Fuzzy Hashing を用いた比較による長期的な Browser Fingerprinting の端末識別手法の提案

石川貴之† 高須 航† 山田智隆† 武居直樹†
細井理央† 高橋和司‡ 安田昂樹‡ 齋藤孝道‡
†明治大学大学院 ‡明治大学

あらまし 端末から収集可能な複数の情報を組み合わせることで端末を識別する Browser Fingerprinting と呼ばれる手法が利用されている．しかし，この手法は端末のアクセス間隔が長くなるにつれて識別精度が低下することが我々の先行研究により判明した．本論文では，Fuzzy Hashing を用いた比較により Browser Fingerprinting を用いた端末識別の長期化の手法を提案する．また，我々が運用している情報収集サイトで得られたデータをもとに分析を行い，文字列の完全一致による識別と比較して識別精度が向上したことも示す．

A Long Term Identification with Fuzzy Hashing in Browser Fingerprinting

Takayuki ISHIKAWA† Ko TAKASU† Tomotaka YAMADA† Naoki TAKEI†
Rio HOSOI† Kazushi TAKAHASHI‡ Koki YASUDA‡ Takamichi SAITO‡
†Graduate School of Meiji University ‡Meiji University

Abstract The browser fingerprinting is getting popular to identify Web clients by information emitted from the device. However, we previously revealed that the browser fingerprinting was not good at long-term use. In this paper, we propose an identification method for long-term tracking using with the fuzzy hashing in the browser fingerprinting. Furthermore, based on the data collected in our site, we show that identification accuracy of our proposed method is higher than that of exact match.

1 はじめに

Web サーバが Web ブラウザから収集可能な端末の情報を複数組み合わせることで，アクセスする端末を識別する手法に Browser Fingerprinting（以降，Fingerprinting と呼ぶ）がある．この手法はアクセス解析における利用者識別などにおいて利用されつつある．

しかし，Fingerprinting に用いる情報は時間経過に伴い変化することが示されており，収集した情報の単純な完全一致による長期的な端末識別は困難であることが示された[1]．これは，Fingerprinting の利用において適切な識別でないと，期間を空けてアクセスする端末を識別できないことを示している．

そこで，本論文では，ハッシュした文字列間を比較して類似度を算出することが可能なハッシュアルゴリズムである Fuzzy Hashing [2]を用いて，処理速度が高速でかつ長期的な端末識別を可能とする手法を提案する．また，我々が運営する Fingerprinting に用いる情報を収集する Web サイト[3]（以降，Fingerprint 収集サイトと呼ぶ）で得られたデータを用い

て，他の類似する端末識別手法と比較して，提案手法の識別精度が高いことを示す.

2 Browser Fingerprinting

2.1 Browser Fingerprint

Web ブラウザが Web サーバへアクセスした際に，JavaScript や Flash などを用いることにより Web サーバが Web ブラウザから採取できる端末の情報を特徴点と呼ぶ．特に，UserAgent やインストール済みプラグインなど端末の特定に繋がる情報を指す. 特徴点を 1 つ以上組み合わせたものを Browser Fingerprint と呼ぶ．本論文では，以降 Fingerprint と呼ぶ.

2.2 既存の Fingerprinting に関する研究

Eckersley[4]は Fingerprint を収集する Web サイトを開設し，Fingerprint の分析を行った. その結果, 収集したサンプルの 83.6%がユニークな Fingerprint をもち，Flash や Java が実行できる端末に限定した場合では，94.2%がユニークな Fingerprint を持つことを示した．また，同一端末の Fingerprint が時間経過に伴い変化することから，これを同一と推定するアルゴリズムを提案した.

Mowery[5]は HTML5 の Canvas API を用いて，文字列や画像を描画し，GPU ごとのレンダリングの差から，GPU，OS，Web ブラウザの組合せを特定できることを示した.

Nikiforakis[6]は実際に企業が行っている Fingerprinting について調査を行い，用いられている採取手法について明らかにした．また，実際に存在する UserAgent の偽装を行う拡張機能を調査し，UserAgent の偽装では，偽装したブラウザと異なる挙動を示すケースが存在し，これが特徴点となることを示した.

Boda[7]はフォントリストを JavaScript のみで採取する手法を提案し，Web ブラウザ間で共通のフォントリストなど一部の特徴点のみで端末を識別することによるクロスブラウザに対応した Fingerprinting を提案した.

