデバイス起動時に確認する技術であるセキュアブートでは，Root of Trust として耐タンパモジュールなどの特別なハードウェアが必要とされる．しかし，一般的に IoT 機器は単一機能を安価に提供するという制約を受けるため，耐タンパモジュールを搭載することが困難な場合が多い. この問題に対し, 伊藤らの手法では, Proof of Space を用いて特別なハードウェアを必要としないホワイトリスト型ソフトウェア改ざん検知技術を提案している．

3. 改ざん耐性を有するデバイスフィンガープリント

2.2 節で概説した通り,デバイスフィンガープリントの応用例は多様であり，改ざん耐性を備えていることが好ましい．しかし，静的な情報を利用したデバイスフィンガープリントは，容易に改ざん可能である．ベンチマークスコアなどの動的な情報を利用したデバイスフィンガープリントも，実際にユーザがベンチマークを実行しているかどうかを検証する術が伴わない限り，スコア自体を改ざんすることができてしまう．すなわち，デバイスフィンガープリントをより高信頼な情報源にするためには，以下のような要件が満たされていることが必要であると考えられる．

- 要件１：デバイスフィンガープリントの特徴量として，改ざんが困難な特徴量を用いること
- 要件２：デバイスフィンガープリントが改ざんされていないかという真正性の検証が可能であること

この課題に対し，著者らは，Proof of Work (PoW)を利用したデバイスフィンガープリントを提案する（図 1）. PoW の実行は，計算資源の消費を保証する．よって，PoW の実行時間をデバイス識別の特徴量（識別子）として用いることで，攻撃者が「自身の所持するデバイスの性能を超える特徴量」を提示することができないフィンガープリント（要件 1）が実現される．また，PoW は「求解は困難だが，その検証は容易である」という一方向性を有するため，フィンガープリントの真正性検証の要件（要件 2）も達成される．

本稿では，提案方式の第一報として，CPU バウンドな PoW である HashCash とメモリバウンドな PoW である Equihash を利用したフィンガープリントを扱う．

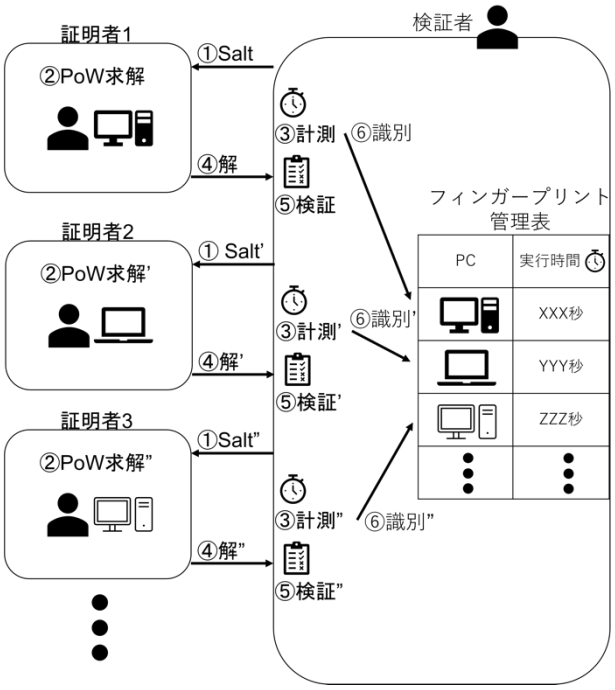

図 1：提案手法

4. 検証

4.1 調査項目

HashCash は CPU バウンドなアルゴリズムであり，Equihash はメモリバウンドなアルゴリズムであることが知られている．したがって，HashCash，Equihash の実行時間を計測することで，CPU や RAM を識別できることが期待される. そこで本章では, HashCash と Equihash の実行時間をフィンガープリントとして用いた場合に，CPU および RAM が異なる PC の識別が可能か否かを確認するための検証を行う．

本稿では，その第一報として，以下の調査項目（RQ）を解明することを目的として実験を行う．

- RQ：異なる CPU，RAM を搭載している PC において HashCash や Equihash を実行させた場合に，その実行時間の差異によって PC の識別は可能なのか？

4.2 調査方法

前節で述べた調査項目を明らかにするために，複数の PC において HashCash および Equihash の 400 回の求解をそれぞれ 20 セット行い，その平均実行時間について一元配置

分散分析およびTukey HSD法による多重比較を行う. なお, 今回は CPU，RAM の区別をメーカと型番で行うことにする．以降，CPU，RAM の種類は，メーカ名:型番（規格）の表記とする．また，今回は HashCash と Equihash の計算には GPU を使用しない. 更に, プログラムのマルチスレッド化も行わない．したがって，HashCash，Equihash の求解速度にはCPU のシングルコアと RAM の処理性能のみが反映されることが期待される.

