



HAL
open science

Leveraging browser fingerprinting for web authentication

Tompoariniaina Nampoina Andriamilanto

► **To cite this version:**

Tompoariniaina Nampoina Andriamilanto. Leveraging browser fingerprinting for web authentication. Systems and Control [cs.SY]. Université Rennes 1, 2020. English. NNT: 2020REN1S045 . tel-03150590

HAL Id: tel-03150590

<https://theses.hal.science/tel-03150590>

Submitted on 23 Feb 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
*Mathématiques et Sciences et Technologies
de l'Information et de la Communication*
Spécialité : *Informatique*

Par

Tompoariniaina Nampoina ANDRIAMILANTO

Leveraging Browser Fingerprinting for Web Authentication

Thèse présentée et soutenue à Rennes, le 17/12/2020
Unité de recherche : IRISA, IRT b<>com

Rapporteurs avant soutenance :

Arnaud LEGOUT Chargé de recherche (HDR), INRIA Sophia Antipolis
Luc BOUGANIM Directeur de recherche, INRIA Saclay Île de France

Composition du Jury :

Président :	Gildas AVOINE	Professeur, INSA Rennes
Rapporteurs :	Arnaud LEGOUT	Chargé de recherche (HDR), INRIA Sophia Antipolis
	Luc BOUGANIM	Directeur de recherche, INRIA Saclay Île de France
Examineurs :	Sonia BEN MOKHTAR	Directeur de recherche CNRS, laboratoire LIRIS Lyon
	Pierre LAPERDRIX	Chargé de recherche CNRS, laboratoire CRISTAL Lille
	Clémentine MAURICE	Chargée de recherche CNRS, laboratoire IRISA Rennes
Dir. de thèse :	David GROSS-AMBLARD	Professeur, Université de Rennes 1
Encadr. de thèse :	Tristan ALLARD	Maître de conférences, Université de Rennes 1

Invité(s) :

Dir. de thèse :	Benoît BAUDRY	Professeur, KTH Royal Institute of Technology
Co-enc. de thèse :	Gaëtan LE GUELVOUIT	Responsable de laboratoire, IRT b<>com

Acknowledgements

Many people have contributed in one way or another to my PhD studies. I take this section to thank them.

First and foremost, I would like to thank my thesis supervisors, Tristan Allard and Gaëtan Le Guelvouit, for the three enjoyable years spent learning and working alongside them. I would also like to thank my thesis directors, Benoît Baudry and David Gross-Amblard, for trusting me to carry out this thesis work.

Second, I would like to thank the members of my thesis jury : Gildas Avoine for chairing the jury ; Arnaud Legout and Luc Bouganim for reviewing the manuscript ; Clémentine Maurice, Pierre Laperdrix, and Sonia Ben Mokhtar for their participation in the jury ; all the members of the jury for their interest in my works.

I spent these three years of PhD studies between the Institute of Research and Technology b<>com and the IRISA laboratory, in both of which I was surrounded by warm people. I would like to thank the members of the Trust and Security team of the IRT b<>com for the good atmosphere that reigned. I enjoyed the discussions over a cup of coffee with Arnault Besnard and David Liouville, the lunchtime board game sessions with Valérie Denis, and Alexandre Garel for introducing me to the domain of machine learning. I would also like to thank the members of the DRUID team of the IRISA laboratory, notably Joris Duguéperoux, Louis Béziaud, Antonin Voyez, and Javier Rojas Balderrama for the interesting discussions during the lunchtime coffees and their reviews of our works.

Next, I would like to thank my friends for their support throughout my thesis studies. In particular, Malo and Christophe with whom I shared this three years adventure, and Evan for the warm exchanges over a sandwich, a tea, or a beer. Finally, I would like to thank my family, near or far away, for their encouragement and their confidence in my ability to succeed this thesis.

Résumé en français

Contexte

Depuis la genèse du *World Wide Web* à la fin des années quatre-vingt, les technologies qui l'entourent n'ont cessé de se développer. Il est aujourd'hui possible d'écouter de la musique, de jouer à des jeux-vidéos, de regarder des vidéos en direct ou en différé, et tout ceci au travers d'un simple navigateur web. En parallèle du développement des technologies web, les services qui s'y reposent se sont aussi développés. Il est désormais possible de réaliser des démarches administratives en ligne, mais aussi des transactions bancaires ou des visioconférences. Malgré le fait que les sites web soient accessibles publiquement, certains services ne devraient être accessibles que par des utilisateurs désignés. Par exemple, seul le propriétaire d'un compte bancaire devrait être autorisé à y effectuer des opérations en ligne. Se pose alors le problème de l'*authentification*, qui consiste à ce que le gestionnaire du site web, aussi appelé le *vérifieur*, vérifie qu'un utilisateur soit bien le détenteur d'un compte. Pour ce faire, le vérifieur demande des éléments que seul le détenteur du compte devrait être capable de fournir, éléments que nous appelons des *facteurs d'authentification*. Le facteur d'authentification couramment utilisé par les sites web est le mot de passe, grâce à sa facilité d'utilisation et de déploiement. Cependant, les mots de passe souffrent de multiples faiblesses. Par exemple, les utilisateurs ont tendance à utiliser les mêmes mots de passe, facilitant alors la tâche aux pirates qui réalisent des attaques par dictionnaires. Ces attaques consistent à déduire quels sont les mots de passe les plus courants, et à les présenter pour essayer d'accéder de manière frauduleuse à un compte. Cette déduction peut se faire à travers les mots de passe qui ont fuités ou été volés lors d'attaques informatiques.

À cause des faiblesses des mots de passe, les vérifieurs intègrent désormais de l'*authentification multi-facteur*. Cela consiste à utiliser plusieurs facteurs d'authentification, de sorte que chaque facteur ajoute une barrière de sécurité supplémentaire. Cependant, le gain en sécurité des facteurs additionnels vient au prix d'une perte de facilité d'utilisation ou de déploiement. Les utilisateurs doivent se souvenir d'une information supplémentaire, avoir un objet sur eux, ou effectuer une action. Les vérifieurs doivent déployer des logiciels ou du matériel supplémentaires, les maintenir, et apprendre aux utilisateurs à s'en servir. Selon des estimations réalisées en 2015 et en 2018, moins de dix pour cent des utilisateurs s'authentifient auprès des sites web à l'aide de plusieurs facteurs d'authentification.

Le développement des technologies web a aussi mené à une diversité des configurations de navigateurs web et à une grande quantité d'information communiquée par ceux-ci. Effectivement, les utilisateurs ont aujourd'hui le choix entre différents systèmes d'exploitation et modèles de navigateur. Ces logiciels étant mis à jour régulièrement, ils existent alors sous différentes versions. Les navigateurs fournissent un accès à des informations concernant l'appareil, le système d'exploitation, le modèle de navigateur, et leurs versions respectives. En 2010, Peter Eckersley a lancé une expérimentation via le site Panopticlick afin de vérifier si, à partir des informations communiquées par les navigateurs, ceux-ci ne seraient pas reconnaissables. Parmi les 470,161 navigateurs qui font partie de l'expérimentation, 83.6% étaient reconnaissables de manière unique à partir des 8 attributs collectés. La technique utilisée est alors appelée *browser fingerprinting*, ou prise d'empreinte de navigateur en français, et consiste à collecter des attributs auprès d'un navigateur afin de constituer une empreinte de celui-ci. Les empreintes de navigateur ont été utilisées à des fins de pistage web. En suivant un utilisateur au fil de sa navigation via l'empreinte de son navigateur, les agences publicitaires peuvent lui proposer des publicités ciblées. Mais cette empreinte peut aussi servir de facteur d'authentification supplémentaire à bas coût : l'utilisateur n'a rien à retenir, rien à installer, et aucune action supplémentaire à effectuer. Il suffit de vérifier que l'empreinte de son navigateur soit suffisamment ressemblante à l'empreinte enregistrée pour le compte demandé, ce qui peut être fait de manière transparente.

Adéquation des Empreintes de Navigateur pour l'Authentification Web

La plupart des études sur l'utilisation des empreintes de navigateur à des fins d'authentification se concentrent sur la conception du mécanisme d'authentification. De plus, les études empiriques sur les empreintes de navigateur se concentrent sur son utilisation pour du pistage web, ignorant alors des propriétés primordiales pour l'authentification comme leur efficacité à reconnaître un navigateur. Enfin, la capacité des empreintes à distinguer les navigateurs sur une grande population et en utilisant un grand nombre d'attributs est, à notre connaissance, encore méconnue. Les études empiriques portent soit sur une population large mais en utilisant moins de trente attributs, sous-estimant alors la distinguabilité des empreintes, soit sur une faible population en utilisant une centaine d'attributs.

Au travers de cette thèse, nous proposons la première étude empirique à large-échelle des propriétés des empreintes de navigateur utilisées à des fins d'authentification web. Nous nous reposons sur l'analyse de quatre jeux de données, dont un constitué de 4,145,408 empreintes composées de 216 attributs initiaux et de 46 attributs déduits. Nous faisons le lien entre les empreintes digitales distinguant les êtres humains et celles des navigateurs distinguant ces derniers, et évaluons les empreintes de navigateur selon des propriétés de facteurs d'authentification biométriques. Celles-ci comprennent la distinguabilité des empreintes, leur stabilité, leur temps de collecte, leur taille en mémoire, l'efficacité d'un mécanisme de vérification d'empreintes, la perte d'efficacité au fil du temps ou en fonction de populations de navigateurs, et l'acceptabilité de la part des utilisateurs. Afin de comprendre les résultats obtenus au sujet des empreintes, nous discutons aussi de l'apport des attributs à chaque propriété, des corrélations observées entre attributs, et présentons en annexe la liste exhaustive des attributs utilisés. En considérant nos attributs et populations de navigateurs, nous atteignons un haut niveau de distinguabilité et de stabilité. Ce niveau de distinguabilité et de stabilité permet d'obtenir une haute précision en simulant un mécanisme simple de comparaison d'empreintes : nous obtenons un taux d'erreur égal compris entre 0.61% et 4.30%. Un jeu de données provenant de machines standardisées d'une université sort du lot et présente une forte baisse de la distinguabilité, de la stabilité, et de la précision.

Méthode de Sélection d'Attributs

Des centaines d'attributs sont à disposition des vérificateurs pour concevoir leur sonde de prise d'empreinte de navigateur. L'ajout d'un attribut peut améliorer la reconnaissance des navigateurs en apportant une information supplémentaire pouvant distinguer deux navigateurs différents. Cependant, l'ajout d'un attribut a aussi un coût d'utilisation : les empreintes requièrent un espace de stockage plus grand, un temps de collecte plus long, et peuvent être plus difficiles à reconnaître si l'attribut en question est instable. Dû aux corrélations survenant entre les attributs, les sélectionner un par un serait peu efficace. De plus, l'apport d'un attribut dépend de ceux déjà sélectionnés. Par exemple, certains attributs sont collectés en parallèle, le temps requis pour les collecter n'est donc pas la somme de leur temps de collecte individuel mais le temps de collecte du plus lent. Explorer toutes les possibilités est aussi à proscrire puisque le nombre de possibilités croît de manière exponentielle en fonction du nombre d'attributs.

Au travers de cette thèse, nous proposons une méthode de sélection d'attributs tels qu'ils satisfassent un niveau de sécurité et réduisent les contraintes d'utilisation. Le niveau de sécurité est mesuré selon la proportion d'utilisateurs usurpés étant donnés les attributs utilisés, une population de navigateurs, et un attaquant modélisé. Nous considérons l'attaquant qui connaît la distribution exacte des empreintes des utilisateurs à protéger et qui est capable de présenter un nombre limité d'empreintes contrefaites. Les contraintes sur l'utilisation sont mesurées selon le temps de collecte des empreintes, leur taille, et leur instabilité. Nous comparons notre méthode avec les méthodes courantes de sélection d'attributs reposant sur l'entropie et l'entropie conditionnelle. Nous exécutons les méthodes sur des échantillons de deux jeux de données, avec plusieurs niveaux de sécurité souhaités, et plusieurs attaquants en fonction du nombre d'empreintes qu'ils puissent présenter. Comparé à l'utilisation de tous nos attributs et en moyenne, les attributs sélectionnés par notre méthode génèrent des empreintes qui sont de 12 à 1663 fois plus petites, de 9 à 32330 fois plus rapide à collecter, et avec 4 à 30 fois moins d'attributs changeants entre deux observations. Notre objectif de réduction du coût d'utilisation des attributs est atteint en utilisant notre méthode. Nous obtenons un coût jusqu'à trois ordres de grandeur plus bas comparé aux méthodes usuelles.

Abstract

Web authentication is the verification that a user claiming an account legitimately owns this account. It widely relies on passwords, but several authentication factors were proposed such that each factor provides an additional security barrier. Browser fingerprinting notably came out as a promising candidate. It is the collection of attributes from a web browser to build its fingerprint which is potentially unique. In this thesis, we provide two contributions to the field of browser fingerprinting for web authentication :

1. We investigate the adequacy of browser fingerprints for web authentication. We make the link between the browser fingerprints that distinguish browsers, and the biometric fingerprints that distinguish Humans, to evaluate browser fingerprints according to properties inspired by biometric authentication factors. We assess these properties on four real-life browser fingerprint datasets, which include one of nearly two million browsers. To comprehend the properties of the fingerprints, we enrich our results with the contribution of the attributes to the properties and the correlations between them.
2. We propose FPSelect, an attribute selection framework to find the attribute set that satisfies a security requirement and reduces the usability cost. The security is measured as the proportion of impersonated users given a fingerprinting probe, a user population, and a modeled attacker. The usability is quantified by the collection time of the browser fingerprints, their size, and their instability. We compare our framework with common baselines, based on two real-life fingerprint datasets, and find out that it selects attribute sets of lower usability cost in our experimental settings.

Contents

Acknowledgements	iii
Résumé en français	v
Abstract	ix
Contents	xi
List of Figures	xvii
List of Tables	xxiii
List of Algorithms	xxvii
1 Introduction	1
1.1 Motivations	3
1.1.1 Adequacy of Browser Fingerprints for Authentication	3
1.1.2 Selection of Browser Fingerprinting Attributes	4
1.2 Contributions	5
1.2.1 Assessing the Adequacy of Browser Fingerprints for Web Authentication	5
1.2.2 Attribute Selection according to a Security and Usability Trade-off	6
1.3 Scientific Publications	7
1.4 Outline of the Thesis	8

2	Background and Context	11
2.1	Web Authentication	12
2.1.1	Definitions	13
2.1.2	Wide Usage and Flaws of Passwords	16
2.1.3	Multi-Factor Authentication	20
2.1.4	Evaluation of Authentication Schemes	24
2.1.5	Evaluation of Biometric Authentication Schemes	24
2.1.6	Robustness of Authentication Schemes against Attacks	27
2.2	Browser Fingerprinting	29
2.2.1	Discovery of Browser Fingerprinting	30
2.2.2	Browser Fingerprinting Attributes	32
2.2.3	Usages of Browser Fingerprints	38
2.3	Authentication by Browser Fingerprinting	40
2.3.1	Analysis of Browser Fingerprints for Identification	41
2.3.2	Browser Fingerprinting into Authentication Mechanisms	42
2.3.3	Adequacy of Browser Fingerprints for Authentication	44
2.3.4	Selection of Attributes	46
2.3.5	Challenge-response Mechanisms	48
2.4	Conclusion	49
3	Experiments and Survey	53
3.1	Browser Fingerprinting Probe	54
3.1.1	JavaScript Attributes	54
3.1.2	Dynamic Attributes	55
3.1.3	HTTP Headers	57
3.1.4	Error and Inaccessibility Handling	57
3.2	Browser Fingerprinting Experiments	57
3.2.1	General Audience Experiment	60
3.2.2	Intranet Experiment	62
3.2.3	University Experiment	63
3.2.4	Enrolled Experiment	65
3.2.5	Comparison with Previous Large-Scale Studies	66
3.3	Browser Fingerprints Preprocessing	67

3.3.1	Dataset Cleaning	68
3.3.2	Unique IDs Resynchronization	68
3.3.3	Deduplication	68
3.3.4	Extracted attributes	69
3.4	Acceptability Survey	70
3.4.1	Respondents Population	70
3.4.2	Respondents Authentication Habits	71
3.5	Conclusion	79
4	Browser Fingerprints for Authentication	81
4.1	Authentication Factor Properties	82
4.1.1	Studied Properties	83
4.1.2	Distinctiveness	84
4.1.3	Stability	86
4.1.4	Performance	87
4.1.5	Acceptability	89
4.2	Evaluation of Browser Fingerprints Properties	89
4.2.1	Distinctiveness	89
4.2.2	Stability	95
4.2.3	Collection Time	99
4.2.4	Fingerprint Size	102
4.2.5	Accuracy of the Simple Verification Mechanism	104
4.2.6	Acceptability	111
4.2.7	Conclusion	114
4.3	Attribute-wise Analysis	116
4.3.1	Attributes Distinct Values	117
4.3.2	Attributes Distinctiveness	120
4.3.3	Attributes Sameness Rate	126
4.3.4	Attributes Collection Time	131
4.3.5	Attributes Size	138
4.3.6	Correlation between the Attributes	142
4.3.7	Focus on the Dynamic Attributes	149
4.4	Conclusion	152

5	Attribute Selection Framework	157
5.1	Problem Statement	160
5.1.1	Authentication Mechanism	160
5.1.2	Attack Model	161
5.1.3	Attribute Selection Problem	163
5.2	Attribute Selection Framework	165
5.2.1	Similarity to the Knapsack Problem	165
5.2.2	Lattice Model and Resolution Algorithm	167
5.2.3	Illustrative Sensitivity and Usability Cost Measures	171
5.3	Experimental Validation	175
5.3.1	Instantiation of the Experiments	175
5.3.2	Attribute Selection Framework Results	178
5.4	Conclusion	190
6	Conclusion	193
6.1	Contributions	195
6.1.1	Assessing the Adequacy of Browser Fingerprints for Web Authentication	195
6.1.2	Attribute Selection according to a Security and Usability Trade-off	197
6.2	Future Works	198
6.2.1	Extension of FPSelect to Attackers having Targeted Knowl- edge	198
6.2.2	Challenge-Response Mechanisms and Dynamic Attributes	199
6.2.3	Web Environment Components Impacting Attributes	199
6.2.4	Matching Function based on Fingerprints Evolution	200
6.3	Perspectives on Browser Fingerprinting for Authentication	201
A	Browser Fingerprinting Attributes	203
A.1	JavaScript Properties	203
A.2	HTTP Headers	204
A.3	Enumeration or Presence of Browser Components	206
A.3.1	List Attributes	206

A.3.2	Support of Codecs	206
A.3.3	List of Video Codecs	206
A.3.4	List of Audio Codecs	207
A.3.5	List of Fonts	207
A.4	Extension Detection	207
A.4.1	Detection of Ad Blockers	208
A.5	Size and Color of Web Page Elements	208
A.5.1	Bounding Boxes	208
A.5.2	Width and Position of a Created <code>div</code> Element	209
A.5.3	Colors of Layout Components	209
A.6	WebGL Properties	210
A.7	WebRTC Fingerprinting	210
A.8	HTML5 Canvases	211
A.9	Audio Fingerprinting	211
A.9.1	Simple Process	212
A.9.2	Advanced Process	213
B	Updates of the Fingerprinting Probe	215
B.1	For the Intranet Experiment	215
B.1.1	Attributes Removed	216
B.1.2	Attributes Added	216
B.1.3	Instability Fix of our Custom HTML5 canvas	217
B.2	For the Enrolled Experiment	217
B.2.1	Attributes Removed	217
B.2.2	Attributes Added	218
B.2.3	Instability Fix of our Custom HTML5 canvas	218
B.3	For the University Experiment	219
C	Browser Classification Keywords	221
C.1	Robot Keywords	221
C.2	Device Type	221
C.3	Browser and Operating System Families	222

D	Advanced Verification Mechanism	225
D.1	Attributes Matching	225
D.2	Comparison to the Simple Verification Mechanism	228
D.3	Accuracy for the General Audience Dataset	229
D.4	Accuracy for the Intranet Dataset	229
D.5	Accuracy for the University Dataset	230
D.6	Accuracy for the Enrolled Dataset	232
E	Acceptability Survey	233
E.1	Questions and Possible Answers	233
E.2	Description of the Test Authentication Mechanism	237
F	Test Authentication Mechanism	239
F.1	Enrollment	239
F.2	Authentication	240
F.3	Account Recovery and Multi-browser Support	240
F.3.1	Password Recovery	240
F.3.2	Browser Fingerprint Recovery and Multi-browser	241
G	Demonstrations of Measures Monotony	243
G.1	Monotony of the Illustrative Sensitivity	243
G.2	Monotony of the Illustrative Usability Cost	244
H	Explanations to the Experimenters	247
H.1	Intranet Experiment	247
H.2	University Experiment	249
H.3	Enrolled Experiment	251
H.3.1	Informational page	251
H.3.2	Terms and conditions of use	253
I	Attributes List and Properties	265
	Bibliography	281
	List of Author's Publications	307

List of Figures

2.1	Web authentication process between the verifier’s web platform and a user.	15
2.2	Data breaches that occurred over the last three years that were recorded by informationisbeautiful.net at the date of the 1st April 2020. The warmer the color is, the more sensitive is the data. . . .	19
2.3	Example of an HTML5 canvas that displays overlapping texts, rectangles, emojis, curves, ellipses, and a shading.	34
2.4	Example of a two-dimensional WebGL canvas that displays a succession of connected triangles.	35
3.1	The gender reported by the respondents.	71
3.2	The age reported by the respondents.	71
3.3	The number of people in the household reported by the respondents.	72
3.4	The level of technical skill reported by the respondents.	72
3.5	The socio-professional categories of the respondents.	72
3.6	The type of the devices that the respondents use. Mobile phone does not include the smartphones.	73
3.7	The authentication methods that the respondents know and use. Fingerprint here refers to the biometric fingerprints.	73
3.8	The satisfaction of the respondents with the current authentication methods. The scores are divided between the detractors, the passives, and the promoters according to the Net Promoter Score methodology.	75

3.9	The reasons reported by the 261 unsatisfied respondents (20% of the respondents) about why they are dissatisfied with the current authentication methods.	76
3.10	The difficulties regarding the current authentication methods that the 489 respondents (38% of the respondents) have already encountered.	77
3.11	The ways the respondents retrieve their passwords.	78
4.1	Anonymity set sizes and frequency of browser arrivals through the time-partitioned datasets obtained for each of the four datasets. The new browsers of the general audience dataset are displayed in hundreds of thousands.	90
4.2	Unicity rate for the overall, the mobile, and the desktop browsers, through the time-partitioned datasets obtained from the general audience and enrolled datasets.	90
4.3	Average similarity between the pairs of consecutive fingerprints as a function of the time difference, together with the number of compared pairs.	96
4.4	Cumulative distribution of the collection time of the fingerprints in seconds.	100
4.5	Cumulative distribution of the fingerprint size in bytes.	103
4.6	The number of identical attributes between the same-browser comparisons and the different-browsers comparisons. The figures on the left start from the lowest observed value, and those on the right start from the value for which 0.5% of the same-browser comparisons are below (20% for the university dataset).	106
4.7	False match rate (FMR) and false non-match rate (FNMR) given the required number of identical attributes, averaged among the month samples.	107
4.8	Interest of the respondents in the authentication mechanism according to its description only.	111

4.9	The advantages of the authentication mechanism that are perceived by the 1,035 respondents that are interested, according to its description.	113
4.10	The weaknesses of the authentication mechanism that are perceived by the 250 respondents that are not interested, according to its description.	113
4.11	The satisfaction of the respondents with their experience on the test authentication mechanism. The scores are divided between the detractors, the passives, and the promoters according to the Net Promoter Score methodology.	114
4.12	Cumulative distribution of the number of distinct values of the attributes in logarithmic scale.	117
4.13	Cumulative distribution of the normalized entropy, and of the entropy in bits, among the attributes. The dashed gray line is the entropy of the most entropic attribute.	120
4.14	Cumulative distribution of the sameness rate of the attributes among the consecutive fingerprints.	127
4.15	Median collection time of the attributes that have a median collection time higher than 5ms, ranked from the slowest to the fastest to collect for the overall browsers.	132
4.16	Median size of the attributes that weigh more than 100 bytes, ranked from the heaviest to the lightest.	138
4.17	Minimum, average, and maximum normalized conditional entropy of an attribute when the value of another attribute is known, ordered by the average normalized conditional entropy.	143
5.1	Example of a web authentication mechanism based on browser fingerprinting and a failed attack.	158

5.2	Example of an attacker instantiated with his knowledge of a probability mass function (PMF) over the fingerprints F , and a web platform protecting a user population U with their fingerprint. We consider a limit of two submissions and a strict comparison between the fingerprints. The attack dictionary is composed of f_1 and f_2 , resulting in the shown impersonated users.	161
5.3	Example of a lattice of attribute sets, with their cost c , their sensitivity s , and their efficiency e . The blue node satisfies the sensitivity, the white nodes do not, and the green node with a diamond satisfies the sensitivity and minimizes the cost. The red line is the satisfiability frontier.	167
5.4	Cost of the attribute sets found by the ASF with 1 explored path (ASF-1), the ASF with 3 explored paths (ASF-3), the entropy, and the conditional entropy. The gray horizontal line is the cost when considering all the candidate attributes.	180
5.5	Number of explored attribute sets by the attribute selection methods.	183
5.6	The proportion of impersonated users when considering the candidate attributes, and as a function of the number of submissions. The sensitivity thresholds α are displayed.	184
A.1	HTML5 canvas inspired by the AmIUnique [127] study in PNG format.	211
A.2	HTML5 canvas similar to the Morellian [124] study in PNG format.	211
A.3	Architecture of the network of <code>AudioNode</code> objects for the simple audio fingerprinting method.	212
A.4	Architecture of the network of <code>AudioNode</code> objects for the simple audio fingerprinting method.	214

D.1 The number of matching attributes between the same-browser comparisons and the different-browsers comparisons for the four datasets. The figure on the left starts from the lowest observed value, and the figure on the right starts from the value for which 0.5% of the same-browser comparisons are below (20% for the university dataset). 226

D.2 False match rate (FMR) and false non-match rate (FNMR) given the required number of matching attributes, averaged among the month samples for the four datasets. 227

H.1 Consent banner of the intranet experiment 247

List of Tables

2.1	Example of HTTP headers sent during a request	33
2.2	Usages of browser fingerprinting, classified between the usages to recognize a browser in the crowd, and the ones to recognize the type of browser.	38
3.1	Comparison between the browser fingerprint datasets obtained from the four experiments, and the large-scale datasets of previous studies. Previous studies include the studies Panopticlick (PTC) [56], AmIUnique (AIU) [127], Hiding in the Crowd (HitC) [81], and Long-Term Observation (LTO) [178]. The returning browsers are the browsers from which multiple fingerprints have been collected. The unicity is the proportion of the fingerprints that were observed for a single browser only. The attributes comprise the derived attributes. The number of fingerprints and of distinct fingerprints are obtained after dataset preprocessing. As the LTO study tests various attribute sets, we present the ranges of the results obtained by their attribute selection method. - denotes missing information, and * denotes deduced information. The proportion of mobile and desktop fingerprints do not sum to 1.0 as other device types were observed (e.g., tablets, TVs, video game consoles).	58
3.2	The share of browser and operating system families among the browser populations of the four experiments. The categories are ordered from the most common to the least for the general audience dataset.	59

3.3	The number of initial raw entries, together with the number of entries having a bad format, coming from robots, having the cookies disabled, or having a duplicated fingerprint, which are filtered out. The proportion of browsers observed with an interleaved fingerprint is also shown.	70
4.1	Comparison of the normalized entropy between the studies Panoptlick (PTC) [56], AmIUnique (AIU) [127], Hiding in the Crowd (HitC) [81], and our four datasets. The attributes are ranked from the most to the least distinctive for the general audience dataset. We provide the normalized entropy equivalent to 1 bit of entropy for comparison.	121
4.2	The minimum normalized conditional entropy (MNCE) of the attributes that have a MNCE higher than the equivalent of a conditional entropy of 1 bit, for the general audience, intranet, and enrolled datasets. <i>W</i> refers to the <code>window</code> object, and <i>N</i> to the <code>navigator</code> object. - denotes an attribute that is not included in, or that do not have a MNCE sufficiently high for, a given dataset. . . .	144
4.3	The minimum normalized conditional entropy and the minimum conditional entropy in bits of the attributes that have a non-null MNCE for the university dataset. <i>A</i> refers to an initialized audio context, <i>N</i> to the <code>navigator</code> object, <i>WG</i> to an initialized WebGL context, and <i>S</i> to the <code>screen</code> object.	148
4.4	The dynamic attributes included in the four datasets. - denotes a non-included attribute. * indicates that the attribute is generated using randomized instructions, hence is always unique.	149
5.1	Example of fingerprints shared by users.	164
5.2	Example of the execution of Algorithm 1 on the lattice of Figure 5.3, with the sensitivity threshold $\alpha = 0.15$ and the number of explored paths $k = 2$. Stage i is the state at the end of the i -th <code>while</code> loop.	171
5.3	The cost of the candidate attributes, together with the maximum, the average, and the minimum cost of a single attribute for each cost dimension.	177

5.4	The attributes selected by the attribute selection framework for each experimentation setup of the general audience dataset. We denote k the number of explored paths, β the number of fingerprint submissions, and α the sensitivity threshold.	186
5.5	The attributes selected by the attribute selection framework for each experimentation setup of the enrolled dataset. We denote k the number of explored paths, β the number of fingerprint submissions, and α the sensitivity threshold.	187
A.1	The HTTP headers that are stored into a dedicated attribute, and the headers that are ignored, for each experiment.	205
A.2	Extensions detected by the changes they bring to the page content.	208
A.3	Extensions detected by the availability of their web accessible resource. C stands for chrome, and R stands for resource.	209
A.4	Properties of the <code>div</code> element measured for the attributes related to the bounding boxes.	210
C.1	The keywords and the exact <code>userAgent</code> values that we consider as indicating a robot browser. The long values are cut at a blank space and displayed with indentations.	222
C.2	The keywords that we consider as indicating each device type.	223
C.3	The keywords that we consider as indicating each browser family.	223
C.4	The keywords that we consider as indicating each operating system family.	223
D.1	Comparison between the simple verification mechanism that uses identical attributes and the advanced verification mechanism that uses matching attributes for the general audience dataset.	230
D.2	Comparison between the simple verification mechanism that uses identical attributes and the advanced verification mechanism that uses matching attributes for the intranet dataset.	231
D.3	Comparison between the simple verification mechanism that uses identical attributes and the advanced verification mechanism that uses matching attributes for the university dataset.	231

D.4	Comparison between the simple verification mechanism that uses identical attributes and the advanced verification mechanism that uses matching attributes for the enrolled dataset.	232
I.1	The attributes of the general audience dataset, together with their distinct values, their normalized entropy, their minimum normalized conditional entropy (MNCE), their stability, their median size, and their median collection time. The attributes from which we derive the extracted ones are ignored for the MNCE.	267
I.2	The additional attributes of the intranet dataset, together with their distinct values, their normalized entropy, their minimum normalized conditional entropy (MNCE), their stability, their median size, and their median collection time.	278
I.3	The additional attributes of the enrolled dataset, together with their distinct values, their normalized entropy, their minimum normalized conditional entropy (MNCE), their stability, their median size, and their median collection time.	279

List of Algorithms

- 1 Greedy algorithm to find good solutions to the Attribute Selection Problem. 169
- 2 Illustrative sensitivity measure. 173

Chapter 1

Introduction

Since the genesis of the World Wide Web – or web for short – at the end of the eighties, the technologies surrounding the web never stopped developing. It is now possible to hold video conferences [222], play video games¹, or manipulate three-dimensional scenes [221] using a common web browser. As the functionalities of the web technologies evolve, a large variety of services become accessible on the web. Examples are the social networks that help people connect each other, the government websites that help citizens perform administrative procedures, or the bank websites that help their clients execute financial transactions. Although websites are usually publicly accessible, some services should be only accessible by designated users. For example, the owner of a bank account should be the only user that is authorized to perform transactions on her account. Therefore, website owners have to verify that a user legitimately owns the account for which he wants to perform an operation before authorizing him to do so. The problem that the website owner has to solve is the authentication. It consists for the website owner, called the verifier, to check that the user owns the claimed account.

To authenticate a user, the verifier asks him to provide pieces of evidence – called authentication factors – that only the account owner should be able to present. The most used authentication factor on the web is currently the password thanks to its ease of use and deployment [28]. However, passwords have been shown to suffer from severe security flaws when used without any additional au-

¹<http://xproger.info/projects/OpenLara>

thentication factor. Real-life users indeed use common passwords [94], which paves the way to brute-force or guessing attacks [28]. Moreover, they tend to use similar passwords across different websites [44], which increases the impact of successful attacks. Phishing attacks are also a major threat to the use of passwords. Over the course of a year, Thomas et al. [210] achieved to retrieve 12.4 million credentials stolen by phishing kits. These flaws bring the need for supplementary security layers, primarily through multi-factor authentication [30] such that each additional factor provides an additional security barrier. However, this usually comes at the cost of usability (i.e., users have to remember, possess, or do something) and deployability (i.e., verifiers have to deploy dedicated hardware or software, teach users how to use them, and maintain the deployed solution). According to previous estimations [173, 149], the usage rate of multi-factor authentication is below 10%.

In the meantime, browser fingerprinting gains more and more attention. It is the collection of attributes from a web browser to build its fingerprint, which is potentially unique. In his thesis, Mayer [145] is the first to ask if "with all the customizations now available [...] are any two web browsing environments identical?" By collecting four attributes from 1,328 web browsers, he observes that 96.23% of them show a unique combination of these four attributes. Shortly after, Eckersley [56] published the results from the Panopticlick website [72] which invited users to provide their browser fingerprint. Among the 470,161 collected fingerprints composed of 8 attributes, 83.6% were observed for a single browser. This result showed the effectiveness of browser fingerprints to recognize browsers, and lead the way of the research on browser fingerprinting. Since the discovery of the technique, a plethora of attributes have been reported. They started from the information provided by common JavaScript objects and HTTP headers [145, 56, 127], to images drawn by the browser [153, 36], to the installed fonts inferred by the size of text boxes [65], to the technical information of a real-time communication [66], to the installed browser extensions [205, 199], and to audio signals manipulated in the browser [179]. The first studies on browser fingerprinting focus on the threat posed by browser fingerprinting to the privacy of web users, whether it is about its effectiveness [56, 25, 228, 2, 96, 127], its use on the web [3, 57, 64], or counter-measures to mitigate this threat [68, 61, 128, 211, 125]. In addition

to being widely used for web tracking purposes (raising legal, ethical, and technical issues), browser fingerprinting is also used for web authentication. Verifiers typically include browser fingerprints as a supplementary factor that is verified at login. Browser fingerprints are indeed a promising candidate as an additional web authentication factor thanks to their distinctive power, their frictionless deployment (e.g., no additional software or hardware to install), and their usability (no secret to remember, no additional object to possess, and no supplementary action to carry out). As a result, companies like MicroFocus [70] or SecureAuth [196] include this technique into their authentication mechanisms.

1.1 Motivations

1.1.1 Adequacy of Browser Fingerprints for Authentication

To the best of our knowledge, no large-scale study rigorously evaluates the adequacy of browser fingerprints as an additional web authentication factor. On the one hand, most works about the use of browser fingerprints for authentication concentrate on the design of the authentication mechanism [212, 177, 79, 203, 124, 184]. On the other hand, the large-scale empirical studies on browser fingerprints focus on their effectiveness as a web tracking tool [56, 127, 81, 178]. Such a mismatch between the understanding of browser fingerprints for authentication – currently poor – and their ongoing adoption in real-life is a serious harm to the security of web users. The lack of documentation from the existing authentication tools (e.g., about the used attributes, about the distinctiveness and the stability of the resulting fingerprints) only adds up to the current state of ignorance, all this whereas security-by-obscurity directly contradicts the most fundamental security principles. Moreover, the distinctiveness of browser fingerprints that can be achieved when considering a wide-surface of fingerprinting attributes on a large population is, to the best of our knowledge, unknown. On the one hand, the studies that analyze browser fingerprints in a large-scale (more than 100,000 fingerprints) consider fewer than thirty attributes [56, 127, 214, 81]. This underestimates the distinctiveness of the fingerprints (e.g., [81] reports a rate of

33.6% of unique fingerprints) as it increases the chances for browsers to share the same fingerprint. All this whereas more than a hundred attributes are accessible. On the other hand, the studies that consider more than fifty attributes either work on less than two thousand users [111, 178], or do not analyze the resulting fingerprints at all [64]. The current knowledge about the hundreds of accessible attributes (e.g., their stability, their collection time, their correlation) is, to the best of our knowledge, also incomplete. Indeed, previous studies consider few attributes [56, 153, 65, 36, 127, 199, 81, 179] or focus on a single aspect of them (e.g., their stability [214]). Furthermore, to the best of our knowledge, only Blake et al. [24] studied two browser populations, but not according to the same properties as the second population only gets the stability of their fingerprints studied. The properties of the browser fingerprints that are collected during an experiment vary according to the attributes that are used, but also according to the studied browser population. To the best of our knowledge, no study compares the properties of browser fingerprints collected from various browser populations.

1.1.2 Selection of Browser Fingerprinting Attributes

The verifiers are given a large choice of attributes² to build their browser fingerprinting probe, and face the problem of which attributes to use. On the one hand, the addition of an attribute to the probe can strengthen the distinctiveness of browsers, which helps to distinguish a legitimate user from an impostor. On the other hand, each addition comes with a usability cost that may render the probe impractical in an online authentication context. Indeed, each attribute consumes storage space (up to hundreds of kilobytes [36]), collection time (up to several minutes [152, 155, 157, 191, 193, 179]), and can increase the instability of the generated fingerprints [214]. Moreover, some attributes are strongly correlated together, and including them only increases the usability cost without increasing the distinctiveness. Due to these correlations, picking attributes one by one independently may

²Most attributes are properties accessed through the browser that are limited by its functionalities. Other attributes are items which presence are checked (e.g., the fonts [83], the extensions [199]) or the computation of specific instructions (e.g., the canvas [36]). These are limited by the available items or instructions, which can be large (e.g., more than 2^{154} for the canvas [124]).

lead to poor distinctiveness and usability. Previous works cope with the problem of attribute selection by considering the well-known attributes [56, 127, 81], removing the attributes of the lowest entropy [214], picking iteratively the attribute of the highest weight (typically the entropy) until a threshold is reached [146, 111, 65, 24, 91, 208], or evaluating every possible set [69]. The entropy measures the skewness of the distribution of fingerprints or attribute values. As pointed out by Acar [4], it does not take the worst cases into account (i.e., the most common values that attackers can submit similarly to dictionary attacks on passwords [28]). Moreover, fingerprints cannot be compared identically like passwords due to their evolution through time. The impersonators do not need to find the exact fingerprint of a victim, but one that is similar enough to deceive the verification mechanism. The problem could be solved by exploring exhaustively the space of the possible attribute sets, evaluating the distinctiveness and the usability cost of each set. This is, however, infeasible as the number of attribute sets grows exponentially with the number of attributes³.

1.2 Contributions

1.2.1 Assessing the Adequacy of Browser Fingerprints for Web Authentication

We conduct the first large-scale data-centric empirical study of the fundamental properties of browser fingerprints when used as an additional web authentication factor. We base our findings on the analysis of our four fingerprint datasets, that include a dataset of 4,145,408 fingerprints composed of 216 attributes plus 46 extracted attributes. In particular, this dataset includes nine dynamic attributes of three types, which values depend on instructions provided by the verifier: five HTML5 canvases [36], three audio fingerprinting methods [179], and a WebGL canvas [153]. The dynamic attributes are used within state-of-the-art web authentication mechanisms to mitigate replay attacks [184, 124]. Each dynamic attribute has been studied singularly, but their fundamental properties have not yet been

³Obviously, this discards as well the manual selection of attributes.

studied simultaneously on the same browser population. To the best of our knowledge, no related work considers several datasets of this scale, in terms of both fingerprints and attributes, together with various dynamic attributes. We formalize, and assess on our datasets, the properties necessary for paving the way to elaborate browser fingerprinting authentication mechanisms. We make the link between the browser fingerprints that distinguish browsers, and the biological fingerprints that distinguish Humans, to evaluate browser fingerprints according to properties inspired by biometric authentication factors [139, 231, 75]. The properties aim at characterizing the adequacy and the practicability of browser fingerprints, independently of their use within future authentication mechanisms. In particular, we measure the size of the browser anonymity sets through time, the proportion of identical attributes between two observations of the fingerprint of a browser, the collection time of the fingerprints, their size, the loss of effectiveness between device types, and the accuracy of a simple illustrative verification mechanism. To comprehend the obtained results on the complete fingerprints, we include an in-depth study of the contribution of the attributes to the fingerprint properties. Moreover, we discuss the correlation between the attributes, make a focus on the contribution of the dynamic attributes, and provide the exhaustive list of the attributes together with their properties. To the best of our knowledge, no previous work analyzed browser fingerprinting attributes at this scale, in terms of the number of attributes, of the number of fingerprints, and of the variety of properties (e.g., stability, collection time). We also present the results of a survey about the authentication habits of users and the acceptability of an authentication mechanism relying on browser fingerprinting.

1.2.2 Attribute Selection according to a Security and Usability Trade-off

We place ourselves in the context of a verifier that aims to protect the users of her web platform using a browser fingerprinting based authentication mechanism. The verifier stores the fingerprint of the usual browser of each user. On each login, the fingerprint of the browser of the user is matched against the fingerprint that is stored for this user. The attacker tries to impersonate the users by submitting

specially crafted fingerprints. The aim of the verifier is to limit the reach of the attacker, also called sensitivity, which is measured as the proportion of impersonated users. We propose the FPSelect framework to help verifiers choose the attributes to include in their fingerprinting probe such that (1) the sensitivity against powerful attackers knowing the fingerprint distribution of the protected users (i.e., the worst-case distribution for the verifier) is bounded and the bound is set by the verifier, and (2) the usability cost of collecting, storing, and using these attributes is close to being minimal. FPSelect is parameterized with the sensitivity requirement, the number of submissions that the attacker is deemed able to execute, and a representative sample of the fingerprints of the users. To design and evaluate the FPSelect framework, we perform the following steps. We formalize the attribute selection problem that a verifier has to solve to dimension her probe, and show that this problem is NP-hard because it is a generalization of the Knapsack Problem. We define the model of the dictionary attacker, whose adversarial power depends on the knowledge of a fingerprint distribution and the number of fingerprint that he is able to submit. We propose an illustrative measure to quantify the sensitivity of a probe given a browser population and the number of fingerprints that the attacker is able to submit. We propose an illustrative measure of the usability cost that combines the size of the generated fingerprints, their collection time, and their instability. We propose a heuristic algorithm for selecting an attribute set that satisfies a higher bound on the sensitivity and reduces the usability cost. We express this as a search problem in the lattice of the power set of the candidate attributes. This heuristic is inspired by the Beam Search algorithm [103] and is part of the Forward Selection algorithms [194]. We evaluate the FPSelect framework on two real-life fingerprint datasets, and compare it with common attribute selection methods based on the entropy and the conditional entropy. We show experimentally that FPSelect finds attribute sets that have a lower usability cost compared to both the candidate attributes and the baselines.

1.3 Scientific Publications

During this thesis, we published the papers that are listed below, from which parts of this thesis were adapted.

- [13] **Nampoina Andriamilanto**, Tristan Allard, and Gaëtan Le Guelvouit. “FPSelect: Low-Cost Browser Fingerprints for Mitigating Dictionary Attacks against Web Authentication Mechanisms”. In: *Annual Computer Security Applications Conference (ACSAC)*. ISBN: 978-1-4503-8858-0. DOI: [10.1145/3427228.3427297](https://hal.archives-ouvertes.fr/hal-02965948/document). URL: <https://hal.archives-ouvertes.fr/hal-02965948/document>.
- [12] **Nampoina Andriamilanto**, Tristan Allard, and Gaëtan Le Guelvouit. “Guess Who? Large-Scale Data-Centric Study of the Adequacy of Browser Fingerprints for Web Authentication”. In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. Ed. by L. Barolli, A. Poniszewska-Maranda, and H. Park. 2021, pp. 161-172. ISBN: 978-3-030-50399-4. DOI: [10.1007/978-3-030-50399-4_16](https://hal.archives-ouvertes.fr/hal-02611624/document). URL: <https://hal.archives-ouvertes.fr/hal-02611624/document>.
- [15] **Nampoina Andriamilanto**, Gaëtan Le Guelvouit, and Tristan Allard. “Browser Fingerprinting for Web Authentication, A Large-scale Empirical Analysis”. In: *Rendez-Vous de la Recherche et de l’Enseignement de la Sécurité des Systèmes d’Information (RESSI)*. National conference without proceedings. 2019.

1.4 Outline of the Thesis

The remainder of this thesis is organized as follows.

- In Chapter 2, we present the current state of web authentication where the usage of passwords is prevalent but flawed. We categorize the authentication methods that can supplement passwords, and remark that their usage rate is limited due to usability and deployability costs. Then, we present the browser fingerprinting technique, and the related works on its usage for authentication. Finally, we position our contributions in the research axis of browser fingerprinting for authentication. We notably make the link between biometric authentication methods and browser fingerprinting.