Khademi[8]は，Fingerprinting を JavaScript Objects Fingerprinting, JavaScript-based Font Detection，Canvas Fingerprinting，Flash-based Fingerprinting の 4 つに分類し，それらを組み合わせ実装した Fybrid という JavaScript ライブラリを作成した. そのライブラリを用いて Fingerprint の収集，分析を行い，JavaScript から収集できるブラウザオブジェクトを組み合わせた JavaScript Objects Fingerprint は 1,523 のサンプルのうち 89.96%がユニークな値を持っていたことを示した.

著者らの先行研究[9]では，Fingerprint 収集サイトで収集している特徴点のなかで，最もエントロピーが高いプラグインリストに注目し，類似度を用いることで端末識別精度の向上を図った．結果としてプラグインリストのみで真陽性率 97.94%，真陰性率 97.95%の精度で端末を識別できることを示した.

3 関連知識

3.1 Fuzzy Hashing

Kornblum[2]は spamsum アルゴリズムを元に，Fuzzy Hashing（Context Triggered Piecewise Hashing：CTPH とも呼ばれる）と呼ばれるハッシュアルゴリズムを提案，実装した.

Fuzzy Hashing とは，部分文字列をずらしながらハッシュ化を行う Rolling Hashing と，部分文字列をハッシュ化して結合する Piecewise Hashing を組み合わせた Hash アルゴリズムである．図 1 に Fuzzy Hashing の概念を示す.

Fuzzy Hashing では，入力文字列の長さに応じてウインドウサイズ，トリガー値が決まる. Rolling Hashing でウインドウサイズの部分文字列をハッシュ化し，ハッシュ値がトリガー値と一致したとき, その位置を Piecewise Hashing を行う部分文字列の起点（トリガー）

とする．これを文字列の最後尾まで繰り返す．そして，トリガーから次のトリガーまでの文字列で Piecewise Hashing を行い，それらを結合した結果が Fuzzy Hashing によりハッシュ化された文字列となる．

2 つの文字列を Fuzzy Hashing でハッシュ化することで，ハッシュをそれぞれ得たのち，それらを比較する．このとき，文字列の一部が異なっていた場合に生成されるハッシュは一部のみが異なり，類似度を求めることができる．類似度は 0~100 の値をとり，値が大きいほど二つの文字列は類似している．図 2 に，本論文のあらましに記述した文字列に対し Fuzzy Hashing を適用した結果と，あらましに記述した文字列を，”の”を”を”として一部変更してハッシュ化した結果，及び，それらの 2 つの値の類似度を算出した結果を示す．

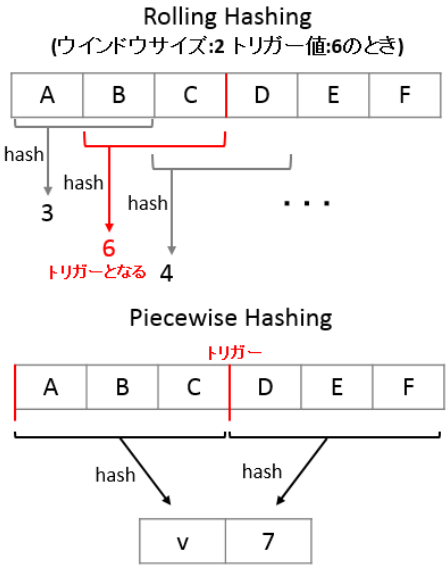

図 1 Fuzzy Hashing 概念図

本論文では，Kornblum によって実装された Fuzzy Hashing を行うソフトウエアである ssdeep[10]の Python ラッパーである ssdeep ライブラリを用いた．ssdeep では，文字列のハッシュ化とハッシュした 2 つの文字列の比較による類似度の算出を行うことができる．

あらましのハッシュ値

12:LY5qecOtggQ3/GCwggmG+vNqdLwrX9Ghair/c2nkE54kbSx6x0CmCfLsJAmiMn:85PToOGgWNHG8krkE5/nmeYiMn

あらましを一部変換した時のハッシュ値(“の”→”を”)

12:LY5qecOtggQ3/GLD2G+vNqZrX9Ghair/c2nN054kbSx6x0CmCjJAmi3:85PToOLyN8G8krC5/nmci3