4.3 実行環境

計測に利用したソフトウェア環境ならびにハードウェア環境を，それぞれ，表 2，表 3 に示す．なお，表 3 の「laptop」の製品名は Lenovo IdeaPad Slim 560i Pro (82L300D3JP)である.

表 2：計測に利用したソフトウェア環境

PoW	OS	使用言語
HashCash	Ubuntu 20.04.4 LTS	Python 3.8.10
Equihash		C++11(g++9.4.0)

4.4 計算方法

HashCash の計測においては，難易度 D を 5 bit とした．求解ごとに 256 bit のランダムビット列を生成し，これを Salt として指定する. Equihash の計測においては, n を 80, k を 4 とした．求解ごとに 0xFFFFFF 以内の正整数をランダムに生成し，これを Salt として指定する．以下の計測を各デバイスにおいて 20 セット行う.

1. PC 起動する.
2. HashCash の求解を 400 回実行し，その合計実行時間を計測する．（Salt は各求解でランダムに指定する.）
3. PC を再起動する.
4. Equihash の求解を 400 回実行し, その合計実行時間を計測する．（Salt は各求解でランダムに指定する.）

4.5 結果

表 4 に各デバイスにおける Hashcash と Equihash の合計実行時間の平均及び標準偏差を示す．表 4 の計測結果に対して統計分析を行い，4.1 節で設定した RQ を解明する．3 群以上の比較として一元配置分散分析を行った後に，各 2 群間で TukeyHSD 法による多重比較を行う．有意差水準は 5%とする.

一元配置分散分析の結果，$p<0.05$ となり，群間で何らかの差があることが確認された．どの群間に差があるのか確認をするために，各 2 群間で TukeyHSD 法による多重比較を行った．その結果をそれぞれ表 5，表 6 に示す．表中の値はp値である. 小数第3位以下を切り捨て表示してある.

表 5，表 6 より，desktop1-1 と desktop1-2 の HashCash の検定結果を除いた 2 群間で有意差が認められる結果になった．すなわち，HashCash と Equihash の実行時間から，CPU および RAM が異なる PC の識別が可能であることが確かめられた。以降では特筆すべき点について論じる．表 3 より，desktop1-1 と desktop1-2 は，CPU は同一，RAM は不同である．これに対し，表 5，表 6 より、HashCash の実行時間には有意差が認められず、Equihash の実行時間には有意差が認められた．この結果より，CPU が同一，RAM が不同という条件下において，「HashCash の実行時間を用いることで，CPU が同一であると判定できる」,「Equihash の実行時間を用いることで不同の RAM を識別できる」ことが確かめられた．表 3 より，desktop1-2 と desktop2 は、CPU は不同であり, 性能も大きく異なる. RAM も不同であるが, 規格上の性能は同じである. これに対し, 表 5, 表 6 より、HashCash も Equihash も実行時間に有意差が認められた．この結果より，「(CPU が不同である場合には）RAM の規格上の性能が同じであったとしても，Equihash の実行時間を用いることで不同の RAM を識別できる」ことが確かめられた．今後、CPU が同一である場合にも同様の結果が得られるか確かめる必要がある.