- In Chapter 3, we present the four experiments that we performed. They are the collection of browser fingerprints from four browser populations. The populations are the browsers that visited one of the most visited French website, those visiting the internal website of a company, the standardized browsers of a university, and the browsers that participated in the test of an authentication platform relying on browser fingerprints. As irrelevant data appear in the collected datasets, we describe the preprocessing steps performed to obtain the working datasets. We also introduce the survey about the acceptability of browser fingerprinting for authentication that we performed, by discussing respondent population and their current authentication habits.
- In Chapter 4, we present our contribution about the adequacy of browser fingerprinting for web authentication. It takes the form of the first large-scale data-centric empirical study of the properties of browser fingerprints when used as an additional web authentication factor. We base our findings on the analysis of our four fingerprint datasets. We make the link between browser fingerprints and biometric authentication factors, and analyze the former according to properties inspired by the latter. We complement our results with the study of the contribution of the attributes to the properties of the fingerprints.
- In Chapter 5, we present our contribution to the methods for selecting the attributes to use: the FPSelect framework. This framework helps the verifier select the attributes to include in his fingerprinting probe according to a trade-off between the security and the usability of the generated fingerprints. The security captures the robustness against a modeled attacker, who leverages external knowledge to present specifically crafted fingerprints to impersonate users. The usability is quantified according to the collection time, the size, and the instability of the generated fingerprints. The framework works by modeling the set of possibilities as a lattice, and by exploring a parameterized number of paths in this lattice. We experimentally show that the framework performs better than the common baselines.

- Finally, Chapter 6 concludes this thesis, and discusses future works and perspectives on browser fingerprinting for web authentication.

Chapter 2

Background and Context

In this chapter, we begin by presenting the current state of web authentication and by defining the terms related to authentication. We show that the most used web authentication method is the password, and argue that it suffers from several flaws. We classify the other authentication methods that can supplement passwords to mitigate these flaws, and remark that their usage rate is estimated as low. Indeed, each of these methods comes with a usability cost or a deployability cost that can prevent people from implementing or using them. We present the related works that propose frameworks to evaluate and compare authentication methods. Among these methods, we focus on the biometric methods as they have the particularity of being inexact (e.g., a digital fingerprint is not matched exactly like passwords as changes can occur through time). One important property of authentication methods is their robustness against attacks. Hence, we highlight the common attacks that can be executed on authentication methods.

Second, we focus on a promising candidate: browser fingerprinting. It aims at distinguishing browsers based on attributes that can be collected transparently from the browser. We present the history of browser fingerprinting, the attributes discovered through time, and the usual measures of distinctiveness and stability to evaluate them. We provide the known usages of browser fingerprinting, and acknowledge that it is mainly used for web tracking purposes. Although browser fingerprinting is mainly used for identification, we are interested in its contribution to web authentication. We provide related works on the usages for both identification

and authentication. The related works on the latter propose to include the browser fingerprint as additional verified information in authentication mechanisms, study the adequacy of browser fingerprints for authentication, propose methods to reduce the number of collected attributes, and design challenge-response mechanisms that leverage the dynamic attributes to thwart replay attacks. We position our contributions in relation to the research axis of browser fingerprinting for authentication. We notably make the link between biometric authentication factors and browser fingerprints to evaluate the latter according to properties inspired by the former. Indeed, they both consist into extracting features (e.g., a digital fingerprint, a browser fingerprint) from a unique entity (i.e., a person, a browser) to authenticate the entity. Imperfections can occur as we manipulate digital representation of the features (e.g., two persons or two browsers can share the same fingerprint). Some evaluated properties capture the severity of these imperfections.

This chapter is organized as follows. First, Section 2.1 presents the current state of web authentication, the wide usage of passwords, and the candidate authentication factors to supplement them. Then, we present the browser fingerprinting technique in Section 2.2, the known attributes, and its various uses. Section 2.3 focuses on browser fingerprinting for authentication, for which we provide related works, and in which we position of our contributions. Finally, Section 2.4 concludes.

2.1 Web Authentication

In this section, we begin by defining the terms of authentication, user authentication, and web authentication. Then, we present the current state of web authentication that widely relies on passwords. Third, we present the several flaws that passwords suffer from, which lead to the use of additional authentication factors. We argue that the factors proposed can decrease the usability or the deployability of the authentication mechanism. To comprehend how authentication factors affect a mechanism, previous works have proposed frameworks to evaluate them. Fourth, we present an overview of these frameworks, and focus on the desirable properties of biometric factors. The biometric factors interest us, as the scheme that integrate them have to cope with imperfections. For example, the same nu-

merical representation of the biometric factor – called the biometric template – can be shared by two persons, and the biometric template of a person can evolve through time.

2.1.1 Definitions

User Authentication

According to the Merriam-Webster online dictionary¹, authentication is the “act, process, or method of showing something to be real, true, or genuine”. In computer science, the term usually refers to the verification of the identity of a user. It consists into a *user* claiming an *identity* (e.g., a bank account, a student account) to a *verifier* (e.g., a bank, a university). The claiming user is called the *claimant* for simplicity. To prove that the claimed identity belongs to the claimant, the claimant presents *credentials* that are proofs that the identity belongs to him (e.g., a password, a student card). The verifier compares the provided credentials with the credentials associated to the claimed identity. If the verifier deems the credentials genuine, he grants the claimed identity to the claimant. Otherwise, the verifier does not grant the identity, and can take restrictive measures towards this user as it can be an attempt of fraudulent access. *User authentication* is the process of verifying that the claimed identity belongs to the claimant.

Authentication Factor, Scheme, and Mechanism

The credentials are more specifically called authentication factors. An *authentication factor* is a piece of evidence that the claimant present to the verifier, to support his claim that the identity belongs to him. Examples of authentication factors are passwords, phone numbers, and fingerprints.

An *authentication scheme* is the set of processes that are needed to verify an authentication factor. The same authentication factor can be used by multiple authentication schemes. For example, several authentication schemes that use the fingerprint authentication factor have been proposed [118]. They typically differ on the quantization of the fingerprints and on the parameters used (e.g., the threshold

¹<https://www.merriam-webster.com/dictionary/authentication>

on the matching score, the parameters of the machine learning model).

An *authentication mechanism* is the complete system that integrates one or several authentication schemes and manages their combination. The verifier designs the authentication mechanism that she implements in her web platform. For example, she implements a mechanism that relies on a password and on a digital fingerprint, which only asks for the fingerprint if the password is correct, and which verifies the fingerprint according to a given fingerprint verification mechanism.

Difference with Identification and Authorization

As authentication is sometimes mistaken with identification or authorization, we highlight here the differences between these terms.

Identification is, according to the Cambridge online dictionary², the “act of recognizing and naming someone or something”. User authentication refers to the verification that the identity claimed by a user legitimately belongs to her, answering the question “is this user who she claims to be?”. User identification consists into the retrieval of the identity of a user, answering the question “who is this user?”. Authentication is a one-to-one matching, as the verifier receives a claimed identity, and compares the information of the claimed identity with the information provided by the claimant. Identification is a one-to-many matching, as the verifier searches for the identity which matches with the information of the user to identify. We stress that authentication factors can also be used for identification (e.g., facial or fingerprint recognition).

Authorization occurs after the claimed identity is granted to the claimant. It consists into the verification that the identity is authorized to execute actions (e.g., access, modification) on resources (e.g., files, information), answering the question “is this identity authorized to do this action on this resource?”.

Enrollment, Verification, and Recovery

An authentication mechanism is composed of three separate stages. The initial stage is the *enrollment*, which is the association of the authentication factors to an identity. For example, a user that enters his chosen password at the creation of

²<https://dictionary.cambridge.org/us/dictionary/english/identification>

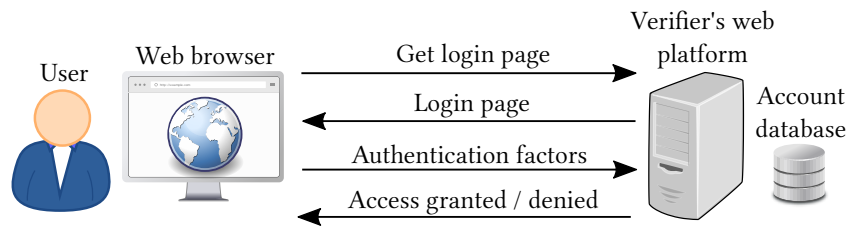


Figure 2.1: Web authentication process between the verifier’s web platform and a user.

his account, or a bank that binds a set of security codes to a customer’s account before sending them to him. After linking the authentication factors to an identity, users can claim this identity by presenting the factors (e.g., entering the password). This stage is called the *verification*, as the verifier checks the presented factors. Finally, the *recovery* is a stage during which users use another set of factors, due to the inability to use one or many primary authentication factors. It occurs when users are unable to present one of the required factor, e.g., when they forget their password or lose their physical token.

Web Authentication

In this thesis, we focus on the authentication of a user in a web context, that we call *web authentication* for simplicity. The *verifier* controls a *website* that contains resources that are private to specific users. A *user* navigates to the verifier’s website using a standard *web browser*, and claims an *identity* to the verifier. The identity is typically identified by a username, an email, or a phone number [180]. The claimant also provides *authentication factors* as proofs that the identity belongs to him. If the authentication factors are deemed genuine by the verifier, the verifier grants the identity to the claimant, and the claimant can access the resources that are private to this identity. Figure 2.1 depicts a web authentication process between the verifier’s web platform and a user.

A concrete example is a bank that controls a banking site, and that only authorizes the owner of a bank account to carry out operations on it. The users identify themselves on the banking site using their customer number, and provide a password as credential. The bank compares the submitted password with that of the claimed account, and grants the identity of the claimed account to the user

only if the passwords match. Otherwise, the bank does not grant the identity to the user, prevents any authentication attempt for a given amount of time, and notifies the account owner about the failed attempt.

2.1.2 Wide Usage and Flaws of Passwords

Nowadays, the authentication factor most implemented by verifiers is the *password* [27, 180]. This is mostly due to its ease of use, its ease of deployment, and the familiarity of users to it. Indeed, passwords are strings of characters that are chosen by users, and submitted to the verifier through a simple form on his website on each authentication. On the other side, verifiers only have to process a strict comparison between the submitted password and the password stored for the claimed account. Although passwords are widely used, they are known to suffer from several flaws. We describe these flaws in this section.

Password Guessability

Passwords are chosen by human users and are not randomly picked in the space of possibilities. They are usually composed of common words (e.g., common names) or common sequences of digits (e.g., dates, zip codes). This makes them easier for an attacker to guess, either through an offline guessing attack, or to a lesser extent, through an online guessing attack. We describe below these two types of guessing attack.

An *offline guessing attack* [223] consists into getting couples of identifier and password hash, and retrieving the plain text of the hashed passwords. This is done by hashing selected plain texts, and comparing the obtained hash with the hashes to retrieve. This attack is then limited by the computational power of the attacker. An *online guessing attack* [220] consists into submitting couples of identifier and password directly to the authentication service of the website. This is done by submitting the most probable passwords given the knowledge of the attacker about the victims' password. This attack is then limited by the capacity of the verifier to limit the authentication attempts to a given identity. To execute both of these guessing attacks, attackers need to find the passwords to try. They can do this by brute force attacks, or more efficiently by dictionary attacks. We

describe these two attacks below.

Brute force attacks consist into trying every possible password. Although rate limiting policies mitigate this threat for online attacks, advances in computation power enable the generation of every password composed of 8 characters in less than a day in an offline attack³. Brute force attacks can be enhanced using rainbow tables, that are precomputed lookup tables that provide a time-memory trade-off. Already in 2003, using such tables, Oechslin managed to break 99.9% of alphanumeric passwords for Windows operating systems in a matter of seconds [165].

Dictionary attacks work by trying the most probable passwords first, given knowledge about passwords or their patterns. Weir et al. [223] proposed a method to infer the most common password patterns from a password dictionary (e.g., a digit followed by five letters), and to generate passwords according to these common patterns. The researchers used a dataset of more than 60,000 real passwords, that they cut in two to form the training set, and the set of the passwords to crack (i.e., that would have been hashed and which plaintext would be to retrieve). They achieve an increase of the cracked passwords up to 129% compared to the state-of-the-art tool in 2009, namely John the Ripper⁴, and manage to crack more than 10 thousand passwords among the 33 thousand passwords to retrieve. Most recent dictionary attacks leverage targeted knowledge about victims (e.g., another used password, personal information) to increase the number of cracked passwords (e.g., Wang et al. [220] manages to crack 77% of their passwords with 100 tries). This effectiveness questions the robustness of websites against online guessing attacks. Indeed, Lu et al. [136] examined the rate-limiting policies of 182 websites among the 500 most visited websites in the United States⁵ at the date of December 21, 2017. They found that 131 of these websites allow more than 100 authentication attempts.

Password Reuse and Data Leaks

Due to the guessing attacks being facilitated by the skewness of the distribution of user-chosen passwords, verifiers enforce *stricter password policies*. These policies

³https://www.theregister.co.uk/2019/02/14/password_length

⁴<https://www.openwall.com/john>

⁵<https://www.alexa.com/topsites/countries/US>

typically require users to use a password longer than a specific size, and that is composed of various characters including lowercase, uppercase, digits, and symbols. However, more complex password policy hardens password creation and recall [198]. This leads to users undergoing the process of account recovery more often, or to wrong behavior like writing passwords on a piece of paper.

The problem of *password memorability* is exacerbated by users having more and more accounts, hence more and more passwords to remember. The average number of online accounts that a user have is not well-known, and estimations vary. In 2014, a report from Experian stated that Britons have 19 online accounts on average [60]. Another report from Dashlane in 2015 stated that an email address was associated to more than 90 online accounts on average⁶. More recently, a report from Okta in 2019 stated that, on average, workers have 10 passwords to remember in their everyday professional life.

Users tend to *reuse the same password* for multiple online accounts, or slight modification of an already used password. Wang et al. [218] studied the password reuse and modification patterns of 28 million users. They found that 38% of the users reuse at least one password on several accounts, and 21% use at least one modification of an already used password. The leak of a user's password can then highly impact the security of his online accounts.

In a world where companies tend to store a large amount of data, *data leaks* are common. These leaks even regularly contain passwords. The BreachAlarm project⁷ keeps track of data leaks, and hardly a day goes by without a new leak being added to the long list. The HaveIBeenPwned project⁸ keeps track of data leaks in a less granular way, but allows users to check whether their email address is present in a leak. To date, the project allows searching among 438 leaks that concern more than 9 billion accounts. Figure 2.2 displays the visualization of the data breaches occurring over the last three years, that were recorded by *informationisbeautiful.net*⁹ at the date of the 1st April 2020. The reuse of passwords,

⁶<https://web.archive.org/web/20191211045559/https://blog.dashlane.com/infographic-online-overload-its-worse-than-you-thought>

⁷<https://breachalarm.com/all-sources>

⁸<https://haveibeenpwned.com>

⁹<https://www.informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks>

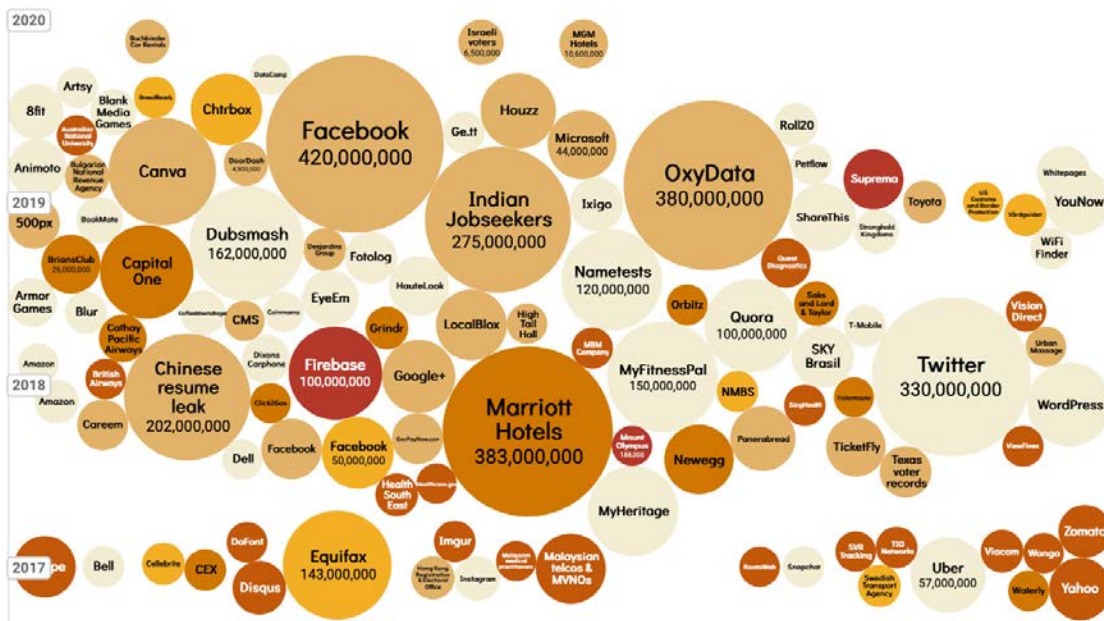


Figure 2.2: Data breaches that occurred over the last three years that were recorded by informationisbeautiful.net at the date of the 1st April 2020. The warmer the color is, the more sensitive is the data.

combined with the large amount of password leaks, increases the effectiveness of credential stuffing attacks¹⁰, which consist into using stolen couples of identifier and password to authenticate to websites. The reach of the dictionary attacks based on targeted knowledge [220] is also enhanced.

Phishing Attacks

Instead of waiting for password leaks to happen, or stealing the passwords stored by a website, an attacker can directly ask users for their password. Attackers do not present themselves as such, but instead use subterfuges to present themselves as an entity trusted by the victim. This is called a *phishing attack*, and can be effective as deceiving unaware users is easier than attacking a well-secured website.

The Google Transparency Report¹¹ publishes the number of detected phishing websites per week. To date, around 30 thousand new phishing websites are

¹⁰<https://www.wired.com/story/what-is-credential-stuffing>

¹¹<https://transparencyreport.google.com/safe-browsing/overview>

detected every week, compared to less than 10 thousand 10 years ago. Moreover, phishing websites manage to get valid certificates [117], deceiving even more the users as common browsers used to display a green padlock in their interface. Since, most common browsers switched to a gray padlock instead of a green one (see Chrome¹² and Firefox [89]). Other browsers keep the green padlock when the entity behind the website is also verified by *Extended Validation*, which links the certificate with a company's name that is then displayed in the interface (see Edge¹³). However, users can still be deceived by the impression of trustfulness induced by the padlock, and phishing websites can get an *Extended Validation* certificate with a name similar to the name of the entity owning the impersonated website [95].

Thomas et al. [210], over the period of March 2016 to March 2017, compiled passwords stolen through leaks, keyloggers¹⁴, and phishing kits¹⁵. They managed to retrieve 3 million victims of phishing kits, and an additional 12 million Gmail users that were potentially victims of these kits. They explain that, although they compiled more passwords coming from leaks than phishing kits, a password stolen through a phishing kit is more likely to match the password currently in use by the victim. Indeed, they report a match rate of 24.8% for the passwords stolen through phishing kits, against 6.9% for the leaked passwords, and 11.9% for the passwords stolen by a keylogger. They conclude that having his password stolen through a phishing kit increases the chances of getting his account hijacked by more than 400 times.

2.1.3 Multi-Factor Authentication

Because of the multiplicity of password flaws, verifiers need to replace or supplement passwords with other ways of verifying users' identity. We classify authentication factors into the four well-known categories that are described below, and for which we provide examples. These factors can supplement passwords in what is

¹²<https://blog.chromium.org/2018/05/evolving-chromes-security-indicators.html>

¹³<https://expeditedsecurity.com/blog/dv-ssl-in-microsoft-edge>

¹⁴A keylogger is a malware that spies on the actions of the user on his computer, typically by recording the pressed keys, the mouse movements, and screenshots.

¹⁵Phishing kits are packages that allow attackers to easily deploy phishing attacks. They comprise the fake website and the tool to report the stolen passwords.

called *multi-factor authentication* (MFA), which is an authentication process that relies on factors from at least two different categories. If relying on exactly two categories, it is called *two-factor authentication* (2FA). According to estimations done in 2015 [173] and 2018 [149], the usage rate of MFA is below 10%.

Classification of Authentication Factors

We provide below the well-known categories of authentication factors. We stress that in a web context, not all authentication factors are usable (e.g., the ones that require a specific scanner), but web browsers evolve to include more APIs that allow implementers to access a wide range of factors (e.g., the W3C Web Authentication standard¹⁶).

What you know *Knowledge factors* consist into a secret shared between the user and the verifier. The verifier authenticates the user by verifying that he knows the secret. Passwords are part of this category, together with unlock patterns of mobile phones [23], and answers to personal questions [181, 29].

What you have *Possession factors* consist into an object or an entity that the user possesses. The verifier authenticates the user by verifying that he has the object or the entity. Examples are security tokens [48] or mobile phones [116].

What you are or what you can do *Biometric factors* comprise anatomical and behavioral factors. *Anatomical* factors consist into the measurements of physical properties of the user's body. The verifier authenticates the user by verifying parts of his body. Examples are fingerprint [139], face [161], or iris recognition [119]. *Behavioral* factors consist into how the user interacts with the physical world. The verifier authenticates the user by verifying that he behaves the same way. Examples are keystroke dynamics [202], mouse dynamics [202], touch-based gestures [189], or gait analysis [159].

When and where you authenticate Context-based factors concern the location from where, and the time when, the user authenticates. The verifier authenti-

¹⁶<https://www.w3.org/TR/webauthn>

cates the user by verifying that the process takes place in a geographical location, and in a time window, that is in line with the user's usual behavior. Examples are the hour of the day [202], and the geographical location inferred from the IP¹⁷ address of the user [224] or obtained from a GPS sensor [7].

Types of Web Authentication

Explicit and Implicit Authentication Schemes Authentication schemes can be classified into the explicit and the implicit schemes. Explicit authentication schemes require the user to undergo an action, and typically use authentication factors of the first three categories. Indeed, users have to enter the secret, prove that they possess the object or the entity, or undergo the scanning process of their body parts. Implicit authentication schemes are executed transparently, without the user noticing the verification [112]. They usually leverage an authentication factor that is either a behavioral factor, or a context-based factor. For example, the mouse movements of users can be verified during their interaction with the web page, and their location can be inferred by their IP address that is transmitted with their communications.

Static and Continuous Authentication Mechanisms The authentication process can be executed at two different times. It can be done at log-in for *static* authentication mechanisms. The verifier checks the identity of the user at log-in, and assigns the claimed identity to the user for this browsing session if she is deemed legitimate. It is typically performed by assigning a session cookie to the user's browser, that allows the verifier to maintain the granted identity. The authentication mechanism can also be *continuous*, in which case the verifier assesses the legitimacy of the user throughout his browsing session. Such mechanisms seek to thwart account hijacking attacks by leveraging implicit authentication schemes.

Risk-Based Authentication The sensitivity of the resources to which an identified user can have access can be variable. A verifier can allow users to use

¹⁷*IP* stands for Internet Protocol. The IP address is a label assigned to each device, allowing it to communicate on the Internet.

less constraining authentication schemes for insensitive resources (e.g., solely passwords), and can require users to use more robust and more constraining schemes for sensitive resources (e.g., a one-time password). This is the main idea of *risk-based authentication* [224]. It consists into assigning a risk level to users depending on how they authenticated, and requiring the risk level to be higher than a threshold to allow the access to a specific resource. Quermann et al. [180] observed real-world usages of risk-based authentication by German banks. The websites of the studied banks ask users for their password when users access their transactions or their balances, and require users to present a second factor when accessing other functionalities like the processing of a financial transaction.

Fallback Authentication It happens that users are unable to present one of the requested authentication factors, like when they forget their password or lose their physical token. To allow a user to recover the access to his account, the verifier has to authenticate the user by other factors than the ones that the user is unable to provide. It is typically done by sending an email to the address registered to the account, containing a code or a link that allows the user to reinitialize the primary factors (e.g., a password). This process is called *fallback authentication* [181, 180], and is more rarely executed than the primary authentication process. For example, Brostoff and Sasse [33] estimated the rate of password recovery to 1 per 4-5 users per month.

Low Usage of Multi-Factor Authentication

Verifiers have several candidate authentication factors to replace or to supplement passwords. Even though websites already propose to use additional factors to passwords [180], few users authenticate using multiple factors. Petsas et al. [173], in 2015, examined over 100 thousand Google accounts and found that only 6.4% of them activated two-factor authentication. Milka Grzergor, a Google engineer, later confirmed their findings during a talk at Enigma 2018 [149]. He presented a use rate of two-factor authentication by active Google accounts that is lower than 10%.

2.1.4 Evaluation of Authentication Schemes

The obstacles to the wide usage of MFA stem from the lack of usability, security, or deployability of the candidate authentication schemes [27] compared to passwords. To select the authentication schemes to implement among the candidates, a verifier needs a way to compare them on a common ground. In this section, we present the related works about benchmarks to quantify or qualify the desirable properties that an ideal authentication scheme should have.

Ruoti et al. [188] analyzed the usability of seven web authentication mechanisms according to the metric of the System Usability Scale (SUS) [32]. The SUS metric measures the perception of the usability of a system by users on a numerical scale that goes from 0 to 100. The metric is measured by asking users 10 questions about their perceived usability of the authentication mechanism.

Bonneau et al. [27] proposed the Usability Deployability Security (UDS) framework that embarks 25 properties qualifying the usability, deployability, and security of authentication mechanisms. Moreover, they evaluated 35 authentication mechanisms according to the UDS framework. They concluded that most evaluated mechanisms provide a better security than passwords, that some offer a better usability, but that all of them provide a worse deployability.

Alaca et al. [7] extended the UDS framework with 2 additional usability properties and 4 additional security properties. These new properties focus on inexact authentication schemes, like the *No-False-Accepts* and *No-False-Rejects* properties that are tied to the effectiveness of the verification mechanism, and on the resistance against impersonation attacks.

2.1.5 Evaluation of Biometric Authentication Schemes

Biometric authentication factors and browser fingerprints share strong similarities. They both work by extracting features from a unique entity, which is a person for the former and a browser for the latter, that can be used for identification or authentication. Although the entity is unique, the extracted features are a digital representation of the entity, that can lead to imperfections (e.g., the fingerprints of two different persons can show similar representations). Previous studies [139,

[231, 75] identified the properties for a biometric characteristic to be *usable*¹⁸ as an authentication factor, and the additional properties for a biometric authentication scheme to be *practical*. Moreover, they present the common metrics used to evaluate these authentication schemes, and the visualizations to compare them. These properties are commonly used to compare biometric or behavioral authentication schemes [176, 231, 75, 147], so are the metrics [140, 164, 75, 147, 55] and the visualizations [118].

Usability and Practicability Properties

Required Properties for Usability The four properties needed for an anatomical or a behavioral characteristic to be usable as a biometric authentication factor are described below.

- *Universality*: the characteristic should be present in everyone.
- *Distinctiveness*: two distinct persons should have different characteristics.
- *Permanence*: the same person should have the same characteristic over time.
- *Collectibility*: the characteristic should be collectible and measurable.

Properties for Practicability The three properties that a biometric scheme requires to be practical are the following.

- *Performance*: the scheme should be accurate, consume few resources, and be robust against environmental changes.
- *Acceptability*: the users should accept to use the scheme in their daily lives.
- *Circumvention*: it should be difficult for an attacker to deceive the scheme.

¹⁸Here, *usable* refers to the adequacy of the characteristic to be used for authentication, rather than the ease of use by the users.

Metrics and Visualizations

Metrics A user is either a genuine user (i.e., she owns the claimed identity) or an impostor (i.e., she is claiming an identity that is not hers). The verifier stores the numerical representation of the biometric characteristic of the user during the enrollment process. This numerical representation is called the *biometric template*. The verifier then defines a matching function that, given the template stored for the claimed identity and the one presented by the claimant, returns a *matching score*. The higher this matching score is, the higher is the confidence that the claimant is the genuine user. The verifier defines a *threshold* above which the claimant is deemed legitimate, and below which the claimant is deemed an impostor. The four metrics are related to the ability of the verification mechanism to correctly classify a user into her true class, depending on the configured threshold.

- *False Match Rate* (FMR): the proportion of impostor users that are classified as legitimate, also called the *False Acceptance Rate* (FAR).
- *False Non-Match Rate* (FNMR): the proportion of legitimate users that are classified as impostors, also called the *False Reject Rate* (FRR).
- *Equal Error Rate* (ERR): the rate for the threshold that provides equality¹⁹ between the FMR and the FNMR.
- *Accuracy*: the proportion of users classified as their true class.

Visualizations To evaluate or compare biometric authentication schemes, two visualizations are commonly used. Increasing the threshold decreases the FMR and increases the FNMR, as the verification mechanism considers more cases as impostor and is stricter. The opposite also holds. It is then important to display the results with respect to the threshold.

¹⁹If we seek to minimize the FMR, we could pick the lowest threshold. As a result, every user is deemed an impostor, the FMR is null, and the FNMR is 1.0. On the contrary, if we seek to minimize the FNMR, we could pick the highest threshold. As a result, every user is deemed genuine, the FNMR is null, and the FMR is 1.0. The FMR decreases monotonically, and the FNMR increases monotonically, with the increase of the threshold. As the value of the FMR and the FNMR are impacted by the threshold, the measure of the EER allow us to provide a balanced indicator.

- *Guenuine and impostor matching score distribution*: this visualization displays the distribution of the matching score of the genuine and impostor classes with respect to the threshold.
- *Detection-Error Tradeoff (DET) curve*: this visualization displays the FMR and FNMR, usually in logarithmic scale, for several threshold values.

2.1.6 Robustness of Authentication Schemes against Attacks

One important aspect of authentication schemes is the security they provide, that is usually expressed as their robustness against attacks. We provide here a list of attacks against which an authentication scheme should be robust to. These attacks consist into an attacker seeking to impersonate one or several legitimate users. We compiled the attacks from the security properties of the UDS framework [27], and from studies related to the evaluation of authentication schemes [164, 219, 124].

Guessing An attacker can search for a value to present to the authentication scheme to impersonate a user (e.g., a password, a biometric template). To do so, the attacker can try every possible value, or leverage the skewness of the value distribution to increase his chances of impersonation. The former method is called *brute force* and simply consists into trying every possible value. The reach of this method is limited in an online context as the verifier can enforce a rate limiting policy, whereas it can be more efficient in an offline context. The latter method is called *dictionary attack* and requires the attacker to have the knowledge of a distribution over the possible values. We note that Section 2.1.2 discusses the sensitivity of passwords against these two attacks. To mitigate guessing attacks, there are several requirements. The space of the possible values should be large enough to thwart a brute force attack. The values should be distributed uniformly to mitigate dictionary attacks. The rate limit should be low enough to reduce the number of presentations that an attacker can perform. The reach of the attacker is limited by his computing power for offline attacks, over which the verifier has no control.

Targeted Impersonation To increase his chances of impersonation, an attacker can leverage targeted knowledge about a victim. Such attacks are called *targeted impersonation* attacks or mimicry attacks [7]. The difference between the dictionary attack and the targeted impersonation attack is that, for the former there can be no information about the victim in the attacker’s knowledge, whereas in the latter there is. The scope of targeted knowledge is large, and includes personal information [29], behavioral information [197], and leaks from other verifiers [220].

Replay If an attacker has complete knowledge over the value taken by the authentication factor of a victim, he can execute a *replay attack* to impersonate this victim. It simply consists for the attacker to present the value. The credential stuffing attack over passwords that is described in Section 2.1.2 falls within the family of replay attacks. *Presentation attacks* are replay attacks that specifically target biometric systems, and consist into presenting a spoofed anatomical feature to the biometric sensor. For example, Matsumoto et al. [144] managed to deceive fingerprint biometric systems by presenting fake fingerprints made of easily accessible gelatin, and forged from fingerprint residue.

Relay *Relay attacks* are like phishing attacks with the difference that they work in real-time. In these attacks, the attacker mounts a system that allows him to pose as the verifier to a victim, and as the victim to the verifier. These attacks were already analyzed outside a web context, particularly for wireless authentication mechanisms. Kfir et al. [110] executed relay attacks on contactless smart card systems, Francillon et al. [73] on passive keyless entry systems of cars, and Roland et al. [185] on a mobile contactless payment system. Relay attacks are also already used by thieves in the real-world [122]. In a web context, Amnesty International reported relay attacks targeting privacy-conscious users [99]. The strength of these attacks is that they manage to bypass two-factor authentication²⁰ by relaying the request for an SMS One-Time Password (OTP) from the verifier to the victim, and relaying back the OTP entered by the victim to the verifier. Ways to mitigate relay attacks are distance-bounding protocols [73, 20].

²⁰<https://blog.duszynski.eu/phishing-ng-bypassing-2fa-with-modlishka>

Internal Observation An attacker can get access to a victim’s device, typically by tricking the victim into installing a malware. This attack is called an *internal observation* attack, and allows the attacker to observe the actions of the victim, the computations of the device, and the communications. *Eavesdropping* are the internal observation attacks that focus on analyzing the communications. It can be performed by a man-in-the-middle attack, that consists into the attacker getting access to the network packets on the communication channel between the verifier and the victim. However, due to the wide-usage of encryption for the exchange of sensitive information on the web, it would be easier for attackers to compromise the devices than the communications. A concrete example is an attacker that gets a keylogger installed on a victim’s device, and retrieves the victim’s password by registering the pressed keys.

Physical Attack An attacker that is physically close to the user can execute *physical attacks*, that comprise *shoulder surfing* and *theft*. *Shoulder surfing* [23] consists into an attacker observing a victim during the authentication phase, with his own eyes or with observation tools (e.g., binoculars, cameras). An attacker can simply steal the possession factors of a victim, like a physical token for example. These two attacks are not effective on every authentication schemes, and can be mitigated by the protection of the authentication factor (e.g., protecting the physical token or the entry of the password). Moreover, previous works [86, 23] proposed authentication schemes that are robust against shoulder surfing.

2.2 Browser Fingerprinting

In this section, we present the technique of browser fingerprinting, that consists into distinguishing browsers by collecting browser attributes. An attribute is an information that is accessible through the browser, and which can differentiate two browsers (e.g., the `UserAgent` JavaScript property [186] which gives a technical description of the browser). There is no list of the accessible attributes, as they depend on the evolution of web technologies, and some are the result of side effects (e.g., the drawing of a scheme [153, 36], the inference of installed fonts [65, 83] or extensions [205, 199]). We give an overview of the attributes that were discovered,

together with the properties against which they are usually evaluated, and the various usages of browser fingerprinting.

2.2.1 Discovery of Browser Fingerprinting

At the start of 2020, DataReportal estimated that there are more than 4.5 billion Internet users in the world [46], and that this number keeps increasing (it is 7% more than what was observed in 2019 at the same period). Moreover, the place that Internet takes in our daily lives is important. The same report states that the average Internet user spends more than 6 hours per day online. One of the main usage of the Internet is the access to the web, mostly through mobile applications or web browsers.

Since the release of the first readily accessible web browser in 1991²¹, browsers have evolved to support more and more functionalities. This evolution stems from the need of users to use the same platform to perform their online activities. Modern browsers allow users to watch videos, make video conference²², and even play video games²³. Moreover, users are given a large choice of browsing platform, thanks to the diversity of device types (e.g., smartphones, desktop computers, smart TVs), operating systems (e.g., Linux-based, macOS, Android), and web browsers (e.g., Chrome, Firefox, Edge). Web developers had to adapt to this diversity, and develop their web pages for compatibility between browsing platforms (e.g., by the principles of cross-browser compatibility²⁴ and responsive web design²⁵). To help web developers, browsers are built to give access to information about the browsing platform. Examples are the HTTP²⁶ header named *User-Agent* [187] that provides the browser family and version, and the various screen related information provided by the *screen* JavaScript object [45]. Finally, users can customize their browser, by configuring the settings (e.g., the language, the

²¹<http://line-mode.cern.ch>

²²<https://www.w3.org/TR/webrtc>

²³<http://xproger.info/projects/OpenLara>

²⁴https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Cross_browser_testing

²⁵https://www.w3schools.com/html/html_responsive.asp

²⁶*HTTP* stands for Hyper Text Transfer Protocol. It is the protocol that defines the way to communicate information on the web.

Do Not Track signal [71]), and by installing themes, fonts²⁷, plugins, or extensions [106]. The browser functionalities, the diversity of browsing platforms, and the user customizations all contribute to the overall diversity of browsers.

Jonathan R. Mayer is the first [145], in 2009, to ask if “with all the customizations” are there “any two web browsing environments identical”? By collecting 4 attributes from 1,328 browsers, he found that the values of these attributes were unique for 96.23% of the browsers. Afterward, in 2010, Peter Eckersley published the result of the collection of 8 attributes from 470,161 browsers [56]. He called the technique of collecting browser attributes *browser fingerprinting*, and called the obtained values *browser fingerprints*. The collection took place on the website of Panopticlick [72], a project of the Electronic Frontier Foundation that explores the diversity of browser fingerprints. Eckersley found that 83.6% of the browsers had a unique fingerprint, and that 37.4% of the users that have multiple visits presented different fingerprints through time.

Browser fingerprinting is the process of collecting attributes from a web browser and aggregating their value into a single *browser fingerprint*, to distinguish different browsers. The term of *device fingerprinting* is sometimes used to refer to browser fingerprinting. We emphasize that even if browser fingerprinting can be included in the notion of device fingerprinting, these are two different techniques. Indeed, browser fingerprinting identifies the web browsing environment by ways limited to what is offered by the browser (e.g., network communications, client-side scripts), whereas device fingerprinting identifies a device by ways limited to what is offered by the device [121] (e.g., native applications).

The fingerprinting term refers to the physiological fingerprints, in the way that the latter are generally used to recognize individuals. Simon Cole argued about fingerprints that they “may be unique in the sense that, as Gottfried Wilhelm Leibniz argued, all natural objects can be differentiated if examined in enough detail” [139]. The idea of fingerprinting was applied to a variety of things, from paper sheets [39] to 3D printed objects [132].

²⁷We stress that modern browsers do not provide a way to directly install fonts. However, users can install a font on their operating system, and then configure this font as the default one.

2.2.2 Browser Fingerprinting Attributes

Since the awareness about browser fingerprinting was raised, several studies discovered new fingerprinting attributes. We give an overview of these attributes in this section. First, we present the *fixed attributes*, which value only depends on the web environment into which a browser is running. Then, we present the *dynamic attributes*, which value depends on the web environment but also on specific instructions provided by the fingerprinter. We also give the two properties of the attributes that are usually evaluated, namely the *distinctiveness* and the *stability*. Finally, we discuss the evolution of accessible attributes.

Fixed Attributes

The value of *fixed attributes* only depends on the current web environment into which the browser is running. The *web environment* is composed of the software and hardware components that run the browser, and that impact its behavior. It is at least composed of the browser itself, the customization of the browser, the underlying operating system, the network connection, the graphical card, the screen, and the peripheral drivers. Abgrall et al. [1] classified these attributes into the passive and the active ones.

Passive attributes *Passive attributes* are collected passively from the behavior of the browser, or from information that it systematically sends. Passive attributes mostly consist of the information sent by the browser to a web server during the web communication. They include the IP address, the HTTP headers [187, 56], and TCP²⁸ or ICMP²⁹ timestamps [115].

Active attributes Active attributes need to be explicitly requested, typically by sending a client-side script that is then executed by the browser. Modern browsers embark a JavaScript engine that allows them to execute JavaScript scripts. Browsers used to also support plugins (e.g., Flash Player, Java, Silverlight)

²⁸*TCP* stands for Transmission Control Protocol [98]. It is the main connected protocol used to communicate over the Internet protocol, and the one on which HTTP relies.

²⁹*ICMP* stands for Internet Control Message Protocol [175]. It is the protocol that defines the transmission of error or debugging messages of the underlying Internet protocol.

Table 2.1: Example of HTTP headers sent during a request

Header name	Header value
User-Agent	Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:74.0) Gecko/20100101 Firefox/74.0
Accept	text/css,*/*;q=0.1
Accept-Language	en-US,en;q=0.5
Accept-Encoding	gzip, deflate, br
DNT	1

that enabled them to execute scripts of other languages. However, the major browsers started to remove the support of plugins around 2016 (see Firefox³⁰ and Chrome³¹), and the end of the support of plugins is close (see Silverlight³², Java³³, and Flash [209]). The functionalities of the now obsolete plugins are planned to be replaced by the functionalities provided by HTML5³⁴ (e.g., real time communication [66], audio processing and player [179], rendering of two-dimensional graphics and three-dimensional scenes [153]). These attributes are mostly composed of the JavaScript properties [56, 127] provided by the `navigator`, `screen`, `window`, and `document` JavaScript objects. We also find attributes related to the hardware components, like the screen resolution [127], the quirks of the sensors [26, 79, 226, 141], the state of the battery [166], and information about specific components [207, 191, 193, 192] (e.g., the CPU, the available memory, how the browser processes mathematical functions). Specific APIs³⁵ also provide information, typically for debugging purposes, like the WebRTC [66]³⁶ or the WebGL³⁷ [153, 127]

³⁰<https://blog.mozilla.org/futurereleases/2015/10/08/npapi-plugins-in-firefox>

³¹<https://blog.chromium.org/2014/11/the-final-countdown-for-npapi.html>

³²<https://support.microsoft.com/en-in/help/4511036/silverlight-end-of-support>

³³<https://blogs.oracle.com/java-platform-group/moving-to-a-plugin-free-web>

³⁴*HTML* stands for HyperText Markup Language. It is the descriptive language that tells browsers how to render web pages. *HTML5* refers to the fifth and latest version to date, that was released in 2014. It embarks plenty of new features since the previous version that was released in 1999.

³⁵*API* stands for Application Programming Interface. The APIs are a set of interfaces offered by browsers that provide functionalities (e.g., the audio API [179] provides functions to process and play audio signals).

³⁶WebRTC stands for Web Real-Time Communication. This API provides the utilities to open a real-time communication between browsers. It is notably used for video or audio conferences.

³⁷*WebGL* stands for Web Graphics Library. This API provides the utilities to render and manipulate two-dimensional graphics or three-dimensional scenes.



Figure 2.3: Example of an HTML5 canvas that displays overlapping texts, rectangles, emojis, curves, ellipses, and a shading.

APIs. The installed components can also be retrieved, like the plugins [56], the fonts [65, 83], or the extensions [205, 199].

Dynamic Attributes

The value of *dynamic attributes* depends on the web environment, but also on instructions provided by the fingerprinter. They typically are media objects (e.g., a sound, an image) that are rendered by the browser, encoded (e.g., *base64* encoding), and sent back to the fingerprinter. The dynamic attributes can be efficient to distinguish browsers depending on the set of instructions used, as their value depend on the behavior of the browser which relies on its hardware and software components. For example, the HTML5 canvas is one of the most distinctive attributes (see Section 4.3.2). It is due to the generated image depending on several components that include the fonts and the emojis used to render texts, and the graphical components (e.g., the graphical card and its driver) used to render the overlapping figures and colors. Below, we describe the three dynamic attributes that, to the best of our knowledge, were discovered. We note that fingerprinting a browser using either HTML5 canvas or WebGL canvas is more specifically called *canvas fingerprinting*.

HTML5 Canvas The *canvas*³⁸ API provides the functions to render and manipulate graphics within the browser. To use this API, the fingerprinter first needs to specify a context. The *2d* context provides the functionalities to draw and manipulate two-dimensional graphics. It is used to fingerprint browsers by differences

³⁸<https://html.spec.whatwg.org/multipage/canvas.html>

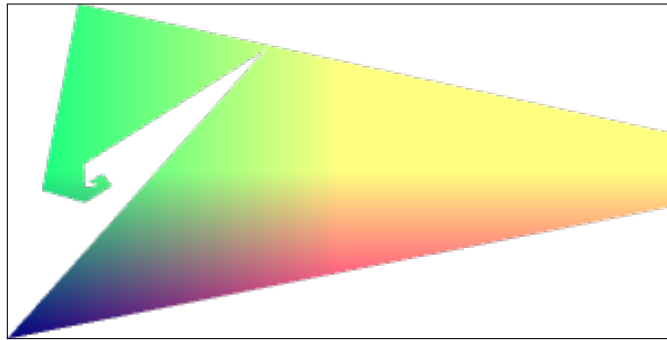


Figure 2.4: Example of a two-dimensional WebGL canvas that displays a succession of connected triangles.

in the rendered graphics. Mowery et al. [153] were the first to fingerprint browsers using the inconsistencies among graphics rendered by the canvas API, by rendering a chosen text in a specified font. They noticed that browsers have different ways to render a pangram, which is a sentence that contains all the alphabet letters, in the common Arial font. Moreover, they remarked that browsers use a fallback font when the specified font is unavailable. Laperdrix et al. [127] analyzed the rendering of canvas images displaying two pangrams of different fonts and colors, together with an emoji and a rectangle of another color. They acknowledged that these images were among the most distinctive attributes of their study.

WebGL Canvas The canvas API also includes the *webgl* and the *webgl2* contexts. These two contexts use the *WebGL* library [221] that leverages hardware accelerations to render and manipulate two-dimensional graphics or three-dimensional scenes. To differentiate the two usages of the canvas API, we call *HTML5 canvas* the method that relies on the 2d context, and *WebGL canvas* the method that relies on the *webgl* context. Mowery et al. [153] were also the first to use the *webgl* context to fingerprint browsers. They render a specifically chosen image on a hyperbolic paraboloid structure, and illuminate the scene from a specific angle with a given lighting. Finally, they extract a snapshot of the rendered scene, and use this as a fingerprinting attribute.

Audio Fingerprinting The counterpart of the canvas API for the manipulation of audio signals is the *Web Audio API*³⁹. It works by creating an *Audio Context*, into which we manipulate *Audio Nodes*. These nodes are of three types. Source nodes generate an audio signal (e.g., from a microphone or an audio stream), destination nodes send the signal to be rendered (e.g., by speakers), and manipulation nodes manipulate the signal (e.g., increasing the volume). These nodes are linked together to form a network, that goes from source nodes to destination nodes through manipulation nodes. Englehardt and Narayanan [57] discovered the use of this API by fingerprinters when assessing the usage of browser fingerprinting on the web. Queiroz et al. [179] tested various methods using this API for fingerprinting. By analyzing the fingerprints obtained from 122 devices, they found that only 6 devices had unique fingerprints. They explain that this can come from their use of basic features of the API, coupled with the use of the same audio rendering engines by different browsers.