上記 2 つを比較したときの類似度

82

図 2 Fuzzy Hashing のハッシュ値と比較の例

3.2 識別精度の評価尺度

本論文での識別精度の定義を説明する．同一端末を正しく同一とみなすことを真陽性(True Positive：TP)，同一の端末を異なるとみなすことを偽陽性（False Negative：FN）という．また，異なる端末を正しく異なるとみなすことを真陰性(True Negative：TN)，異なる端末を同一とみなすことを偽陰性(False Positive：FP)という．これらをまとめた表を以下に示す．

表 1　言葉の定義

		事実	
		同一	異なる
端末識別を行った結果	同一	TP	FP
	異なる	FN	TN

また，同一の端末を正しく識別できた割合を TP 率とし，(1)式とする．同様に，異なる端末を正しく異なると識別できた割合を TN 率とし，(2)式とする．

$$\text{TP 率} = \frac{TP}{TP + FP} \quad (1) \qquad \text{TN 率} = \frac{TN}{TN + FN} \quad (2)$$

本論文では，TP 率，TN 率が高いほど識別精度が高いとみなす．

なお，誤検出の割合である偽陽性率(FP 率)と偽陰性率(FN 率)はそれぞれ(3)(4)式とする．

$$\text{FP 率} = \frac{FP}{TP + FP} \quad (3) \qquad \text{FN 率} = \frac{FN}{TN + FN} \quad (4)$$

4 収集環境

本節では Fingerprint の収集環境と分析に用いるサンプル，特徴点について述べる．

4.1 Fingerprint 収集サイト

我々が運用する Fingerprint 収集サイトでは，端末に関する様々な情報を採取している[3]．また，同一端末からのアクセスであることを確認するために，Fingerprint の収集に伴い端末識別用の HTTP クッキー（以降，UID と呼ぶ）を生成し，Fingerprint 収集サイトのデータベースに保存している．同時に，アクセス時刻も保存している．

4.2 データセット

2013/12/6 から 2015/6/20 までの期間収集し，サンプル数は 3,674，ユニークなサンプル数(UID 数)は 1,513 となった．また，複数回アクセスしたユーザは 565，最長アクセス期間は 411 日であった．また，同一 UID 間の組み合わせ数は 14,032，異なる UID 間の組み合わせ数は 3,330,959 となった．

4.3 分析に用いる特徴点

本論文では，表 2 の特徴点を用いた Fingerprint について分析を行う．また，フォントリストは Flash を用いて採取しており，Flash Player 14(2014 年 6 月)からは結果がソートされて出力されるようになっているため，すべてのフォントリストをソートしている．

表 2　本論文で用いる特徴点

No.	特徴点
1	プラグインリスト
2	グローバル IP アドレス
3	JavaScript UserAgent
4	フォントリスト
5	HTTP UserAgent
6	プライベート IP アドレス
7	画面解像度と色深度
8	sse2 の有無
9	http accept language
10	http accept
11	http origin
12	http connection
13	http referer
14	device pixel ratio
15	http accept encoding
16	タッチ機能の有無
17	タイムゾーン
18	http accept charset
19	local storage の利用可否
20	session storage の利用可否

5 既存の Fingerprinting における端末識別手法

本章では，既存の Fingerprinting による端末識別手法を我々のサンプルに適用し，それらの TP 率と FP 率を表 5 に示す．なお，本論文での実験においては，すべての端末識別手法で FP が必ず存在する．これは，UID で同一端末と異なる端末を識別しているので，HTTP クッキーを削除して再度アクセスしたユーザが存在した場合，異なる端末が同一 Fingerprint を持つと認識されてしまうことが原因である．

5.1 Fybrid

Fybrid[11]はオープンソースの Browser Fingerprinting ライブラリである．このツールでは，JavaScript オブジェクト，Canvas Fingerprint，プラグインリスト，フォントリ

ストを Fingerprint として収集し，それらを MD5 でハッシュ化することで識別子を生成している．ハッシュの比較による識別を行うので，Fingerprint の比較は完全一致である．