表 3：計測に利用したデバイスのハードウェア環境

デバイス名	CPU	RAM:
desktop1-1	Intel: i9-10850K	Crucial: CT2K8G4DFS832A (DDR4-3200, 8GB×2)
desktop1-2	Intel: i9-10850K	Crucial: W4U2666CM-8GR (DDR4-2666, 8GB×2)
desktop2	Intel: i5-6600	Corsair: CMK16GX4M2A2666C16 (DDR4-2666, 8GBx2)
laptop	Intel: i7-1195G7	lenovo:オンボード (DDR4-3200, 16GB)

表 4：各デバイスにおける Hashcash と Equihash の合計実行時間の平均及び標準偏差

デバイス	合計実行時間の平均: HashCash [ms]	標準偏差: HashCash [ms]	合計実行時間の平均: Equihash [s]	標準偏差: Equihash [s]
desktop1-1	9.518	0.4452	61.22	19.62
desktop1-2	9.542	0.5437	63.06	15.40
desktop2	13.34	0.7239	116.7	31.34
laptop	8.150	0.6830	65.84	18.22

表 4 と表 6 より，今回の計測では，Equihash の実行時間によって，すべてのデバイスを識別できることが確認できた．しかし，Equihash は求解の際の実行時間の標準偏差が大きい．従って，求解を大量に繰り返さなければデバイス識別の精度を確保することができず，現時点においては識別に 60 秒以上を要するため，フィンガープリントとして運用するためには計測手法を改良する必要がある．

表 5：HashCash の多重比較の結果

	desktop 1-1	desktop 1-2	desktop 2	laptop
desktop1-1	-	-	-	-
desktop1-2	0.99	-	-	-
desktop2	0.00**	0.00**	-	-
laptop	0.00**	0.00**	0.00**	-

**:<0.1, *:<0.05

表 6：Equihash の多重比較の結果

	desktop 1-1	desktop 1-2	desktop 2	laptop
desktop1-1	-	-	-	-
desktop1-2	0.04*	-	-	-
desktop2	0.00**	0.00**	-	-
laptop	0.00**	0.00*	0.00**	-

**:<0.1, *:<0.05

5. まとめ

本稿では，PoW の実行時間を識別子として用いることで，改ざんが困難，かつ，真正性の検証が可能なデバイスフィンガープリントを提案した．提案方式の第一報として，CPU バウンドな PoW である HashCash を利用したフィンガープリントと，メモリバウンドな PoW である Equihash を利用したフィンガープリントを扱った．検証の結果，HashCash と Equihash の実行時間から，CPU および RAM が異なる PC の識別が可能であることが確かめられた．また，HashCash と Equihash の実行時間は CPU の性能に依存するため，低性能な PC の所持者が高性能な PC の所持者になりすますことは困難であることが確認された．今後は，以下の事項について取り組む．

- Equihash に関して，短い実行時間でデバイスを識別が行えるよう，計測手法を改良する．
- 計測対象のデバイス数を増やすことで，提案手法の解像度を確かめる．
- HashCash や Equihash のプログラムをマルチスレッド化することで結果にどのような影響が及ぼされるのか確認を行う．
- 仮想マシンや GPU が異なる PC についても識別可能であるか否かを調査する．

参考文献

[1] P. Eckershley.:How Unique Is Your Web Browser?,PETS’10, pp1-18 (2010)

[2] Al-Fannah, N.M.， Mitchell, C.J.：Beyond Cookie Monster Amnesia: Real World Persistent Online Tracking，ISC 2018，pp.481-501（2018）.

[3] ForgeRock:Implementing Device Fingerprints With Intelligent Authentication Trees in AM，（オンライン），入手先 <https://developer.forgerock.com/docs/platform/how-tos/implementing-device-fingerprints-intelligent-authentication-trees-am>，（参照 2022-03-22）.

[4] Sivakorn, S., Polakis, J., Keromytis, A. D.: I’m not a human: Breaking the Google reCAPTCHA, Black Hat 14(2016).

[5] neuroradiology：InsideReCaptcha，github.com，（オンライン），入手先〈https://github.com/neuroradiology/InsideReCaptcha〉（参照 2022-5-16）

[6] Microsoft：ハードウェア構成の変更後に Windows のライセンス認証をもう一度行う，Mirosoft サポート，（オンライン），入手先〈https://support.microsoft.com/ja-jp/windows/ハードウェア構成の変更後に Windows のライセンス認証をもう一度行う-2c0e962a-f04c-145b-6ead-fb3fc72b6665〉（参照 2022-5-16）

[7] Vastel, A.，Laperdrix, P.，Laperdrix, P.ほか：Fp-Scanner: The Privacy Implications of Browser Fingerprint Inconsistencies，USENIX Security 18，pp.135-150（2018）.