Properties of Attributes

Browser fingerprinting attributes are evaluated following two properties: their *distinctiveness*, that is the ability of the attribute to distinguish two different browsers, and their *stability*, that is to which extent the attribute keeps the same value through time.

Distinctiveness The distinctiveness of an attribute is usually measured by the entropy, the normalized entropy, the anonymity sets, and the unicity. The *Shannon's entropy* [56] measures the quantity of information provided by a random variable, mostly in bits. To keep it short, each additional bit of entropy divides the population of browsers in two when searching to recognize a given browser. The *normalized entropy* was introduced by Laperdrix et al. [127]. It is the ratio between the entropy, and the maximum achievable entropy that depends on the size of the population. The *anonymity set* of a fingerprint, or an attribute value, is the set of browsers that share it. The *unicity* is simply the percentage of the fingerprints that have been seen for a single browser only.

³⁹<https://www.w3.org/TR/webaudio>

Stability The stability of an attribute is its ability to provide the same value through time. Peter Eckersley [56] first measured the stability on complete fingerprints as the percentage of the fingerprints that stay identical after a given time. Alaca et al. [8] evaluated the stability of an attribute given its nature, that is whether it is expected to change or not. Vastel et al. [214] measured the stability of an attribute as the days that pass before a percentage of the browsers has this attribute changed. Pugliese et al. [178] defined the stability of a fingerprint, given the set of considered attributes, as its ability to stay stable at least between two observations.

Evolution of Accessible Attributes

The functionalities of web browsers evolve constantly, with new ones being added, and obsolete functionalities being removed. The attributes that can be used for fingerprinting follow these evolutions. Most attributes are information accessed through the browser, that are limited by the functionalities of the browser. Other attributes are items which presence are checked (e.g., the fonts [83], the extensions [205, 199]) or the computation of specific instructions (e.g., the canvas [36]). These are limited by the available items or instructions, which can be large (at least 2^{154} for the canvas [124]). We can then acknowledge that, to date, we have more than hundreds of available attributes.

New functionalities are regularly added to browsers, some following the release of new web standards⁴⁰ published by the World Wide Web Consortium (W3C). A functionality can be removed from browsers due to its obsolescence and replacement by newer technologies (e.g., the plugin removal explained in Section 2.2.2). The privacy threats posed by the identification of browsers, and by the leakage of information, are also causes of removal. For example, the *Battery Status* API, that gives access to the state of the battery of the device, was removed from Firefox [21] due to the privacy threats demonstrated by Olejnik et al. [166]. The rendering of visited links also used to leak history through a side-channel. Since, its implementation was altered in major browsers, but new attacks are still discovered [201].

⁴⁰<https://www.w3.org/TR/?status=rec>

In the crowd	Of the type
Web tracking	Anti-robots protection
Authentication	Anomaly or intrusion detection
Anti-fraud protection	Service customization
Forensic analysis	User advice
	Targeted attacks

Table 2.2: Usages of browser fingerprinting, classified between the usages to recognize a browser in the crowd, and the ones to recognize the type of browser.

2.2.3 Usages of Browser Fingerprints

Abgrall et al. [1] emphasize that browser fingerprinting techniques can be used to identify *browsers*, or their *type* (e.g., family, version). We classify the usages of browser fingerprinting into two categories, that are the recognition of a browser *in the crowd*, and the recognition *of the type* of a browser. This classification is displayed in Table 2.2.

In the Crowd

Web Tracking Web tracking is the process of tracking a user’s activities on the web, mostly by following the web pages that she visits through time. The tracking is done either by using stateful techniques that store an identifier in one or several storage mechanisms of the browser (e.g., cookies, evercookies⁴¹), or by using stateless techniques that leave no trace in the browser. Browser fingerprinting is the main, if not the only, stateless technique, raising privacy concerns⁴² about this technique [56, 127, 57].

Authentication Authentication is the usage of browser fingerprinting that interests us. The Section 2.3 is dedicated to discuss this usage.

Anti-fraud Protection Various companies include the browser fingerprinting into their anti-fraud services (e.g., EarlyWarning⁴³, Sift⁴⁴). The usage of browser

⁴¹<https://samy.pl/evercookie>

⁴²<https://www.w3.org/TR/fingerprinting-guidance>

⁴³<https://www.earlywarning.com/products/browser-iq>

⁴⁴<https://blog.sift.com/2015/introducing-formulas-and-device-fingerprinting>

fingerprinting in such services typically works by assigning a risk level to a browser that is recognized by its fingerprint. This risk level is afterward provided to the customer of the service (e.g., an online store), that can make decisions based on this knowledge (e.g., ask for more verification if the browser is of a high risk level).

Forensic Analysis Browser fingerprints, like their biometric counterparts, can be used during investigation as a clue. Previous works [133, 100] proposed to use browser fingerprinting to trace attackers. Moreover, investigators working for TripAdvisor managed to create a link between a company and fake reviews submitted on their website, by using “hundreds of review attributes such as IP addresses, browser types and even the screen resolution of a reviewers device”⁴⁵.

Of the Type

Anti-robots protection It is estimated that in 2018, 20% of the web traffic was done by malicious robots [51]. Robots are usually running inside a headless browser, that is a browser without graphical interface and having fewer functionalities. As the fingerprints of these headless browsers differ from the fingerprints of the browsers that humans use, it is possible to detect robots based on the fingerprint of their browser. Vastel et al. [215] studied this usage, and found that it is already present on the web.

Anomaly or Intrusion Detection When the population of browsers that is expected to connect to a website is known (e.g., an intranet website expecting browsers running on specific systems), it is possible to detect, and deny the access to, external browsers by using browser fingerprinting. Moreover, it is also possible to detect altered browsers (e.g., emulated browsers). Bursztein et al. [36] proposed Picasso, a protocol that relies on HTML5 canvas to identify the browser and operating system families of browsers. The objective of this protocol is to detect anomalous browsers, like non-mobile browsers accessing mobile application marketplaces.

⁴⁵<https://www.tripadvisor.com/TripAdvisorInsights/w4237>

Service Customization Websites can offer a personalized service depending on the browsing environment of the user. Hupperich et al. [97] searched for price discrimination based on browser fingerprinting on hotel and car rental websites. They argue that even if news articles reported such cases⁴⁶, they did not find any systematic price discrimination on the studied websites.

User Advice The knowledge of the versions of the browser and the operating system of a user can help advise him. For example, a system administrator can warn the users to update their outdated browser or operating system. A website can also warn a user that his browsing environment is not compatible with functionalities offered by the website (e.g., YouTube that warns users that their browser is outdated⁴⁷).

Targeted Attacks Fingerprinting is commonly used in targeted attacks to infer the targeted environment (e.g., operating system, firmware) before sending a malicious payload to which the environment is sensitive. Fingerprinting tools rely on the behavior of devices, typically by their network communication, to infer their software or hardware stack. An example of such tool is *poF*⁴⁸, that dates back to 2000, and uses the TCP/IP communications to infer information about a device. Since the third version, it also embarks browser identification capabilities by the analysis of the HTTP headers.

2.3 Authentication by Browser Fingerprinting

As most studies on browser fingerprinting focus on its use for identification, we first present an overview of these studies. We notably focus on the analysis of browser fingerprints for identification purpose, and show that properties necessary for their use for authentication are not addressed. Afterward, we present an overview of the studies that focus on browser fingerprinting for web authentication, and position our contributions to each research axis. We argue that browser fingerprinting is a

⁴⁶<https://www.wsj.com/articles/SB10001424052702304458604577488822667325882>

⁴⁷<https://support.google.com/youtube/thread/27621880?hl=en>

⁴⁸<https://lcamtuf.coredump.cx/p0f3>

good candidate as an authentication factor thanks to its distinguishing power, its frictionless deployment (e.g., no additional software to install), and its usability (no secret to remember, no additional object to possess, and no supplementary action to carry out).

2.3.1 Analysis of Browser Fingerprints for Identification

Most studies on browser fingerprinting focus on its use for identification, mostly for web tracking purposes. These studies emphasize the threat posed to privacy by browser accessible information [56, 153, 65, 127, 66, 205, 81, 179, 106], assess the usage of browser fingerprinting over the web [2, 57, 43], or propose counter-measures to defend against it [163, 128, 22, 125]. The studies that analyze browser fingerprints for identification lack strong insights into their use for authentication (e.g., the stability necessary for the verification of fingerprints, the accuracy of a matching mechanism).

Peter Eckersley [56] analyzed the fingerprints of 470,161 browsers collected from the Panopticlick website [72]. He defines the stability as the proportion of changing fingerprints given the time elapsed between two observations, and proposes a simple matching algorithm. Due to this study being 10 years old, the set of available attributes since evolved as more attributes are available and browsers do not support plugins anymore.

Laperdrix et al. [127], and Gómez-Boix et al. [81], respectively analyzed 118,934 and 2,067,942 fingerprints composed of the same set of 17 attributes. These studies focus on the identification effectiveness of fingerprints, and on the evolution of web technologies that would reduce this effectiveness. They only consider a small portion of the available attributes, which does not reflect well what a fingerprinter can really achieve. Moreover, they do not analyze the evolution of fingerprints through time, whereas the stability of fingerprints is necessary for their verification in an authentication context.

Vastel et al. [214] studied the evolution of 98,598 fingerprints from 1,905 browsers. They measure the stability of each attribute as the duration until the attribute value changes, and propose a matching algorithm that relies on both rules and machine learning techniques. Even if they provide a good insight on

stability, their study only comprises 18 attributes, and the matching is different between identification and authentication. Indeed, identification consists into linking a presented fingerprint to one, or optionally none, of a set of candidate browsers (i.e., it is a 1 to N matching). For authentication, the set of candidate browsers is reduced to these of the claimed account, and we seek to verify whether the presented fingerprint belongs legitimately to one of them (i.e., 1 to 1, or 1 to n matching if multiple devices are allowed).

Pugliese et al. [178] analyzed 88,088 fingerprints collected over three years, performed a survey on users' perception of browser fingerprinting, and proposed an attribute selection method that we discuss in Section 2.3.4. They provide the analysis of the distinctiveness and the stability of fingerprints, and use these two parameters to select the subset of attributes to implement. However, they provide few data on the resource consumption of fingerprints (e.g., their collection time), and no insight into the proportion of attributes that change in the fingerprints.

2.3.2 Browser Fingerprinting into Authentication Mechanisms

The first studies about the use of browser fingerprinting for security purposes proposed to integrate it into authentication mechanisms. All of these mechanisms are either continuous [212, 203], risk-based [202], or even both [177]. The adequacy of browser fingerprinting for continuous authentication stems from the possibility to collect the attributes transparently in the background. As for risk-based authentication, browser fingerprinting allows the verifier to gain insight about the user's device, which is captured in the verified context.

Unger et al. [212] proposed the Session Hijacking Prevention Framework, which is a continuous authentication framework to protect users against session hijacking. The framework works by verifying that the fingerprint of a browser stays the same during the authenticated session. The used attributes are the HTTP headers, the IP address, and the supported CSS⁴⁹ features. We note that the supported features do not provide distinctiveness between two browsers of the same version,

⁴⁹ *CSS* stands for Cascading Style Sheets. It is a language designed to describe the presentation of web pages.

as they only depend on the version.

Preuveneers et al. [177] proposed SmartAuth, an authentication mechanism that is continuous, risk-based⁵⁰, and directed toward mobile devices. It works by storing the information about the previous contexts into which a user has authenticated. The context comprises browser fingerprinting attributes (JavaScript properties and HTTP headers), the location, the IP address, and the time of access. The attributes collected on the browser are hashed using similarity preserving hash functions before being sent. This way, the value of these attributes is hidden. They extended the open source identity management OpenAM [174] with SmartAuth, and added data to the communications (e.g., a counter, a timestamp) to counter replay and phishing attacks.

Spooren et al. [203] proposed to include the state of the battery as a factor in a continuous authentication mechanism. They proposed two binary-classifiers to determine the authenticity of a user. The first classifier leverages the pattern of the battery state throughout the day. It provides the likelihood that a given user, at a given time of the day, has his device charged to a given percentage. The second classifier leverages previous measurements of the battery charge, and provides the likelihood that, after a given time, the battery is charged to a given percentage. Afterward, they evaluate the effectiveness of each classifier when considering a naïve attacker, by constructing the attack attempts from random users. They also evaluate the effectiveness in an adversarial context, by considering that the attacker presents a specifically chosen spoofed value. They present four different strategies, including the use of global knowledge to infer the most likely value at a given time of the day, or the presentation of a completely charged battery that they consider common in the morning. However, the first proposed classifier has an equal error rate of 41.3%, which is a small gain compared to a random guess (50%). As for the second classifier, they obtain an equal error rate of 27.3% when the battery charge was observed one hour before, and of 42.8% for an observation done two hours ago.

⁵⁰SmartAuth relies on configurable scores depending on the authentication context, which comprises the browser fingerprint among other factors (e.g., IP address, geolocation). If the scores exceed a configured threshold, the user has to undergo additional authentication steps. The example that the authors give is a one-time password sent by email or SMS that the user has to provide.

Solano et al. [202] proposed a risk-based authentication method that authenticates a user by his behavior and the context from which he authenticates. The context information that they consider comprises the location, the browser fingerprint, and the time of connection. By combining behavioral dynamics (mouse, keyboard), possession (web environment), and context-based factors (location, time of connection), they manage to hinder the attackers that simulate the context of a victim, or that have physical access to a victim's device. Although the authors argue that the duration of a password log-in is sufficient to analyze the behavioral interactions, we argue that the user would have few interactions with the log-in page. The interactions would only consist of the password entry – which is typically less than twelve characters [28, 172, 217] – and a mouse click on the submit button. Moreover, users can use the password auto-filling functionality of the browser or the one of a password manager.

These studies focus on the integration of browser fingerprinting into an authentication mechanism, and do not provide insight into how efficiently is browser fingerprinting when used that way (e.g., their distinctiveness, their stability, their resource consumption). They consider a small fraction of the hundred of available attributes, that does not include the dynamic attributes that can be used in a challenge-response mechanism to thwart replay attacks. Moreover, some proposed mechanisms require the usage of a mobile device [177, 203], putting aside the desktop users that still constitute a significant part of the web users.

2.3.3 Adequacy of Browser Fingerprints for Authentication

Several studies assessed the adequacy of browser fingerprints for authentication. They mostly consist into the estimations of the quality of browser fingerprints according to desirable properties, and lack empirical analysis of actual fingerprints to support these estimations.

Spooren et al. [204] studied the adequacy of the fingerprints of mobile browsers for authentication. They analyzed the fingerprints of 59 mobile devices, and conclude that although their sample is small, these fingerprints lack distinctiveness and are predictable. Indeed, the devices of the same model present similar fin-

gerprints, and several attributes can be inferred directly from the device model (e.g., the plugins are mostly empty, the screen resolution is publicly available in the description of a model).

Alaca and van Oorschot [8] provided a classification of the fingerprinting attributes that can be used for authentication, evaluated their properties, and identified the attacks on such mechanisms. The properties that they consider are the stability, the repeatability, the resource usage, the passiveness, and the resistance to spoofing. The attacks they identified on the mechanism include guessing attacks that consist into presenting the most common fingerprints, and targeted attacks that leverage knowledge over the victim. We note that the properties of attributes are mostly qualified, or quantified from the results of previous studies that work on different populations of different sizes. Moreover, the attackers are only identified, and no analysis of the robustness of a supposed mechanism is presented.

Alaca et al. [7] extended the UDS framework [27] (see Section 2.1.4) to consider the resistance against mimicry attacks, and evaluated implicit authentication schemes. These schemes include several sets of fingerprinting attributes, comprising the usual JavaScript properties, the dynamic attributes, and hardware sensors [43]. The added properties to the UDS framework capture the resistance to errors of the verification mechanism, the variability of the scheme that helps thwart simple replays of previously leaked values, and its resistance to spoofing. They performed an evaluation of the several sets of fingerprinting attributes following the UDS framework and their extension.

Laperdrix et al. [124] identified various attacks that can be executed on an authentication mechanism that includes browser fingerprinting. These attacks notably include the submission of the most common fingerprints, and targeted attacks using the knowledge of static attributes to infer dynamic ones.

Markert et al. [142] recently presented a work in progress about the long-term analysis of fallback authentication methods. They plan to measure the recovery rate of the evaluated methods, after an elapsed time of 6, 12, and 18 months. These methods include the browser fingerprinting, for which they acknowledge that “not much about browser fingerprinting-based security systems is known”.

Our contribution: “Guess Who”

In Chapter 4, we present our contribution to the assessment of the adequacy of browser fingerprinting for authentication. This contribution provides the first large-scale empirical study of browser fingerprints with a focus on authentication, by evaluating them according to the desirable properties of biometric authentication factors presented in Section 2.1.5 and on four fingerprint datasets. The scale of our largest dataset is the largest to date: it was collected over six months from nearly two million browsers, and it contains more than four millions fingerprints that are composed of 216 attributes, which is closer to what a verifier can achieve.

2.3.4 Selection of Attributes

Due to the large number of accessible attributes, related works have focused on reducing the number of collected attributes.

Previous studies proposed to remove the attributes of the lowest entropy [214], or proposed greedy algorithms that iteratively pick the attributes of the highest entropy until a threshold is reached [146, 111, 24, 91]. These methods do not take into account the correlation between the attributes. Effectively, two attributes can separately have a high entropy, but when taken together provide a lower entropy than another attribute set.

To cope with the problem of correlation between attributes, several studies weight the attributes conditionally to these already picked. The method is then to iteratively select the attributes by taking the one that has the highest weight at each step. Taking two correlated attributes is therefore avoided. Fifield and Egelman [65] proposed to weight attributes by the conditional entropy. The entropy of each attribute is measured according to the ones that are already picked. Pugliese et al. [178] proposed an attribute selection method that iteratively selects the best attribute given the objective function. They propose two objective functions that capture the duration for which a fingerprint stays stable, and the unicity of fingerprints.

Another solution is to explore exhaustively the possible set of attributes, which requires to have enough computing power to evaluate every possible attribute set. Although it is possible for few candidate attributes, it becomes quickly impractical

as the number of possible sets grows exponentially with the candidate attributes. Indeed, there are 2^n possible attribute sets for n candidate attributes. Flood and Karlsson [69] evaluated every set of their 13 attributes to find the attribute set that provides the best classification results, and found that 7 attributes do not higher the accuracy. An exhaustive search is feasible on a small set of candidate attributes, but unrealistic on a larger set. Tanabe et al. [208] selected attributes based on the classification effectiveness of an attribute set. For each size of possible sets (i.e., from 1 to n for n available attributes), they evaluated every attribute set of this size, and held the set that provides the highest classification effectiveness (measured as the Area Under the Curve or AUC). Given a size, they obtain the attribute set of this size that provides the best classification effectiveness. To select the final attribute set, they take the most efficient one among the results obtained for each size. We note that this method explores exhaustively each possible attribute set.

Gulyás et al. [83] studied the problem of finding the set of s -items (e.g., fonts, plugins, applications) for which to check for presence on a device, to reduce the number of devices that share the same value. They proved that this problem is NP-hard by reducing it to the Maximum Coverage Problem [113], and proposed greedy algorithms to find the closest approximation in polynomial time.

Our contribution: FPSelect

In Chapter 5, we present our contribution to the methods of attribute selection. It is the FPSelect framework that allows a verifier to select the attributes to use, according to a trade-off between the security and the usability of the generated fingerprints. The security captures the robustness against a modeled dictionary attacker, who leverages external knowledge to present specifically crafted fingerprints to impersonate users. The usability is quantified according to the collection time, the size, and the instability of the generated fingerprints. The framework works by modeling the set of possibilities as a lattice, and by exploring a parameterized number of paths in this lattice. Even if the obtained solution is not optimal, due to the NP-hardness of the problem, we empirically show that the framework performs better than the common baselines of entropy and conditional entropy.

2.3.5 Challenge-response Mechanisms

Several works noted that some attributes could receive instructions from the verifier, which impacts the generated result. These works proposed to leverage these dynamic attributes to design a challenge-response mechanism that thwarts replay attacks. Our two contributions study several instances of these dynamic attributes, including six HTML5 canvas [36], one WebGL canvas [153], and three audio fingerprinting methods [179].

To the best of our knowledge, the first work that proposes to challenge a device for web authentication was done by Van Goethem et al. [79]. They proposed to leverage the sensor quirks of the accelerometer as an additional authentication method directed toward mobile devices. To do so, the enrollment consists into making the mobile device play vibrations of different durations, and simultaneously collecting data from the embedded accelerometer. The verifier then stores the behavior of the device for each vibration duration. During the authentication process, a random subset of the stored vibration durations is fetched, and sent to the device as a challenge. The resulting accelerometer data forms the response, and is checked against the expected behavior that was previously stored. The limit of their method is that it can only be used on mobile devices, and requires to place the device on a hard surface, reducing the real-life contexts into which it can be used (e.g., it is unusable in external environments).

To the best of our knowledge, the first work that proposes to vary the instructions to draw the HTML5 canvas images is the work of Daud et al. [47]. They seek to introduce canvas fingerprinting into the authentication service of their organization. After executing a preliminary test, they found that the obtained canvas hashes were uniform on most of their tested browsers. Because of this uniformity, they decided to use different set of instructions for each user, and for each authentication process. The goal is to avoid using the same instructions, which prevents users to fall on the same canvas value. However, an attacker that uses a browsing environment similar to the one of a victim – which is included in their attack model – would probably fall on the same canvas value as the victim’s browser. Even if this attacker and her victim would answer different canvas values from the other users, they would probably answer the same value.

Rochet et al. [184] proposed a challenge-response protocol that leverages the HTML5 canvas and deep learning techniques. The enrollment works by asking 2,000 35x280 canvas images containing random texts to be drawn by the user's browser. These images form the genuine class for the training of a binary-class Convolutional Neural Network (CNN) model, whereas the impostor class is forged using random images of the other users. During authentication, the user's browser is asked to draw several images, constituting the challenge. Each of these images are passed as input to the user's CNN model, which outputs a prediction value per image. Their mechanism has the drawback of consuming high resources during the enrollment due to the generation and the storage of the 2,000 images. Moreover, requiring storing and training one model per user can become cumbersome when the user base grows.

Laperdrix et al. [124] also proposed a challenge-response that leverages the HTML5 canvas. Their canvas images are more complex than the ones of [184], as they contain gradients, text shadows, mathematical curves, and the texts are displayed in various sizes and rotations. At the enrollment, they only ask one 1900x30 canvas image to be drawn by the user's browser, contrary to the 2,000 35x280 canvas images of [184]. The instructions and the resulting image are both stored. During the authentication, they challenge the user's browser with two set of instructions: the previously stored instructions and newly generated ones. There are two resulting images corresponding to each set of instructions. If the image generated using the previously stored instructions matches the expected image previously stored, the challenge is considered as successful. In which case, the newly generated instructions and the associated image are stored for the next authentication.

2.4 Conclusion

In this chapter, we presented first the current state of the authentication on the web, where the most used authentication factor is the password. We argued that passwords suffer from several flaws that include their guessability, phishing attacks, and their reuse coupled with the commonness of data leaks. We presented the categories of authentication factors that can replace or supplement passwords, and

showed that each factor comes with a price in usability or deployability. To qualify this price and compare various candidate factors, we provided previous works that seek to evaluate authentication factors. We notably focus on the desirable properties of biometric factors, as these fit the factors that are inexact (e.g., both human and browser fingerprints are not matched identically contrary to passwords). One important property is the robustness against attacks. Hence, we highlight the common attacks against authentication schemes. Second, we presented browser fingerprinting, which consists into collecting several attributes from a web browser to build its fingerprint. We unraveled the history of browser fingerprinting, the attributes discovered over time, and the usual measures of distinctiveness and stability to evaluate them. We classified the use cases of browser fingerprinting as depending on the objective: recognizing a browser in a crowd of browsers or recognizing its type. We provided related works on the analysis of browser fingerprints for identification, which most studies on browser fingerprinting focus on. We emphasized that we are rather interested in its contribution to web authentication, for which we gave related works and in relation to which we positioned our contributions. The related works propose to include the browser fingerprint as additional verified information in authentication mechanisms, study the adequacy of browser fingerprints for authentication, propose methods to reduce the number of collected attributes, and design challenge-response mechanisms that leverage the dynamic attributes to thwart replay attacks.

Our contributions are the following. In Chapter 4, we present our contribution to the assessment of the adequacy of browser fingerprinting for authentication. We provide the first large-scale empirical study of browser fingerprints with a focus on authentication, by evaluating them according to desirable properties of biometric authentication factors on four fingerprint datasets. We assess the properties required for browser fingerprints to be usable, that include their distinctiveness and stability. We also assess the properties for browser fingerprints to be practical, that include the accuracy of a matching mechanism, their resource consumption, and their acceptability by users. In Chapter 5, we present our contribution to the methods of attribute selection. It takes the form of a framework that allows verifiers to select the attributes to use, according to a trade-off between the security and the usability of the generated fingerprints. The security captures the robust-

ness against a modeled dictionary attacker, and the usability captures the resource consumption and the stability of the generated fingerprints. The framework works by searching for a solution in the set of possibilities modeled as a lattice.

Chapter 3

Experiments and Survey

The browser fingerprints are closely dependent on the browser population being studied. For example, browsers running on the standardized devices of a company are expected to show less distinctiveness than a population of general public browsers which shows more diversity. To study the properties of browser fingerprints, we conducted four experiments that consist into the collection of fingerprints from various browser populations. The browser fingerprints also depend on the attributes that are collected by the fingerprinting probe. We compiled a large set of browser fingerprinting attributes, and designed a fingerprinting probe that collects them. Due to the experimental aspect of the fingerprints combined with the large-scale of our fingerprint collection experiments, irrelevant data appear in the resulting datasets. Hence, we undergo four preprocessing steps. The cleaning step removes irrelevant and erroneous data. The resynchronization step merges the identifiers that we deem as belonging to the same browser with a high confidence. The deduplication step removes the duplicated fingerprints of each browser. The derivation step extracts the extracted attributes from source attributes. We also present the acceptability survey about browser fingerprinting for authentication that we conducted with the help of the *Le Lab* experimentation platform. We focus this chapter on the preliminary questions about the current authentication habits of the respondents, and leave the results on the acceptability of browser fingerprinting for authentication for Section 4.2.6.

This chapter is organized as follows. First, Section 3.1 describes the browser

fingerprinting probe used to collect the fingerprints. Second, Section 3.2 describes the four browser fingerprinting experiments that we conducted. Third, we provide a description of the preprocessing steps that are applied to the raw datasets in Section 3.3. Fourth, in Section 3.4, we describe the acceptability survey in terms of the respondents' population, their authentication habits, and their opinion about the current authentication methods. Finally, Section 3.5 concludes this chapter.

3.1 Browser Fingerprinting Probe

We designed a browser fingerprinting probe, in the form of a JavaScript script, that collects over a hundred attributes. The attributes are either collected from information accessible via JavaScript or contained in the HTTP headers. No plugins are used due to their obsolescence (see Section 2.2.2). We sought to obtain browser fingerprints composed of as many attributes as possible, to estimate more precisely their distinctive power. Hence, we compiled the attributes from previous studies and open source projects (e.g., fingerprintjs2 [67]). In addition to the probe, we set a fingerprint collection server up, which receives the fingerprints collected by the probe. As web technologies evolve through time, we made changes to the fingerprinting probe between the experiments. These evolutions are described in Appendix B. In this section, we give a brief overview of the attributes collected from JavaScript and HTTP headers, together with how the encountered errors can be used as additional information. We refer the reader to Appendix A for a taxonomy of the attributes, and Appendix I for the complete list of the attributes together with their properties (e.g., their number of distinct values).

3.1.1 JavaScript Attributes

The JavaScript attributes of the fingerprinting probe mostly consist of textual properties that are collected from common JavaScript objects (e.g., `navigator`, `screen`). They also comprise list attributes (e.g., the list of plugins [56]), the presence of browser components (e.g., the presence of fonts [65], of extensions [205, 199]), and the properties of web page elements (e.g., the default size and origin of a created `div` element). Three types of dynamic attributes are also part of

the JavaScript attributes. They include five HTML5 canvases [36], three audio fingerprinting methods [179], and a WebGL canvas [153]. In addition to the values of the JavaScript attributes, we also store their collection time. We save a Unix timestamp when the fingerprinting probe starts, and we save a timestamp each time an attribute is collected. The difference between these two timestamps allow us to deduce the elapsed time until the attribute is collected. Moreover, by ordering the collected attributes by their timestamp and evaluating their difference two by two, we can deduce the time taken to collect each attribute¹.

3.1.2 Dynamic Attributes

Our fingerprinting probe includes up to nine dynamic attributes: five HTML5 canvases [36], three audio fingerprinting methods [179], and a WebGL canvas [153]. These attributes are particular as they can be integrated within challenge-response mechanisms that mitigate replay attacks [184, 124].

HTML5 Canvases

Our fingerprinting probe includes up to five HTML5 canvases, generated following three sets of instructions and in two image formats (PNG and JPEG). We name the canvases given the set of instructions used and their format. We call *AmIUnique canvas* the canvas generated by the set of instructions inspired by the AmIUnique² study [127], and extracted in PNG format. We call *Morellian canvas* the canvas generated by the set of instructions that is similar to the Morellian study [124], which is extracted in PNG and JPEG formats. We call *custom canvas* the canvas generated by a set of instructions that we designed, which is extracted in PNG and JPEG formats. We collect the canvases by using the `toDataURL` function. The `quality` parameter of this function goes from 0.0 to 1.0, and allow us to control

¹Most attributes are collected sequentially (one after the other). However, some attributes are asynchronous and are collected simultaneously with the others. These asynchronous attributes are the first ones to get their collection process launched. Their collection time is then the difference between the timestamp collected when the probe starts and the one when they are collected.

²Contrarily to the AmIUnique study [127], we only have one sentence and one smiling face emoji instead of two, we do not draw a colored rectangle, and the sentence is drawn using a color gradient.

the level of compression of the JPEG versions. As we seek to compare the PNG canvases that are compressed without loss with their JPEG counterparts by using a high level of compression, we set the `quality` to 0.1 for the JPEG versions. An example of the custom canvas in PNG format is displayed in Figure 2.3. Examples of the AmIUnique and Morellian canvases are given in Appendix A.

WebGL Canvas

The WebGL canvas is an image that is also drawn using the HTML5 API, but within the `webgl` or `webgl2` contexts. These contexts use the WebGL library [221] that leverages hardware accelerations to render and manipulate two-dimensional graphics, but also three-dimensional scenes. Canvas fingerprinting was first introduced by Mowery et al. [153] by using the `webgl` context, but afterward most studies focused on the `2d` context [22, 36, 63, 214]. This can result from the unreliability of the method encountered by Laperdrix et al. [127], for which Cao et al. [38] proposed a remedy by setting specific parameters. The WebGL canvas of our fingerprinting probe is composed of sequential triangles drawn using a color gradient. Figure 2.4 displays an example of our WebGL canvas.

Web Audio Fingerprinting

Web audio fingerprinting was discovered by Englehardt et al. [57] when assessing the use of web tracking methods on the web, and more recently studied thoroughly by Queiroz et al. [179]. It consists into processing audio data into the browser, and getting the rendered result. Similarly to canvases, this processing relies on software and hardware components, and variations occur between different component stacks. Our fingerprinting probe includes three web audio fingerprinting attributes: the *audio fingerprinting simple* (AFS) attribute relies on a simple process, together with the *audio fingerprinting advanced* (AFA) and *audio fingerprinting advanced frequency data* (AFA-FD) attributes that rely on a more advanced process. Their concrete implementation is described in Appendix A.

3.1.3 HTTP Headers

The HTTP headers are collected from the POST requests sent by the probe to the server, which contain the fingerprint. Due to this collection method, we have no fingerprint collected from browsers having JavaScript disabled. We extract specific fields from the HTTP headers, which are stored into dedicated attributes. Among the remaining fields, we ignore a set of fields, and store the name and the value of the other fields into a dedicated attribute. Appendix A provides the collected and the ignored HTTP header fields for each version of the fingerprinting probe.

3.1.4 Error and Inaccessibility Handling

If the value of an attribute is not accessible, a flag explaining the reason is stored instead, as it is still exploitable information. Indeed, two browsers can be distinguished if they behave differently on the inaccessibility of an attribute (e.g., returning a null value is different from throwing an exception). We also configure a timeout after which the fingerprint is sent without waiting for every attribute to be collected. The attributes that were not collected are then set to a specific flag.

3.2 Browser Fingerprinting Experiments

We conducted four experiments that consisted into collecting browser fingerprints from various browser populations that depend on the collection website. For clarity, we name each experiment according to the browser population. We collected fingerprints (1) from a general audience French website during the *general audience experiment*, (2) from an internal website of the Institute of Research and Technology b<>com during the *intranet experiment*, (3) from standardized computers of Université de Rennes 1 during the *university experiment*, and (4) from a publicly accessible website where users could test an authentication mechanism which includes browser fingerprints as an authentication factor during the *enrolled experiment*. The fingerprinting probe was updated between the experiments, as described in Appendix B. Table 3.1 displays a comparison between the browser fingerprint datasets of the four experiments and the large-scale datasets of previous studies. By matching the `userAgent` JavaScript property with manually selected

Table 3.1: Comparison between the browser fingerprint datasets obtained from the four experiments, and the large-scale datasets of previous studies. Previous studies include the studies Panoptichick (PTC) [56], AmUnique (AIU) [127], Hiding in the Crowd (HitC) [81], and Long-Term Observation (LTO) [178]. The returning browsers are the browsers from which multiple fingerprints have been collected. The unicity is the proportion of the fingerprints that were observed for a single browser only. The attributes comprise the derived attributes. The number of fingerprints and of distinct fingerprints are obtained after dataset preprocessing. As the LTO study tests various attribute sets, we present the ranges of the results obtained by their attribute selection method. - denotes missing information, and * denotes deduced information. The proportion of mobile and desktop fingerprints do not sum to 1.0 as other device types were observed (e.g., tablets, TVs, video game consoles).

	PTC	AIU	HitC	LTO	G. Aud.	Intranet	Univ.	Enrolled
Beginning	01/27/10	11/-/14	12/07/16	02/09/16	12/07/16	04/17/18	09/02/19	12/09/19
Ending	02/15/10	02/15/15	06/07/17	02/23/19	06/07/17	06/30/20	06/30/20	06/30/20
Duration	3 weeks	3 months*	6 months	3 years	6 months	2 years	9 months	6 months
Attributes	8	17	17	305	262	260	243	259
Browsers	-	-	-	-	1,989,365	489	129	980
Returning brws	-	-	-	-	0.275	0.734	0.473	0.281
Fingerprints	470,161	118,934	2,067,942	88,088	4,145,408	9,422	268	1,787
Distinct fps	409,296	142,023 ³	-	9,822-16,541	3,578,196	9,342	85	1,761
Desktop fps	-	0.890*	0.879	0.697*-0.707*	0.805	1.000	1.000	0.874
Mobile fps	-	0.110*	0.121	0.293-0.303	0.134	-	-	0.118
Unicity	0.836	0.894	0.336	0.954-0.958	0.818	0.984	0.235	0.973
Unicity (mobile)	-	0.810	0.185	0.916-0.941	0.399	-	-	0.962
Unicity (desktop)	-	0.900	0.357	0.974-0.978	0.884	0.984	0.235	0.974

Table 3.2: The share of browser and operating system families among the browser populations of the four experiments. The categories are ordered from the most common to the least for the general audience dataset.

	General Audience	Intranet	University	Enrolled
Firefox	0.268	0.342	1.000	0.337
Chrome	0.265	0.586	-	0.519
Internet Explorer	0.260	0.002	-	0.008
Edge	0.078	0.005	-	0.046
Safari	0.064	0.061	-	0.059
Samsung Browser	0.048	-	-	0.013
Others	0.017	0.004	-	0.018
Windows 10	0.338	0.253	0.298	0.551
Windows 7	0.312	0.190	-	0.059
Other Windows	0.096	$5.5 \cdot 10^{-4}$	-	-
Android	0.091	-	-	0.106
Windows 8.1	0.075	0.052	-	0.011
Mac OS X	0.051	0.422	0.006	0.206
iOS	0.026	-	-	0.034
Ubuntu	0.005	0.065	-	0.013
Other Linux	0.003	0.019	0.696	0.020
Others	0.003	-	-	0.001

keywords (see Appendix C), we infer the browser and operating system family of each browser. Table 3.2 presents the share of browser and operating system families among the browser population of each experiment.

The browsers of our four datasets mostly belong to French users. Counter-intuitively, considering an international population may not reduce the distinctiveness. Indeed, we can expect foreign users to have a combination of contextual attributes (e.g., timezone, languages) different from the French users, making them distinguishable even if the remaining attributes have identical values. Al-Fannah and Li [63] found out that browser families are not equally fingerprintable (e.g., Safari browsers are less distinguishable than Chrome browsers). Although the fingerprintability of the browser families of the population studied impacts the obtained distinctiveness, studying this aspect is out of the scope of this thesis.

3.2.1 General Audience Experiment

The first experiment was done in collaboration with the authors of the Hiding in the Crowd study [81] and an industrial partner⁴ that controls one of the 15 most visited French websites⁵. The authors of [81] only held the 17 attributes of their previous work [127], and focused on the issue of web tracking. On the contrary, we consider more than one order of magnitude more attributes, and focus on the use of browser fingerprints as a web authentication factor. We installed the fingerprinting probe on two general audience web pages that are controlled by our industrial partner, which subjects are political news and weather forecast. The probe was active between December 7, 2016, and June 7, 2017. The browser fingerprints collected during this experiment compose the general audience dataset.

³This number is displayed in Figure 11 of [127] as the number of distinct fingerprints, but it also corresponds to the number of raw fingerprints collected. Every fingerprint would be unique if the number of distinct and of collected fingerprints are equal. Hence, we are not confident in this number, but it is the number provided by the authors.

⁴These works were done in the context of a partnership with the industrial partner that did not go all the way, hence we do not disclose the name of this partner.

⁵<https://www.alexa.com/topsites/countries/FR>

User and Browser Population

The website audience of the general audience experiment is mainly general French-speaking users, which leads to biases of which we provide examples here. As displayed in Table 3.2, Firefox browsers are the most common, followed by Internet Explorer browsers, and Chrome browsers. Even if French users tend to use Firefox as their desktop browser more than the world average [50], the share of browsers is different from what is reported by Statcounter with Chrome being the most common [49]. The distribution of browser and operating system families can be explained by the population visiting the website being less technically savvy, hence using more common web environments (e.g., a Firefox on a Windows operating system) than technical environments (e.g., Linux-based operating systems), contrary to the population of previous studies [56, 127, 178].

The contextual attributes related to the time zone or to the configured language are less distinctive than for previous studies (see Appendix 4.3.2). For example, the normalized entropy of the `Timezone` JavaScript property is of 0.008, against 0.161 for the Panopticlick study [56], and 0.201 for the AmIUnique study [127]. These attributes also tend towards the typical French values: 98.48% of the browsers have a `Timezone` value of `-1`, 98.59% of them have the daylight saving time enabled, and `fr` is present in 98.15% of the value of the `Accept-Language` HTTP header.

Privacy Concerns

We complied with the European directives 2002/58/CE [58] and 2009/136/CE [59] in effect at the time, and took additional measures to protect the participating users. First, the script was set on two web pages of a single domain in a first-party manner, hence providing no extra information about users' browsing. The content of the web pages are generic, hence they leak no information about users' interests. Second, we restricted the collection to the users having consented to cookies⁶. Third, a unique identifier (UID) was set as a cookie with a 6-months lifetime, corresponding to the duration of the experiment. Fourth, we deleted the fingerprints for which the `cookieEnabled` JavaScript property is not set to `true`. Finally, we hashed the IP addresses by passing them through a HMAC-SHA256

⁶As required by the European directives 2002/58/CE [58] and 2009/136/CE [59].

using a key that we threw afterward. It was done using the `secret` and the `hmac` libraries of Python superior to version 3.6. These one-way hashed IP addresses are only used for the UIDs resynchronization (see Section 3.3.2), and are not used as an attribute in the working dataset.

3.2.2 Intranet Experiment

We installed the fingerprinting probe into an internal website⁷ of the Institute of Research and Technology b<>com, which is only accessible to the users connected to the internal network. In the following, we refer to this institute as the *company*. The website presents news about the company and its subjects of interest to the employees. The probe was in place from April 4, 2018, and we consider the fingerprints collected up to June 30, 2020. The browser fingerprints collected during this experiment compose the intranet dataset.

User and Browser Population

The website audience is limited to the users connected to the internal network. The audience is mainly composed of the employees of the company, but visitors can also access the website. The work computers are managed by the staff of the technical support of the company, but the users are authorized to install software (e.g., native applications, browsers, extensions, fonts). As the company is located in France, the employees mainly speak French, but some employees are foreigners who also speak other languages. During the experiment, we observed no mobile browser, as the employees tend to access the website only from their workstation.

Privacy Concerns

We complied with the European directives 2002/58/CE [58] and 2009/136/CE [59] in effect at the time, and took additional measures to protect the participating users. First, the script is set on the home page of the internal website in a first-party manner, hence providing no extra information about users' browsing nor users' interests. Second, we display a banner asking for the consent of the user to

⁷As the website is accessible only from the internal network of the institute, we do not provide its link.

participate in the fingerprint collection experiment, and provide a link for more information about the subject. The link points to a blog post that explains the goal of the experiment which is to study browser fingerprints, the elements about which we collect data (e.g., operating system, browser, plugins), the fact that we set a cookie to follow the evolution of the fingerprints, and what we do not collect (e.g., the IP address, the browsing history). We also provide an email address if users have further questions. Third, we communicated about the experiment to the employees through an internal presentation, several emails, and a blog post. Fourth, we deleted the fingerprints for which the `cookieEnabled` property was not set to `true`.

3.2.3 University Experiment

The fourth experiment was done in collaboration with Université de Rennes 1. We installed the fingerprinting probe as an extension into the default browser of the workstations located in the practical work rooms of the university, and also proposed to the students who were lent laptops to install the extension. We use an extension because it is easier to deploy on the workstations of the university. Moreover, it is more convenient for the users to provide their browser fingerprint automatically through the extension than going regularly on a collection web page. The probe was in place from the beginning of the academic year on September 2, 2019, and we consider the fingerprints collected up to June 30, 2020. The browser fingerprints collected during this experiment compose the university dataset.

User and Browser Population

This experiment concerns two populations of students and university staff. The first population is the users who browse the web through the default browser of the workstations located in the practical work rooms of the university. The default browser of these workstations is Firefox, and is shipped with the extension containing the fingerprinting probe. There are approximately 300 concerned workstations, which are standardized by the support staff to have the same software and hardware components. The second population is the students who are lent laptops by the university. These laptops are re-initialized and are shipped with the

extension – which is not installed – before being lent to the students. The students are provided a flyer explaining the experiment and how to install the extension. The extension is in the WebExtension format, which allows the students to install it on the browsers that accept this format. These browsers are typically the most common, and include Firefox and Chrome.

As the browsers of the first population are running in standardized web environments, they lack diversity. Indeed, all these browsers run on Firefox either on Windows or on Linux⁸. This lack of diversity can be seen by the unicity falling down to 0.235, even on a small population of 129 browsers. However, it also shows that differences occur even between uniformized browsers that share the same software and hardware stacks.

Privacy Concerns

We complied to the General Data Protection Regulation [77] in effect at the time, and took additional measures to protect the participating users. After the extension is installed, it displays a web page explaining the experimentation, and asks for the user consent to participate in the experiment. The explanation notably comprises the types of the collected information, the duration of the storage of the fingerprints, and the rights held by the user (e.g., the access to their data) together with an email address for them to assert their rights. The experiment was launched in collaboration with Université de Rennes 1, in contact with the supervisor of the information security, with the technical support staff, and with the data protection officer. We notably completed a declaration form for the processing of personal data. We communicated to the involved students and teachers through various means that include posts on the intranet of the university, emails, flyers, and posters.

⁸On browser reports running on Mac OS X. However, the workstations of the university only consist of Linux and Windows operating systems.

3.2.4 Enrolled Experiment

The third experiment was done through a publicly accessible website⁹ for experimenters to test an authentication mechanism that includes passwords and browser fingerprints as authentication factors. The working of this authentication mechanism is described in Appendix F. The collected fingerprints, necessary for the authentication to take place, are also stored in a fingerprint dataset. We communicated about this experiment to colleagues and on social networks. We also collaborated with *Le Lab*¹⁰ that manages an experimentation platform to which experimenters can subscribe to participate to experiments about new technologies. The experimenters earn points that they can exchange with gifts. *Le Lab* also posted a survey about authentication to the experimentation platform (see Section 3.4). The website is up since December 9, 2019, and we consider the fingerprints collected until June 30, 2020. The browser fingerprints collected during this experiment compose the enrolled dataset.

Canvas challenges

The authentication mechanism integrates a challenge-response mechanism [124] based on our custom HTML5 canvas. The instructions to draw the canvas are changed on each fingerprint collection. Due to this, the values of the canvases obtained during this experiment are always unique. To avoid the biases that this generates (e.g., the fingerprints are all unique), we replace the value of the HTML5 canvas by the same value for all the fingerprints.

User and Browser Population

The user population of the enrolled experiment consists of the people that have heard of the experiment and wanted to participate. It includes colleagues, the experimenters to which *Le Lab* advertised the experiment, and the people that came across one of our posts on social media.

⁹The link to the experimentation website: <https://demo.barbacan.irt-b-com.org>.

¹⁰<https://lelab.orange.fr>.