識別精度としては，サンプル全体での TP 率は 40.3%と低く，単純なハッシュ化による端末識別は困難であると考えられる．

5.2 OpenAM

OpenAM[12]はオープンソースのシングルサインオンソフトウエアである．OpenAM 11.0 から Fingerprinting を用いたリスクベース認証の一種である「デバイスプリント認証」が導入されている．デバイスプリント認証では，それぞれの特徴点にペナルティポイントを設定することができる．認証時に Fingerprint を採取し，保存された Fingerprint と特徴点が異なっていた場合，ポイントを加算する．ポイントの累計が閾値を超えた場合，（ワンタイムパスワードによる）追加の認証が必要となる．用いられる特徴点とデフォルトのペナルティポイントを表 3 に示す．

表 3 OpenAM 11.0 におけるペナルティポイント

No.	特徴点	ペナルティポイント
1	プラグインリスト	35
3	UserAgent	35
4	フォントリスト	35
7	画面解像度	35
7	色深度	35
17	タイムゾーン	35
N/A	位置情報(許可有)	100
閾値		50

表 3 より，OpenAM 11.0 での端末識別では，（利用者に明示的な許可を求める）位置情報を除き，特徴点が 2 つ以上変化した場合に異なる端末であるとみなす．また，OpenAM 12.0 では，デバイスプリント認証の名称を Device ID 認証に変更し，デフォルトのペナルティポイントの閾値をより低い値である 30 に変更している．このため，特徴点が 1 つでも変化した場合に異なる端末と見なされるので，5.1 節の Fybrid でのハッシュ化による完全一致による比較手法と同一の結果となる．

表 5 より，サンプル全体での TP 率は 69.79%となった．また，14 日経過時点で 24.35%に低下しており，2 週間以上期間が空いた場合の端末識別は困難であると考えられる．

5.3 Eckersley による識別手法

Eckersley は同一端末の類似 Fingerprint を同一と推定する手法として，独自のアルゴリズムを提案した．

このアルゴリズムでは Fingerprint 間で特徴点ごとに比較を行う．特徴点が 1 つだけ異なった場合，その特徴点が HTTP クッキーの利用可否，画面解像度，タイムゾーン，及び，画面解像度のいずれかであれば同一端末とみなす．それ以外の特徴点の場合は，Python のスタンダードライブラリである difflib の SequenceMatcher 関数を用いてその文字列間の一致している比率を算出，その値が 0.85 を超えているときは同一端末とみなしている．

Eckersley は，このアルゴリズムによって同一端末と推定できる割合は 65%，FP 率は 0.56%と記述している．この手法では，Fingerprint に用いる特徴点として表 4 の特徴点を用いている．

HTTP クッキーの利用可否の特徴点は我々の収集したサンプルでは端末ごとの差が生じなかったので，それを除いている．また，スーパークッキーは Eckersley の分析時においても実装されていなかったので，これらを除いた 6 個の特徴点を用いて分析を行った．

その結果，表 5 より識別精度は TP 率 68.48%，TN 率 99.99%となり，Eckersley の

実験と類似した結果となった．しかし，長期的な端末識別は考慮されておらず，2 週間経過時点で TP 率 23.57%となり，長期的な識別は困難である．

表 4 Eckersley [1]の実験で用いる特徴点

No.	特徴点
1	プラグインリスト
3	UserAgent
4	フォントリスト
7	画面解像度
9,10,11,12,13,15,18	HTTP Headers
17	タイムゾーン
N/A	スーパークッキー
N/A	HTTP クッキーの利用可否

5.4 YIKS 類似度を用いた手法

我々の先行研究[9]では，プラグインリストをプラグインごとに分割し，それぞれを一文字とみなして類似度を求める YIKS 類似度を用いることで，プラグインリストのみでの端末識別を提案した．TP 率 97.94% ，FP 率 97.95%の精度で端末を識別できることを示した．

表 5 では，先行研究の結果よりも識別精度が低下している．これは，Fingerprint 収集サイトに複数回アクセスしたユーザが Fingerprinting の対策技術を試すことでプラグインリストが採取できなくなったケースも含まれていることが原因と考えられる．

この手法による識別精度は 2 週間経過時点においても TP 率 88.12%, TN 率 94.54%と高い状態を保てている．しかし，プラグインを使用していない端末では識別が不可能になる．また，処理速度について，それぞれのプラグインに対して距離を計算しているので，処理速度は低速である．