[8] Jakobsson, M.，Juels, A. ：Proofs of work and bread pudding protocols，Secure information networks，pp.258-272（1999）.

[9] A.Back.: Hashcash - A Denial of Service Counter-Measure. (オンライン)，入手先 <http://www.HashCash.org/HashCash.pdf>,(2002).

[10] A.Biryukov , D. Khovratovich.:Equihash: asymmetric proof-of-work based on the Generalized Birthday problem，NDSS(2016).

[11] Laperdrix, P.，Bielova, N.，Baudry, B.ほか：Browser fingerprinting: A survey，ACM Transactions on the Web，Vol.14，No.2，pp.1-33（2020）.

[12] 北條大和，齋藤祐太，齋藤孝道：深層学習を用いたパッシブフィンガープリンティング手法の提案と実装，コンピュータセキュリティシンポジウム 2019 論文集，pp.252-259（2019）. .

[13] Mowery, K.， Bogenreif, D.， Yilek, S.ほか：Fingerprinting information in JavaScript implementations，Proceedings of W2SP（2011）.

[14] Nakibly, G., Shelef, G., Yudilevich, S.：Hardware fingerprinting using HTML5，arXiv preprint arXiv:1503.01408.，（オンライン），入手先〈https://arxiv.org/abs/1503.01408〉（参照 2022-5-16）.

[15] 齋藤 孝道，安田 昂樹，石川 貴之ほか：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 2017,2C3-2,(2017).

[16] Durey, A.， Laperdrix, P.，Rudametkin, W.ほか：FP-Redemption: Studying browser fingerprinting adoption for the sake of web security，International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment，pp.237-257（2021）.

[17] majestic.：The Majestic Million，Majestic，（オンライン），入手先〈https://majestic.com/reports/majestic-million〉（参照 2022-5-16）.

[18] NEC Solution Innovators, Ltd：リスクベース認証，サイバーセキュリティ セキュリティ用語集，（オンライン），入手先〈https://www.nec-solutioninnovators.co.jp/ss/insider/security-

words/24.html〉（参照 2022-5-16）.
[19] 瀬戸洋一：認証技術の種類と動向，電気設備学会誌，Vol.30，No.10，pp.809-812（2010）.
[20] Google Developers：reCAPTCHA v3，reCAPTCHA，（オンライン），入手先〈https://developers.google.com/recaptcha/docs/v3〉（参照 2022-5-16）.
[21] Biryukov, A.，Khovratovich, D.：Egalitarian computing，USENIX Security 16，pp.315-326（2016）.
[22] Nakamoto, S：Bitcoin: A peer-to-peer electronic cash system. ，Bitcoin，（オンライン），入手先〈http://www.bitcoin.org/bitcoin. pdf〉（参照 2022-5-16）.
[23] Dwork, C.，Goldberg, A.，Naor, M.：On memory-bound functions for fighting spam.，CRYPTO’03，Vol.2729，pp.426-444（2003）
[24] Wagner, D：A generalized birthday problem，CRYPTO’02，Vol.2442，pp.288-303（2002）.
[25] 福田竜央，面和成：プライバシーを考慮したブロックチェーンを用いた柔軟なコンタクトトレーシング手法，Symposium on Cryptography and Information Security2022, 2D3-3，（2022）.
[26] Lv, W., Wu, S., Jiang, C., Cui, Y., Qiu, X.,Zhang, Y. ：Decentralized blockchain for privacy-preserving large-scale contact tracing，arXiv preprint arXiv:2007.00894.，（オンライン），入手先〈https://arxiv.org/abs/2007.00894〉（参照 2022-5-16）.
[27] 高木誠也，柿崎淑郎，廣瀬幸ほか：ブロックチェーンを使用したクラウド上でのソフトウェア著作権保護システムの提案，インターネットと運用技術研究会，Vol.2019-IOT-44，No.35，pp.1-6（2019）.
[28] Lee, H.，Seo, C.，Shin, S. U.：DRM Cloud Architecture and Service Scenario for Content Protection，JSIS2013，Vol.3，pp.94-105（2013）.
[29] 伊藤真奈美， 山越公洋，瀧口浩義ほか：Proof of Space を活用した改ざん検知システムの提案，コンピュータセキュリティシンポジウム 2019 論文集，pp.1448-1453（2019）.