Privacy Concerns

We complied to the General Data Protection Regulation [77] in effect at the time, and took additional measures to protect the participating users. For an experimenter to participate in the experiment, she has to create an account on the authentication platform. During the activation of this account, the experimenter is displayed the terms of service which includes an explanation of browser fingerprinting, a brief description of the collected data, and the fact that browser fingerprints are stored for statistical analysis additionally to their use by the authentication mechanism. The access to the authentication phase during which the fingerprint is collected requires the experimenter to give her consent. Finally, at the end of the experimentation phase, we deleted the accounts that the experimenters created to test the authentication platform, and only held the fingerprint dataset.

3.2.5 Comparison with Previous Large-Scale Studies

We compare the datasets using the unicity rate which is the proportion of the fingerprints that are unique. The general audience dataset has a lower unicity rate compared to previous studies [56, 127, 178] due to the larger browser population. Even if its unicity rate is lower than that of AmIUnique, the gap is not extremely wide. This is due to the fact that considering a larger browser population leads to higher chances of collision, reducing the unicity rate. However, by considering a wider surface of fingerprinting attributes, we reduce these chances. These two effects produce this slight decrease of the unicity rate. Although the university dataset has a small browser population, its unicity rate is lower than any of the presented datasets. This is due to the browsers of the university experiment that are uniformized. The intranet and the enrolled datasets have a slightly higher unicity rate compared to previous studies [127, 178]. This results from the browser population being smaller and diverse, and from the higher number of considered attributes. We also observe a lower unicity rate for the fingerprints of mobile browsers compared to the fingerprints of desktop browsers, confirming the findings of previous studies [204, 127, 81].

The authors of the Hiding in the Crowd study [81] worked on fingerprints

collected from the same experiment as the general audience experiment. However, to stay consistent with their previous study [127], they consider the same set of 17 attributes. This explains the higher number of fingerprints, as two browsers that have different fingerprints for a given set of attributes can come to the same fingerprint if a subset of these attributes is considered. Hence, they remove more duplicated fingerprints than us due to the higher chances for a browser to present the same fingerprint for 17 attributes than for 216 attributes. The unicity rate of the general audience dataset is also higher, being at 81.8% for the complete dataset against 33.6% for their study. This is due to the larger set of considered attributes that distinguish browsers more efficiently, as each additional attribute can provide a way to distinguish browsers. Our little drops on the proportion of desktop and mobile browsers come from a finer-grained classification, as we have 4.8% of the browsers that are classified as belonging to touchpads, smart TVs, and game consoles.

3.3 Browser Fingerprints Preprocessing

Given the experimental aspect of browser fingerprints, the raw datasets contain erroneous or irrelevant samples. That is why we perform several preprocessing steps before any analysis. The datasets are composed of entries in the form of (f, b, t) tuples so that the fingerprint f was collected from the browser b at the time t . We talk here about entries (i.e., (f, b, t) tuples) and not fingerprints (i.e., only f) to avoid confusion. The preprocessing is divided into four steps that are described below. They comprise the dataset cleaning, the resynchronization of the unique identifiers (UIDs), the deduplication of the fingerprints, and the derivation of the extracted attributes. The four datasets are passed through the cleaning, the deduplication, and the derivation steps. Only the general audience dataset gets its UIDs resynchronized. Table 3.3 displays the number of initial raw entries, and the entries filtered out during the cleaning or the deduplication steps.

3.3.1 Dataset Cleaning

The fingerprinting probe prepares, sends, and stores the entries in string format consisting of the attribute values separated by semicolons. We remove the entries that have a wrong number of fields, mainly due to truncated or unrelated data (e.g., the body of a post request, empty string). We filter out the entries that come from robots, by checking that blacklisted keywords are present in the `UserAgent` HTTP header (see Appendix C for the list of keywords). We reduce the entries that have multiple exact copies (down to the same moment of collection) to a single instance. Finally, we remove the entries that have the cookies disabled, and the entries that have a time of collection that falls outside the time window of the corresponding experiment.

3.3.2 Unique IDs Resynchronization

The cookies are considered an unreliable browser identification solution, hence we undergo a cookie resynchronization step for the general audience dataset, similarly to the Panopticlick study [56]. We consider the entries that have the same (fingerprint, IP address hash) pair to belong to the same browser, and assign them the same unique identifier (UID). Similarly to the Panopticlick study, we do not synchronize the interleaved UIDs, that are the UIDs related to the entries having the same (fingerprint, IP address hash) pair, but showing UID values b_1 , b_2 , then b_1 again. Indeed, the interleaved UIDs is a strong indicator that several browsers are operating behind the same public IP address. The UIDs resynchronization step is only applied to the general audience dataset due to its size and the availability of the IP address hash. The general audience dataset gets 181,676 UIDs replaced with 116,708 UIDs.

3.3.3 Deduplication

To avoid storing duplicates of the same fingerprint observed several times for a browser, the usual way is to ignore a fingerprint if it was already seen for a browser during the collection [56, 127]. Our script collects the fingerprint on each visit, no matter if it was already seen for this browser or not. To stay

consistent with common methodologies, we deduplicate the fingerprints offline. For each browser, we hold the first entry that contains a given fingerprint, and ignore the following entries if they also contain this fingerprint. This method takes the interleaved fingerprints into account, that are the fingerprints so that we observe f_1 , f_2 , then f_1 again. For example, if a browser b has the entries $\{(f_1, b, t_1), (f_2, b, t_2), (f_2, b, t_3), (f_1, b, t_4)\}$, after the deduplication step we only hold the entries $\{(f_1, b, t_1), (f_2, b, t_2), (f_1, b, t_4)\}$.

We hold the interleaved fingerprints to realistically simulate the state of the fingerprint of each browser through time. The interleaved fingerprints can come from attributes that switch between two values. An example is the screen size that changes when an external screen is plugged or unplugged. Previous studies discarded the fingerprints that were already encountered for a given browser [56, 127, 81], hiding these interleaved fingerprints.

3.3.4 Extracted attributes

From the attributes that we collect, we derive extracted attributes that are parts of the original attributes (e.g., the height and width of the screen derived from the screen size) or inferred information (e.g., the number of plugins from the list of plugins). For all the datasets, we derive 46 extracted attributes of two types from 9 original attributes. First, we have the extracted attributes that are parts of an original attribute, like an original attribute that is composed of 28 triplets of RGB (Red Green Blue) color values that we split into 28 single attributes. Then, we have the extracted attributes that are derived from an original attribute, like the number of plugins derived from the list of plugins. The extracted attributes do not increase the distinctiveness as they come from an original attribute, and they are, at most, as distinctive as their original attribute. However, the extracted attributes can offer a higher stability than their original attribute, as the latter is impacted by any little change among the extracted attributes. For example, if exactly one of the 28 RGB values changes between two fingerprint observations, the original attribute is counted as having changed, but only one of the extracted attributes will be.

Table 3.3: The number of initial raw entries, together with the number of entries having a bad format, coming from robots, having the cookies disabled, or having a duplicated fingerprint, which are filtered out. The proportion of browsers observed with an interleaved fingerprint is also shown.

	Gen. Aud.	Intranet	University	Enrolled
Raw entries	8,205,416	43,493	2,793	2,778
Bad format	769	28,991 ¹¹	0	21
Robot entries	53,251	0	0	0
Cookies disabled	18,591	6	0	0
Duplicated entries	2,420,217	3,398	2,300	887
Remaining entries	5,714,738	11,098	493	1,870
Remaining fingerprints	4,145,408	9,422	268	1,787
Browsers with interleaved fps	0.106	0.307	0.318	0.023

3.4 Acceptability Survey

In this section, we describe the population of the respondents to the acceptability survey published by *Le Lab* on its experimentation platform during the enrolled experiment. We designed the survey in collaboration with *Le Lab* to measure authentication factor usage, the interest in the proposed authentication mechanism, and the user satisfaction after testing it. We discuss the preliminary questions that were asked to the respondents about their authentication habits. We focus here on the answers to these preliminary questions, and let the results about the acceptability of the authentication mechanism that includes browser fingerprints for Section 4.2.6. Appendix E lists the questions of the survey and the answers proposed to the participants.

3.4.1 Respondents Population

The respondents of the survey are the members of the experimentation platform of *Le Lab* on the enrolled experiment that accepted to participate in the survey. The survey was posted when the enrolled experiment began on the December 9,

¹¹The number of entries showing a bad format is very high for the intranet dataset. This is mainly due to the fingerprinting probe that was updated through the experiment. As a result, the fingerprints of the raw intranet dataset are in different formats. We choose to hold only the fingerprints of the latest format, as they are the ones that embark the most attributes.

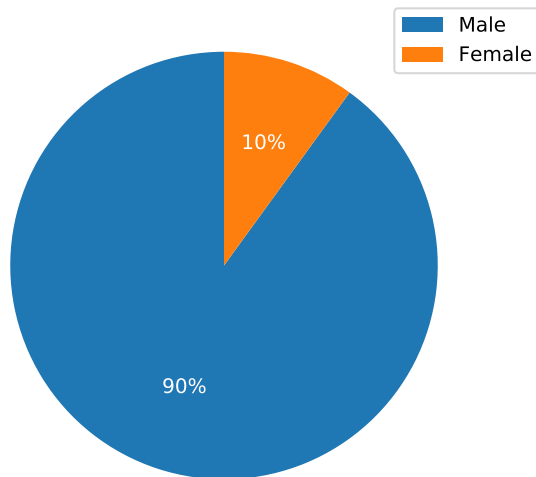


Figure 3.1: The gender reported by the respondents.

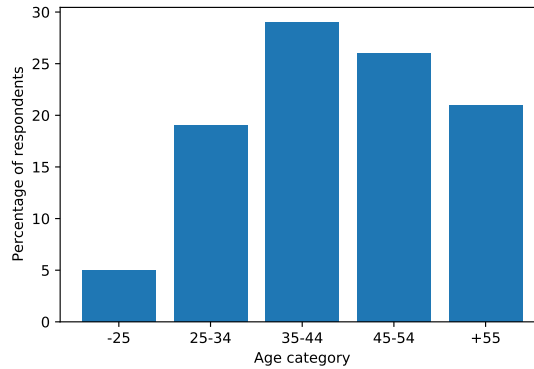


Figure 3.2: The age reported by the respondents.

2019. The respondents were required to have either a smartphone, a mobile phone, a tablet, or a computer. We have a total of 1,285 respondents, among which 682 (53%) also participated in the enrolled experiment.

Figure 3.1 presents the gender reported by the respondents, Figure 3.2 displays the age that they reported, Figure 3.3 shows the number of people in the household that they reported, Figure 3.5 displays their socio-professional categories, Figure 3.4 presents the technical skill level that they reported, and Figure 3.6 shows the types of the devices that they use. Most respondents are men, technically skilled, and have a higher managerial or professional occupation. This can be explained by experimenters of *Le Lab* occupying higher professional occupations in information technologies industry. Moreover, this population may be more interested in participating in an experiment about testing new authentication technologies.

3.4.2 Respondents Authentication Habits

Before surveying the experimenters about their opinion on our authentication mechanism that leverages browser fingerprinting, we first surveyed their current authentication habits.

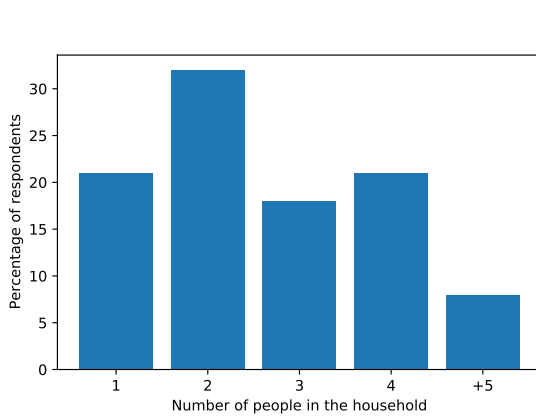


Figure 3.3: The number of people in the household reported by the respondents.

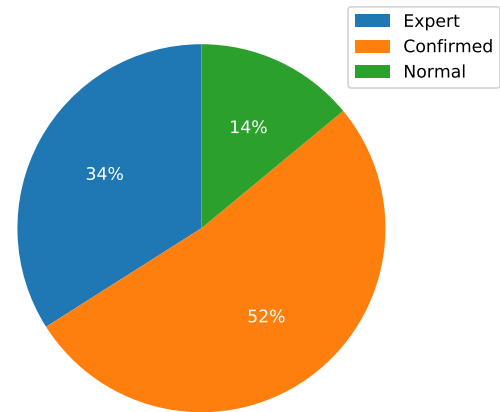


Figure 3.4: The level of technical skill reported by the respondents.

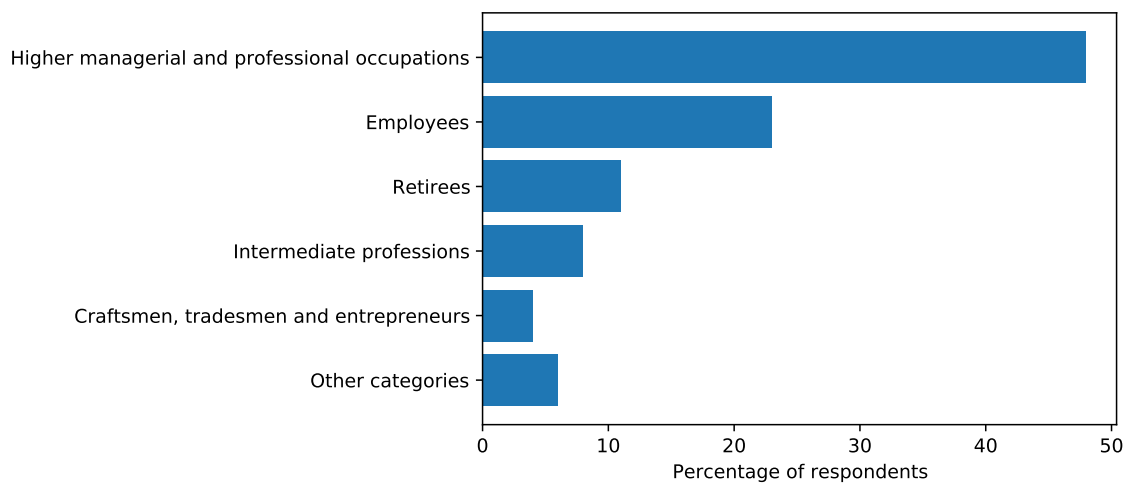


Figure 3.5: The socio-professional categories of the respondents.

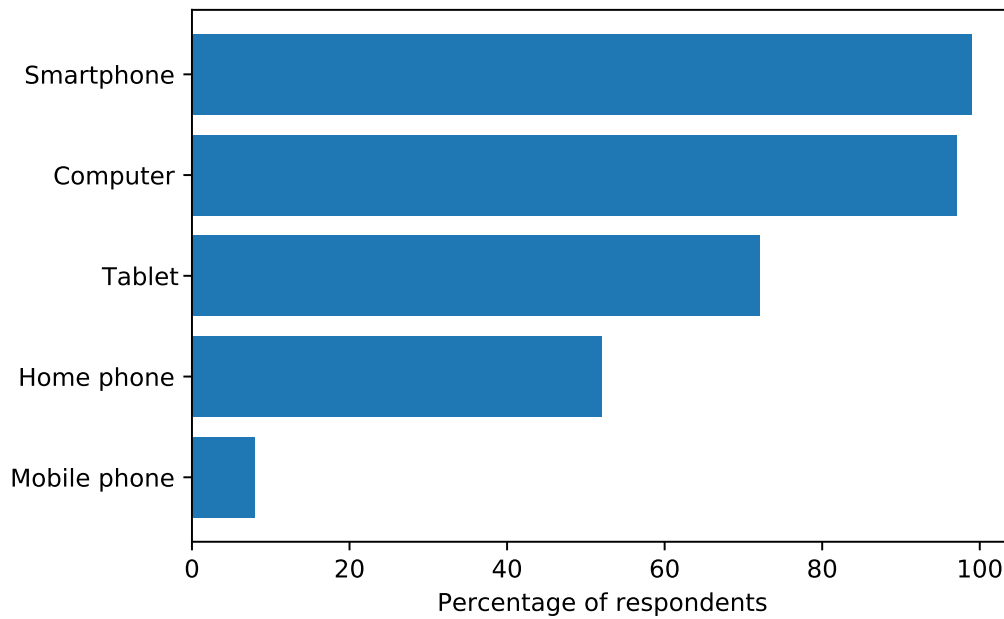


Figure 3.6: The type of the devices that the respondents use. Mobile phone does not include the smartphones.

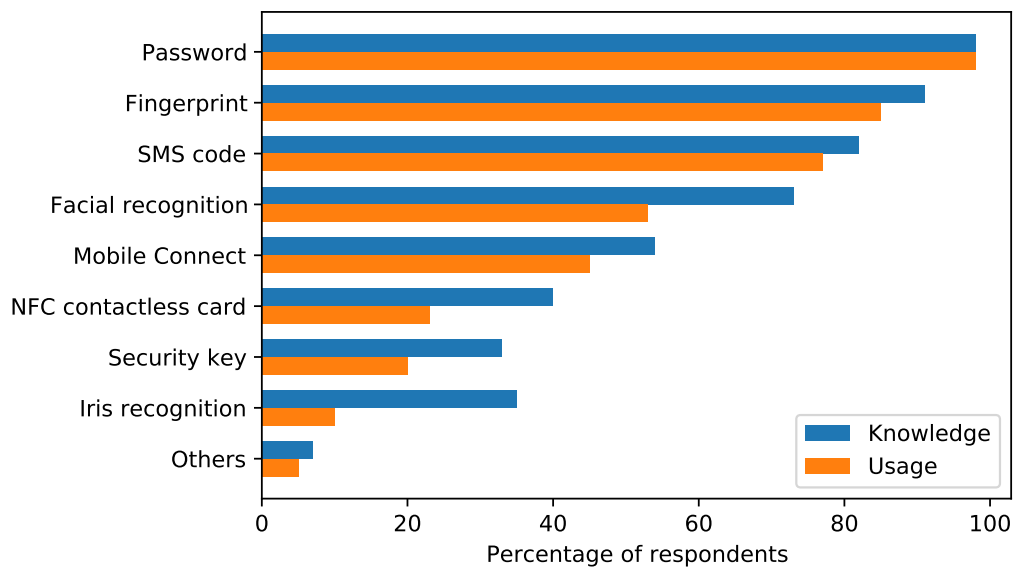


Figure 3.7: The authentication methods that the respondents know and use. Fingerprint here refers to the biometric fingerprints.

Known and Used Authentication Methods

We began by asking the participants which authentication methods do they know and which ones do they use. Figure 3.7 shows the authentication methods that the respondents reported knowing and using. The password, the fingerprint, and the SMS code are the three most known authentication methods with a respective knowledge rate of 98%, 91%, and 82%. They are also the three most used methods with a respective usage rate of 98%, 85%, and 77%. The gap between the respondents that reported knowing them, and those that reported using them is smaller compared to the other methods. These three authentication methods are typically available on smartphones, notably the fingerprint and the SMS code. As the respondents have a high usage rate of smartphones, this can explain the high usage rate of these authentication methods. The remaining authentication methods are typically physical authentication factors (security key, NFC contactless card), biometric authentication factors (facial and iris recognition), Mobile Connect, and authentication methods that were rarely cited by the respondents (less than 8% of knowledge and usage rate). Mobile Connect [90] is a standard of the Global System for Mobile Communications Association (GSMA) to share user data between a service provider and the user's mobile network operator. The user data can be used for authentication [150]. The authentication methods that are rarely cited by the respondents include Google Authenticator¹², password managers¹³, certificates, and one-time passwords.

Satisfaction with Current Authentication Methods

After surveying the authentication methods that the participants know and use, we asked them if they were satisfied with these methods. Figure 3.8 displays the satisfaction of the respondents with the current authentication methods, on a scale from 0 to 10, with 0 being the greatest dissatisfaction and 10 the greatest satisfaction. We measure the satisfaction of the respondents by the Net Promoter Score (NPS) [107] that is calculated using the following methodology. It consists

¹²<https://support.google.com/accounts/answer/1066447?hl=en>

¹³We stress that password managers are not an authentication method, but a helping tool to store passwords which are the actual authentication method.

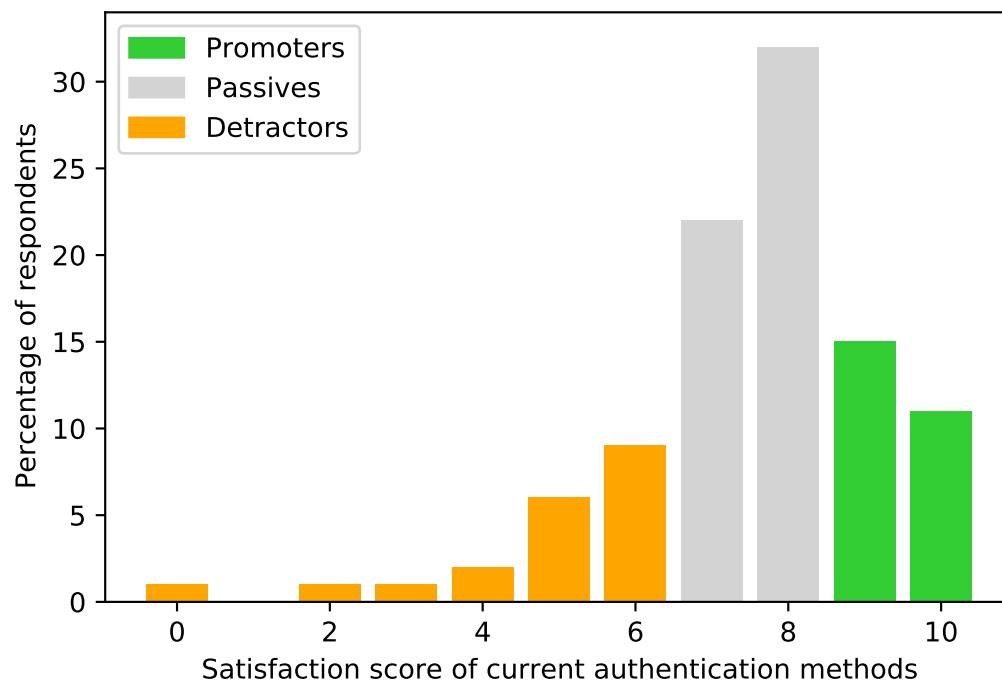


Figure 3.8: The satisfaction of the respondents with the current authentication methods. The scores are divided between the detractors, the passives, and the promoters according to the Net Promoter Score methodology.

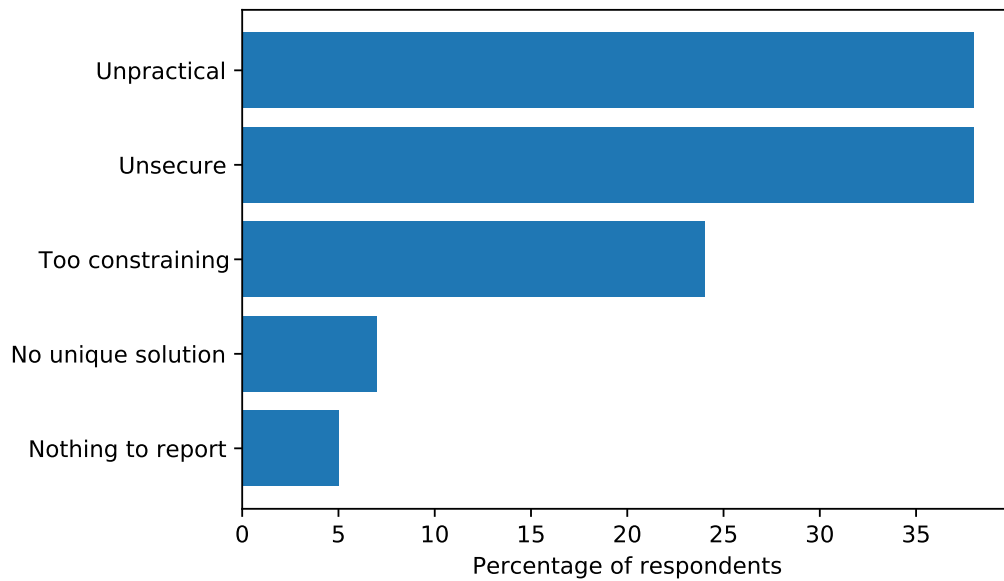


Figure 3.9: The reasons reported by the 261 unsatisfied respondents (20% of the respondents) about why they are dissatisfied with the current authentication methods.

into splitting the scores into three categories to identify three types of respondents: the promoters (scores of 9 and 10), the passives (scores of 7 and 8), and the detractors (scores of 6 or lower). The NPS is the difference between the percentage of promoters and the percentage of detractors. The NPS is then comprised between -100 when all the respondents are detractors, and 100 when they are all promoters. The greater the NPS is, the more the respondents are satisfied. When asked about their satisfaction with the current authentication methods, 26% of the respondents are promoters, 20% of them are detractors, and 54% of them are passives. The resulting NPS is then 6, which is a low positive value.

Reasons for Dissatisfaction

We asked the 261 respondents (20% of the respondents) that are dissatisfied with the current authentication methods (i.e., that reported a score between 0 and 6) the reasons for this dissatisfaction. Figure 3.9 presents the reasons why the unsatisfied respondents are dissatisfied with the current authentication methods. Among the unsatisfied respondents, 38% found that the current authentication

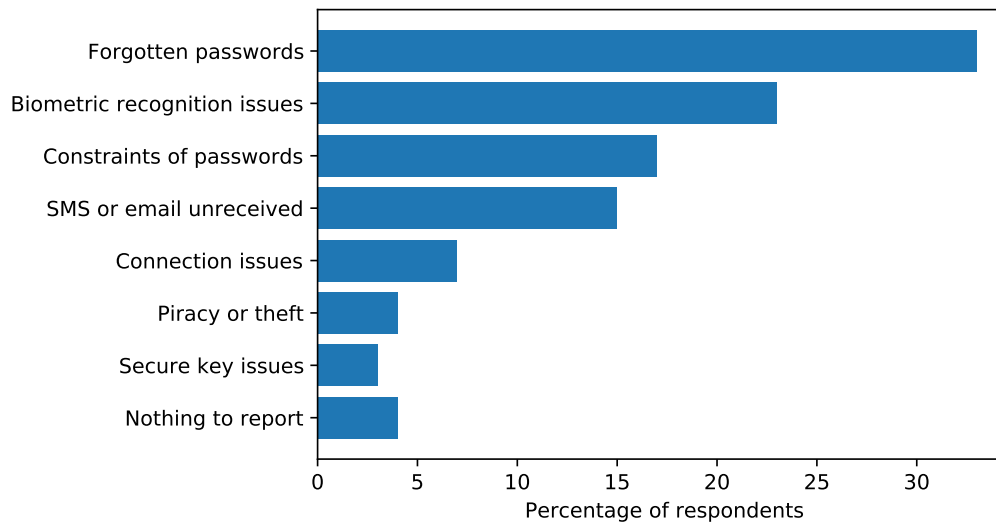


Figure 3.10: The difficulties regarding the current authentication methods that the 489 respondents (38% of the respondents) have already encountered.

methods are unpractical. They mostly complained about the need to remember numerous and complex passwords. They stressed that these constraints were enhanced by the password creation policies that differ between websites. As many unsatisfied respondents (38%) found that the current authentication methods were unsecure, mostly due to the lack of two-factor authentication and the risk of piracy that remained. For 24% of the unsatisfied respondents, the current authentication methods were too constraining. They mostly complained about the constraints on password policies, and authentication mechanisms being too constraining. Unsatisfied respondents (7%) also reported the lack of a unique authentication solution as a dissatisfaction factor. Finally, 5% of the respondents had no reasons to report.

Difficulties with Current Authentication Methods

We asked the respondents if they already encountered difficulties regarding the current authentication methods. To this question, 489 respondents (38% of the respondents) answered positively. Figure 3.10 summarizes the difficulties that these respondents encountered. Among the respondents that encountered difficulties, 33% have already had difficulties to remember passwords, resulting in errors in their entry or in their forgetfulness. When processing biometric authentication,

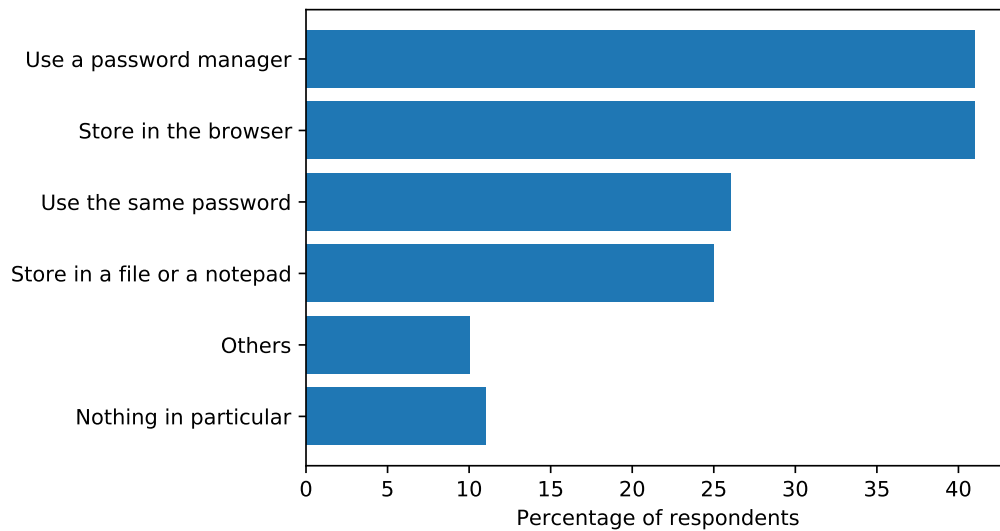


Figure 3.11: The ways the respondents retrieve their passwords.

23% of the respondents that encountered difficulties have already been unrecognized, resulting in longer connection time or in the impossibility to connect. Other difficulties with passwords are the number of passwords to remember, and the requirements regarding their creation, that 17% of the respondents that encountered difficulties have faced. The remaining difficulties are the non-reception of an SMS or an email (15% of the respondents that encountered difficulties), network or connection issues (7%), piracy or theft (4%), and secure key issues (3%). Finally, 4% of the respondents that encountered difficulties answered positively but reported no difficulties.

How Respondents Cope with Memorizing Passwords

As memorizing passwords is an issue that website users face, we asked the participants how do they cope with this issue. Figure 3.11 presents the ways the respondents retrieve their passwords. Among the respondents, 41% reported using a password manager, and as many reported relying on the default password management functionality of their web browser. These are good practices, although using a dedicated password manager is better than relying on the browser, as their functionalities are more secure (e.g., passwords are stored encrypted and are removed from memory after use). A non-negligible number of respondents (26%) admitted

to always use the same password to remember it more easily. Password reuse is known as a bad practice that users still rely on [44, 135, 134, 172], which eases the task for a pirate to illegitimately access their accounts [220, 210, 84]. Among the respondents, 25% reported storing their passwords in a file or a notepad. Although this is both easy and quick, it is a bad practice as any person having access to the files can retrieve the password. A better alternative is the use of a password manager. More physical ways to retrieve passwords have been reported by 10% of the respondents. They include memorizing passwords and writing them down on a piece of paper. These rates are similar to what was reported by Das et al. [44], being that 13% of the respondents to their survey reported storing their passwords on their computer, and 13% of them wrote their passwords down. Finally, 11% of the respondents reported doing nothing in particular to retrieve their passwords.

The respondents seem to have better practices than common web users, which can be explained by their technical skills. Indeed, their password reuse rate is of 26%, against 31.44% [84] and 51% [44] for previous studies. Moreover, they tend to use more password managers, with a use rate of 41% against 6% [44] and 12.34% [172] for previous studies.

3.5 Conclusion

To empirically study browser fingerprints, we designed a fingerprinting probe that embark more than a hundred attributes in the form of a JavaScript script. Using this fingerprinting probe, we performed four experiments that consisted into collecting the browser fingerprints of four browser populations. These populations are the general public users that visited one of the most popular French website, the employees that visited the internal website of a company, the students and university staff that navigated on the standardized computers of a university, together with the visitors and experimenters that participated in the testing of an authentication mechanism that leverages browser fingerprints. The fingerprint datasets obtained from these experiments contained irrelevant data, hence we passed them through preprocessing steps before any analysis. These steps are the cleaning of the bogus and bot-related data, the resynchronization of the unique identifiers, the fingerprint deduplication, and the derivation of the extracted attributes. Finally,

to study the acceptability of browser fingerprints for authentication, we conducted a survey with the help of the *Le Lab* experimentation platform. We describe the respondent population, their authentication habits, and their opinion about current authentication methods. We let the results about the acceptability of browser fingerprinting for authentication for the next chapter.

Chapter 4

Browser Fingerprints for Authentication

Most studies about browser fingerprinting for authentication concentrate on the design of the authentication mechanism [212, 177, 79, 203, 124, 184] and the large-scale empirical studies on browser fingerprints focus on their effectiveness as a web tracking tool [56, 127, 81, 178]. Moreover, the distinctiveness of the browser fingerprints that can be achieved when considering a wide-surface of attributes on a large population is, to the best of our knowledge, unknown. The studies that analyze browser fingerprints in a large-scale (more than 100,000 fingerprints) consider fewer than thirty attributes [56, 127, 214, 81] and usually focus on a single aspect of the fingerprints (e.g., their distinctiveness, their stability). This underestimates the distinctiveness of the fingerprints (e.g., [81] reports a rate of 33.6% of unique fingerprints), as it increases the chances for browsers to share the same fingerprint. All this whereas more than a hundred attributes are accessible. The current knowledge about the hundreds of accessible attributes (e.g., their stability, their collection time, their correlation) is, to the best of our knowledge, also incomplete.

In this chapter, we conduct a large-scale data-centric empirical study of the fundamental properties of browser fingerprints when used as an additional web authentication factor. We base our findings on the analysis of our four fingerprint datasets, that include a dataset of 4,145,408 fingerprints composed of 216 base

attributes plus 46 extracted ones. We formalize, and assess on our dataset, the properties necessary for paving the way to elaborate browser fingerprinting authentication mechanisms. We make the link between the digital fingerprints that distinguish browsers, and the biological fingerprints that distinguish Humans, to evaluate browser fingerprints according to properties inspired by biometric authentication factors [139, 231, 75]. The properties aim at characterizing the adequacy and the practicability of browser fingerprints, independently of their use within future authentication mechanisms. To comprehend the obtained results on the complete fingerprints, we include an in-depth study of the contribution of the attributes to the fingerprint properties.

This chapter is organized as follows. In Section 4.1, we formalize the properties for evaluating the browser fingerprints. Section 4.2 provides the results of the analysis of our four fingerprint datasets according to these properties. We break down the analysis to the attributes in Section 4.3, discuss their correlation, and focus on the properties of the dynamic attributes. Finally, Section 4.4 concludes this chapter.

4.1 Authentication Factor Properties

Biometric authentication factors and browser fingerprints share strong similarities. Both work by extracting features from a unique entity, which is a person for the former and a browser for the latter, that can be used for identification or for authentication. Although the entity is unique, the extracted features are a digital representation of the entity, that can lead to imperfections (e.g., the biometric fingerprints of two different persons can show similar representations [170, 55]). Previous studies [139, 231, 75] identified the properties for a biometric characteristic to be *usable*¹ as an authentication factor, and the additional properties for a biometric authentication scheme to be *practical*. We evaluate browser fingerprints according to these properties, because of their similarity with biometric authentication factors. In this section, we list these properties, formalize how to measure some, and explain why the others are not addressed in this thesis.

¹Here, *usable* refers to the adequacy of the characteristic to be used for authentication, rather than the ease of use by users.

4.1.1 Studied Properties

The four properties needed for a biometric characteristic to be *usable* [139] as an authentication factor are the following:

- *Universality*: the characteristic should be present in everyone.
- *Distinctiveness*: two distinct persons should have different characteristics.
- *Permanence*: the same person should have the same characteristic over time. We rather use the term *stability*.
- *Collectibility*: the characteristic should be collectible and measurable.

The three properties that a biometric authentication scheme requires to be *practical* [139] are the following:

- *Performance*: the scheme should be accurate, consume few resources, and be robust against environmental changes.
- *Acceptability*: the users should accept to use the scheme in their daily lives.
- *Circumvention*: it should be difficult for an attacker to deceive the scheme.

The properties that we study are the *distinctiveness*, the *stability*, and the *performance*. We consider that the *universality* and the *collectibility* are satisfied, as the HTTP headers that are automatically sent by browsers constitute a fingerprint. The major browsers embark a JavaScript engine to run JavaScript client-side scripts, and few users disable the execution of these scripts². However, we stress that a loss of distinctiveness occurs when no JavaScript attribute is accessible.

About the *circumvention*, we refer the reader to Laperdrix et al. [124] that analyzed the security of an authentication mechanism based on browser fingerprints that implements a challenge-response mechanism that relies on HTML5 canvases³. They identify the attacks that can be executed on such mechanism and the reach of each attack. They show that such mechanism is not sensible to replay attacks, that

²<https://deliberatedigital.com/blockmetry/javascript-disabled>

³Although they focus on HTML5 canvases, we emphasize that several dynamic attributes can be combined to design such challenge-response mechanism.

the number of possible challenges is sufficiently large (more than 2^{154} possibilities) to thwart preplay attacks, and that the number of possible values is sufficiently large to avoid blind guessing attacks. They discuss guessing attacks which can reach 9.9% of the browser population for a strong attacker. They also explain that configuration recovery attacks are efficient as grouping the browsers according to their complete fingerprint leads to a unique canvas rendering for around 60% of the groups. However, they stress that such attacks require extra works for the attackers. Although they acknowledge that such mechanism is fallible to relay attacks, we emphasize that two-factor authentication mechanisms are also sensible to relay attacks^{4,5,6}.

4.1.2 Distinctiveness

To satisfy the *distinctiveness*, the browser fingerprints should distinguish browsers. The two extreme cases are every browser sharing the same fingerprint, which makes them indistinguishable from each other, and no two browsers sharing the same fingerprint, making every browser distinguishable. The distinctiveness falls between these extremes, depending on the attributes and the browser population. We consider the use of browser fingerprinting as an additional authentication factor. Hence, we do not require a perfect distinctiveness, as it is used in combination with other authentication factors to improve the overall security.

The dataset entries are composed of a fingerprint, the source browser, and the time of collection in the form of a Unix timestamp in milliseconds. We denote B the domain of browsers, F the domain of the fingerprints, and T the domain of the timestamps. The fingerprint dataset is denoted D , and is formalized as:

$$D = \{(f, b, t) \mid f \in F, b \in B, t \in T\} \quad (4.1)$$

We use the size of the browser anonymity sets to quantify the distinctiveness, as the browsers that belong to the same anonymity set are indistinguishable. We

⁴<https://blog.duszynski.eu/phishing-ng-bypassing-2fa-with-modlishka>

⁵<https://techcommunity.microsoft.com/t5/azure-active-directory-identity/all-your-creds-are-belong-to-us/ba-p/855124>

⁶<https://www.csoonline.com/article/3399858/phishing-attacks-that-bypass-2-factor-authentication-are-now-easier-to-execute.html>

denote $\mathcal{B}(f, D)$ the function that gives the browsers that provided the fingerprint f in the dataset D . It is formalized as:

$$\mathcal{B}(f, D) = \{b \in B \mid \forall (g, b, t) \in D, f = g\} \quad (4.2)$$

We denote $\mathcal{A}(\epsilon, D)$ the function that provides the fingerprints that have an anonymity set of size ϵ (i.e., that are shared by ϵ browsers) in the dataset D . It is formalized as:

$$\mathcal{A}(\epsilon, D) = \{f \in F \mid \text{card}(\mathcal{B}(f, D)) = \epsilon\} \quad (4.3)$$

A common measure of the fingerprint distinctiveness is the *unicity rate* [56, 127, 81], which is the proportion of the fingerprints that were observed for one browser only. We denote $\mathcal{U}(D)$ the unicity rate of the dataset D , which is formalized as:

$$\mathcal{U}(D) = \frac{\text{card}(\mathcal{A}(1, D))}{\text{card}(D)} \quad (4.4)$$

Previous studies measured the anonymity set sizes on the whole dataset [56, 127, 81]. We measure the anonymity set sizes on the fingerprints currently in use by each browser, and not on their whole history. A browser that runs in a fancy web environment (e.g., having a custom font) and that has several fingerprints in the dataset (e.g., fifty) would bloat the proportion of unique fingerprints. Indeed, fifty fingerprints would be unique, whereas they all come from a single browser. Moreover, two browsers that share the same fingerprint but on different time windows (e.g., a browser is updated before the other) would fall into the same anonymity set. A verifier that observes the fingerprint of these two browsers would still be able to distinguish them.

We evaluate the anonymity set sizes on the time-partitioned datasets composed of the last fingerprint seen for each browser at a given time. Let $\mathcal{S}_\tau(D)$ be the time-partitioned dataset originating from D that represents the state of the fingerprint of each browser after τ days. With t_τ the last timestamp of this day, we have:

$$\mathcal{S}_\tau(D) = \{(f_i, b_j, t_k) \in D \mid \forall (f_p, b_q, t_r) \in D, b_j = b_q, t_r \leq t_k \leq t_\tau\} \quad (4.5)$$

4.1.3 Stability

To satisfy the *stability*, the fingerprint of a browser should stay sufficiently similar between two observations to be recognizable. Browser fingerprints have the particularity of evolving through time, due to changes in the web environment like a software update or a user configuration. We measure the stability by the average similarity between the consecutive fingerprints of the browsers, according to the elapsed time between their observation. The two extreme cases are every browser holding the same fingerprint through its life, and the fingerprint changing completely with each observation. A lack of stability makes it harder to recognize the fingerprint of a browser between two observations.

We denote $\mathcal{C}(\Delta, D)$ the function that provides the pairs of consecutive fingerprints of D that are separated by a time-lapse comprised in the Δ time range. It is formalized as:

$$\begin{aligned} \mathcal{C}(\Delta, D) = \{ & (f_i, f_p) \mid \forall ((f_i, b_j, t_k), (f_p, b_q, t_r)) \in D^2, b_j = b_q, t_k < t_r, (t_r - t_k) \in \Delta, \\ & \nexists (f_c, b_d, t_e) \in D, b_d = b_j, f_c \neq f_i, f_c \neq f_p, t_k < t_e < t_r \} \end{aligned} \quad (4.6)$$

We consider the Kronecker delta $\delta(x, y)$ which gives 1 if x equals y and 0 otherwise. We consider the set Ω of the n used attributes. We denote $f[a]$ the value taken by the attribute a for the fingerprint f . Let $\text{sim}(f, g)$ be a simple similarity function between the fingerprints f and g , which is formalized as:

$$\text{sim}(f, g) = \frac{1}{n} \sum_{a \in \Omega} \delta(f[a], g[a]) \quad (4.7)$$

We define the function $\text{avsim}(\Delta, D)$ that provides the average similarity between the pairs of consecutive fingerprints⁷, for a given time range Δ and a dataset D . It is formalized as:

$$\text{avsim}(\Delta, D) = \frac{\sum_{(f,g) \in \mathcal{C}(\Delta, D)} \text{sim}(f, g)}{\text{card}(\mathcal{C}(\Delta, D))} \quad (4.8)$$

⁷More complex similarity functions can be designed (e.g., relying on machine learning techniques [214]). We refer to Appendix D for a discussion about a verification mechanism that authorizes small differences when comparing the attributes. The similarity function that we use to evaluate the stability provides a lower bound as the identical attributes should also match when using more advanced similarity functions.

4.1.4 Performance

Browser fingerprinting can easily be deployed by adding a script on the authentication page, and by preparing the servers to handle the reception, the storage, and the verification of the fingerprints. The users solely rely on their regular web browser, and do not have to run any dedicated application, nor possess specific hardware, nor undergo a configuration step. The main additional load is on the supplementary consumption of memory and time resources. The web environments also differ between device types (e.g., mobile browsers have more limited functionalities than desktop browsers) and through time (e.g., modern browsers differ from the ones from ten years ago). These differences impact the effectiveness of browser fingerprinting. Finally, as the fingerprints evolve through time, their verification is badly impacted by this particularity. We consider four aspects of the *performance* of browser fingerprints for web authentication: their *collection time*, their *size* in memory, the *loss of effectiveness* between different device types, and the *accuracy* of a simple illustrative verification mechanism.

Collection Time

The browser fingerprints can be solely composed of passive attributes (e.g., HTTP headers) that are transmitted along with the communications with the server. In this case, the fingerprints are collected without the user perceiving any collection time, but major attributes are set aside. The client-side properties collected through JavaScript provide more distinctive attributes, at the cost of an additional collection time. We measure the collection time of the fingerprints considering only the JavaScript attributes, and ignore the HTTP headers that are transmitted passively.

Size

Browser fingerprinting consumes memory resources on the clients during the buffering of the fingerprints, on the wires during their sending, and on the servers during their storage. The memory consumption depends on the storage format of the fingerprints. For example, a canvas can be stored as an image encoded in a base64 string, or shorter as a hash. A trade-off has to be done between the quantity of

information and the memory consumption. The less memory-consuming choice is to store the complete fingerprint as a single hash. However, the fingerprints evolve through time – even more when they are composed of many attributes – which results in the use of a hash of the complete fingerprint being impractical. Due to the unspecified size of the attributes (e.g., the specification of the `User-Agent` HTTP header does not define a size limit [186]), we measure their size on a fingerprint dataset.