6 Fuzzy Hashing を用いた端末識別手法

本節では，我々の提案手法の識別精度と提案手法に関する考察を述べる．

本手法によって，端末識別精度はサンプル全体で TP 率 97.42%，TN 率 97.69%となり，2 週間経過時点においても TP 率 99.42%, TN 率 90.11%と既存の手法を上回る結果となった．

6.1 閾値の決定

Fuzzy Hashing はハッシュを比較した結果として類似度を求めるので，類似度がいくつを超えたら同一端末とみなすかを決定する必要がある．そのため，サンプルから類似度の分布を求め，閾値を決定する．

まず，サンプルの Fingerprint を ssdeep でハッシュ化する．次に同一 UID を持つハッシュ間と，異なる UID を持つハッシュ間で比較を行い，それぞれ類似度を算出する．算出された類似度の分布を図 3 に示す．類似度の分布を元に閾値を定義した場合の FP 率と FN 率を求め，FP 率+FN 率が最少となるものを閾値として定める．厳密に閾値を定めた場合，類似度 67 のとき FP 率+FN 率が最少の値 4.89%となった．

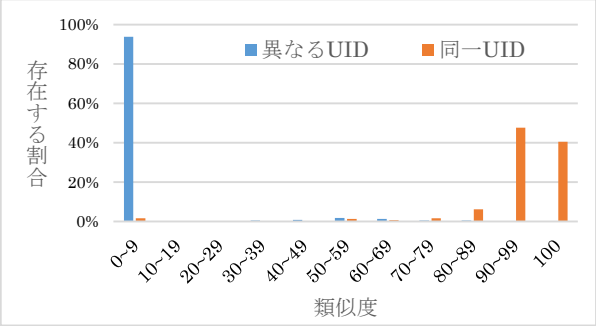

図 3 ssdeep による類似度の分布

6.2 識別精度

6.1 節で決定した閾値を用いて期間ごとの

TP 率と TN 率を求める．Fingerprint 収集サイトのサンプルで交差検証による比較を行い，同一 UID，異なる UID のそれぞれに対して，2 つのサンプル間の時刻差を算出し，一定期間ごとに分類を行った．それらの期間ごとのサンプル間で ssdeep による類似度の算出と閾値による TP，TN を求めた．その結果を表 5 に示す．

結果として，すべての期間において 80%以上の精度を保ち続け，既存の手法と比較して長期的な識別を行うことが可能となった．また，サンプル全体の識別精度についても既存の手法に比べて高い精度を示した．

6.3　本手法における考察

本手法における処理時間について記述する．システムのハードウエア環境として，CPU は AMD Phenom II X4 940 Processor，メモリは 4G バイトとした．ソフトウエア環境として，PHP で ssdeep のライブラリを用いた．あらかじめ我々の Fingerprint 収集サイトのデータベースに使用する Fingerprint のハッシュを生成・格納しておく．データベース上のすべてのハッシュで比較を行うための処理時間を計測した．その結果，3,674 個のハッシュ値との比較にかかる処理時間は 17ms 程度であった．これは実用に耐えうる処理時間であると考えられる．

本手法では，文字列に対して Fuzzy Hashing を行う汎用的なソフトウエアを用いている．汎用的に利用しやすい方法で文字列を分割し，分割した文字列のハッシュを比較しているので，結果の類似度は特徴点の文字列長にある程度依存する．そのことから，Fingerprinting に対し最適にはなっていないと考えられる．たとえば，タッチの有無やストレージの利用可否など 2 値の結果を出力する特徴点に関しては，文字列の長さは短くなる．本論文で用いた特徴点，サンプルでは高い識別精度となったが，今後短い文字列長で高い情報量（または，ほかの特徴点に依存しない情報）を持つ特徴点が出現した際には，その特徴点の類似度に及ぼす値は小さくなってしまう可能性がある．

ssdeep というツールの問題点として，文字列が短いと正確な識別ができないというものがある．しかし，Fingerprinting で多くの特徴点が採取できない場合，Fingerprint 文字列が短くなる可能性があるが，それは特徴点の選択自体が間違っているということが考えられるので，大きな問題にはならない．

本論文では閾値は FP 率+FN 率が最少となるように値をとったが使用用途によって閾値を変更することで，要件に合わせた利用が可能であると考えられる．例えば，リスクベース認証の場合 FP 率を低くなるように設定することで，正しい利用者を異なると判断する確率が上がり，ユーザビリティが低下する一方で，異なる端末を同一とみなす確率を低下させ，よりリスクの小さい認証にすることができる．