Loss of Effectiveness

The loss of effectiveness is the loss of stability, of distinctiveness, or of performance of the fingerprints. It can occur either for a group of browsers (e.g., mobile browsers), or resulting from changes brought to web technologies. First, previous works showed differences in the properties of the fingerprints coming from mobile and desktop devices [204, 127, 81], notably a limited distinctiveness for the mobile browsers. Following these findings, we compare the properties shown by the mobile and by the desktop browsers. Second, browser fingerprinting is closely dependent on the evolution of web technologies. As new technologies are integrated into browsers, new attributes are accessible, and conversely for removal. Similarly, functionality updates can lead to a change in the fingerprint properties. For example, Kurtz et al. [121] detected an iOS update by the sudden instability of an attribute that provides the iOS version. Following their finding, we verify whether the evolution of web technologies provokes major losses in the properties of the fingerprints.

Accuracy of a Simple Verification Mechanism

We evaluate the accuracy of a simple illustrative verification mechanism under the evolution of fingerprints. This mechanism counts the identical attributes between the presented and the stored fingerprint, and considers the evolution legitimate if this number is above a threshold Θ . The simplicity of this mechanism gives us an idea of the accuracy that can be easily achieved, without having to engineer more complex rules. More elaborate mechanisms can obviously be designed (see Appendix D).

4.1.5 Acceptability

We evaluate the acceptability of the usage of browser fingerprinting for authentication with the help of the *Le Lab* experimentation platform on the enrolled experiment. *Le Lab* conducted a survey about the test authentication mechanism that integrates browser fingerprints as an authentication factor. The questions were asked in two phases. First, the participants were shown a brief description of the test authentication mechanism (available in Appendix E) and were asked their opinion about the mechanism according to the description. Then, the participants were invited to test the authentication mechanism and give their opinion on their experience.

4.2 Evaluation of Browser Fingerprints Properties

In this section, we evaluate the browser fingerprints of our four datasets according to the distinctiveness, the stability, the performance, and the acceptability properties. We present here the results on the complete fingerprints, and let Section 4.3 provide insights on the contribution of the attributes to each property. We use the terms of mobile fingerprints and desktop fingerprints to refer to the fingerprints coming respectively from the mobile and from the desktop browsers.

4.2.1 Distinctiveness

In this section, we provide the results on the distinctiveness of the fingerprints of our four datasets. Figure 4.1 presents the size of the anonymity sets alongside the frequency of browser arrival for the time-partitioned datasets obtained from the four datasets. Figure 4.2 presents the unicity rate through the time-partitioned datasets obtained from the general audience and enrolled datasets, for the overall, the mobile, and the desktop browsers. The time-partitioned datasets are designed so that each browser has the last fingerprint observed at the end of the τ -th day. The overall fingerprints comprise those collected from desktop and from mobile browsers, but also those of tablets, consoles, and smart TVs.

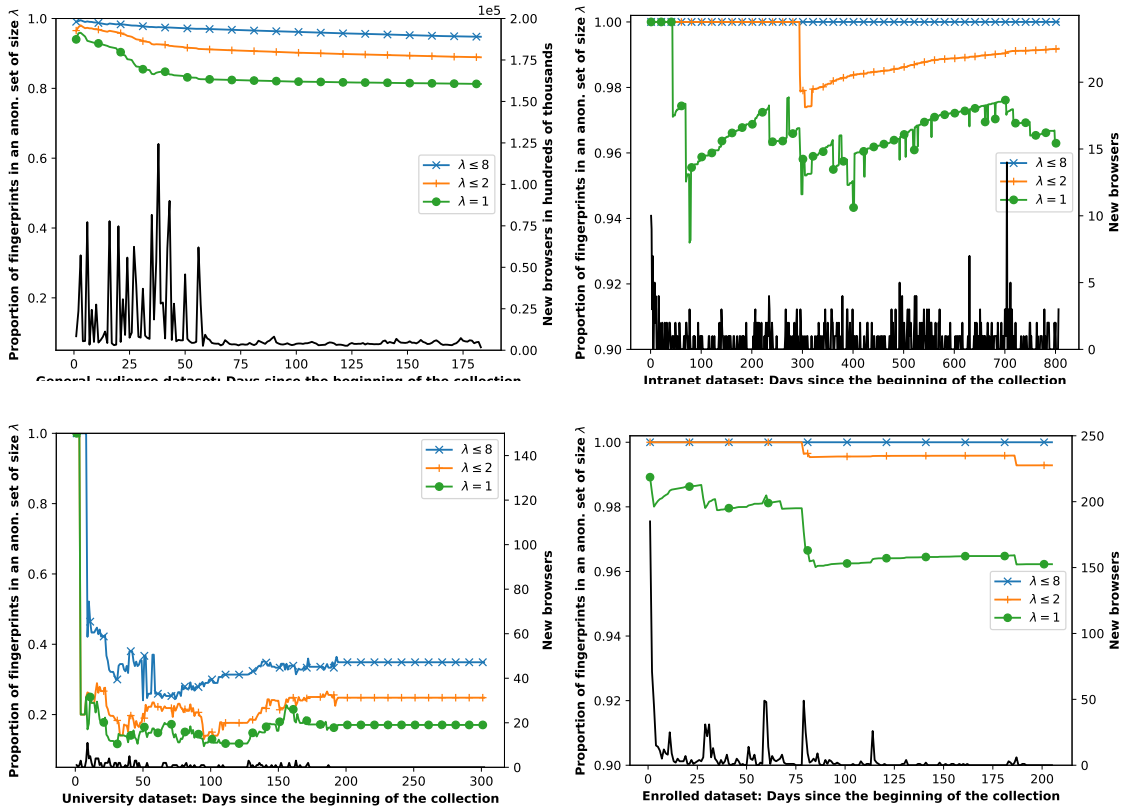


Figure 4.1: Anonymity set sizes and frequency of browser arrivals through the time-partitioned datasets obtained for each of the four datasets. The new browsers of the general audience dataset are displayed in hundreds of thousands.

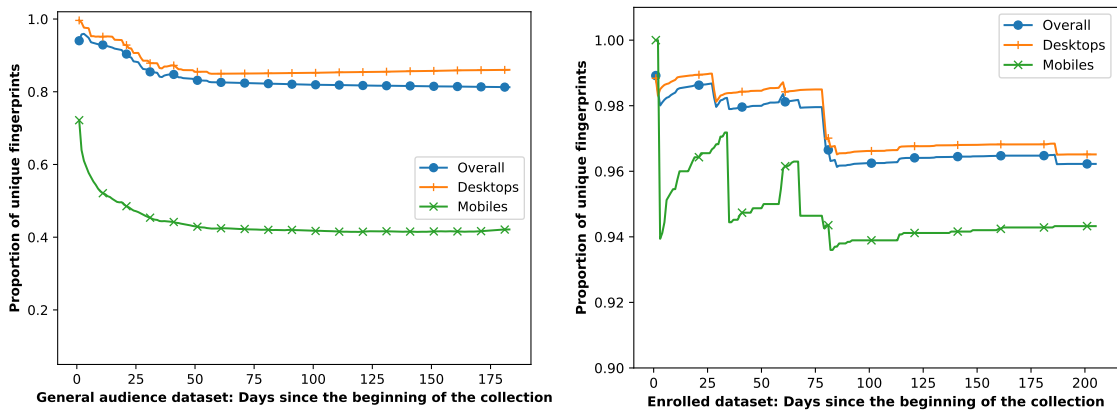


Figure 4.2: Unicity rate for the overall, the mobile, and the desktop browsers, through the time-partitioned datasets obtained from the general audience and enrolled datasets.

Summary of the Distinctiveness Results

For the general audience, intranet, and enrolled datasets, more than 81.3% of the fingerprints are unique on the long run, considering the time-partitioned datasets. Moreover, more than 94.7% of the fingerprints of these three datasets are shared by 8 browsers of fewer. The university dataset shows a lower distinctiveness due to the standardized browser population. The unicity rate of the time-partitioned datasets goes down to 11%, and stabilizes at 17.1% on the long run. Although the browsers of the university dataset are standardized, some are still distinguishable by their fingerprint.

The unicity rate of the time-partitioned datasets is lower than the unicity rate of the complete datasets. The loss is lower than 5.18% of the unicity rate of the complete datasets, at the exception of the university dataset for which it is 53.19%. This is due to browsers having multiple unique fingerprints in the complete dataset, which typically occurs when a browser having a unique web environment is fingerprinted multiple times. Considering the time-partitioned datasets removes this over-counting effect. This effect is enhanced for the university due to few browsers having a unique web environment.

New browsers join the experiments over time, with several days for which we observe a high number of arrival due to external events (e.g., major news for the general audience dataset, or advertisement on social medias for the enrolled dataset). The anonymity sets are impacted by the arrival of new browsers in the datasets. Indeed, the more browsers there are in the dataset, the greater the chance of collision. The COVID-19 pandemic only impacted the university dataset. The containment that occurred in France prevented users to connect to the university computers. As a result, no new browsers joined the university experiment after the 187th day.

The mobile fingerprints of the general audience and enrolled datasets show a lower distinctiveness compared to the desktop fingerprints. For the general audience dataset, the unicity rate of the mobile fingerprints goes down to 41%, whereas the lowest unicity rate of its desktop fingerprints is two times higher (84%). The loss is less significant for the enrolled dataset which has fewer browsers. The enrolled dataset shows a unicity rate of 93.6% for the mobile fingerprints against

96.5% for the desktop fingerprints. We confirm the conclusions of the previous studies [204, 127, 81] about the fingerprints of mobile browsers being less distinctive. However, we remark that the loss is less important for the enrolled dataset than for the general audience dataset which has a larger mobile browser population.

Detailed Analysis of the Distinctiveness of the General Audience Dataset

The fingerprints of the time-partitioned datasets of the general audience dataset have a stable unicity rate of more than 81.3%, and more than 94.7% of them are shared by at most 8 browsers. The anonymity sets grow as more browsers are encountered, due to the higher chances of collision. However, the fingerprints tend to stay in small anonymity sets, as can be seen by the growth of the anonymity sets of size 2 being more important than the growth of the anonymity sets of size 8 or higher. Due to the over-counting effect, the unicity rate of the time-partitioned datasets (81.3%) is lower than the unicity rate of the complete dataset (81.8%). The general audience experiment occurred before the COVID-19 pandemic and was not impacted by the containment that took place in France.

New browsers are encountered continually during the general audience experiment. However, starting from the 60th day, the arrival frequency stabilizes around 5,000 new browsers per day. Before this stabilization, the arrival frequency is variable, and has major spikes that seem to correspond to events that happened in France. These events could lead to more visits, hence explaining these spikes. For example, the spike on the 38th day corresponds to a live political debate on TV, and the spike on the 43rd correlates with the announcement of a cold snap.

The mobile fingerprints are more uniform than the desktop fingerprints. For the time-partitioned datasets, the former shows a unicity rate of approximately 41%, against 84% for the latter. The unicity rate of the desktop fingerprints slightly increases by 1.04 points from the 60th to the 183th day, from 84.99% to 86.03%. On the contrary, the unicity rate of the mobile fingerprints slightly decreases by 0.29 points on the same period, from 42.42% to 42.13%.

Detailed Analysis of the Distinctiveness of the Intranet Dataset

All the fingerprints of the intranet dataset are in anonymity sets of size 5 or lower. In the time-partitioned datasets, no fingerprint is shared by more than two browsers until the 295th day of the experiment. After this day, more than 97.4% of the fingerprints are shared by at most two browsers. On the long run, more than 93.3% of the fingerprints are shared by a single browser. Due to the overcounting effect, the unicity rate of the time-partitioned datasets (93.3%) is lower than the unicity rate of the complete dataset (98.4%). The intranet dataset shows no significant impact from the containment that took place in France due to the COVID-19 pandemic⁸.

When few new browsers are encountered, the size of the anonymity sets tends to decrease. This results in the increase of the proportion of fingerprints in small anonymity sets (e.g., of size one or two). This is caused by the fingerprints of the new browsers falling in small anonymity sets, or by the fingerprints of the browsers already in the population leaving their anonymity set for smaller ones. The latter can be explained by the fingerprints of the same anonymity set evolving differently⁹.

As the experiment is limited to the browsers having access to the internal network of the company, the browser population does not grow as much as for the general audience experiment. Through time, few new browsers are encountered. For the majority of the experiment days, no new browser is encountered, and only 15 days have more than 3 new browsers. They comprise the first days of the experiment, the days around 500 days after the beginning of the experiment, and a spike on the 704th day with 14 new browsers encountered.

⁸The containment took place between March 17 and May 11, 2020. It corresponds to the 701st and the 756th day of the intranet experiment. The employees continued to visit the intranet website from their homes using the VPN of the company. The intranet website helped the employees get news about the company and stay in touch with their colleagues. News about the company and the employees regarding the sanitary context was published on the 704th day of the intranet experiment, which could explain the spike of new browsers.

⁹For example, one fingerprint of a given anonymity set stays identical if the browser has not been fingerprinted for a long time. If another fingerprint of this anonymity set is updated (e.g., the browser was updated and then fingerprinted), then it leaves this anonymity set. Another example is two fingerprints of the same anonymity set being updated differently (i.e., their web environments are modified in different ways).

Detailed Analysis of the Distinctiveness of the University Dataset

Before the 9th day of the university experiment, few fingerprints have been observed, and the anonymity sets are small. From the 9th to the 40th day, more than half of the browsers of the university experiment are encountered. This results in a drastic decrease of the proportion of the anonymity sets of small size, down to a unicity rate of 11%. The university dataset is impacted by the containment that took place in France due to the COVID-19 pandemic¹⁰. After the 187th day, no new browser is encountered, but fingerprints are still collected from the enrolled browsers. Seven days later, the anonymity set sizes stabilize at 34.9% for the sets of 8 fingerprints or fewer, at 24.8% for the sets of 2 fingerprints or less, and at 17.1% for the unique fingerprints. Both the lowest (11%) and the stabilized (17.1%) unicity rates of the time-partitioned datasets are lower than the unicity rate of the complete dataset (23.5%) due to the over-counting effect. Although these results are not surprising due to the standardized browser population, it is interesting to see that the fingerprints are still able to distinguish some browsers.

Detailed Analysis of the Distinctiveness of the Enrolled Dataset

All the fingerprints of the enrolled dataset are shared by at most 4 browsers. The unicity rate of the time-partitioned datasets goes from 98.9% on the first day, to the lowest value of 96.1% on the 85th day, and stabilizes at 96.2% from the 187th day to the end of the experiment. No fingerprint is shared by two browsers until the 79th day. From the 79th day to the 187th day, 99.6% of the fingerprints are shared by two browsers or fewer. Afterward, this rate stabilizes at 99.3%. The enrolled dataset shows no significant impact from the containment that took place in France due to the COVID-19 pandemic¹¹.

¹⁰The containment took place between the 198th and the 253th day of the university experiment. Starting from few days before the containment, no new browser is encountered as no student nor university staff used the computers of the university during the containment. The end of the academic year occurred shortly after the containment was over, which explains why no user enrolled to the experiment after the containment. Fingerprints were still collected from the registered experimenters during the containment. They surely come from the students that were lent a laptop by the university and installed the extension to participate in the experiment.

¹¹The containment took place between the 100th and the 155th day of the enrolled experiment. The experimenters had access to the website during the containment as it was up and publicly accessible. We even observe a spike of 26 new browsers on the 114th day.

Many new browsers are encountered on the first days of the experiment, up to 185 browsers on the first day. Afterward, several spikes of 25 to 50 new browsers occur. As we communicated several times about the experiment, these spikes can be the result of new experimenters coming after having heard of the experiment.

The mobile fingerprints are slightly less distinctive than the desktop fingerprints, but the difference is not as significant as for the general audience dataset. The unicity rate of the mobile fingerprints goes down to 93.6% and stabilizes at 94.3%. The unicity rate of the desktop fingerprints goes down to, and stabilizes at, 96.5%. This is partly explained by the lower browser population for these two groups. The enrolled dataset has a number of browsers up to 3 orders of magnitude lower than the general audience dataset, and fewer browsers reduces the chances of fingerprint collision.

4.2.2 Stability

In this section, we provide the results on the stability of the fingerprints of our four datasets. Figure 4.3 displays the average similarity between the pairs of consecutive fingerprints as a function of the time difference, together with the number of compared pairs, for the four datasets. When mobile browsers are in the dataset, the results are displayed for the overall, the desktop, and the mobile browsers. The ranges Δ are expressed in days, so that day d on the x-axis represents the fingerprints that are separated by $\Delta = [d; d + 1[$ days. Two consecutive fingerprints are necessarily different as we remove the consecutive fingerprints that are duplicated¹² (see Section 3.3.3). The stability results are a lower bound, as the consecutive fingerprints are necessarily different (i.e., their similarity is strictly lower than 1). For each type of browser, we ignore the comparisons of the time ranges that have less than 3 pairs, to have samples of sufficient size without putting too many comparisons aside. We also ignore the bogus comparisons that have a time difference higher than the duration of the experiment. These two sets of comparisons compose the outliers.

¹²Considering the fingerprints (f_1, f_2, f_3) that were collected for a browser and ordered by the time of collection, the set of consecutive fingerprints is $\{(f_1, f_2), (f_2, f_3)\}$.

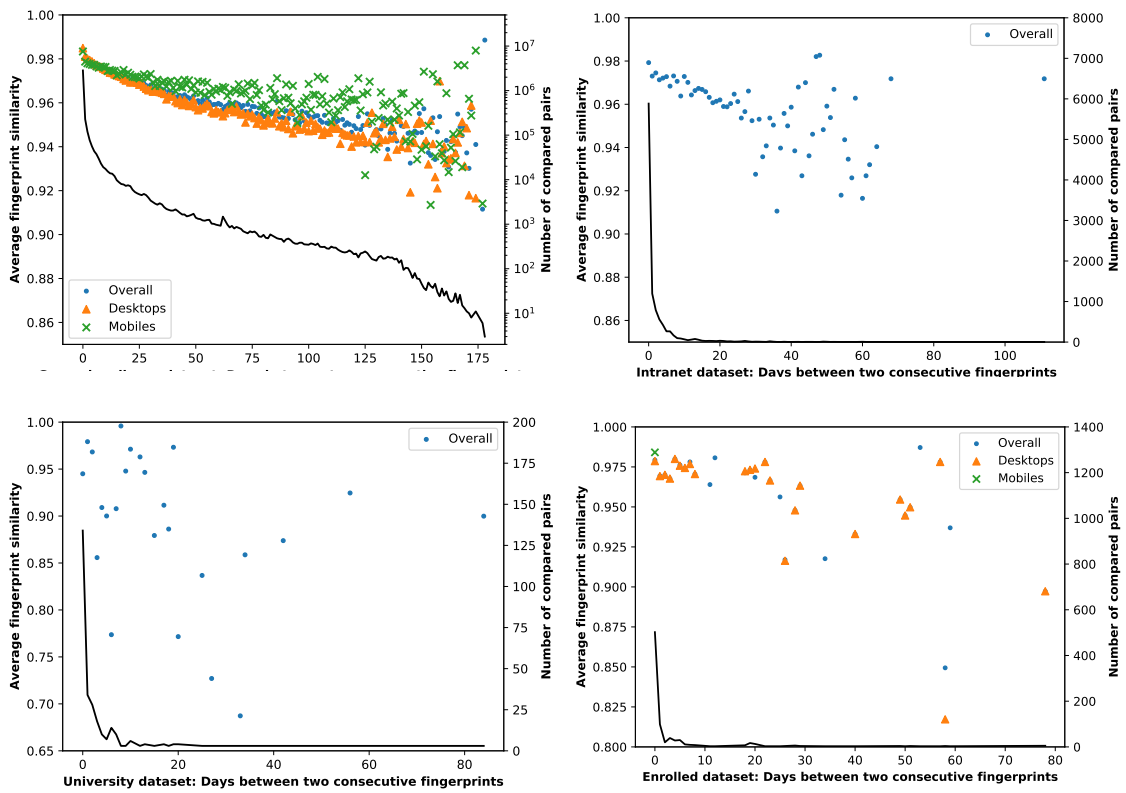


Figure 4.3: Average similarity between the pairs of consecutive fingerprints as a function of the time difference, together with the number of compared pairs.

Summary of the Stability Results

Except for the university dataset, the majority of the consecutive fingerprints have more than 91% of identical attributes on average, even when several months have passed between their observation. For the general audience and intranet datasets, more than 91% of the attributes are expected to stay identical between two observations of the fingerprint of a browser, even when separated by respectively six and three months. For the enrolled dataset, the average fingerprint similarity is also above 91% when considering up to 57 days between two fingerprint observations. However, from 58 to 78 days, the average fingerprint similarity falls down to 84.9% for the overall browsers, and to 81.7% for the desktop browsers. The university dataset shows a more scattered average fingerprint similarity, which is comprised between 68.7% and 99.6% when considering up to 84 days elapsed between two fingerprint observations. During the university experiment, the browser profile of the students and university staff was shared between the computers of the campus. As a result, the browsers from which the fingerprints were collected could run on different web environment (e.g., a Windows and a Linux), resulting in this lower similarity for the university dataset. Only the general audience dataset contains comparisons of mobile fingerprints over several time ranges. For this dataset, the mobile fingerprints tend to be more stable than the desktop fingerprints.

Detailed Analysis of the Stability for the General Audience Dataset

The outliers account for less than 0.01% of each type of browser for the general audience dataset. The results are obtained by comparing 3,725,273 pairs of consecutive fingerprints, that include 2,912,805 pairs of desktop fingerprints, and 594,542 pairs of mobile fingerprints. On average more than 91% of the attributes are expected to stay identical, considering up to 178 days (nearly 6 months) between two fingerprint observations. The mobile fingerprints are generally more stable than the desktop fingerprints as suggests their respective similarity curve.

Detailed Analysis of the Stability for the Intranet Dataset

The outliers account for less than 0.5% of the consecutive fingerprints for the intranet dataset. The results are obtained by comparing 10,562 pairs of consecutive fingerprints. On average, more than 91% of the attributes are expected to stay identical, considering up to 111 days (more than 3 months) between two fingerprint observations.

Detailed Analysis of the Stability for the University Dataset

The outliers account for 16.29% of the consecutive fingerprints for the university dataset. This is due to several time ranges having less than 3 comparisons. This comes from only 61 browsers having multiple fingerprints, and from the long duration of the experiment. The results are obtained by comparing 313 pairs of consecutive fingerprints. The results are more scattered for the university dataset, for which the consecutive fingerprints have up to 84 days between them. The lowest average similarity is at 68.7% for 33 days between two fingerprint observations, and the highest is at 99.6% for 8 days. We have 3 days for which the average similarity is between 70% and 80%, 15 days for which it is between 80% and 95%, and 5 days for which it is between 95% and 99%.

By manually checking the changes of some attributes, we remark that a unique identifier (UID) could have been observed on different operating systems or using different graphical cards. The explanation is the particularity of the probe used during the university experiment, that makes it possible for two different hardware and computer stacks to present the same UID. The probe was integrated into an extension, that was deployed in the default browser of the computers managed by the university. The browser profiles of the users are shared among these computers, allowing a user to connect on any of them to retrieve the same browser configuration. As these profiles contain the cookies, an experimenter that connects on different computers then presents the same UID. The lower stability observed for the university dataset can be linked to this bias. Indeed, among the consecutive fingerprints of this dataset, we observe 76 changes of the `UserAgent` switching between a Windows and a Linux operating system. This results in a lower similarity between the consecutive fingerprints as the attributes can show different values on

the different web environments.

Detailed Analysis of the Stability for the Enrolled Dataset

The enrolled dataset shows more outliers about the consecutive fingerprints: 5.08% for all the browsers¹³, 6.57% for the desktop browsers, and 32.69% for the mobile browsers. This is also due to several time ranges having less than 3 comparisons, as only 275 browsers have multiple fingerprints over the long duration of the experiment. The results are obtained by comparing 847 pairs of consecutive fingerprints, that include 761 pairs of desktop fingerprints, and 52 pairs of mobile fingerprints. The pairs of mobile fingerprints are all separated by less than a day. On average, more than 91% of the attributes are expected to stay identical, considering up to 57 days between two fingerprint observations. However, when looking at the comparisons separated by 58 days, this rate falls down to 84.9% for the overall browsers, and to 81.7% for the desktop browsers. Afterward, it goes up to 93.7% for the overall browsers separated by 59 days, and then goes down to 89.7% for the overall and desktop browsers separated by 78 days.

4.2.3 Collection Time

In this section, we provide the results on the collection time of the fingerprints of our four datasets. Figure 4.4 displays the cumulative distribution of the collection time of the fingerprints in seconds, with the outliers removed, and for the four datasets. When mobile browsers are in the dataset, the results are displayed for the overall, the desktop, and the mobile browsers. We measure the collection time by the difference between two timestamps. The first is recorded at the starting of the script, and the second just before sending the fingerprint. Some values are extremely high and take from several hours to days. They can come from a web page put in background, or accessed after a long time. We limit the results to

¹³The outlier rate is lower for the overall fingerprints than for the mobile and desktop fingerprints. The overall fingerprints comprise the fingerprints of all the device types, and can exceed the required number of consecutive fingerprints for a time range. Hence, the overall fingerprints have more chance to not be considered an outlier. We take the example of a time range that has 2 comparisons of mobile fingerprints and 1 comparison of desktop fingerprints. The overall fingerprints have 3 comparisons and is not deemed an outlier, whereas the two subgroups have fewer and are deemed outliers.

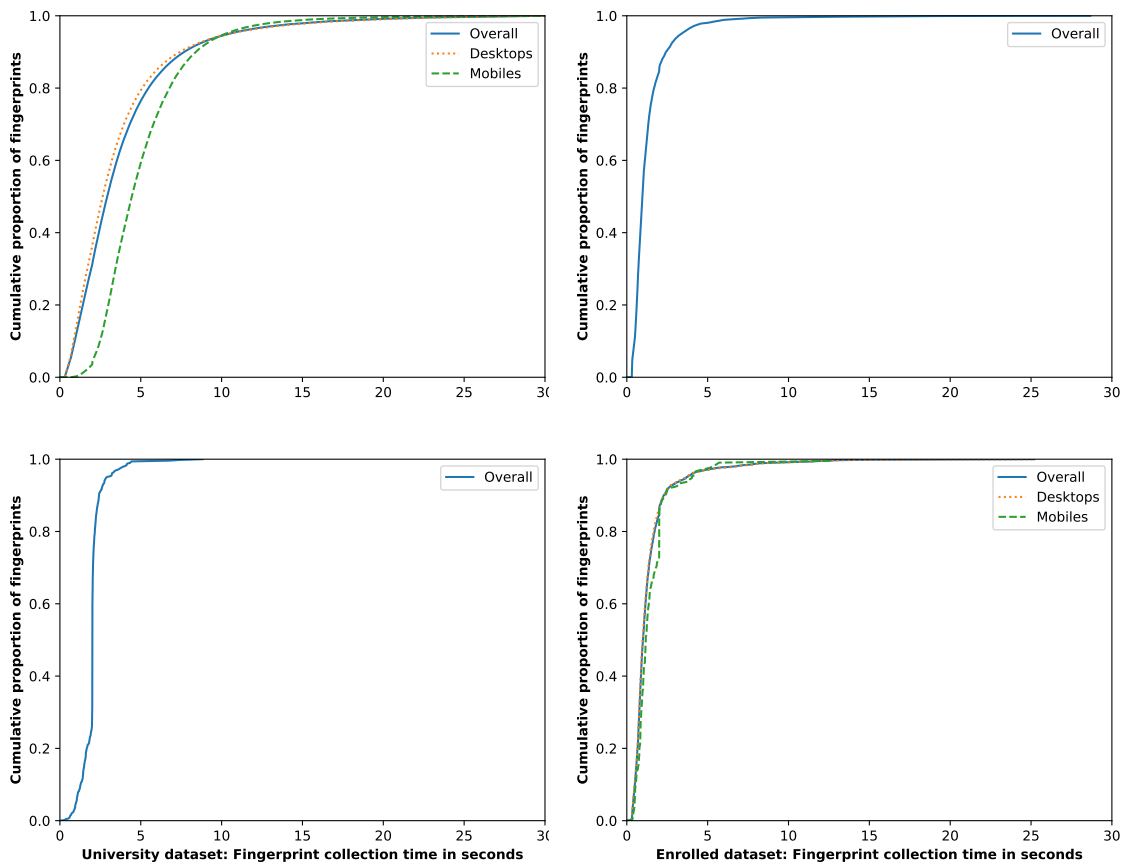


Figure 4.4: Cumulative distribution of the collection time of the fingerprints in seconds.

the fingerprints that take less than 30 seconds to collect, and consider the higher values as outliers. We do not expect users to wait more than 30 seconds for a web page to load. They account for less than 0.5% for each dataset.

Summary of the Collection Time Results

The median collection time of the fingerprints of the intranet, the enrolled, and the university datasets are below 2 seconds. Moreover, the majority (95%) of the fingerprints of these datasets are collected in less than 3.69 seconds. The fingerprints of the general audience dataset show a higher median collection time of 2.92 seconds, and the majority (95%) of them are collected in less than 10.42 seconds. This can be explained by three factors. First, the web pages that hosted

the fingerprinting probe are heavier and require more computing power to load. Second, browsers running on devices having less computing power are part of the population. Third, the general audience experiment took place few years before the other experiments. The median collection time of the fingerprints of our datasets is less than the estimated median time taken by web pages to load completely [18], being at 6.7 seconds for the desktop browsers, and at 18.9 seconds for the mobile browsers, at the date of August 1, 2020. The mobile fingerprints generally take more time to collect than the desktop fingerprints. Half the mobile fingerprints of the general audience dataset take 4.44 seconds to collect, against 2.64 seconds for the desktop fingerprints. The difference is smaller for the enrolled dataset, for which half of the mobile fingerprints take 1.18 seconds to collect, against 0.98 seconds for the desktop fingerprints. This higher collection time can be explained by the limited computing power of mobile devices.

Detailed Analysis of the Collection Time for the General Audience Dataset

The outliers account for less than 0.5% of the fingerprints of each device type for the general audience dataset. Half of the fingerprints are collected in less than 2.92 seconds, and the majority (95%) in less than 10.42 seconds. The mobile fingerprints are generally longer to collect than the desktop fingerprints. Half of the desktop fingerprints are collected in less than 2.64 seconds, and the majority (95%) in less than 10.45 seconds. These numbers are respectively of 4.44 seconds and 10.16 seconds for the mobile fingerprints.

Detailed Analysis of the Collection Time for the Intranet Dataset

The outliers account for less than 0.1% of the entries of the intranet dataset. Half of the fingerprints are collected in less than 0.95 seconds, and the majority (95%) in less than 3.38 seconds.

Detailed Analysis of the Collection Time for the University Dataset

The university dataset has no outliers when it comes to the collection time, as all of its fingerprints are collected in less than 8.83 seconds. Half of the fingerprints are

collected in less than 2 seconds, and the majority (95%) in less than 2.95 seconds.

Detailed Analysis of the Collection Time for the Enrolled Dataset

The outliers account for less than 0.2% of the entries of each device type for the enrolled dataset. Half of the fingerprints are collected in less than 1 second, and the majority (95%) in less than 3.69 seconds. The mobile fingerprints are slightly longer to collect than the desktop fingerprints. Half of the desktop fingerprints are collected in less than 0.98 seconds, and the majority (95%) in less than 3.65 seconds. These numbers are respectively of 1.18 seconds and 4.03 seconds for the mobile fingerprints.

4.2.4 Fingerprint Size

In this section, we provide the results on the size of the fingerprints of our four datasets. Figure 4.5 displays the cumulative distribution of the size of the fingerprints of the four datasets in bytes. The fingerprints are encoded in UTF-8 (with ASCII characters only), hence one character takes exactly one byte, and the results can be expressed in both units. The canvases are stored as sha256 hashes of 64 bytes long. The fingerprint sizes are measured by excluding the metadata fields (e.g., the UID, the timestamp) and the source attributes of the extracted attributes.

Summary of the Fingerprint Size Results

Half of the fingerprints of the four datasets weigh less than 7,776 bytes, and 95% weigh less than 12 kilobytes. At the exception of an outlier in the general audience dataset, all the fingerprints weigh less than 22.3 kilobytes. Such a fingerprint size is negligible given the current storage and bandwidth capacities. The mobile fingerprints also tend to be lighter than the desktop fingerprints. For the general audience dataset, 95% of the mobile fingerprints weigh less than 8,020 bytes, against 12,082 bytes for the desktop fingerprints. The difference is smaller for the enrolled dataset, for which 95% of the mobile fingerprints weigh less than 8,752 bytes, against 9,606 bytes for the desktop fingerprints. This is due to

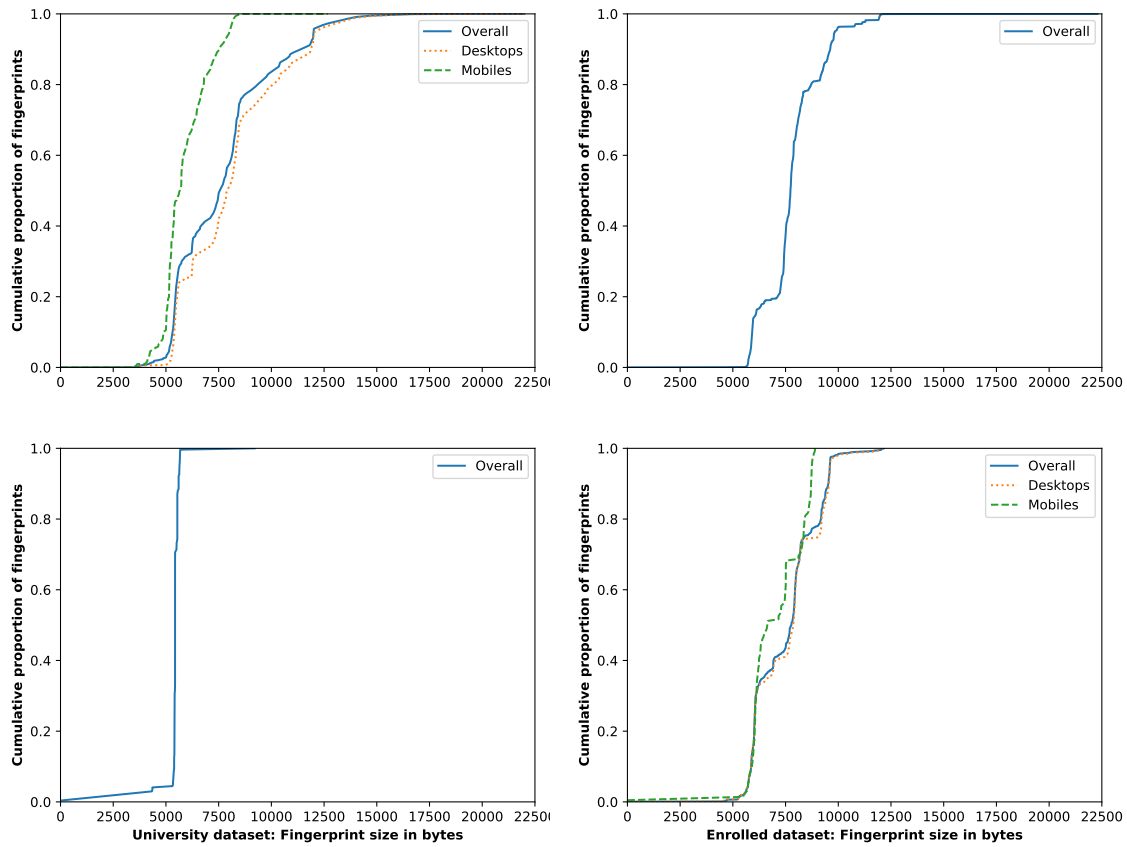


Figure 4.5: Cumulative distribution of the fingerprint size in bytes.

heavy attributes being lighter on mobiles, like the list of plugins or of mime types that are most of the time empty. We discuss these cases in Section 4.3.5.

Detailed Analysis of the Fingerprint Size for the General Audience Dataset

The average fingerprint size for the general audience dataset is of $\mu = 7,692$ bytes, and the standard deviation is of $\sigma = 2,294$. We remove a desktop fingerprint considered an outlier due to its size being greater than $\mu + 15 \cdot \sigma$. Half of the fingerprints of the general audience dataset take less than 7,550 bytes, 95% less than 12 kilobytes, and all of them less than 22 kilobytes. We observe a difference between the fingerprints of mobile and desktop browsers, with 95% of fingerprints weighing respectively less than 8,020 bytes and 12,082 bytes.

Detailed Analysis of the Fingerprint Size for the Intranet Dataset

Half of the fingerprints of the intranet dataset take less than 7,723 bytes, 95% less than 9,832 bytes, and all of them less than 22.3 kilobytes.

Detailed Analysis of the Fingerprint Size for the University Dataset

Half of the fingerprints of the intranet dataset take less than 5,433 bytes, 95% less than 5,643 bytes, and all of them less than 9,212 bytes.

Detailed Analysis of the Fingerprint Size for the Enrolled Dataset

Half of the fingerprints of the enrolled dataset take less than 7,776 bytes, 95% less than 9,600 bytes, and all of them less than 12.17 kilobytes. We observe a difference between the fingerprints of mobile and desktop browsers, with 95% of fingerprints weighing respectively less than 8,752 bytes and 9,606 bytes.

4.2.5 Accuracy of the Simple Verification Mechanism

The accuracy of the simple illustrative verification mechanism is measured on the four datasets according to the following methodology. First, we split the datasets into samples of one month for the general audience dataset, and of two months

for the other datasets which have fewer fingerprints per month. We assume that a user would spend at most two months between two connections, and otherwise would accept to undergo a heavier fingerprint update process. Two sets of pairs of compared fingerprints, also called *comparisons*, are afterward extracted from each sample. The *same-browser* comparisons are composed of the consecutive fingerprints of each browser, and the *different-browsers* comparisons are composed of two randomly picked fingerprints collected from different browsers. After constituting the same-browser comparisons for each month sample, we sample the different-browsers comparisons to have the same size as the same-browser comparisons. The sampling also helps the different-browsers comparisons to be realistic by pairing fingerprints that are separated by at most two months. We measure the accuracy by the false match rate (FMR) which is the proportion of different-browsers comparisons that are classified as same-browser comparisons, by the false non-match rate (FNMR) which is the inverse, and by the equal error rate (EER) which is the rate where the FMR and the FNMR are equal. These measures are discussed further in Section 2.1.5.

Figure 4.6 displays the distribution of the identical attributes between the same-browser comparisons and the different-browsers comparisons for the four datasets. Figure 4.7 displays the false match rate and the false non-match rate for the four datasets. The attributes include the extracted attributes without their source attribute. The displayed results are the average for each number of identical attributes among the month samples.

Summary of the Accuracy Results

We observe that the same-browser comparisons of the four datasets have from 154% to 248% more identical attributes on average, compared to the different-browsers comparisons. By simulating the simple verification mechanism, we achieve an equal error rate of 0.61% on the general audience dataset, of 2.19% on the intranet dataset, of 4.30% on the enrolled dataset, and of 29.42% on the university dataset. We emphasize that the lower equal error rate of the university dataset is due to the standardized browser population, and to unique identifiers being assigned to different browsers. Although the verification mechanism

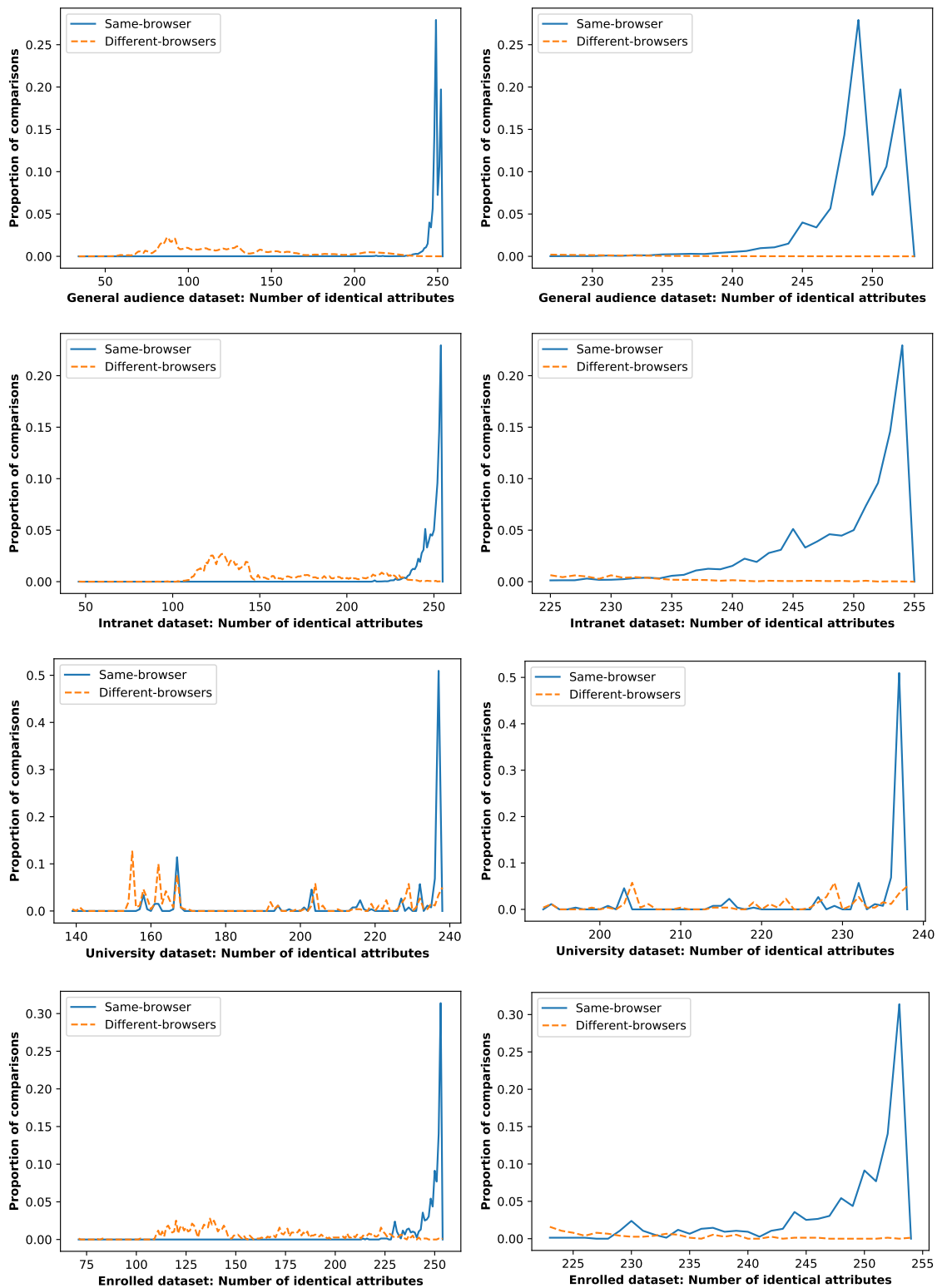


Figure 4.6: The number of identical attributes between the same-browser comparisons and the different-browsers comparisons. The figures on the left start from the lowest observed value, and those on the right start from the value for which 0.5% of the same-browser comparisons are below (20% for the university dataset).

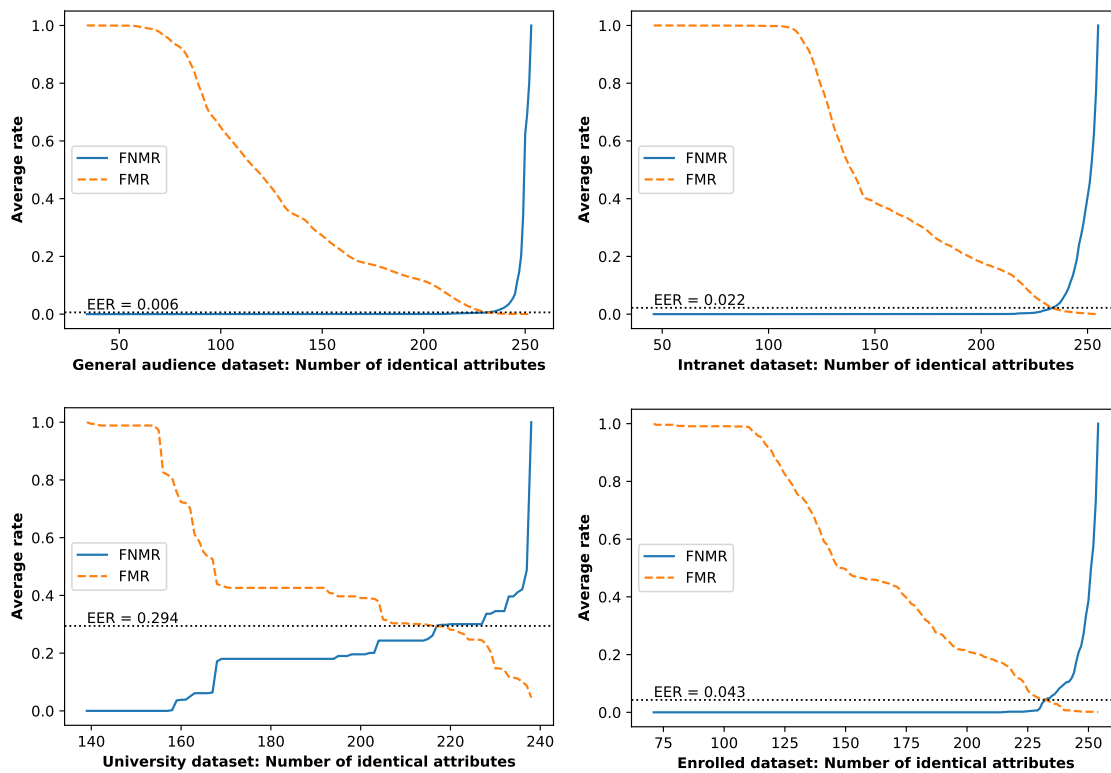


Figure 4.7: False match rate (FMR) and false non-match rate (FNMR) given the required number of identical attributes, averaged among the month samples.

does not have a perfect accuracy, this is acceptable. Indeed, a user getting his browser unrecognized can undergo the fallback authentication process [180, 142]. Moreover, we consider the use of browser fingerprinting as an additional authentication factor, hence the other factors can prevent a falsely recognized browser. Both these cases are expected to rarely occur as can be seen by the low equal error rate.

Detailed Analysis of the Accuracy for the General Audience Dataset

Each of the two sets of comparisons for the general audience dataset contains 3,467,289 comparisons over the six months of the experiment. We can observe that the two sets are well separated, as 99.05% of the same-browser comparisons have more than 234 identical attributes – over a total of 253 attributes – and 99.68% of the different-browsers comparisons have fewer. The different-browsers comparisons have between 34 and 253 identical attributes, with an average of 127.41 attributes, and a standard deviation of 44.06 attributes. The same-browser comparisons have between 72 and 252 identical attributes, with an average of 248.64 attributes, and a standard deviation of 3.91 attributes. Few same-browser comparisons have fewer than 234 identical attributes. They are correctly identified as same-browser comparisons, hence the FNMR is null for a threshold below 234 identical attributes. After exceeding this threshold, the FNMR increases as same-browser comparisons begin to be classified as different-browsers comparisons. The equal error rate is of 0.61% and is achieved for 232 identical attributes.

These results are tied to the distinctiveness and to the stability of the fingerprints of the general audience dataset. Indeed, as more than 94.7% of the fingerprints are shared by less than 8 browsers, two random fingerprints have little chances to be identical. Moreover, more than 96.64% of the attributes are identical between the consecutive fingerprints of a browser, on average and when separated by less than 31 days.

Detailed Analysis of the Accuracy for the Intranet Dataset

The same-browser comparisons of the intranet dataset is composed of 9,667 comparisons, and the different-browsers comparisons contains 9,206 comparisons¹⁴. The different-browsers comparisons have between 46 and 254 identical attributes – over a total of 255 attributes – with an average of 154.87 attributes, and a standard deviation of 37.35 attributes. The same-browser comparisons have between 110 and 254 identical attributes, with an average of 248.87 attributes, and a standard deviation of 6.23 attributes. No same-browser comparisons have fewer than 110 identical attributes, hence the FNMR is null for a threshold below 110 identical attributes. After exceeding this threshold, the FNMR increases as same-browser comparisons begin to be classified as different-browsers comparisons. The equal error rate is of 2.19% and is achieved for 233 identical attributes.

Detailed Analysis of the Accuracy for the University Dataset

The same-browser comparisons of the university dataset is composed of 263 comparisons, and the different-browsers comparisons contains 261 comparisons¹⁵. The two sets of comparisons are not as well-separated as for the other datasets. The different-browsers comparisons have between 139 and 238 identical attributes – over a total of 238 attributes – with an average of 88.05 attributes, and a standard deviation of 32.43 attributes. The same-browser comparisons have between 157 and 238 identical attributes, with an average of 218.46 attributes, and a standard deviation of 28.63 attributes. No same-browser comparisons have fewer than 158 identical attributes, hence the FNMR is null for a threshold below 158 identical attributes. After exceeding this threshold, the FNMR increases as same-browser comparisons begin to be classified as different-browsers comparisons. The equal

¹⁴The user population is the employees, and rarely the visitors, of the company. As browsers return regularly, there are numerous same-browser comparisons. The different-browsers comparisons can be fewer when there are not enough different browsers for some months.

¹⁵Only 80 fingerprints were collected in March and April 2020, and most of them come from the same browser. Due to the few diversity in terms of browser for the sample corresponding to these months, only 38 different-browsers comparisons have been generated, against 40 comparisons for the same-browser class. Moreover, only 18 fingerprints were collected in May and June 2020, and they all come from the same browser. Due to this, no different-browsers comparisons can be generated from the sample corresponding to these months, which are then ignored.

error rate is of 29.42% and is achieved for 217 identical attributes.

These results are linked to the low distinctiveness and the low stability of the fingerprints of the university dataset. Indeed, the unicity rate of the time-partitioned datasets of the university dataset goes as low as 11%, and stabilizes at 17.1% on the long run. Moreover, the stability of the university dataset is sparse and goes as low as 68.7%.

Detailed Analysis of the Accuracy for the Enrolled Dataset

Each of the two sets of comparisons for the enrolled dataset contains 755 comparisons over the first six months of the experiment¹⁶. The different-browsers comparisons have between 71 and 254 identical attributes – over a total of 254 attributes – with an average of 160.23 attributes, and a standard deviation of 38.47 attributes. The same-browser comparisons have between 214 and 254 identical attributes, with an average of 248.18 attributes, and a standard deviation of 6.67 attributes. No same-browser comparisons have fewer than 214 identical attributes, hence the FNMR is null for a threshold below 214 identical attributes. After exceeding this threshold, the FNMR increases as same-browser comparisons begin to be classified as different-browsers comparisons. The equal error rate is of 4.30% and is achieved for 232 identical attributes. The EER is lower than for the general audience dataset, as the intersection of the FMR and the FNMR curves occur at a higher value. This is due to 5.83% of the same-browser comparisons having 232 identical attributes or less, and to 3.97% of the different-browsers comparisons having 232 identical attributes or more. Although the enrolled dataset has a high unicity rate, the fingerprints of two different browsers can share several identical attributes. Moreover, the consecutive fingerprints of the enrolled dataset show a lower stability for some elapsed time between two observations (e.g., down to 84.9% identical attributes between the consecutive fingerprints separated by 58 days).

¹⁶The enrolled experiment was processed over seven months (see Section 3.2 for a description of this dataset), and the samples have a size of two months. Due to this sampling, the seventh month is not considered.

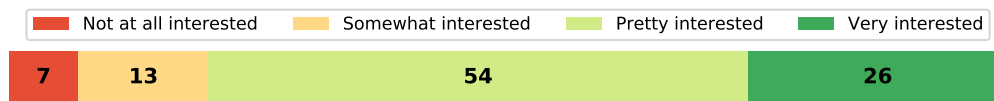


Figure 4.8: Interest of the respondents in the authentication mechanism according to its description only.

4.2.6 Acceptability

In this section, we provide the results of the survey about the acceptability of the authentication mechanism that experimenters of the enrolled experiment tested. The questions were put to the participants in two phases: after reading a brief description of the authentication mechanism, and after testing the actual authentication mechanism.

Opinion on the Description of the Authentication Mechanism

After reading the description of the test authentication mechanism, which is presented in Appendix E, the participants were asked about their interest in the described mechanism. Figure 4.8 shows the level of interest of the respondents in the described authentication mechanism. We have 1,035 respondents (80%) that are interested in the described mechanism, and 26% of the respondents even report being very interested.

We asked the 1,035 respondents that are interested which advantages do they perceive in the described authentication mechanism. Their answers are summarized in Figure 4.9. The most perceived advantage is the ease of use, which is reported by 43% of the interested respondents. They note that the mechanism does not require any additional software, nor any specific action. The enhancement of the security is reported by 41% of the interested respondents. They note that the mechanism provides additional security thanks to the hardware recognition. Among the interested respondents, 25% deem the mechanism trustworthy, and 4% deem it innovative. Finally, 9% of the interested respondents report no advantage.

For the 250 respondents that are not interested, we asked them the weaknesses that they perceive in the described authentication mechanism. Their answers are summarized in Figure 4.10. Among the uninterested respondents, 24% do not

trust the described authentication mechanism. They note that the fraudulent access or theft of the device deceive the mechanism. The fingerprint is linked to a device, and 17% of the uninterested respondents report this as a weakness. Some explain that they use several devices in different places (e.g., at home, at work), and others complain about the mechanism requiring to always use the same device. The description of the mechanism did not emphasize this aspect, but the authentication mechanism allowed users to register several devices, as described in Appendix F. Among the uninterested respondents, 15% have concerns about the collection of browser fingerprints that they consider as personal or sensitive data. The uninterested respondents also report already using a satisfying solution (13%), finding the described authentication mechanism too complex (13%), the solution being only available on Android platforms¹⁷ (13%), and not needing or not convinced by the described mechanism (4%). Finally, 7 of the uninterested respondents report no reason for their lack of interest.

Opinion on the Experience of the Authentication Mechanism

After being invited to participate in the public test of the authentication mechanism, 682 participants (53% of the respondents to the survey) tested it and answered the questions about their experience. Figure 4.11 shows the satisfaction score reported by the participants with their experience on the test authentication mechanism. We measure the satisfaction of the respondents by the Net Promoter Score (NPS) [107] as described in Section 3.4.2, and we compare the obtained results with the satisfaction about the current authentication methods presented in this same section. When asked about their satisfaction with their experience on the test authentication mechanism, 40% of the respondents are promoters (against 26% for the current authentication methods), 22% of them are detractors (against 20% for the current methods), and 38% of them are passives (against 54% for the current methods). The test authentication mechanism polarizes more the respondents than the current authentication methods, as can be seen by the fewer passive

¹⁷We emphasize that this refers to the description of the solution that stipulates that it is "compatible with most Android browsers and mobile applications". However, the testing platform is actually compatible with any browser that supports JavaScript, but its device fingerprinting counterpart which leverages mobile applications was developed to only support Android platforms.

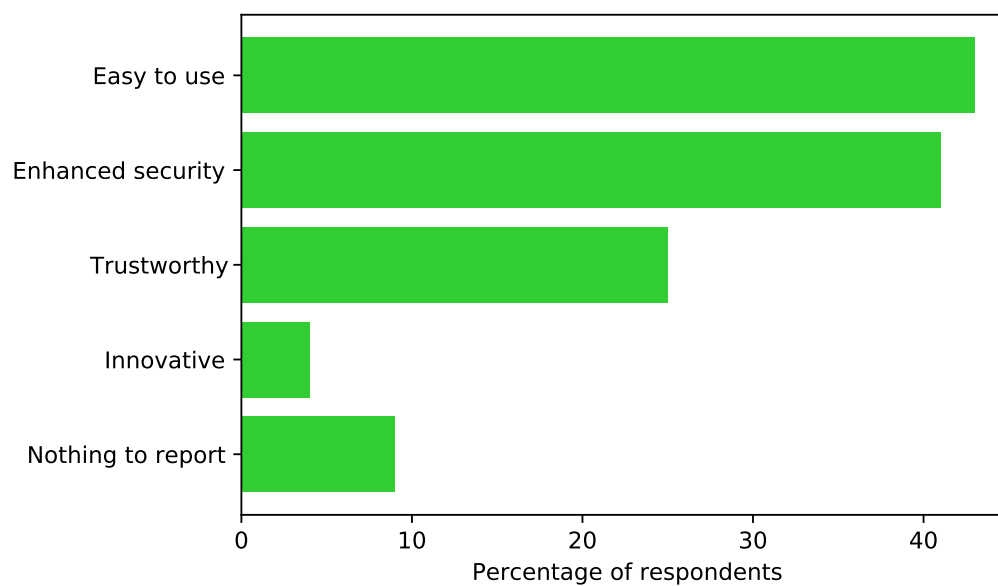


Figure 4.9: The advantages of the authentication mechanism that are perceived by the 1,035 respondents that are interested, according to its description.

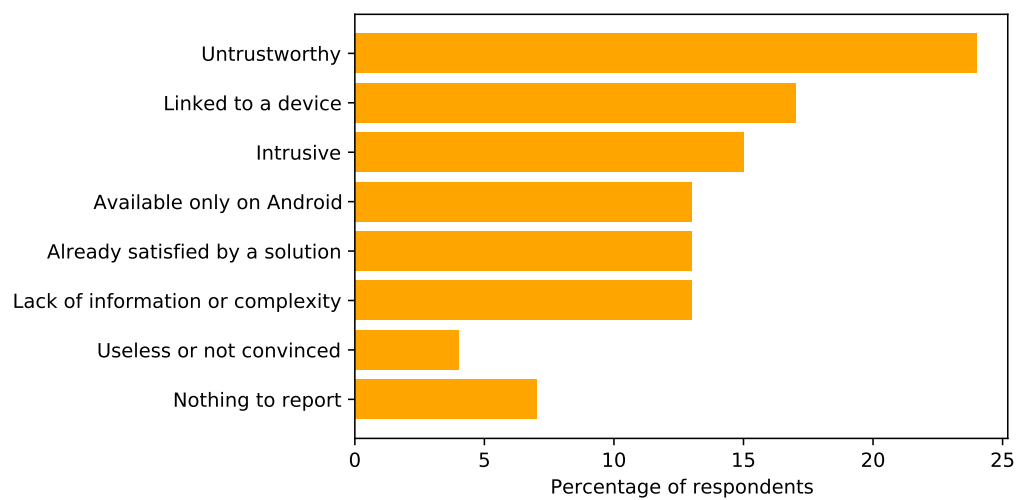


Figure 4.10: The weaknesses of the authentication mechanism that are perceived by the 250 respondents that are not interested, according to its description.

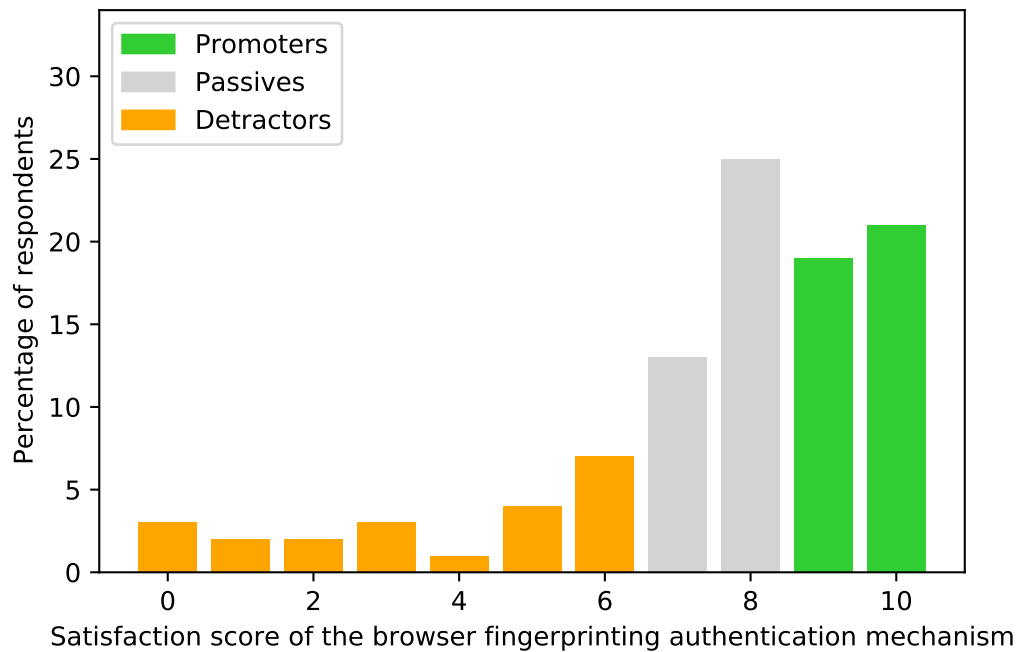


Figure 4.11: The satisfaction of the respondents with their experience on the test authentication mechanism. The scores are divided between the detractors, the passives, and the promoters according to the Net Promoter Score methodology.

respondents. Passive respondents switch more to promoters than detractors, and we have nearly two more times promoters than detractors. We obtain a NPS of 18, which is three times more than the NPS of the current authentication methods. The satisfaction of the respondents with the test authentication mechanism that includes browser fingerprints as an authentication factor is higher¹⁸ than with the current authentication methods¹⁹.

4.2.7 Conclusion

In this section, we analyze the fingerprints of our four datasets according to the properties of authentication factors that we identified. Generally, the properties

¹⁸Not all the respondents to the survey tested the authentication mechanism, resulting in the respondents to the satisfaction about the mechanism being a subset. This can bias the results towards a higher number of promoters, as a respondent that finds the description of the authentication mechanism interesting can tend to confirm his interest in it [160].

¹⁹The current authentication methods are these that the respondents use and are presented in Section 3.4.2. They notably include the password, the biometric fingerprint, and the SMS code.

of the university dataset are different from those of the three other datasets because the browsers of the university experiment are standardized and managed by the university staff. Moreover, the browser profile of the experimenters is shared between the computers of the university, and the same browser identifier can reference browsers running on different web environment (e.g., on a Linux and on a Windows operating system). The intranet, university, and enrolled experiments were still running during the containment in France due to the COVID-19 pandemic. Only the university experiment was impacted as no student nor university staff used the university computers. Experimenters still accessed the fingerprint collection websites of the intranet and enrolled experiments.

About the distinctiveness when considering the time-partitioned datasets, the general audience, the intranet, and the enrolled datasets show a proportion of unique fingerprints above 80%. Their lowest unicity rate are respectively of 81.3%, of 93.3%, and of 96.1%. On the contrary, the unicity rate of the university dataset goes down to 11%, and stabilizes at 17.1% on the long run.

About the stability, the average similarity between the consecutive fingerprints of the general audience, the intranet, and the enrolled datasets is above 81%. At the exception of two occurrences for the enrolled dataset, this rate is even above 91%. The average similarity between the consecutive fingerprints of the university dataset shows a high variability and goes from 68.7% to 99.6%.

The median collection time of the fingerprints of the intranet, the enrolled, and the university datasets are below 2 seconds. The majority (95%) of the fingerprints of these datasets are collected in less than 3.69 seconds. The general audience dataset shows a higher median collection time of 2.92 seconds, and the majority (95%) of its fingerprints are collected in less than 10.42 seconds.

Half of the fingerprints of the four datasets weigh less than 7,776 bytes, 95% of them weigh less than 12 kilobytes, and all of them weigh less than 22.3 kilobytes at the exception of an outlier in the general audience dataset.

By simulating the simple verification mechanism on the general audience, the intranet, and the enrolled datasets, we achieve a respective equal error rate of 0.61%, of 2.19%, and of 4.30%. The university dataset, due to its specificities, shows a higher equal error rate of 29.42%.

We confirm the conclusions of the previous studies [204, 127, 81] about the

mobile fingerprints being less distinctive. Their unicity rate in the time-partitioned datasets of the general audience dataset falls down to 41%, compared to 84% for the desktop fingerprints. The loss is less significant for the enrolled dataset, which shows a lower unicity rate of 93.6% for the mobile fingerprints, against 96.5% for the desktop fingerprints. Only the general audience dataset comprises consecutive mobile fingerprints over several time ranges. The mobile fingerprints of this dataset tend to be more stable through time than the desktop fingerprints. About the size and the collection time of mobile fingerprints, they are generally lighter and take more time to collect than the desktop fingerprints. We do not remark any significant loss in the properties offered by the fingerprints of the four datasets through the duration of their respective experiment.

We surveyed 1,285 experimenters about their interest in an authentication mechanism that relies on browser fingerprints. Among them, 80% report being interested, and 26% report being very interested. The interested respondents perceive the mechanism as easy to use (43%), as enhancing the security (41%), and as trustworthy (25%). The uninterested respondents perceive the mechanism as untrustworthy (24%), as tightly linked to a device (17%), and as intrusive (15%). We also measure the satisfaction of the 682 participants who tested the test authentication mechanism that leverages browser fingerprints by using the Net Promoter Score. The test authentication mechanism polarizes the experimenters more than the current authentication methods. Only 38% of the experimenters are classified as passive, compared to 54% for the current methods. This loss in the number of passives results in an increase of both the promoters and the detractors. About the test authentication mechanism, 40% of the experimenters are promoters and 22% of them are detractors, compared to 14% of promoters and 20% of detractors about the current methods. The net promoter score of the test authentication mechanism is of 18, which is three times more than the score of the current methods (6).

4.3 Attribute-wise Analysis

In this section, we discuss the contribution of the attributes to the properties of the fingerprints. We also discuss the correlation between the attributes, and provide

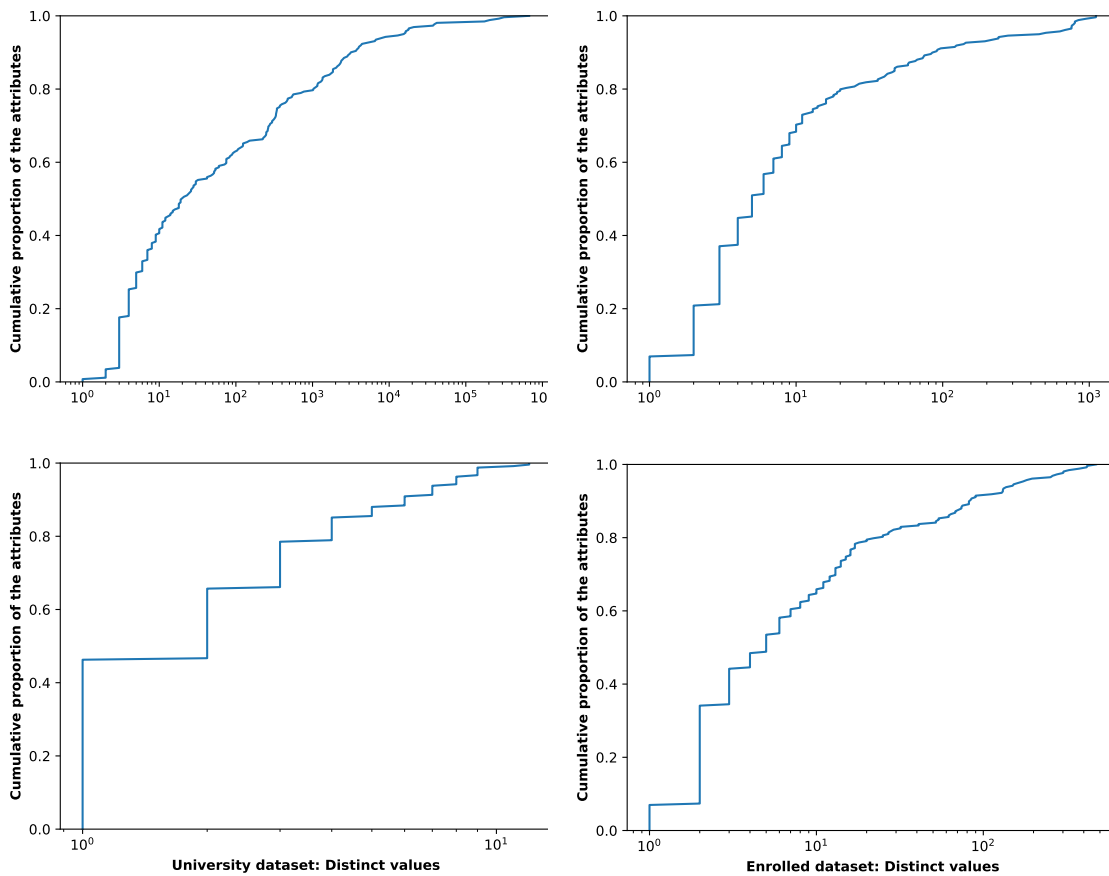


Figure 4.12: Cumulative distribution of the number of distinct values of the attributes in logarithmic scale.

a focus on the dynamic attributes. We refer the reader to Appendix A for more information about the implementation of each attribute, and to Appendix I for the exhaustive list of the attributes with their properties.

4.3.1 Attributes Distinct Values

In this section, we present the number of distinct values of the attributes of the four datasets, as it impacts the distinctiveness that they provide. Figure 4.12 displays the distribution of the number of distinct values of the attributes, for the four datasets and considering the extracted attributes. As an attribute has at most as many distinct values as there are distinct fingerprints in the dataset, we provide the latter for comparison.

Summary of the Attributes Distinct Values Results

Although thousands to millions of fingerprints were collected during the general audience, intranet, and enrolled experiments, between 63% and 91% of their attributes got at most 100 distinct values observed. Due to the standardized browser population of the university experiment, its attributes got at most 12 distinct values observed. The values of the attributes are of different nature, impacting the distinct values that can be observed among different populations or time ranges. Some attributes have a fixed number of values, like the Boolean attributes or the categorical attributes that have a fixed set of possibilities. Other attributes are composed of elements having a fixed set of possibilities, but their combination provides a high number of values. It is the case for the attributes that are related to languages that are typically composed of language identifiers (e.g., `fr`) that can be valued (e.g., $q = 0.80$). These attributes also comprise the list of fonts, that is composed of Boolean values that indicates the presence of a given font. Other attributes consist in an integer, or in a real number represented as a floating-point number, resulting in a large set of possible values. This category comprises the size attributes (e.g., the screen width and height), and our audio fingerprinting method which values are floating-point numbers with a high precision of more than 8 digits. Finally, some attributes are textual information that can include version number, which results in them having a high number of distinct values. Moreover, as new values appear through time (e.g., new versions, new software components), the set of the observed values is expected to grow over the observations. Examples are the `userAgent` or the list of plugins, as these attributes are composed of the name and the version of hardware and software components.

Detailed Analysis of the Attributes Distinct Values for the General Audience Dataset

The general audience dataset has 3,578,196 distinct fingerprints. Among the 253 attributes, 42% have at most 10 distinct values, 63% have at most 100 distinct values, and 79% have at most 1,000 distinct values. Only 5 attributes have more than 100,000 distinct values. They are the WebRTC fingerprinting method (671,254

values), the list of plugins (314, 518 values), the custom canvas in the PNG format (269, 874 values) and in the JPEG format (205, 005 values), together with the list of mime types (174, 876 values).

Detailed Analysis of the Attributes Distinct Values for the Intranet Dataset

The intranet dataset has 9,342 distinct fingerprints. Among the 260 attributes, 70% have at most 10 distinct values, 91% have at most 100 distinct values, and more than 99% have at most 800 distinct values. Two attributes have between 800 and 1,000 distinct values: the `innerWidth` (844 values) and the `screenX` (957 values) properties of the `window` object. Two attributes have more than 1,000 distinct values: the `width` (1,110 values) and the `position` (1,113 values) of the created `div` element.

Detailed Analysis of the Attributes Distinct Values for the University Dataset

The university dataset has 85 distinct fingerprints. All the 243 attributes of this dataset have at most 12 distinct values. Among them, 46% have only a single value, 79% have at most 3 distinct values, and 99% have at most 10 distinct values. Only 3 attributes have between 10 and 12 distinct values: the `UNMASKED_RENDERER_WEBGL` property has 11 distinct values, and the `UserAgent` collected from JavaScript and from HTTP headers both have 12 distinct values.

Detailed Analysis of the Attributes Distinct Values for the Enrolled Dataset

The enrolled dataset has 1,761 distinct fingerprints. Among the 259 attributes, 66% have at most 10 distinct values, 91% have at most 100 distinct values, and 99% have at most 400 distinct values. Only 3 attributes have more than 400 distinct values: the `UserAgent` collected from JavaScript (419 values) and from HTTP headers (417 values), together with the `innerHeight` property of the `window` object (474 values).

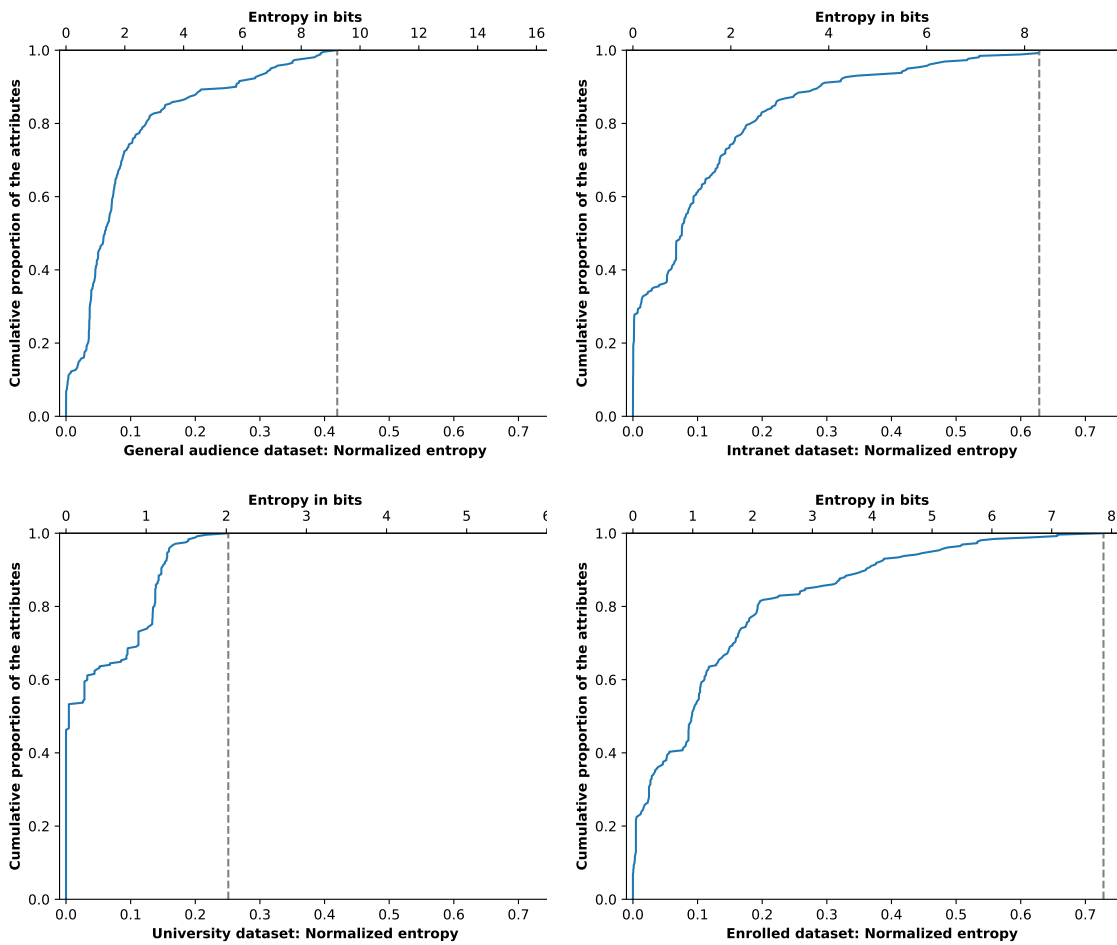


Figure 4.13: Cumulative distribution of the normalized entropy, and of the entropy in bits, among the attributes. The dashed gray line is the entropy of the most entropic attribute.

4.3.2 Attributes Distinctiveness

In this section, we present the distinctiveness of the attributes measured by the normalized entropy. Figure 4.13 displays the cumulative distribution of the normalized entropy, and of the entropy in bits, among the attributes of the four datasets. Table 4.1 compares the normalized entropy of the attributes between the studies Panopticlick [56], AmIUnique [127], Hiding in the Crowd [81], and our four datasets.

Table 4.1: Comparison of the normalized entropy between the studies Panoptick (PTC) [56], AmUnique (AIU) [127], Hiding in the Crowd (HitC) [81], and our four datasets. The attributes are ranked from the most to the least distinctive for the general audience dataset. We provide the normalized entropy equivalent to 1 bit of entropy for comparison.

Attribute	PTC	AIU	HitC	Gen. Aud.	Intranet	University	Enrolled
Canvas (PNG)	-	0.491	0.407	0.420	0.599	0.199	-
Canvas (JPG)	-	-	0.391	0.399	-	-	-
List of plugins	0.817	0.656	0.452	0.394	0.288	0.000	0.354
User agent	0.531	0.580	0.341	0.350	0.629	0.189	0.656
List of fonts	0.738	0.497	0.329	0.305	0.278	0.218	0.448
Content language	-	0.351	0.129	0.124	0.199	0.142	0.294
List of HTTP headers	-	0.249	0.085	0.095	0.131	0.026	0.316
Renderer WebGL	-	0.202	0.264	0.089	0.079	0.029	0.113
Do Not Track	-	0.056	0.091	0.085	0.134	0.026	0.159
Vendor WebGL	-	0.127	0.109	0.080	0.079	0.028	0.111
Platform	-	0.137	0.057	0.068	0.119	0.147	0.168
Accept	-	0.082	0.035	0.028	0.000	0.000	0.191
Content encoding	-	0.091	0.018	0.019	0.054	0.000	0.018
Timezone	0.161	0.198	0.008	0.008	0.001	0.000	0.004
AdBlock	-	0.059	0.002	0.002	0.064	0.004	0.083
Use of local storage	-	0.024	0.002	0.001	0.000	0.000	0.001
Use of session storage	-	0.024	0.002	0.000	0.000	0.000	0.001
Cookies enabled	0.019	0.015	0.000	0.000	0.000	0.000	0.000
Equivalent of 1 bit of entropy	0.053	0.059	0.048	0.045	0.076	0.124	0.093
Maximum entropy H_M	18,843	16,859	20,980	21,983	13,202	8,066	10,803
Number of Fingerprints	470,161	118,934	2,067,942	4,145,408	9,422	268	1,787

Normalized Entropy Measure

We measure the distinctiveness of the attributes as the normalized entropy for comparability with previous studies. The normalized entropy was proposed by Laperdrix et al. [127] to cope with the problem of comparing the entropy of attributes between fingerprint datasets of dissimilar sizes. The normalized entropy h_n of an attribute is defined as the ratio of its entropy h to the maximum entropy $H_M = \log_2(N)$ when considering N fingerprints. The entropy can be calculated by $h = h_n * H_M$.

Summary of the Attributes Distinctiveness Results

A minority of the attributes provide a high distinctiveness. Looking at the attributes distinctiveness figures, we observe that the 80% of the less distinctive attributes provide a normalized entropy that is several times lower than the 20% most distinctive attributes. This can be seen by the long tail at the top of the figures. The most distinctive attributes are generally the `UserAgent`, the canvases, the list attributes, and the attributes about the size of the browser window. We observe that the most distinctive attributes of previous studies [56, 127] also belong to the most distinctive attributes of our datasets. We also explain the three factors that impact the normalized entropy of our attributes in our datasets compared to the previous studies. We report four attributes that are highly distinctive in our datasets that were not identified as such by the previous large-scale studies [56, 127, 81]. They concern the browser window and the properties of elements drawn in the web page.

Comparison with Previous Studies

The most distinctive attributes of previous studies [56, 127] also belong to the most distinctive attributes of our datasets. Although, three factors greatly impact the obtained normalized entropy for our datasets, when compared with previous studies. First, the maximum entropy H_M increases with the number of fingerprints. However, an attribute that has n possibilities (e.g., a Boolean attribute has only two possible values) is limited to a normalized entropy of at most $\log_2(n)$. As the

number N of fingerprints increases, the normalized entropy decreases due to the entropy of the attribute being capped at $\log_2(n)$, whereas the ratio is to $\log_2(N)$. Second, the contextual attributes are biased towards the French-population, as described in Section 3.2, hence they provide a lower normalized entropy compared to previous studies. For example, the time zone and the `Accept-Language` HTTP header (named `Content language` in Table 4.1) provide a respective normalized entropy that is comprised in $[0.000; 0.008]$ and $[0.124; 0.294]$ for our datasets, against 0.198 and 0.351 for the AmIUnique study [127]. The third reason is the evolution of web technologies since the Panopticlick and the AmIUnique study. For example, the list of plugins is less distinctive for our datasets due to the replacement of plugins by HTML5 functionalities or extensions [209]. Another example is the list of fonts that could be collected through plugins [56], but now has to be inferred from the size of text elements [65].

New Distinctive Attributes

Interestingly, four attributes unreported by the previous large-scale studies [56, 127, 81] are found to be highly distinctive. We find new attributes related to the browser window that are among the most distinctive in our datasets. The `innerWidth`, `innerHeight`, `outerWidth`, and `outerHeight` properties collected from the `windows` JavaScript object are mentioned by [195, 215] without any distinctiveness measure. They have a normalized entropy respectively comprised in $[0.263; 0.482]$, $[0.388; 0.728]$, $[0.293; 0.509]$, and $[0.327; 0.613]$ for our datasets excluding the university dataset due to its specific browser population. The `availWidth` and `availHeight` properties of the `screen` JavaScript object are already known as they are mentioned by [145, 83, 38, 114, 214, 97], but no distinctiveness measure is provided. They have a normalized entropy respectively comprised in $[0.202; 0.328]$ and $[0.268; 0.507]$ for our datasets excluding the university dataset. The `AvailTop` and `AvailLeft` properties, for their part, are less distinctive with a respective normalized entropy comprised in $[0.060; 0.149]$ and in $[0.048; 0.129]$ for these same datasets. The size of bounding boxes is used by [65] as a method of font fingerprinting. From the entropy reported by the authors, we compute a normalized entropy of 0.761. In our datasets, this attribute is di-

vided into the list of the values of widths and of heights, which have a respective normalized entropy comprised in [0.299; 0.424] and [0.264; 0.439] for our datasets excluding the university dataset. To the best of our knowledge, no previous study uses the width and the position of a newly created `div` element as a fingerprinting attribute. However, they are highly distinctive as they achieve a normalized entropy respectively comprised in [0.316; 0.535] and [0.316; 0.536] for our datasets excluding the university dataset.

Detailed Analysis of the Attributes Distinctiveness for the General Audience Dataset

Among the attributes of the general audience dataset, 10% provide a normalized entropy lower than 0.003, and another 10% provide a normalized entropy between 0.25 and 0.42. The majority of the attributes (80%) provide a normalized entropy comprised between 0.003 and 0.25. The three canvases are among the most distinctive attributes. Our designed canvas in PNG has a normalized entropy of 0.420, the canvas similar to [124] has a normalized entropy of 0.385, and the canvas inspired by [127] shows a normalized entropy of 0.353. The `UserAgent` collected from the JavaScript property is more distinctive than its HTTP header counterpart, as they respectively have a normalized entropy of 0.394 and 0.350. Finally, the list attributes are also among the most distinctive attributes. The list of plugins has a normalized entropy of 0.394, the list of the supported mime types has a normalized entropy of 0.311, and the list of fonts has a normalized entropy of 0.305.

Detailed Analysis of the Attributes Distinctiveness for the Intranet Dataset

Among the attributes of the intranet dataset, 36% provide a normalized entropy of at most 0.041, 56% provide a value between 0.041 and 0.322, and 8% provide a value above 0.322. The `userAgent` properties collected from JavaScript and HTTP header are the most distinctive attributes, and both show a normalized entropy of 0.629. The properties related to the browser window are also among the most distinctive. The `innerHeight` property provides a normalized entropy of 0.628,

the `outerHeight` one shows a value of 0.517, the `innerWidth` one shows a value of 0.482, and the `outerWidth` one provides a value of 0.442. The three canvases are part of the most distinctive attributes. Our designed canvas in PNG has a normalized entropy of 0.599, the canvas similar to [124] has a value of 0.471, and the canvas inspired by [127] shows a value of 0.459. The position and the width of a created `div` element follow with a respective normalized entropy of 0.536 and 0.535. The `appVersion` property shows a normalized entropy of 0.519, and the combination of the width and the height of first bounding box provides a normalized entropy of 0.455.

Detailed Analysis of the Attributes Distinctiveness for the University Dataset

The attributes of the university dataset, due to the standardized browser population, provide a lower distinctiveness compared to the other datasets. Among the attributes of the university dataset, 46% have a null normalized entropy and provide no distinctiveness at all. Moreover, 74% of the attributes provide a normalized entropy lower than 0.124, which is equivalent to 1 bit of entropy. Only 7 attributes provide a normalized entropy above 0.185. The two most distinctive attributes of the university dataset concern the WebGL library. Indeed, the browsers of the university experiment are standardized, but from the collected fingerprints we observe that several graphics drivers are reported by the `UNMASKED_RENDERER_WEBGL` property. It provides a normalized entropy of 0.251, and the list of WebGL extensions shows a normalized entropy of 0.203. The list of fonts is part of the most distinctive attributes, with a normalized entropy of 0.218. Our HTML5 canvas provides a normalized entropy of 0.199. The textual attributes that describe the browser and its version are also part of the most distinctive attributes. The two `userAgent` versions provide a normalized entropy of 0.189, and the `buildID` property shows a value of 0.186.

Detailed Analysis of the Attributes Distinctiveness for the Enrolled Dataset

Among the attributes of the enrolled dataset, 52% provide a normalized entropy of at most 0.093, which is equivalent to 1 bit of entropy. Most of the attributes 90% provide a normalized entropy of at most 0.361, and 10% of them show a normalized entropy comprised between 0.361 and 0.728. Among the most distinctive attributes, we retrieve the properties related to the browser window. The `innerHeight`, the `outerHeight`, the `outerWidth`, and the `availHeight` properties respectively provide a normalized entropy of 0.728, 0.613, 0.509, and 0.507. Then, we have the textual information about the browser and its version. The `UserAgent` collected from JavaScript shows a normalized entropy of 0.657, the one collected from HTTP headers provides a value of 0.656, and the `appVersion` property shows a value of 0.559. We also have the `UNMASKED_RENDERER_WEBGL` property that provides a normalized entropy of 0.538, the width of the text box that is displayed with the fallback font (0.533), and the position of the created div element (0.533).

4.3.3 Attributes Sameness Rate

We express the stability of the attributes as the proportion of the consecutive fingerprints for which the value of the attribute stays identical. We call this proportion the sameness rate. In this section, we present the sameness rate of the attributes of the four datasets, as it impacts the stability of the complete fingerprints. Figure 4.14 displays the cumulative distribution of the sameness rate among the attributes.

Summary of the Attributes Sameness Rate Results

Among the attributes of the general audience, intranet, and enrolled datasets, few are highly unstable. Indeed, there are between 3% and 6% of the attributes of these three datasets that show a sameness rate lower than 80%. The university dataset falls apart as 31% of its attributes have a sameness rate lower than 80%, due to the sharing of the same browser identifier among the computers

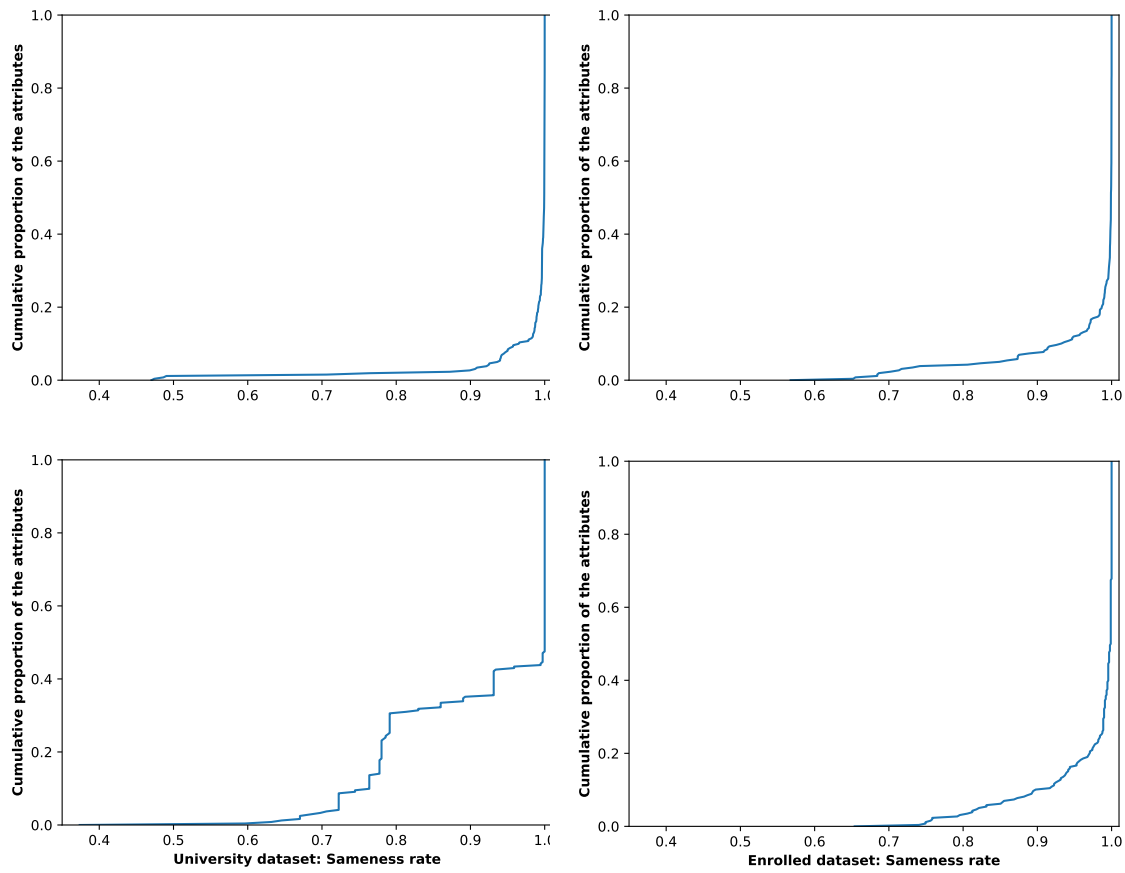


Figure 4.14: Cumulative distribution of the sameness rate of the attributes among the consecutive fingerprints.

of the university. The most unstable attributes concern the size of the browser window, the components drawn in the web page (including canvases), the peripherals connected to the device (including the screen), the network connection, the textual information about the browser (including the `UserAgent`), WebRTC fingerprinting, and list attributes. As few attributes are highly unstable, getting rid of them would improve the stability of the fingerprints, but could reduce their distinctiveness.

Detailed Analysis of the Attributes Sameness Rate for the General Audience Dataset

Among the 253 attributes of the general audience dataset, 6% provide a sameness rate comprised in $[80; 95]\%$, 11% provide a sameness rate comprised in $[95; 99]\%$, and more than 80% of the attributes have a sameness rate higher than 99%. Only 6 attributes show a sameness rate lower than 80%. They are the bounding boxes, three extracted attributes derived from them, the `Cache-Control` HTTP header, and the WebRTC fingerprinting method.