7　まとめ

本論文では，Browser Fingerprinting における，長期間にわたる端末識別手法を提案した．提案手法では，既存の手法に比べて高い識別精度を示し，サンプル全体での FP 率+FN 率は 4.89%となった．今後の課題として，識別精度の向上がある．Fuzzy Hashing のトリガー値などを Browser Fingerprinting により適した値に変更することで識別精度の向上が見込める．

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 論文集 p.378-p.385
[2] J. Kornblum, Identifying almost identical

files using context triggered piecewise hashing, in Digital Investigation, vol. 3S, pp. 91-97,2006.
[3] https://www.saitolab.org/fingerprint
[4] P. Eckersley, “How Unique is Your Web Browser?” In Proc. Privacy Enhancing Technologies Symposium (2010), LNCS vol.6205, 2010
[5] K Mowery, H Shacham, “Pixel Perfect: Fingerprinting Canvas in HTML5”, In Proc. of Web 2.0 Security and Privacy (W2SP), 2012.
[6] N Nikiforakis, A Kapravelos, W Joosen, C Kruegel, F Piessens, G Vigna, ”Cookieless Monster:Exploring the Ecosystem of Web-based Device Fingerprinting” , in Proc. of 34th IEEE Symposium of Security and Privacy (IEEE S&P 2013), 2013.
[7] K Boda, A Foldes, G Gulyas, S Imre, “User Tracking on the Web via Cross-Browser Fingerprinting” , in Proc. of 16th Nordic Conference on Information Security Technology for Applications, 2011.
[8] A.F Khademi, M Zulkernine, K Weldemariam, “An Empirical Evaluation of Web-based Fingerprinting”, IEEE Software pp. 46-52, 2015.
[9] 山田智隆，磯侑斗，桐生直輝，塚本耕司，高須航，武居直樹，齋藤孝道, “編集距離を用いたロバストな Browser Fingerprint 間の識別方式の提案”, 暗号と情報セキュリティシンポジウム 2015 ,2015
[10] http://ssdeep.sourceforge.net/
[11] Fybridjs https://github.com/ammosh/fybridjs
[12] OpenAM http://openam.forgerock.org

表 5 それぞれの識別手法における期間ごとの TP,TN （90%以上を灰色とする）

手法		全体の識別精度	期間									
			0, 1	1, 7	7, 14	14, 21	21, 28	28, 35	35, 42	42, 49	49, 56	56, 411
Fybrid	TP	40.29%	78.27%	18.69%	7.52%	3.34%	2.10%	0.00%	0.00%	0.89%	3.15%	0.00%
	TN	100%	99.98%	99.99%	100%	100%	100%	100%	100%	100%	100%	100%
OpenAM	TP	69.79%	94.89%	82.45%	77.17%	24.35%	9.25%	8.01%	6.63%	5.98%	9.84%	3.38%
	TN	99.99%	99.83%	99.91%	99.89%	99.93%	99.97%	99.99%	99.99%	99.99%	99.98%	100%
Eckersley	TP	68.48%	93.76%	78.88%	76.16%	23.57%	9.12%	7.52%	6.63%	5.98%	9.84%	3.38%
	TN	99.99%	99.91%	99.94%	99.93%	99.96%	99.97%	99.99%	99.99%	99.99%	99.98%	100%
YIKS 距離を用いた手法	TP	95.07%	98.63%	98.95%	98.14%	88.12%	93.46%	89.32%	82.14%	82.91%	82.79%	69.72%
	TN	97.37%	95.20%	95.43%	93.86%	95.45%	95.70%	95.77%	95.86%	95.38%	96.65%	98.27%
提案手法	TP	97.42%	98.99%	98.82%	99.10%	99.42%	98.40%	97.09%	100.00%	89.74%	95.08%	85.40%
	TN	97.69%	93.58%	93.93%	89.40%	90.11%	91.58%	91.74%	95.00%	96.76%	97.49%	98.35%
組み合わせ数	同一	14032	6440	2296	1774	1035	811	412	196	117	122	829
	異なる	3330959	115969	215681	146747	133417	111553	105333	84117	59988	62257	2295897