The bounding boxes attribute is composed of the width and the height of three rectangles, from which we extract several parts. When looking at these parts, they have a sameness rate higher than 90%, at the exception of the height of the first bounding box that has a sameness rate of 49.04%. This illustrates the necessity of breaking down some attributes to parts, as removing this part from the original attribute would drastically increase its sameness rate. The `Cache-Control` HTTP header allows the browser to specify the cache policy used during requests. It is the second most unstable attribute with a sameness rate of 70.63%. This is due to this header not being sent sometimes, and to some values containing the `max-age` parameter which can vary between requests. The third most unstable attribute is the WebRTC fingerprinting method that has a sameness rate of 76.46%, because of three factors. First, the experimental state of this attribute leads it to be unsupported, undefined, or erroneous for 75.23% of the observed fingerprints. Then, the collected information contains local IP addresses²⁰ (i.e., the IP address

²⁰Our script hashes these local IP addresses by the MD5 hash function directly on the client for privacy reasons. We refer to Takasu et al. [207] for more information about how the WebRTC API gives access to this information.

of the device in the internal network) which can change between two observations. Finally, it is composed of numerous information about the instance of a WebRTC connection, and the change of any of them changes the value of the whole attribute.

Detailed Analysis of the Attributes Sameness Rate for the Intranet Dataset

Among the 260 attributes of the intranet dataset, 4% have a sameness rate comprised in [56; 75]%, 18% comprised in [80; 99]%, 59% comprised in [99; 100]%, and 17% are completely stable with a sameness rate of 100%. The attributes that present a sameness rate lower than 80% are related to the browser window, to the network connection, and to the screen size.

Although the attributes related to the browser window provide a high distinctiveness, they are highly unstable in the intranet dataset. The attributes related to the inner and outer width or height of the browser window present a sameness rate comprised in [56.76; 71.35]%. The width and the position of the created `div` element both have a sameness rate of 68.43%, and the `availHeight` property shows a sameness rate of 71.70%. The fact that the employees of the company use a laptop, to which they plug an external screen when they are not in the move, contributes to this instability. Some attributes related to the network connection also show a high instability. The `downlink` and the `rtt` properties of the `navigator.connection` JavaScript object have a respective sameness rate of 65.20% and 74.17%. The value of these two attributes are estimated from recent active connections before being rounded. Their instability can be explained by their estimation method that is based on the recent state of the network connection. The attributes related to the screen size are also highly unstable. The `screenX` and `screenY` properties of the `window` object have a respective sameness rate of 65.49% and 73.19%.

Detailed Analysis of the Attributes Sameness Rate for the University Dataset

Among the 260 attributes of the university dataset, 31% have a sameness rate comprised in [37; 80]%, 17% comprised in [80; 100]%, and 52% are completely stable with a sameness rate of 100%. Several attributes are highly unstable and show a

sameness rate below 80%. They concern the peripherals, the WebGL properties, the information about the browser and its version, the components of the browser and its window, and graphical elements drawn in the browser. We emphasize that part of the instability of these attributes is tied to the unique identifiers being assigned to different browsers in the university dataset.

The most unstable attribute is the list of the input and output media devices obtained from the `enumerateDevices` function, which shows a sameness rate of 37.36%. Several WebGL properties are part of the most unstable attributes. For example, the `UNMASKED_RENDERER_WEBGL` property, which provides debugging information about the graphic driver, shows a sameness rate of 59.62%. Moreover, 17 other WebGL properties have a sameness rate between 63.19% and 72.25%. The textual attributes that give information about the browser and its version are also among the most unstable. For example, the two `UserAgent` versions have a sameness rate of 74.45%, the `buildID` property provides a sameness rate of 68.41%, and the `platform` property shows a sameness rate of 76.37%. The attributes that list browser components show a high instability. For example, the list of fonts shows a sameness rate of 69.78%, the list of the properties of the `navigator` object provides a sameness rate of 76.37%, and the list of the speech synthesis voices has a sameness rate of 79.12%. The attributes that consist into the colors of components of the browser window (e.g., the scroll-bar) are also among the most unstable. Among them, only the color of the texts when applied the `HighlightText` style shows a sameness rate of 99.73%. The others have a sameness rate comprised between 77.75% and 79.12%. The graphical elements that are drawn in the browser also show instability. For example, our HTML5 canvas has a sameness rate of 70.60%, and all the extracted parts of the size of the bounding boxes have a sameness rate between 76.37% and 79.12%.

Detailed Analysis of the Attributes Sameness Rate for the Enrolled Dataset

Among the 254 attributes of the enrolled dataset, 3% have a sameness rate comprised in [65; 80]%, 29% comprised in [80; 99]%, 35% comprised in [99; 100]%, and 33% are completely stable with a sameness rate of 100%. There are 9 attributes

that have a sameness rate below 80%. They concern the network connection, the browser window and components that are drawn inside it, the screen size, and the textual information about the browser.

The `download` property of the `navigator.connection` object is the most unstable, with a sameness rate of 65.39%. The `innerHeight` and the `outerHeight` properties of the `window` object have a respective sameness rate of 74.04% and 79.21%. Two attributes that consist of components drawn into the browser are also among the most unstable. The width of the text box drawn using the default font has a sameness rate of 74.94%, and the position of a created `div` element has a value of 74.94%. Two attributes related to the screen size show a high instability. The properties `screenX` and `screenY` have a respective sameness rate of 75.62% and 79.55%. Finally, the two `UserAgent` versions both have a sameness rate of 75.84%.

4.3.4 Attributes Collection Time

Figure 4.15 displays the median collection time of the attributes that have a MCT higher than 5ms, ranked from the slowest to the fastest to collect for the overall browsers. We stress that most of these attributes are collected asynchronously, hence the total collection time of the fingerprint is not their sum.

Summary of the Attributes Collection Time Results

Most of the attributes are HTTP headers or JavaScript properties that are collected in a negligible amount of time. For our four datasets, less than 40 attributes have a median collection time (MCT) above 5ms. These attributes are classified into three categories: the detection of extensions, the browser components (e.g., the fonts, the speech synthesis voices, the WebGL extensions), and the attributes related to the manipulation of a media object (e.g., the canvases, the advanced audio fingerprinting method). The attributes that take time to collect are usually part of the longest to collect for the four datasets. The attributes are generally faster to collect on desktop browsers, but in some cases are faster to collect on mobile browsers. The different versions of our script take several seconds to col-

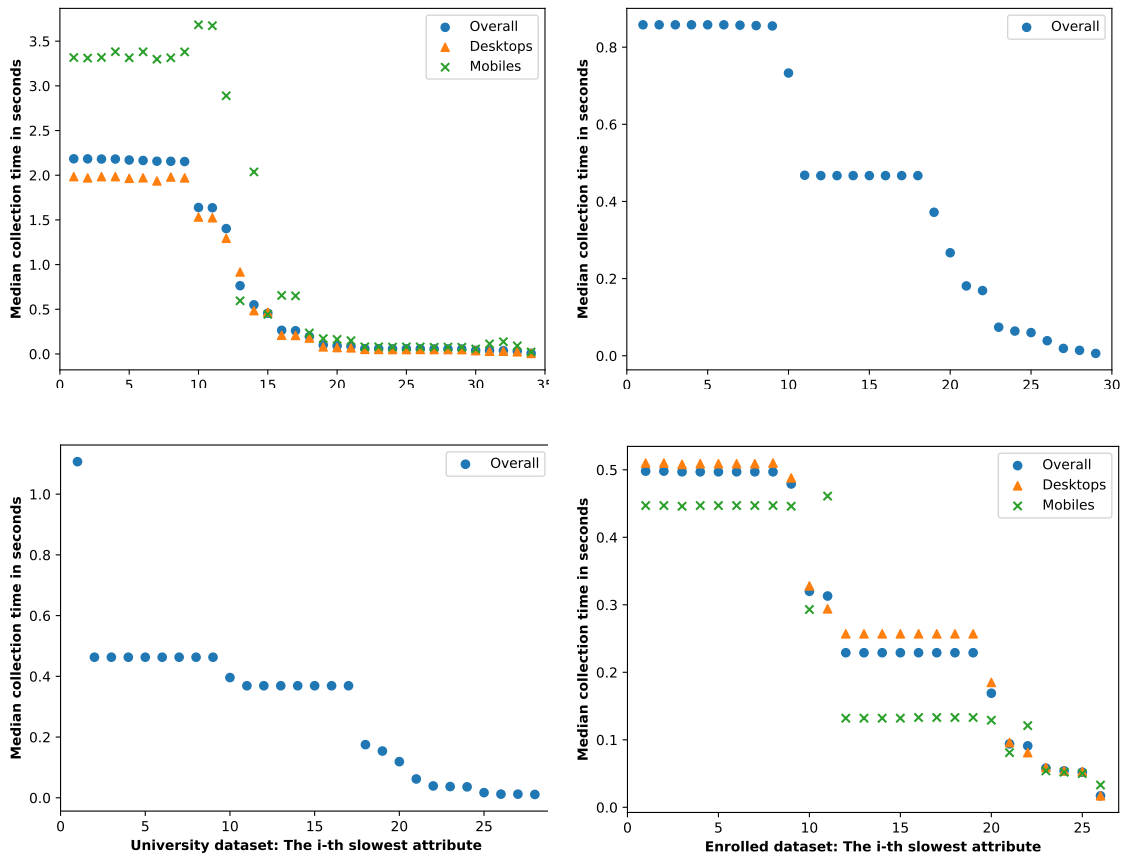


Figure 4.15: Median collection time of the attributes that have a median collection time higher than 5ms, ranked from the slowest to the fastest to collect for the overall browsers.

lect the attributes that compose the fingerprints. However, we emphasize that these scripts are purely experimental, and were developed to collect many attributes to get closer to what a verifier can achieve in real-life. The attributes that are longer to collect and that are less distinctive can be removed to reduce the collection time. For example, our method to detect an advertisement blocker waits a few seconds for a simulated advertisement to be removed, but only provides a Boolean value. Our script can also be updated to leverage the most advanced web technologies, like the `OffscreenCanvas` API^a that migrates the generation of the canvases off the main thread to another one. More generally, we can use the `Service Workers` API^b to collect the attributes concurrently in the background, reducing the perceived collection time. These APIs are available on modern browsers, the collection time should still be monitored for the older browsers.

^a<https://developers.google.com/web/updates/2018/08/offscreen-canvas>

^bhttps://developer.mozilla.org/en-US/docs/Web/API/Service_Worker_API

Detailed Analysis of the Attributes Collection Time for the General Audience Dataset

The 34 attributes that take more than 5ms to collect for the general audience dataset can be separated into three classes: extension detection, browser component, and media related.

The first class of attributes that take time to collect are the methods of extension detection. The 9 slowest attributes detect an extension by the changes it brings to the content of the web page [205]. They have a median collection time (MCT) of approximately 2.2 seconds, due to the waiting time before checking the changes on the web page. There is a clear difference between the desktop and the mobile browsers, with a respective MCT of approximately 2 seconds and 3.4 seconds. The 22nd to the 29th slowest attributes detect an extension based on the web-accessible resources [199], which consist into checking if an image embarked in an extension is accessible or not. They have a MCT of around 60ms, which is lower than the method that relies on detecting changes brought to the web page.

The second class of attributes that take time to collect infer the availability of browser components. The list of the speech synthesis voices is ranked 14th with

a MCT of 568ms. This is due to the collection method that, for some browsers, requires to be done during an `onvoiceschanged` event, which takes time to be triggered. The list of fonts, and the inference of the default font, are ranked 15th and 19th with a respective MCT of 471ms and 103ms. This is due to the detection method that measures the sizes of newly created text boxes [65]. The size of bounding boxes, the colors of browser components, and the width and position of a newly created `div` element are respectively ranked 18th, 20th, and 21st, with a MCT ranging from 60ms to 200ms. This is due to their manipulation of the web page that takes time. The WebRTC fingerprinting method is ranked 13th with a MCT of 783ms. This is due to the creation of a dummy connection that is needed to gather information about the WebRTC configuration. Counter-intuitively, this attribute is faster to collect on mobile (595ms) than on desktop browsers (917ms). This results from the WebRTC API not being implemented on most of the mobile browsers [37] at the time of the experiment, hence avoiding creating the dummy connection. Indeed, this attribute is either empty, undefined, or erroneous for 95.79% of the mobile browsers.

The third class of attributes that take time to collect generate a media file (e.g., a sound, an image). The methods of advanced audio fingerprinting are ranked 10th, 11th, and 12th, and show a MCT between 1.40 seconds and 1.64 seconds. They take longer to collect on mobile browsers, with a MCT comprised between 2.89 seconds and 3.68 seconds, against a MCT between 1.29 seconds and 1.53 seconds for desktop browsers. Our designed canvas in PNG and JPEG format are respectively ranked 16th and 17th with a MCT of 260ms and 265ms. Our canvases take more time to collect on mobile browsers, with a MCT of approximately 650ms against approximately 210ms on desktop browsers. The canvases inspired by the AmIUnique study [127] in PNG and JPEG format are respectively ranked 31st and 33rd with a MCT of 31ms and 39ms. These canvases also show a difference between mobile and desktop browsers, with a respective MCT of approximately 100ms and 30ms. The canvas similar to the canvas of the Morellian study [124] is ranked 32nd with a MCT of 37ms, and shows a MCT of 137ms for the mobile browsers against 31ms for the desktop browsers. Our designed WebGL canvas is ranked 30th with a MCT of 41ms, with a slight difference between mobile (56ms) and desktop browsers (37ms).

Detailed Analysis of the Attributes Collection Time for the Intranet Dataset

The 29 attributes that take more than 5ms to collect for the intranet dataset are the same as for the general audience dataset, at the exception of four attributes.

The first class of attributes that take time to collect are the methods of extension detection. The 9 slowest attributes detect an extension by the changes it brings to the content of the web page [205]. They are the same as for the general audience dataset, and have a MCT of approximately 855ms. The 11th to the 18th slowest attributes detect an extension based on web-accessible resources [199], and have a MCT of 467ms.

The second class of attributes that take time to collect infer the availability of browser components. The list of the speech synthesis voices is ranked 10th, and has a MCT of 733ms. The list of fonts, and the inference of the default font, are ranked 19th and 23rd with a respective MCT of 372ms and 74ms. The size of bounding boxes, the colors of browser components, and the width and position of a newly created `div` element are respectively ranked 22nd, 24th, and 25th, with a MCT ranging from 60ms to 169ms.

The third class of attributes that take time to collect generate a media file. Our designed canvas is ranked 21st with a MCT of 181ms, the canvas inspired by [127] is ranked 28th with a MCT of 14ms, the canvas similar to [124] is ranked 27th with a MCT of 19ms, and the WebGL canvas is ranked 26th with a MCT of 39ms.

Two attributes are among the slowest to collect for the intranet dataset without being in the slowest to collect for the general audience dataset. The first is the list of WebGL extensions, which is ranked 20th and has a MCT of 267ms. This attribute shows a MCT of 8ms in the general audience dataset. The second is the format of date and time, that is collected through the `resolvedOptions` function of an instantiated `DateFormat` object. It is ranked 29th with a MCT of 6ms, which is barely above 5ms.

Detailed Analysis of the Attributes Collection Time for the University Dataset

The 28 attributes that take more than 5ms to collect for the intranet dataset are approximately the same as for the general audience and the intranet dataset.

We retrieve the attributes that detect an extension. The 2nd to the 10th slowest attributes detect an extension by the changes it brings to the content of the web page [205]. They are the same as for the general audience and the intranet dataset, and have a MCT of 463ms, at the exception of the ad blocker detection method that has a MCT of 396ms. The 11th to the 17th slowest attributes detect an extension based on web-accessible resources [199] and have a MCT of 369ms. We also retrieve the attributes that infer the availability of browser components. The list of the speech synthesis voices is ranked 1st, and has a MCT of 1.1 seconds. The list of fonts, and the inference of the default font, are ranked 18th and 22nd with a respective MCT of 175ms and 39ms. The size of bounding boxes, the colors of browser components, and the width and position of a newly created `div` element are respectively ranked 21st, 23rd, and 24th, with a MCT ranging from 36ms to 62ms. We retrieve the attributes that generate a media file. Our designed canvas is ranked 25th with a MCT of 17ms, and the WebGL canvas is ranked 27th with a MCT of 12ms. As we removed the two canvases inspired by two previous studies for the university experiment, we do not retrieve them here. We retrieve the attributes that are among the slowest to collect for the intranet dataset but not for the general audience dataset. The list of WebGL extensions is ranked 19th and has a MCT of 154ms, and the format of date and time is ranked 28th with a MCT of 11ms.

Two attributes are among the slowest to collect for the university dataset, without being in these for the general audience and the intranet dataset. The list of the input and output media devices is ranked 20th with a MCT of 119ms, and the list of the available video codecs is ranked 26th with a MCT of 12ms.

Detailed Analysis of the Attributes Collection Time for the Enrolled Dataset

The 26 attributes that take more than 5ms to collect in the enrolled dataset are approximately the same as for the other datasets. Most attributes are faster to collect on mobile browsers than on desktop browsers, some attributes have a MCT nearly equal between these two groups, and only three attributes are longer to collect on mobile browsers.

We retrieve the attributes that detect an extension. The 8 slowest attributes detect an extension by the changes it brings to the content of the web page [205]. They are the same as for the other datasets, and have a MCT of 497ms and 498ms, at the exception of the ad blocker detection method that has a MCT of 479ms. These attributes are faster to collect on mobile browsers than on desktop browsers, with a respective MCT of approximately 450ms against 500ms. The 12th to the 19th slowest attributes detect an extension based on web-accessible resources [199] and have a MCT of 229ms. These attributes are also faster to collect on mobile devices, with a respective MCT of approximately 130ms against 260ms on desktop browsers.

We retrieve the attributes that infer the availability of browser components. The list of the speech synthesis voices is ranked 10th with a MCT of 320ms. A slight difference occurs between mobile and desktop browsers, with a respective MCT of 293ms and 328ms. The list of fonts, and the inference of the default font, are ranked 20th and 23rd with a respective MCT of 169ms and 58ms. The list of fonts shows a faster collection time on mobile browsers, with a MCT of 129ms against 185ms on desktop browsers. The size of bounding boxes, the colors of browser components, and the width and position of a newly created `div` element are respectively ranked 21st, 24th, and 25th, with a MCT ranging from 52ms to 94ms. We also retrieve the list of WebGL extensions at the 22nd rank, with a MCT of 91ms. This attribute is one of the three that is longer to collect on mobile browsers, with a MCT of 121ms against 81ms on desktop browsers.

We retrieve the attributes that generate a media file. Our designed canvas, which gets its instructions randomized on each fingerprinting, is ranked 26th and has a MCT of 17ms. The WebGL canvas is ranked 11th, and shows a MCT of

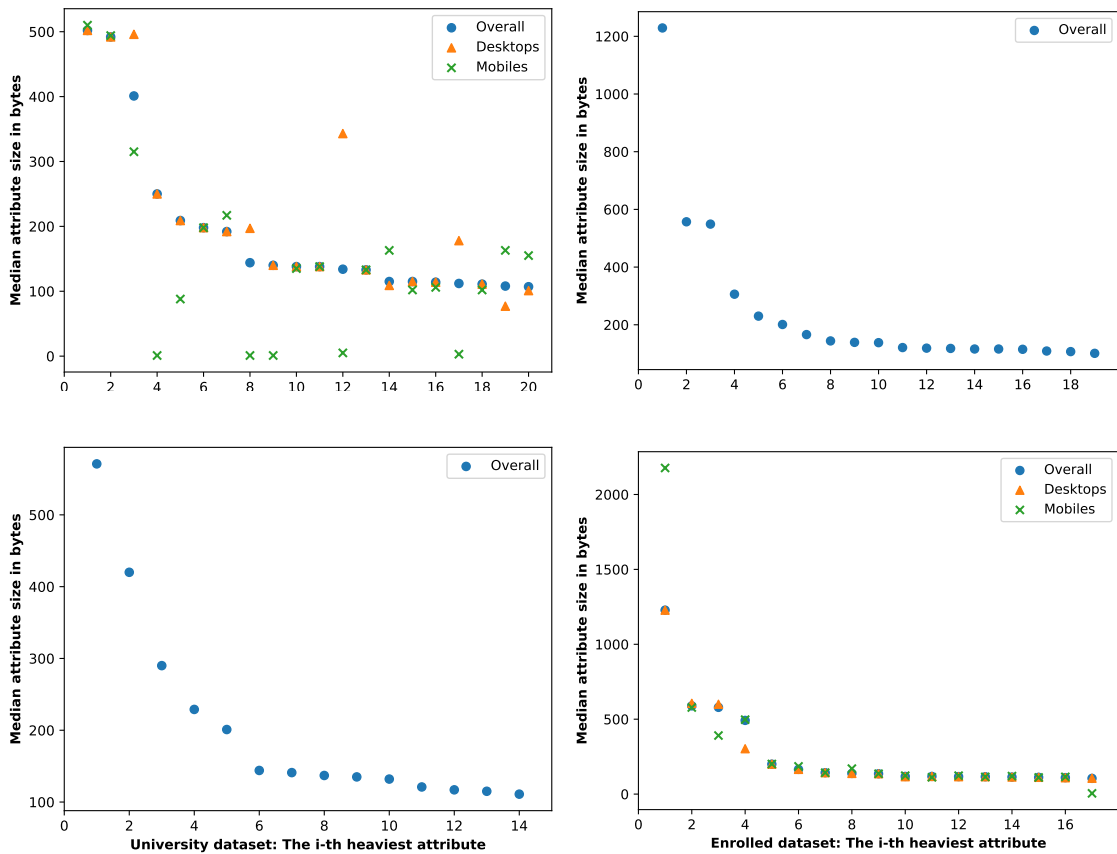


Figure 4.16: Median size of the attributes that weigh more than 100 bytes, ranked from the heaviest to the lightest.

313ms. These two canvases are among the three attributes that are longer to collect on mobile browsers. They show a respective MCT of 33ms and 461ms on mobile browsers, against 17ms and 294ms on desktop browsers.

4.3.5 Attributes Size

In this section, we discuss the heaviest attributes of the four datasets, as they impact the size of the complete fingerprint. Figure 4.16 displays the median size of the attributes that weigh more than 100 bytes, ranked from the heaviest to the lightest for the overall browsers, and for the four datasets.

Summary of the Attributes Size Results

Between 14 and 20 attributes weigh more than 100 bytes for the four datasets. The heaviest attributes are classified into two categories: the list attributes (e.g., the plugins, the WebGL extensions) and the verbose textual attributes (e.g., the `UserAgent`). List attributes are generally lighter on mobile browsers due to the lack of support of some browser components (e.g., the plugins, the WebGL extensions). On the contrary, the verbose textual attributes are slightly heavier on mobile browsers due to them being more verbose (e.g., the `UserAgent` includes the device model). Evolutions regarding the web technologies impact the properties of the attributes. An example is visible when looking at the attributes size: the list of plugins is lighter for the three most recent datasets when compared to the general audience dataset. This is due to the ongoing disappearance of plugins [209].

Detailed Analysis of the Attributes Size for the General Audience Dataset

The 20 attributes of the general audience dataset that weigh more than 100 bytes are composed of list attributes and verbose textual attributes. The heaviest attribute is the list of the properties of the `navigator` object with a MS of 502 bytes. It is followed by the list of the colors of layout components with a MS of 492 bytes, and the list of WebGL extensions with a MS of 401 bytes. Examples of verbose textual attributes are the `appVersion` property that is ranked 18th with a MS of 107 bytes, the `userAgent JavaScript` property that is ranked 14th with a MS of 115 bytes, and its HTTP header counterpart that is ranked 19th with a MS of 108 bytes.

On mobile browsers, some list attributes are most of the time empty due to their lack of customization (e.g., plugins are mostly unsupported). They are the list of the speech synthesis voices which is ranked 4th, the list of the constraints supported by the `mediaDevices` object which is ranked 8th, the list of plugins which is ranked 12th, and the list of the supported mime types which is ranked 17th. On the contrary, the verbose attributes are slightly heavier on the mobile browsers, which is explained by the presence of additional information like the device model in the two versions of the `UserAgent`.

Detailed Analysis of the Attributes Size for the Intranet Dataset

The 19 attributes of the intranet dataset that weigh more than 100 bytes are composed of list attributes and verbose textual attributes. The heaviest attribute is the list of the speech synthesis voices with a MS of 1,229 bytes. It is followed by the list of WebGL extensions with a MS of 557 bytes, and the list of the properties of the `navigator` object with a MS of 549 bytes. Examples of verbose textual attributes are the `appVersion` that is ranked 17th with a MS of 109 bytes, and the two versions of the `userAgent` that are ranked 14th and 15th with a MS of 116 bytes.

We find two new attributes that are among the heaviest for the intranet dataset but absent from those of the general audience dataset. They are the list of heights and widths of the bounding boxes which have a respective MS of 121 bytes and 101 bytes. Three attributes are among the heaviest of the general audience dataset but absent from the intranet dataset. The first is the list of the colors of layout components, which is split into parts and removed from the intranet dataset. The second is the list of plugins, which has a MS of 77 bytes, against 134 bytes for the general audience dataset. This is due to the ongoing disappearance of the plugins [209], resulting in fewer plugins collected for the intranet dataset than for the general audience dataset. The third is the list of the name and values of the HTTP headers that are not specifically stored in a dedicated attribute. This attribute has a MS of 6 bytes for the intranet dataset, against 192 bytes for the general audience dataset. This is due to the update of the fingerprinting probe, as for the intranet experiment we ignore the headers that are automatically added by the proxy of the collection server.

Detailed Analysis of the Attributes Size for the University Dataset

The 14 attributes of the university dataset that weigh more than 100 bytes are only composed of list attributes. The heaviest attribute is the list of WebGL extensions with a MS of 571 bytes. It is followed by the list of the properties of the `navigator` object with a MS of 420 bytes, and the list of the media constraints that is collected through the `getSupportedConstraints` function which shows a MS of 290 bytes.

All the heaviest attributes of the university dataset are also among the heaviest for the general audience and intranet datasets. However, no verbose textual attributes are among the heaviest for the university dataset, and several list attributes weigh less than 100 bytes. The verbose textual attributes are lighter, and most browsers share the same values due to their standardization. For example, the two `UserAgent` versions have a size comprised between 71 bytes and 83 bytes for all the fingerprints. As for examples of lighter list attributes, the list of the supported mime types and of plugins are always empty. This results from the standardized browser population and the lack of personalization allowed to the experimenters of the university experiment (e.g., their ability to configure the operating system or to install software is limited).

Detailed Analysis of the Attributes Size for the Enrolled Dataset

The 17 attributes of the enrolled dataset that weigh more than 100 bytes are composed of list attributes and verbose textual attributes. The heaviest attribute is the list of the speech synthesis voices with a MS of 1,228 bytes. It shows a high difference between mobile and desktop browsers, with a respective MS of 2,177 bytes and 1,228 bytes. The second heaviest attribute is the list of the properties of the `navigator` object with a MS of 589 bytes, and the third is the list of WebGL extensions with a MS of 579 bytes. The latter is lighter on mobile browsers, with a MS of 391 bytes against 600 bytes for the desktop browsers. The list of the constraints supported by the `mediaDevices` object is ranked 4th, shows a MS of 492 bytes, and is heavier on mobile browsers (497 bytes) than on desktop browsers (303 bytes). Examples of verbose textual attributes are the `appVersion` that is ranked 16th with a MS of 110 bytes, and the two versions of the `userAgent` that are ranked 10th and 12th with a MS of 117 bytes and 118 bytes.

Similarly to the intranet dataset and for the same reasons, the list of plugins and of the remaining HTTP headers have a respective MS of 78 bytes and 30 bytes. The list of the widths and of the heights of the bounding boxes are also slightly lighter with a MS of 32 bytes and 98 bytes. Finally, the list of the properties of the `screen` object is also lighter with a MS of 91 bytes.

4.3.6 Correlation between the Attributes

We can expect correlations to occur between the attributes when considering hundreds of them. In this section, we discuss the correlation between the attributes, and refer the reader to Appendix I for the correlation of each attribute. Figure 4.17 displays the minimum, the average, and the maximum normalized conditional entropy (NCE) of an attribute when the value of another attribute is known, ordered by the average NCE, and for the four datasets. We ignore 9 source attributes from which the extracted attributes are derived, and the comparison of an attribute with itself. These cases are irrelevant, as the extracted attributes are completely correlated with their source attribute, and an attribute is completely correlated with itself. We observe that the maximum NCE of an attribute is always equal to the normalized entropy of this attribute, due to the `cookieEnabled` property that is always `true`. As a result, this attribute has a null entropy, and knowing its value does not provide any information on the value of the other attributes. The minimum normalized conditional entropy (MNCE) is also an interesting indicator of the effectiveness to infer the value of an attribute if the value of another attribute is known. Indeed, it represents the NCE when the attribute which helps the most to infer the value of another attribute is known. Table 4.2 displays the less correlated attributes for the general audience, intranet, and enrolled datasets. They consist of the attributes that have a minimum normalized conditional entropy higher than the equivalent of a conditional entropy of 1 bit.

Normalized Conditional Entropy Measure

For comparability with the results of the distinctiveness of the attributes, we express the correlation by the conditional entropy of an attribute a_i when another attribute a_j is known, normalized to the maximum entropy H_M . We call this measure the normalized conditional entropy (NCE). It is comprised between 0.0 if knowing a_i allows to completely infer a_j , and the normalized entropy of a_j if knowing a_i provides no information on the value of a_j (i.e., they are independent). We denote V_i the domain of the attribute a_i , and e_v^i the event that the attribute a_i takes the value v . We consider the relative frequency p of the attribute values among the considered fingerprints. The measure of the conditional

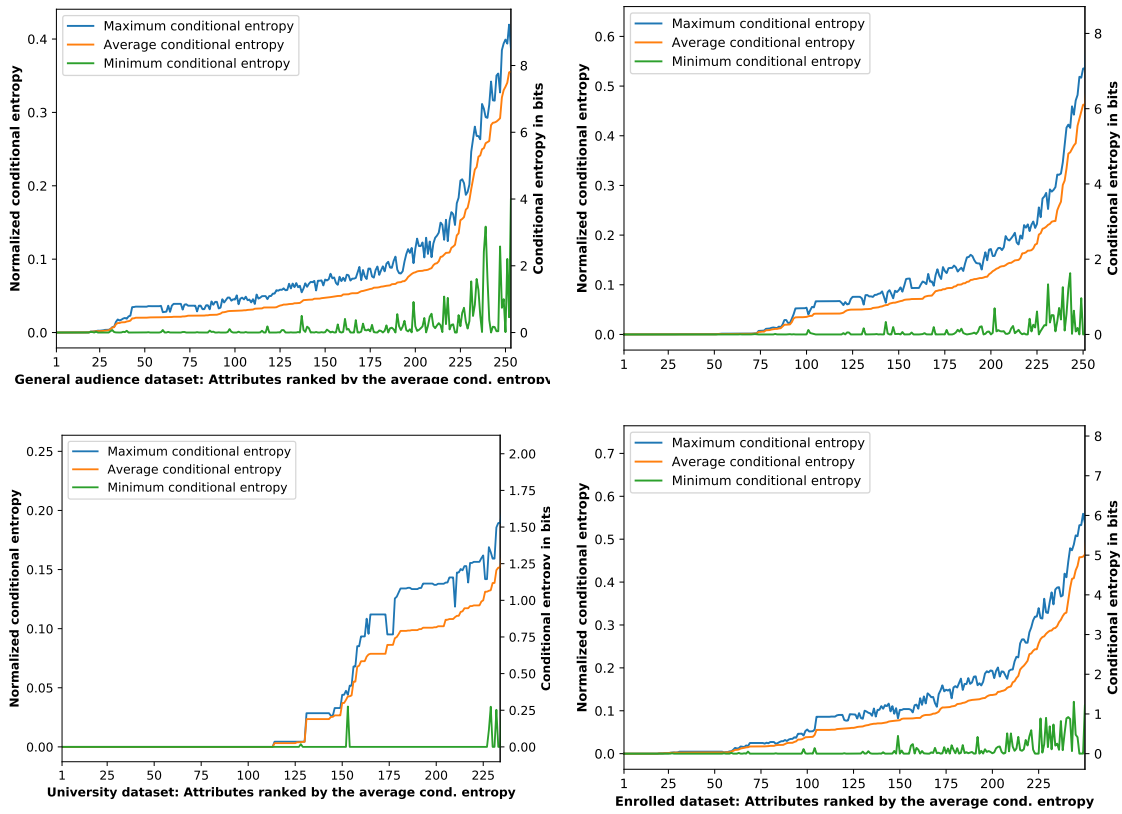


Figure 4.17: Minimum, average, and maximum normalized conditional entropy of an attribute when the value of another attribute is known, ordered by the average normalized conditional entropy.

Table 4.2: The minimum normalized conditional entropy (MNCE) of the attributes that have a MNCE higher than the equivalent of a conditional entropy of 1 bit, for the general audience, intranet, and enrolled datasets. W refers to the `window` object, and N to the `navigator` object. - denotes an attribute that is not included in, or that do not have a MNCE sufficiently high for, a given dataset.

Attribute	Gen. Aud.	Intranet	Enrolled
W.innerHeight	0.181	0.184	0.156
WebRTC fingerprinting	0.144	-	-
W.outerHeight	0.117	-	-
Presence of fonts	0.110	-	-
N.plugins	0.100	-	-
W.outerWidth	0.074	-	-
UNMASKED_RENDERER_WEBGL	0.073	-	0.113
First bounding box height	0.070	-	-
S.availHeight	0.058	0.092	-
W.screenY	0.049	0.095	-
W.screenX	0.047	0.123	-
N.userAgent	0.046	-	-
C.connection.downlink	-	0.101	-
Our HTML5 canvas (PNG)	-	0.108	-
N.[...].enumerateDevices()	-	-	0.121

entropy $H(a_j|a_i)$ of a_j given a_i is expressed as

$$H(a_j|a_i) = - \sum_{v \in V_i, w \in V_j} p(e_v^i, e_w^j) \log \frac{p(e_v^i, e_w^j)}{p(e_v^i)} \quad (4.9)$$

Finally, we normalize the conditional entropy $H(a_j|a_i)$ by the maximum entropy H_M .

Summary of the Attributes Correlation Results

Numerous attributes have a null conditional entropy when the value of another attribute is known: 49 attributes for the general audience dataset, 134 for the intranet dataset, 142 for the enrolled dataset, and 229 for the university dataset. The large difference between the general audience dataset and the university dataset is explained by three factors. First, the general audience dataset contains 15 thousand more attributes than the university dataset, which increases the chances to encounter more combinations of attribute values. Second, the browsers of the university dataset are standardized, hence they tend to have the same attribute values and combination of them. Third, the conditional entropy of an attribute is equal to its entropy if this attribute is not correlated with another one. As the conditional entropy of an attribute is capped by its entropy, and the attributes of the university dataset are lowly distinctive, the obtained conditional entropy are low. Few attributes provide a conditional entropy above 1 bit when the value of another attribute is known: none for the university dataset, 3 for the enrolled dataset, 6 for the intranet dataset, and 12 for the general audience dataset. These attributes concern the browser window size, the browser components (e.g., the list of fonts), the HTML5 canvases, the verbose textual information (e.g., the `UserAgent`), the network, and the hardware components (e.g., the screen). Although the attributes are correlated with at least another, their combination still provides distinctiveness on the fingerprint level.

Detailed Analysis of the Attributes Correlation of the General Audience Dataset

We can see three parts in the figure of the normalized conditional entropy of the general audience dataset. First, 19 attributes have a low normalized entropy of less than 10^{-3} , and the NCE when knowing another attribute is at most as much. Few different values have been observed for these attributes. Then, 194 attributes have an average NCE between 10^{-3} and 10^{-1} . The minimum normalized conditional entropy of these attributes is at most 0.042, hence there exists another attribute that can be used to efficiently infer their value. Finally, 40 attributes have an average NCE higher than 10^{-1} , and generally have a higher MNCE. These attributes help to efficiently distinguish browsers, and are less correlated to other attributes.

We have 49 attributes that have a null MNCE, which can be completely inferred when another attribute is known. Moreover, 192 attributes have a MNCE comprised in the range $]0;0.045]$, hence knowing the value of another attribute helps to infer their value, but not completely. Finally, 12 attributes have a MNCE higher than 0.045, which is equivalent to having a minimum conditional entropy higher than 1 bit. They consist of highly distinctive attributes that concern the size of the screen (e.g., `W.screenX`) or of the window (e.g., `W.innerHeight`), browser components (e.g., fonts, plugins), and verbose information about the browser (e.g., `N.userAgent`) or about external components (e.g., WebRTC fingerprinting).

Detailed Analysis of the Attributes Correlation of the Intranet Dataset

We observe three parts in the figure of the normalized conditional entropy of the intranet dataset. First, 77 attributes have a low normalized entropy of less than 0.01, and the NCE when knowing another attribute is at most as much. Then, 145 attributes have an average NCE between 0.008 and 0.17. The MNCE of these attributes is at most 0.03, hence there exists another attribute that can be used to efficiently infer their value. Finally, 33 attributes have an average NCE higher than 0.17, and generally have a higher minimum normalized conditional entropy. These attributes help to efficiently distinguish browsers, and are less correlated to other attributes.

We have 134 attributes that have a null MNCE, which can completely be inferred when another attribute is known. Moreover, 115 attributes have a MNCE comprised in the range $]0;0.076]$, hence knowing the value of another attribute helps to infer their value, but not completely. Finally, 6 attributes have a MNCE higher than 0.076, which is equivalent to having a minimum conditional entropy higher than 1 bit. They consist of highly distinctive attributes that concern the size of the screen (e.g., `W.screenX`) or of the window (e.g., `W.innerHeight`), our designed HTML5 canvas, and the `download` property of the `navigator.connection` object.

Detailed Analysis of the Attributes Correlation of the University Dataset

We observe three parts in the figure of the normalized conditional entropy of the university dataset. First, 130 attributes have a low normalized entropy of less than 0.01, and the NCE when knowing another attribute is at most as much. Among these attributes, only the `[msw, w]ebdriver` property of the `navigator` object has a non-null minimum normalized conditional entropy of 0.002. Then, 62 attributes have an average NCE between 0.01 and 0.1. The MNCE of these attributes is null, at the exception of the `sampleRate` property of the audio context object which has a MNCE of 0.034. Finally, 46 attributes have an average NCE higher than 0.1, but most of them have a null MNCE.

Due to the specificities of the university dataset, most of its attributes are highly correlated with at least another one. Indeed, 229 attributes have a null MNCE, and can be completely inferred when another attribute is known. Only 9 attributes have a non-null MNCE, and they all show a MNCE that is lower than the equivalent of a conditional entropy of 1 bit. Table 4.3 lists these attributes, their MNCE, and the equivalent minimum conditional entropy in bits. They consist of highly distinctive attributes that concern the size of the window (e.g., the `availHeight` property), elements of the browser (e.g., the fonts, the WebGL extensions), hardware components or configurations (e.g., the graphical card driver, the default sample rate, the input and output media devices), and our designed HTML5 canvas.

Table 4.3: The minimum normalized conditional entropy and the minimum conditional entropy in bits of the attributes that have a non-null MNCE for the university dataset. A refers to an initialized audio context, N to the navigator object, WG to an initialized WebGL context, and S to the screen object.

Attribute	MNCE	Min. Cond. Entropy (bits)
UNMASKED_RENDERER_WEBGL	0.067	0.540
List of fonts (inferred)	0.063	0.506
A.sampleRate	0.034	0.276
N.mediaDevices.enumerateDevices()	0.034	0.274
Our HTML5 canvas (PNG)	0.032	0.256
N.buildID	0.031	0.253
WG.getSupportedExtensions()	0.019	0.153
S.availHeight	0.014	0.115
N.[msW, w]ebdriver	0.002	0.019

Detailed Analysis of the Attributes Correlation of the Enrolled Dataset

We observe three parts in the figure of the normalized conditional entropy of the enrolled dataset. First, 59 attributes have a low normalized entropy of less than 0.006, and the NCE when knowing another attribute is at most as much. Then, 89 attributes have an average NCE between 0.006 and 0.077. The MNCE of these attributes is at most 0.013, hence there exists another attribute that can be used to efficiently infer their value. Finally, 106 attributes have an average NCE higher than 0.077, and generally have a higher minimum normalized conditional entropy. These attributes help to efficiently distinguish browsers, and are less correlated to other attributes.

We have 142 attributes that have a null MNCE, which can completely be inferred when another attribute is known. Moreover, 109 attributes have a MNCE comprised in the range $]0; 0.093]$, hence knowing the value of another attribute helps to infer their value, but not completely. Finally, 3 attributes have a MNCE higher than 0.093, which is equivalent to having a minimum conditional entropy higher than 1 bit. They concern the graphical driver, the input and output media devices, and the size of the browser window.

Table 4.4: The dynamic attributes included in the four datasets. - denotes a non-included attribute. * indicates that the attribute is generated using randomized instructions, hence is always unique.

Dynamic attribute	Gen. Aud.	Intra.	Univ.	Enroll.
Our HTML5 canvas (PNG)	0.420	0.599	0.199	*
Our HTML5 canvas (JPEG)	0.399	-	-	-
HTML5 canvas similar to [124] (PNG)	0.385	0.471	-	-
HTML5 canvas inspired by [127] (PNG)	0.353	0.459	-	-
HTML5 canvas inspired by [127] (JPEG)	0.312	-	-	-
WebGL canvas	0.263	0.274	0.033	0.411
Audio FP simple	0.153	-	-	-
Audio FP advanced	0.147	-	-	-
Audio FP advanced frequency data	0.161	-	-	-

4.3.7 Focus on the Dynamic Attributes

In this section, we discuss the properties of the dynamic attributes that we include in the fingerprinting probe of our experiments (see Section 3.1). As the included dynamic attributes vary between the datasets, Table 4.4 displays the normalized entropy of the dynamic attributes included in each dataset.

HTML5 Canvas

The HTML5 canvas consists into asking the browser to draw an image by using the canvas API within the two-dimensional context. This method is already studied in several works, whether it is about its effectiveness [36, 127, 81], its use on the web [57, 129], or the distinctiveness provided by various sets of instructions [124]. However, to the best of our knowledge, no study evaluates the different properties (e.g., distinctiveness, stability, collection time) offered by various sets of complex instructions (i.e., mixes of texts, emojis, and mathematical curves, drawn using different colors).

Format Comparison We observe that canvases are less distinctive in JPEG format than in PNG format, but are more stable. The numbers given below concern the general audience dataset, which is the only dataset which contains

canvases in JPEG format. For example, the custom canvas has a normalized entropy of 0.420 when exported in PNG, against 0.399 for the JPEG version. However, the PNG version has a sameness rate of 92.16%, against 93.59% for the JPEG version. These differences are due to distinct images in PNG format ending up the same after the lossy compression of the JPEG format. Acar et al. [2] assumes that a canvas generated in a lossy compression format as JPEG is not an indicator of a fingerprinting attempt. Although the JPEG version provides a lower distinctiveness than the PNG version, we show that it is still highly distinctive when generated from a complex set of instructions. Indeed, it is the second most distinctive attribute among the attributes of the general audience dataset (see Section 4.3.2). The time overhead induced by the additional extraction in JPEG format is negligible. For example, the custom canvas has a median collection time (MCT) increased by 5ms for the JPEG version, compared to the PNG version that has a MCT of 260ms. The PNG and the JPEG versions are also highly correlated, as knowing the value in the PNG format of the custom canvas leaves a normalized conditional entropy (NCE) of 5.28×10^{-4} on the value of the JPEG format, whereas the inverse provides a NCE of 0.021.

Instructions Comparison The properties of the three PNG canvases differ, with the custom canvas being the most distinctive. First, the Morellian canvas is an enhanced version of the AmIUnique canvas with additional curves. This enhancement provides an increase of the distinctiveness, but also a decrease of the sameness rate. The Morellian canvas shows a normalized entropy of 0.385 for the general audience dataset and of 0.471 for the intranet dataset, against respectively 0.353 and 0.459 for the AmIUnique canvas. For these same datasets, the Morellian canvas has a respective sameness rate of 94.71% and of 92.55%, against 98.64% and 93.64% for the AmIUnique canvas. Then, the custom canvas is more complex than the Morellian canvas, as it includes several emojis, two strings that include Swedish letters, many overlapping ellipses, a color gradient background, and a rotation of all these elements if the functionality is available. These improvements provide a higher normalized entropy of 0.420 and of 0.599 for the general audience and intranet datasets. As for the university dataset, the normalized entropy is lower at 0.199 due to the standardized browsers, but it is still the fourth most distinctive

attribute. The custom canvas also shows a lower sameness rate of 92.16% and of 80.58% for the general audience and intranet datasets. Due to the specificities of the university dataset, the sameness rate of the custom canvas for it is lower at 70.60%. The main drawback of adding more complexity to the custom canvas is the temporal cost, as it has a MCT of 260ms and 181ms for the general audience and intranet datasets. The MCT of the Morellian canvas is respectively of 37ms and 19ms for these datasets, and the MCT of the AmIUnique canvas is of 31ms and 14ms. Finally, knowing the value of the custom canvas leaves less variability on the value of the two other canvases than the opposite. When knowing the value of the Morellian canvas, the custom canvas has a normalized conditional entropy (NCE) of 0.079 for the general audience dataset, and of 0.134 for the intranet dataset. On the opposite, when knowing the value of the custom canvas, the Morellian canvas has a NCE of 0.044 and of 0.006 for these datasets. As for the AmIUnique canvas, knowing its value results in the custom canvas providing a NCE of 0.103 for the general audience dataset, and of 0.145 for the intranet dataset. On the opposite, knowing the value of the custom canvas results in a NCE of 0.037 and 0.005 on the AmIUnique canvas for these datasets. To conclude, adding more instructions for the canvas drawing usually provides more distinctiveness, as each instruction can induce a difference between browsers, at the cost of additional computation time. Moreover, each additional instruction can constitute an instability factor that negatively impacts the sameness rate.

WebGL Canvas

The WebGL canvas is, for the general audience dataset, the 27th most distinctive attribute with a normalized entropy of 0.263, the 30th for the intranet dataset with a normalized entropy of 0.274, and the 18th for the enrolled dataset with a normalized entropy of 0.411. However, for the university dataset, due to the standardized software and hardware configurations, it is only the 96th most distinctive attribute, with a low normalized entropy of 0.033. It is more stable compared to the HTML5 canvases, but is still among the most unstable attributes. For the university dataset, it provides a sameness rate of 93.13% and is the 146th most stable attribute. For the other datasets, it shows a sameness rate comprised be-

tween 89.33% and 98.97%, and is ranked between the 214th and the 237th most stable attribute. The median collection time (MCT) of the WebGL canvas for the general audience, intranet, and university datasets is comprised between 12ms and 41ms. The WebGL canvas is ranked between the 164th and the 179th fastest to collect for these datasets, according to the MCT. For the enrolled dataset, it shows a higher MCT of 313ms, and is ranked as the 202nd fastest to collect.

Web Audio Fingerprinting

Only the general audience dataset includes audio fingerprinting attributes, hence the results given below are obtained from this dataset. The most distinctive audio attribute is the *audio fingerprinting advanced frequency data* (AFA-FD) attribute that has a normalized entropy of 0.161, followed by the *audio fingerprinting simple* (AFS) (0.153), and the *audio fingerprinting advanced* (AFA) (0.147). They all have a sameness rate of approximately 95%. Their values are the string representations of floating-point numbers, hence they have a median size of 17 bytes. The simple process has a median collection time (MCT) of 1.4 seconds, and the advanced process has a MCT of 1.7 seconds. The AFA-FD is collected from the advanced version, and has a MCT increased by 4ms compared to the AFA.

4.4 Conclusion

In this chapter, we analyze the browser fingerprints of our four datasets according to the properties inspired by biometric authentication factors that we identified. The general audience, intranet, and enrolled datasets show similar results. On the contrary, the university dataset provides particular results due to the standardization of the browser population and the browser profiles being shared between the computers of the university. The intranet, university, and enrolled experiments were still running during the containment in France due to the COVID-19 pandemic. The university experiment was impacted as no one could use the university computers. Experimenters still accessed the websites of the intranet and enrolled experiments, hence these two experiments were not significantly impacted by the containment.

We achieve a unicity rate above 81% when considering the time-partitioned datasets, at the exception of the university dataset for which it goes down to 11%. Between two observations of the fingerprint of a browser, it is expected that more than 81% of their attributes stay identical, even when separated by several months. This rate falls down to 68.7% for the university dataset, mainly due to the browser profiles being shared between various computers. The majority (95%) of the fingerprints are collected in less than 3.7 seconds, at the exception of the general audience dataset for which 95% of them are collected in less than 10.5 seconds. All the fingerprints of our four datasets take at most 22.3 kilobytes to store, at the exception of an outlier in the general audience dataset. By simulating a simple verification mechanism, we achieve an equal error rate comprised between 0.61% and 4.30%. The university dataset, due to the low distinctiveness and stability of its fingerprints, shows a higher equal error rate of 29.42%. We confirm the conclusions of the previous studies [204, 127, 81] about the fingerprints of mobile browsers being less distinctive. However, the loss is dissimilar between the datasets. The unicity rate of the time-partitioned datasets of the general audience dataset falls down to 42% for the mobile browsers, compared to 84% for the desktop browsers. These rates are respectively of 93.6% and 96.5% for the enrolled dataset. Only the general audience dataset includes consecutive mobile fingerprints over several time ranges, which tend to be more stable than the desktop fingerprints. About the size and the collection time of mobile fingerprints, they are lighter and take more time to collect than the desktop fingerprints.

We measured the satisfaction of 682 participants who tested our test authentication mechanism that relies on browser fingerprints by the Net Promoter Score. This method classifies the participants as detractors, passives, or promoters given their rating. The test authentication mechanism polarizes the experimenters more than the current authentication methods: only 38% of the experimenters are classified as passive, compared to 54% for the current methods. About the test authentication mechanism, 40% of the experimenters are promoters and 22% of them are detractors, compared to 14% of promoters and 20% of detractors about the current authentication methods. The net promoter score of the test authentication mechanism is of 18, which is three times more than the score of the current authentication methods being of 6.

When breaking down the analysis to the attributes, we observe that the attributes contribute differently to each property. The university dataset once again differs from the others. The majority of the attributes have at most 100 distinct values, the rate goes from 63% for the general audience dataset to 91% for the intranet and enrolled datasets. As for the university dataset, its attributes have at most 12 distinct values. About the distinctiveness, the majority of the attributes provide a normalized entropy that is strictly higher than 0.0 and lower than 0.3. The university dataset falls apart, as nearly half of its attributes have a null normalized entropy, and all of them have a normalized entropy lower than 0.26. The majority of the attributes provide a sameness rate above 99%. The proportion is comprised between 57% for the university dataset, and 80% for the general audience dataset. Less than 40 attributes have a median collection time above 5ms, they typically consist in the processing of media objects (e.g., the canvas), and in the detection of extensions or of browser components. Few attributes are heavy, as only between 14 and 20 attributes weigh more than 100 bytes. When looking at the correlation between the attributes, we observe that most of the attributes are correlated, as knowing another attribute results in their conditional entropy being lower than 1 bit. Although attributes are correlated with each other, their combination still provides a high distinctiveness. When focusing on the three types of dynamic attributes that we include in our fingerprinting probe, we observe that compared to the other attributes, they are generally more distinctive, less stable, and take more time to collect.

Overall, considering our attributes and browser populations, we achieve a high level of distinctiveness and stability. The university dataset constitutes an exception and shows a decrease of both distinctiveness and stability. This is due to the standardized browser population and the browser profiles being shared between the university computers. As already observed by previous studies, the fingerprints of the mobile browsers are less distinctive than the ones of the desktop browsers. Although the proportion of unique fingerprints is two times lower for the mobile fingerprints than for the desktop fingerprints considering the general audience dataset, the loss is nearly null for the enrolled dataset. The high level of distinctiveness and stability results in a high accuracy when processing a simple verification mechanism on the datasets excluding the university dataset. This

mechanism achieves an equal error rate from 0.61% to 4.30% for these datasets. As for the resource consumption, the storage requirement of the fingerprints can be handled by the actual storage capacities. However, their collection time is rather high for an online authentication context. When breaking the analysis down to the attribute level, we remark that few attributes are highly unstable or take time to collect. We could remove these attributes to increase the stability of the fingerprints and reduce their collection time. However, these attributes tend to provide a high distinctiveness. The choice of the attributes to hold, and consequently to remove, could be done using a trade-off between the distinctiveness, the instability, and the collection time of the attributes. As a non-negligible proportion of the attributes are highly correlated with at least another one, the choice have to be made on complete attribute sets and not individually on each attribute. In the next chapter, we propose an attribute selection method that takes these aspects into account.

Chapter 5

Attribute Selection Framework

Hundreds of attributes are available for a verifier to build her fingerprinting probe. However, including all of them is impractical as each attribute comes with a cost (e.g., additional collection time). In this chapter, we propose a framework for the verifier to dimension her fingerprinting probe according to a trade-off between security and usability. The two adversarial participants are the verifier and the attacker, as depicted in Figure 5.1. The verifier aims to protect the users of her web platform, using an authentication mechanism based on browser fingerprinting. The verifier stores the fingerprint of the usual browser of each user. On each login, the fingerprint of the browser in use is matched against the fingerprint that is stored for the claimed account. The attacker tries to impersonate the users by submitting specially crafted fingerprints. The aim of the verifier is to limit the reach of the attacker, also called *sensitivity* below, which is measured as the proportion of impersonated users. To do so, she builds a fingerprinting probe that integrates one or more attributes selected among the hundreds¹ that are accessible (see Chapter 4). On the one hand, the addition of an attribute to the probe can strengthen the distinctiveness of browsers, hence reducing the sensitivity. On the other hand, each addition comes with a *usability cost* that may render the probe impractical in an online authentication context. Indeed, each

¹Most attributes are properties accessed through the browser that are limited by its functionalities. Other attributes are items whose presence is checked (e.g., the fonts [83], the extensions [199]), or the computation of specific instructions (e.g., the HTML5 canvas [36]). These are limited by the available items or instructions, which can be large (e.g., more than 2^{154} for the canvas [124], nearly 30 thousand detectable extensions [106]).

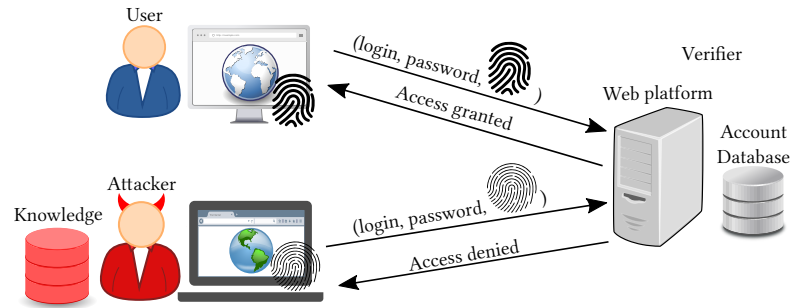


Figure 5.1: Example of a web authentication mechanism based on browser fingerprinting and a failed attack.

attribute consumes storage space (up to hundreds of kilobytes [36]), collection time (up to several minutes [152, 155, 157, 191, 193, 179]), and can increase the instability of the generated fingerprints [214]. For example, considering all of our attributes leads to a fingerprint taking 9.98 seconds on average to collect, which is impractical for the user. Moreover, some attributes are strongly correlated together (see Section 4.3.6), and including them only increases the usability cost without reducing the sensitivity. Due to these correlations, picking attributes one by one independently may lead to poor sensitivity and usability scores.

Previous works only consider the well-known attributes [56, 127, 81], remove the attributes of the lowest entropy [214], iteratively pick the attribute of the highest weight (typically the entropy) until a threshold is reached [146, 111, 65, 24, 91, 208], or evaluate every possible set [69]. The entropy measures the skewness of the distribution of fingerprints or attribute values. As pointed out by Acar [4], it does not take the worst cases into account (i.e., the most common values that attackers can submit similarly to dictionary attacks on passwords [28]). Moreover, fingerprints cannot be compared identically like passwords due to their evolution through time. The attackers do not need to find the exact fingerprint of a victim, but one that is similar enough to deceive the verification mechanism.

In this chapter, we propose FPSelect, a framework that allows a verifier to select the attributes² to include into her fingerprinting probe such that (1) the

²We emphasize that the candidate attributes can contain dynamic attributes, which can be used to implement challenge-response mechanisms that resist fingerprint replay attacks [124, 184]. We study nine instances of three dynamic attributes, which are the HTML5 canvas [36], the WebGL canvas [153], and audio fingerprinting methods [179].

sensitivity against powerful attackers knowing the fingerprint distribution of the protected users (i.e., the worst-case distribution for the verifier) is bounded and the bound is set by the verifier, and (2) the usability cost³ of collecting, storing, and using these attributes is close to being minimal. FPSelect is parameterized with the sensitivity requirement, the number of submissions that the attacker is deemed able to execute, and a representative sample of the fingerprints of the users.

The problem could be solved by exploring exhaustively the space of the possible attribute sets, evaluating the sensitivity and the usability cost of each set. This is, however, infeasible as the number of attribute sets grows exponentially with the number of attributes⁴. Moreover, we show below that the problem of finding the optimal attribute set is NP-hard. To the best of our knowledge, this is the first work that allows verifiers to dimension their fingerprinting probe in a sound manner, by quantifying the security level to reach, and selecting an attribute set that satisfies this level at a low usability cost.

Our key contributions are the following:

- We formalize the *attribute selection problem* that a verifier has to solve to dimension her probe. We show that this problem is NP-hard because it is a generalization of the Knapsack Problem. We define the model of the dictionary attacker, whose adversarial power depends on the knowledge of a fingerprint distribution. We propose a measure to quantify the sensitivity of a probe given a browser population and the number of fingerprints that the attacker is able to submit. We propose a measure of the usability cost that combines the size of the generated fingerprints, their collection time, and their instability.
- We propose a heuristic algorithm for selecting an attribute set that satisfies a higher bound on the sensitivity and reduces the usability cost. We express this as a search problem in the lattice of the power set of the candidate attributes. This algorithm is inspired by the Beam Search algorithm [103] and is part of the Forward Selection algorithms [194].

³Any usability cost can be plugged (e.g, the privacy cost of including an attribute) provided that it is monotonic.

⁴Obviously, this discards as well the manual selection of attributes.

- We evaluate the FPSelect framework on two real-life fingerprint datasets, and compare it with common attribute selection methods based on the entropy and the conditional entropy. We show experimentally that FPSelect finds attribute sets that have a lower usability cost. The attribute sets found by FPSelect generate fingerprints that are 12 to 1,663 times smaller, 9 to 32,330 times faster to collect, and with 4 to 30 times less changing attributes between two observations, compared to the candidate attributes and on average. Compared to the baselines, the attribute sets found by FPSelect generate fingerprints that are up to 97 times smaller, are collected up to 3,361 times faster, and with up to 7.2 times less changing attributes between two observations, on average.

This chapter is organized as follows. Section 5.1 defines the attack model and the attribute selection problem. Section 5.2 describes the resolution algorithm and the proposed illustrative measures of sensitivity and usability cost. Section 5.3 provides the results obtained by processing our framework and the baselines on two real-life fingerprint datasets. Finally, Section 5.4 concludes.

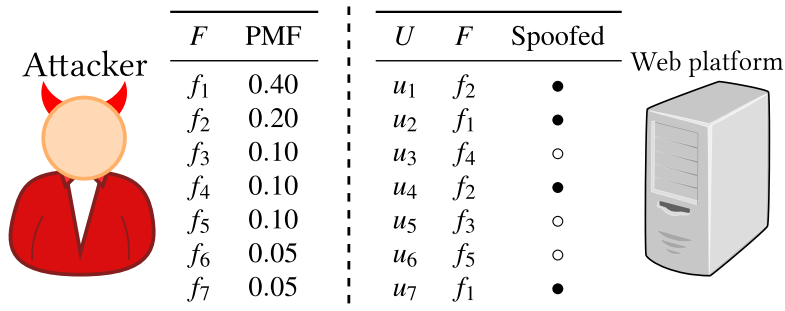
5.1 Problem Statement

In this section, we first present the considered authentication mechanism that relies on browser fingerprinting. Then, we describe how we model the attacker given his knowledge and possible actions. Finally, we pose the attribute selection problem that we seek to solve, and provide an example to illustrate the problem.

5.1.1 Authentication Mechanism

We consider the architecture and the three participants depicted in Figure 5.1. The authentication mechanism is executed on a *trusted web platform* and aims at authenticating *legitimate users* based on various authentication factors, including their browser fingerprint (in addition to, e.g., a password). For the sake of precision, we focus on the browser fingerprint and ignore the other factors.

A user is enrolled by providing his browser fingerprint to the verifier who stores it. During the authentication of a user, the fingerprint of the browser in use is



Attacker	F	PMF	U	F	Spoofed
	f_1	0.40	u_1	f_2	•
	f_2	0.20	u_2	f_1	•
	f_3	0.10	u_3	f_4	○
	f_4	0.10	u_4	f_2	•
	f_5	0.10	u_5	f_3	○
	f_6	0.05	u_6	f_5	○
	f_7	0.05	u_7	f_1	•

Figure 5.2: Example of an attacker instantiated with his knowledge of a probability mass function (PMF) over the fingerprints F , and a web platform protecting a user population U with their fingerprint. We consider a limit of two submissions and a strict comparison between the fingerprints. The attack dictionary is composed of f_1 and f_2 , resulting in the shown impersonated users.

collected by the fingerprinting probe of the verifier, and is compared with the fingerprint stored for the claimed account. If the collected fingerprint matches with the one stored, the user is given access to the account, and the stored fingerprint is updated to the newly collected one. The comparison is done using a *matching function* (i.e., a similarity function between two fingerprints that authorizes differences), as fingerprints are known to evolve (see Section 4.2). Any matching function can be used provided that it is monotonic (i.e., if two fingerprints match⁵ for an attribute set C , they also match for any subset of C). We explain in Section 5.1.3 the need for the monotonicity requirement, and refer to Section 5.3.1 for an example of a matching function. We consider one browser per user, but such authentication mechanism can be designed to support multiple browsers (see Appendix F).

5.1.2 Attack Model

The high-level goal of the attacker is to impersonate legitimate users in a *limited number of submissions* with the help of his knowledge, by forging a *fingerprint attack dictionary* similarly to dictionary attacks on passwords [28]. Figure 5.2 illustrates the attack that we consider. It shows an attacker with his knowledge of

⁵We stress that the monotonic property does not depend on the attributes.

a fingerprint distribution, a population of protected users with their fingerprint, and the impersonated users. We define the attacker model in terms of background knowledge and possible actions, which are provided below and described further in the following subsections.

1. The attacker cannot tamper with the web platform.
2. The attacker cannot tamper with, nor eavesdrop, the communication between the users and the web platform.
3. The attacker knows the attributes of the probe.
4. The attacker knows a fingerprint distribution.
5. The attacker can submit a limited number of arbitrary fingerprints.

Background Knowledge

The attacker can retrieve the attributes of the fingerprinting probe (assumption 3) by reverse-engineering the probe (e.g., static or dynamic analysis of the probe [19], analysis of the network packets).

The attacker knows the *domain of the fingerprints*, and can infer a fingerprint distribution (assumption 4) from documentation [186], datasets [34], or statistics⁶ available online. He can also leverage phishing attacks [210], a pool of controlled browsers [171], or stolen fingerprints [143]. The weakest attacker is the one that lacks knowledge, and considers that the values of the attributes and fingerprints are uniformly distributed. His strategy is then to cover a space as large as possible of the fingerprint possibilities in the number of submissions authorized by the verifier. The strongest attacker is the one that manages to infer the exact fingerprint distribution among the users protected by the verifier. Additionally, our work can be easily extended to the attackers that partially know the fingerprints of targeted users⁷.

⁶<http://carat.cs.helsinki.fi/statistics>

⁷We do not consider the attackers that exactly know the fingerprint of the users they target (or their local configuration) because they are able to bypass trivially any fingerprinting authentication mechanism.

Actions

Tools exist for controlling the attributes⁸ that compose the fingerprint (assumption 5), like Blink [128] or Disguised Chromium Browser [22]. Commercial solutions also exist, like AntiDetect [16] or Multilogin [156]. An attacker can also automatically alter the network packet that contains the fingerprint using tools like BurpSuite⁹. As these attacks are online guessing attacks [28, 220], we assume that the attacker is limited to a number of submissions per user. The verifier instantiates an attacker by his knowledge of a fingerprint distribution, and by the number of submissions to which he is limited, to measure his reach.

5.1.3 Attribute Selection Problem

The defense problem consists into selecting the attribute set that composes the fingerprinting probe, to resist against an instantiated attacker and minimize the usability cost. On the one hand, including an attribute can reduce the reach of an attacker¹⁰, which we call the *sensitivity* and measure as the *proportion of impersonated users*. On the other hand, it increases the *usability cost* of the mechanism. For example, the generated fingerprints take more space to store, can take more time to collect, and can be more difficult to recognize due to the potentially induced instability.

Problem Formulation

The *Attribute Selection Problem* consists in finding the attribute set that provides the lowest usability cost, and keeps the sensitivity below a threshold α set by the verifier¹¹. Let A denote the set of the candidate attributes. We consider

⁸These tools are able to control both the fixed and the dynamic attributes.

⁹<https://portswigger.net/burp>

¹⁰Including an attribute adds one more information to distinguish different browsers. However, an attribute provides no additional distinctiveness if it is correlated with another one. For example and considering the case depicted in Table 5.1, adding the attribute `Timezone` when the attribute `Language` is already selected does not provide any distinctiveness.

¹¹The *sensitivity threshold* α is defined by the verifier according to her security requirements. These requirements depend on the type of website that is to protect (e.g., a bank, a forum) and the contribution of browser fingerprints (e.g., the only secondary authentication factor, an additional verification among others [202]).

User	CookieEnabled	Language	Timezone	Screen
u_1	True	fr	-1	1080
u_2	True	en	-1	1920
u_3	True	it	1	1080
u_4	True	sp	0	1920
u_5	True	en	-1	1080
u_6	True	fr	-1	1920

Table 5.1: Example of fingerprints shared by users.

an attribute set $C \subseteq A$, its usability cost $c(C)$, and its sensitivity $s(C)$. Any measure of usability cost and sensitivity can be plugged in FPSelect provided that it is *monotonic*. Indeed, the usability cost is required to be *strictly increasing* as we add attributes to an attribute set (e.g., the additional attributes are stored, which increases the storage cost). The sensitivity is required to be *monotonically decreasing* as we add attributes to an attribute set¹². Indeed, adding an attribute to an attribute set should not higher the sensitivity because the added attribute either adds distinguishing information to the fingerprints or adds no information if it is strongly correlated with another attribute. For illustrative purposes, we propose measures of sensitivity and usability cost in Section 5.2.3. The ASP is thus formalized as searching for $\arg \min_{C \subseteq A} \{c(C) : s(C) \leq \alpha\}$.

Problem Illustration

To illustrate the problem, we propose an example of a fingerprint distribution in Table 5.1. We consider an attacker who managed to infer the exact same distribution, and who is able to submit one fingerprint per user. If we solely include the `CookieEnabled` attribute which provides no distinctiveness, this attacker can impersonate every user by submitting the `True` value. Whereas including the `Language` and `Screen` attributes leads to unique fingerprints, which reduces the sensitivity to a sixth. Ignoring the `CookieEnabled` attribute reduces the usability cost without increasing the sensitivity. There is also an example of correlation. The

¹²The monotonicity requirement of the matching function comes from the monotonicity requirement of the sensitivity. Indeed, if the matching function was not monotonic, adding an attribute could result in a loss of distinctiveness (i.e., it is harder for the matching function to distinguish two browsers) and consequently in an increase of the sensitivity.

Language is the most distinctive attribute, followed by the **Timezone**. In this case, taking the two most distinctive attributes does not improve the distinctiveness compared to considering **Language** alone.

5.2 Attribute Selection Framework

This section is dedicated to the description of our attribute selection framework. First, we show that the Attribute Selection Problem (ASP) is NP-hard because it is a generalization of the Knapsack Problem (KP), and remark that the ASP can be seen as a lattice of partial KP. Second, and consequently, we propose a greedy heuristic algorithm for finding solutions to the problem. Finally, we propose illustrative measures for the sensitivity and the usability cost.

5.2.1 Similarity to the Knapsack Problem

The *Knapsack Problem* (KP) [108] is a NP-hard problem that consists into fitting valued-items into a limited-size bag to maximize the total value. More precisely, given a bag of capacity W and n items with their value v_i and their weight w_i , we search for the item set that maximizes the total value and which total weight does not exceed W . In this section, we show that the ASP is a generalization of the KP, therefore the ASP is NP-hard. We also provide a way to model the ASP as a lattice of partial KP.

Generalization of the KP to the ASP

First, we remark that the ASP can be solved by picking attributes until we reach the sensitivity threshold, or by starting from the candidate attributes and removing attributes successively without exceeding the threshold. We consider the latter and start from the set A of the candidate attributes. The *value* of an attribute set C is the cost reduction compared to the candidate attributes, formalized as $v(C) = c(A) - c(C)$. The *value* of an attribute a is the cost reduction obtained when removing a from C , formalized as $v(a|C) = c(C) - c(C \setminus \{a\})$. The *weight* of an attribute set C is its sensitivity with $w(C) = s(C)$. The *weight* of an attribute a is the additional sensitivity induced by the attribute removal, formal-

ized as $w(a|C) = s(C \setminus \{a\}) - s(C)$. The *capacity* W is the maximum sensitivity allowed, hence $W = \alpha$. As we remove attributes, the value increases (i.e., the usability cost decreases), and the weight (i.e., the sensitivity) may increase.

Theorem 1. *The Attribute Selection Problem is NP-hard.*

Proof. We consider a simple case where the attributes are not correlated. The weight and the value of the attribute a_i does not depend on the attributes already included in the probe, and is simply defined as w_i and v_i . We obtain a Knapsack Problem consisting into picking the attributes to remove from A , to maximize the total value and keep the weight under the threshold W . The ASP is therefore a generalization of the KP with relative weights and costs, making it at least as hard as the KP which is NP-hard. The Attribute Selection Problem is therefore NP-hard. \square

The Attribute Selection Problem as a Lattice of Partial Knapsack Problems

The Attribute Selection Problem can be modeled as a lattice of partial Knapsack Problems (KP). We consider the deletive way that starts from the set A of the candidate attributes and removes attributes without exceeding the threshold. The initial partial KP consists into picking attributes from A to increase the value and keep the weight under W . The value and weight of each attribute $a \in A$ is $v(a|A)$ and $w(a|A)$. Once we pick an attribute a_p , a new partial KP arises: the item set is $A \setminus \{a_p\}$, the capacity is $W - w(a_p|A)$, and the value and weight of each attribute $a \in A \setminus \{a_p\}$ is now $v(a|A \setminus \{a_p\})$ and $w(a|A \setminus \{a_p\})$. Recursively, it holds for any set R of attributes to remove. The item set is then $A \setminus R$, the capacity is $W - w(R)$, and the value and weight of each attribute $a \in A \setminus R$ are $v(a|A \setminus R)$ and $w(a|A \setminus R)$. Following this, we are given a lattice¹³ of partial KP to solve recursively, each node being a partial solution R , until we reach unfeasible problems (i.e., empty set of items, no more item can fit) and find a final solution among the partial solutions that reach this limit.

¹³This can be seen as a tree, but some paths lead to the same node. Indeed, removing the attributes a_1 then a_2 from A leads to the same partial problem as removing a_2 then a_1 .

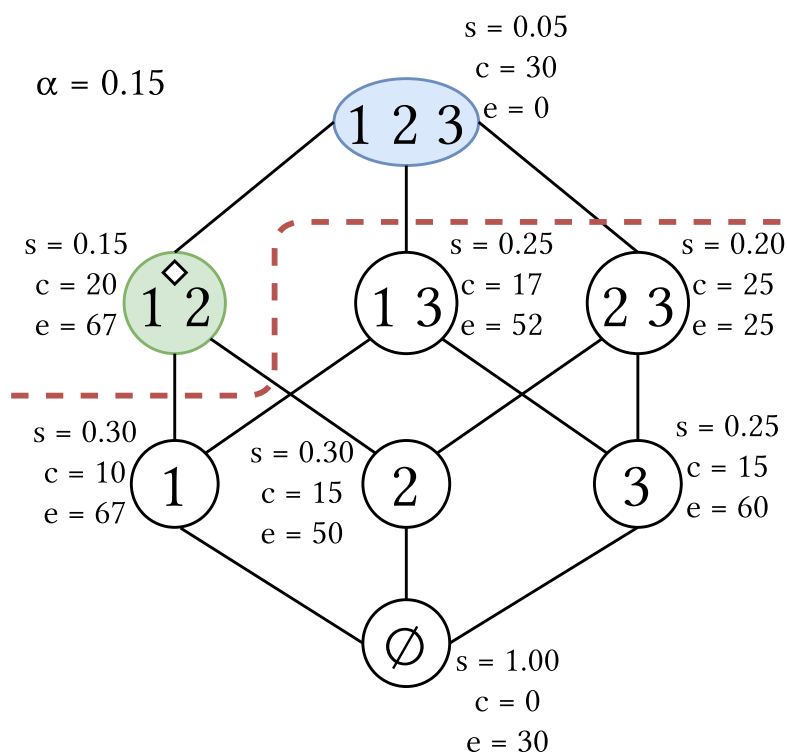


Figure 5.3: Example of a lattice of attribute sets, with their cost c , their sensitivity s , and their efficiency e . The blue node satisfies the sensitivity, the white nodes do not, and the green node with a diamond satisfies the sensitivity and minimizes the cost. The red line is the satisfiability frontier.

5.2.2 Lattice Model and Resolution Algorithm

In this section, we present how we model the possibility space as a lattice of attribute sets, and describe the greedy heuristic algorithm to approximately solve the ASP.

Lattice Model

The elements of the lattice are the *subsets* of A (A included) and the order is the *subset relationship* so that $C_i \prec C_j$ if, and only if, $C_i \subset C_j$. The efficiency of an attribute set C is the ratio between its cost reduction (i.e., $c(A) - c(C)$) and its sensitivity. Figure 5.3 shows an example of such lattice. The *satisfiability frontier* represents the transition between the attribute sets that satisfy the sensitivity threshold, and those that do not. The attribute sets just above this frontier satisfy

the sensitivity threshold at a lower cost than any of their supersets. They comprise the solution found by our resolution algorithm and the optimal solution to the problem.

The sensitivity and the cost are bounded. The lower bound is located at the empty set, which has a sensitivity of 1.0 and a null usability cost. It is equivalent to not using browser fingerprinting at all. On the other end, the set composed of the candidate attributes A is a superset of every attribute set, and provides the lowest sensitivity and the highest usability cost. If A does not satisfy the sensitivity threshold, there is no solution as any other subset has a higher or equal sensitivity.

Greedy Algorithm

We propose the greedy heuristic algorithm presented in Algorithm 1 to find good solutions to the Attribute Selection Problem. It consists into a bottom-up exploration of the lattice by following k paths until reaching the satisfiability frontier. The higher k is, the larger is the explored space, but the higher is the computing time. This algorithm is inspired by the model of the ASP as a lattice of partial Knapsack Problems, and by the Beam Search algorithm [103]. The similarity with the latter lies in the successive expansion of a limited number of nodes, that are chosen according to an order between partial solutions. The order is the efficiency in our case. Our proposed algorithm is part of the Forward Selection algorithms [194], as it iteratively picks attributes according to a criterion, and takes into account those already chosen. However, the proposed algorithm provides the ability to explore several sub-solutions instead of a single one, includes pruning methods that help reduce the computing time, and stops when it reaches the satisfiability frontier instead of when the criterion is not statistically improved.

Algorithm Working Algorithm 1 works by exploring k paths of the lattice. It starts from the empty set and stops when every path reaches the satisfiability frontier. The collection S holds the attribute sets to expand and is initialized to k empty sets. At each stage, the attribute sets to explore are stored in the collection E . They consist of each $S_i \in S$ with one additional attribute. The cost and the sensitivity of each attribute set $C \in E$ is then measured. If C satisfies

Data: The candidate attributes A , the sensitivity threshold α , the number of explored paths k .

Result: The attribute set of the explored paths that satisfies the sensitivity threshold at the lowest cost.

$c_{min}, T, I \leftarrow \text{inf}, \emptyset, \emptyset$

$S \leftarrow$ a collection of k empty sets

if $s(A) > \alpha$ **then**

 | Quit as no solution exists

end

while S is not empty **do**

$E \leftarrow \{C = S_i \cup \{a\} : \forall S_i \in S, \forall a \in A \setminus S_i, \nexists C' \in T \cup I, C' \subset C\}$

$S \leftarrow \emptyset$

for $C \in E$ **do**

if $s(C) \leq \alpha$ **then**

 | $T \leftarrow T \cup \{C\}$

 | $c_{min} \leftarrow c(C)$ if $c(C) < c_{min}$

end

else if $c(C) < c_{min}$ **then**

 | $S \leftarrow S \cup \{C\}$

end

else

 | $I \leftarrow I \cup \{C\}$

end

end

$S \leftarrow$ the k most efficient attribute sets C of S according to $\frac{c(A)-c(C)}{s(C)}$

end

return $\arg \min_{C \in T} c(C)$

Algorithm 1: Greedy algorithm to find good solutions to the Attribute Selection Problem.

the sensitivity threshold α , it is added to the collection T of the attribute sets that reach the satisfiability frontier, otherwise it is added to the collection S of the attribute sets to expand. Finally, the collection S is updated to only hold the k most efficient attribute sets. The efficiency of an attribute set is the ratio between its gain (i.e., the cost reduction compared to the candidate attributes) and its sensitivity. All this process is repeated until S is empty, when all the k paths have reached the satisfiability frontier. The solution is then the attribute set of the lowest cost in T .

Pruning Methods Three properties allow us to reduce the number of attribute sets that are explored. First, we hold the minimum cost c_{min} of the attribute sets T that satisfy the sensitivity. Any explored attribute set that has a cost higher than c_{min} is not added to the collection S of those to explore. Indeed, this attribute set does not provide the lowest cost, nor do its supersets. Then, during the expansion of two attribute sets S_i and S_j of the same size, if S_i satisfies the sensitivity and S_j does not, they can have a common superset S_l . In this case, S_l does not need to be explored as it costs more than S_i . We store S_i in I so that we can check if an attribute set C has a subset in I , in which case we do not explore C . The same holds if S_i costs less than c_{min} and S_j costs more than c_{min} .

Algorithm Complexity Starting from the empty set, we have n supersets composed of one more attribute. From these n supersets, we update S to hold at most k attribute sets. The attribute sets $S_i \in S$ are now composed of a single attribute, and each S_i has $n - 1$ supersets composed of one additional attribute. At any stage, we have at most kn attribute sets to explore. This process is repeated at most n times, as we can add at most n attributes, hence the *computational complexity* of the Algorithm 1 is of $\mathcal{O}(kn^2\omega)$, with ω being the computational complexity of the measures of usability cost and sensitivity of an attribute set. The collection E contains at most kn attribute sets (at most n supersets for each $S_i \in S$). The collections S , T , and I can contain more sets, but are bounded by the number of explored nodes which is kn^2 . The *memory complexity* of the Algorithm 1 is then of $\mathcal{O}(kn^2)$.

Stage	E	T	S
1	$\{\{1\}, \{2\}, \{3\}\}$	$\{\}$	$\{\{1\}, \{3\}\}$
2	$\{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$	$\{\{1, 2\}\}$	$\{\{1, 3\}\}$
3	$\{\}$	$\{\{1, 2\}\}$	$\{\}$

Table 5.2: Example of the execution of Algorithm 1 on the lattice of Figure 5.3, with the sensitivity threshold $\alpha = 0.15$ and the number of explored paths $k = 2$. Stage i is the state at the end of the i -th `while` loop.

Example Table 5.2 displays an example of the execution of Algorithm 1 on the lattice presented in Figure 5.3, with the sensitivity threshold $\alpha = 0.15$ and the number of explored paths $k = 2$. The stage i corresponds to the state at the end of the i -th `while` loop. Initially, the collection S is $\{\emptyset, \emptyset\}$. At stage 1, the two most efficient attribute sets of E are $\{1\}$ and $\{3\}$, which are stored into S . At stage 2, we assume that the attribute set $\{1, 2\}$ is measured first as there is no order among E . In this case, this attribute set is added to the collection T , and the minimum cost is now 20. The attribute set $\{1, 3\}$ is then added to S , but $\{2, 3\}$ is not as it has a higher cost than the minimum cost. At stage 3, the attribute set $\{1, 2, 3\}$ is not added to the collection E as it is a superset of one attribute set of the collection T . The final solution is the less costly attribute set of T , which is $\{1, 2\}$ in this case, and happens to be the optimal solution.

5.2.3 Illustrative Sensitivity and Usability Cost Measures

In this section, we illustrate a sensitivity measure as the proportion of impersonated users given the strongest attacker of our model that knows the fingerprint distribution among the protected users. We also illustrate a usability cost measure according to the fingerprints generated by a fingerprinting probe on a browser population.

Sensitivity Measure

We measure the sensitivity of a given attribute set according to an instantiated attacker and a population of users sharing browser fingerprints. The attacker knows the fingerprint distribution of the protected users, and submits orderly

the most probable fingerprints, until reaching the threshold on the number of submissions. The illustrative *sensitivity measure* evaluates the proportion of users that are impersonated considering the matching function.

From an attribute set C , we retrieve the fingerprint domain F_C such that $F_C = \prod_{a \in C} \text{domain}(a)$, with $\text{domain}(a)$ being the domain of the attribute a and \prod being the Cartesian product. We denote F_A the fingerprints when considering the set A of the candidate attributes. We denote U the set of the users that are protected by the verifier. The set $\mathcal{M} = \{(u, f) : u \in U, f \in F_A\}$ represents the mapping from the users to their fingerprint, so that the user u has the fingerprint f stored.

We denote $\text{project}(f, C)$ the function that projects the fingerprint $f \in F_{C'}$ from the set of attributes C' to the set of attributes C , with the requirement that $C \subseteq C'$. Finally, the function denoted $\text{dictionary}(p, F_C, \beta)$ retrieves the β -most probable fingerprints of F_C given the probability mass function p . We note that it is trivial to retrieve the distribution of the fingerprints composed of any attribute subset $C \subset A$ from the distribution of the fingerprints composed of the candidate attributes A .

We denote $f[a]$ the value of the attribute a for the fingerprint f , and $f[a] \approx^a g[a]$ the matching between the value of the attribute a for the stored fingerprint f and the submitted fingerprint g . It is true only if $f[a]$ matches with $g[a]$, meaning that $g[a]$ is deemed a legitimate evolution of $f[a]$. Finally, we define the set of the matching functions of each candidate attribute as $\Phi = \{\approx^a : a \in A\}$.

We measure the sensitivity as the proportion of impersonated users among a population of protected users, against the attacker that knows the fingerprint distribution among them, using Algorithm 2. The illustrative sensitivity measure is *monotonic* as demonstrated in Appendix G. The *number of submissions* is defined by the verifier according to her rate limiting policy [80] (e.g., blocking the account after three failed attempts). This limit could be set to 1 as a user cannot mistake his browser fingerprint. However, a user can browse from a new or a public browser, and taking preventive action on this sole motive is unreasonable.

Data: The attribute set C , the limit on the number of submissions β , the mapping \mathcal{M} from the users to their browser fingerprint, the probability mass function p , and the set Φ of matching functions.

Result: The proportion of impersonated users.

$R \leftarrow \{\}$

$F_C \leftarrow$ the fingerprint domain when considering C

$V \leftarrow \text{dictionary}(p, F_C, \beta)$

forall $(u, f^*) \in \mathcal{M}$ **do**

$f \leftarrow \text{project}(f^*, C)$
 if $\exists g \in V$ *st.* $\forall a \in C, f[a] \approx^a g[a]$ **then**
 $R \leftarrow R \cup \{u\}$
 end

end

return $\frac{\text{card}(R)}{\text{card}(U)}$

Algorithm 2: Illustrative sensitivity measure.

Usability Cost Measure

There is no off-the-shelf measure of the usability cost of the attributes (e.g., the `UserAgent` HTTP header [186] has no specified size, collection time, nor change frequency). This cost also depends on the fingerprinted population (e.g., mobile browsers generally have fewer plugins than desktop browsers, resulting in smaller values for the list of plugins as described in Section 4.3). As a result, we design an illustrative cost measure that combines three sources of cost (i.e., space, time, and instability), which is computed by the verifier on her fingerprint dataset. The *fingerprint dataset* used to measure the costs is denoted $D = \{(f, b, t) : f \in F_A, b \in B, t \in T\}$, with B being the set of observed browsers.

The *memory cost* is measured as the average fingerprint size. The attribute values are stored and not compressed into a single hash, which is necessary due to their evolution through time. We denote $\text{mem}(C, D)$ the *memory cost* of the attribute set C , and $\text{size}(x)$ the size of the value x . The memory cost is defined as

$$\text{mem}(C, D) = \frac{1}{\text{card}(D)} \sum_{(f,b,t) \in D} \sum_{a \in C} \text{size}(f[a]) \quad (5.1)$$

The *temporal cost* is measured as the average fingerprint collection time, and takes into account the asynchronous collection of some attributes. Although at-

tributes can be collected asynchronously, some require a non-negligible collection time (e.g., the dynamic attributes [153, 179]). We denote $\text{time}(C, D)$ the temporal cost of the attribute set C . Let A_{seq} be the set of the sequential attributes, and A_{async} the set of the asynchronous attributes, so that $C = A_{\text{seq}} \cup A_{\text{async}}$. We denote $\text{ctime}(b, f[a], t)$ the collection time of the attribute a for the fingerprint f collected from the browser b at the moment t . The temporal cost is defined as

$$\text{time}(C, D) = \frac{1}{\text{card}(D)} \sum_{(f,b,t) \in D} \max(\{\text{ctime}(b, f[a], t) : a \in A_{\text{async}}\} \cup \{ \sum_{s \in A_{\text{seq}}} \text{ctime}(b, f[s], t) \}) \quad (5.2)$$

The *instability cost* is measured as the average number of changing attributes between two consecutive observations of the fingerprint of a browser. We denote $\text{ins}(C, D)$ the instability cost of the attribute set C . We denote $\mathcal{C}(\Delta, D)$ the function that provides the pairs of consecutive fingerprints of D that are separated by a time-lapse comprised in the Δ time range, as defined in Section 4.1.3. We denote $\delta(x, y)$ the Kronecker delta being 1 if x equals y and 0 otherwise. We denote $\Psi(D) = \{|t_k - t_r| : ((f_i, b_j, t_k), (f_p, b_q, t_r)) \in D^2\}$ the function that gives the complete time range of the dataset D . The instability cost is defined as

$$\text{ins}(C, D) = \frac{1}{\text{card}(\mathcal{C}(\Psi(D), D))} \sum_{(f,g) \in \mathcal{C}(\Psi(D), D)} \sum_{a \in C} \delta(f[a], g[a]) \quad (5.3)$$

The three dimensions of the cost are weighted by a three-dimensional *weight vector* denoted $\gamma = [\gamma_1, \gamma_2, \gamma_3]$, such that the weights are strictly positive numbers. The verifier tunes these weights according to her needs (e.g., allowing fingerprints to be more unstable, but requiring a shorter collection time). She can do this by defining an equivalence between the three dimensions (e.g., one millisecond of collection time is worth ten kilobytes of size), and setting the weights so that these values amount to the same quantity in the total cost. For a concrete example, we refer to Section 5.3.1.

Finally, we denote $\text{cost}(C, D)$ the cost of the attribute set C given the fingerprint dataset D . The illustrative usability cost measure is *monotonic* as demon-

strated in Appendix G, and is formalized as

$$\text{cost}(C, D) = \gamma \cdot [\text{mem}(C, D), \text{time}(C, D), \text{ins}(C, D)]^\top \quad (5.4)$$

5.3 Experimental Validation

In this section, we describe the experiments that we perform to validate our framework. We begin by describing how the usability cost and the matching function are implemented. Then, we present the results of the attribute selection framework executed with different parameters, and compare them with the results of the common baselines. The experiments were performed on a desktop computer with 32GB of RAM and 32 cores running at 2GHz. We use the datasets that are described in Chapter 3 and analyzed in Chapter 4.

5.3.1 Instantiation of the Experiments

In this section, we present the instantiation of the parameters for the experiments. We first describe how the verifier and the attacker are instantiated by presenting the chosen user population, sensitivity thresholds, and number of submissions. Then, we detail the implementation of the usability cost measure and the matching function between fingerprints, alongside the value of the parameters or weights that they use.

Verifier Instantiation

On the verifier side, we simulate a *user population* by randomly sampling browsers from the first two months of each experiment. The observed fingerprint is considered as the fingerprint stored for the user who owns the browser. We have 70 browsers for the intranet dataset, 81 browsers for the university dataset, and 630 browsers for the enrolled dataset. As for the general audience dataset, we only sample from the first month and limit the sample to 30,000 browsers due to its size. We configure the resolution algorithm to have 1 and 3 *explored paths*

to compare the gain achieved by a larger explored space. We call *ASF-1* and *ASF-3* our attribute selection method with respectively 1 and 3 explored paths. We consider the set of *sensitivity thresholds* $\{0.001, 0.005, 0.015, 0.025\}$. Bonneau et al. [27] defined the resistance against online attacks as a compromise of 1% of accounts after a year when 10 guesses per day are allowed. Hayashi et al [86] estimated that 0.001 is equivalent to a random guess of four-digit. However, to the best of our knowledge, no standard value exists. Hence, we make the choice of these values starting from 0.001 and going to 0.025 to obtain a range from a strict security requirement to one that is less strict. We admit that 0.025 (2.5%) is already high, but it is close to the proportion of users that share the 10 most common passwords in previously leaked datasets [220].

Attacker Instantiation

On the attacker side, an instance is parameterized with the number of fingerprints β that he can submit, and his knowledge over the fingerprint distribution. We consider the strongest attacker of our attack model that knows the fingerprint distribution among the user population.

We consider the set of number of submissions $\{1, 4, 16\}$. To the best of our knowledge, no standard value exists. The choice of 1 is for a strict rate limiting policy that blocks the account on any failure and asks the user to change his password. The choice of 4 is for a policy that would require a CAPTCHA after 3 failed attempts, and would perform the blocking and password change after the fourth failed attempt. Finally, the choice of 16 is for a policy that would let more attempts before performing the blocking and password change. The chosen values are close to the number of submissions allowed into policies enforced in real life [80], and the ones estimated as reasonable values against online guessing attacks [28].

Implementation of the Usability Cost Measure

The implemented *usability cost function* measures the memory in bytes (a character per byte), the time in milliseconds, and the instability as the average number of changing attributes between the consecutive fingerprints. We configure the three-dimensional *weight vector* to the values $\gamma = [1; 10; 10,000]$ to have an equivalence

Dataset	Value	Cost (pts)	Mem. (B)	Time (s)	Inst. (changes)
General Aud.	Candidate	135,171	6,688	9.98	2.86
General Aud.	Max. cost	99,846	1,102	9.98	0.51
General Aud.	Avg. cost	8,794	26	0.87	0.01
General Aud.	Min. cost	1	1	0.00	0.00
Intranet	Candidate	83,078	8,297	1.29	6.18
Intranet	Max. cost	13,388	1,613	1.29	0.43
Intranet	Avg. cost	1,344	33	0.11	0.02
Intranet	Min. cost	5	4	0.00	0.00
University	Candidate	222,878	5,527	1.30	20.44
University	Max. cost	15,157	532	1.30	0.63
University	Avg. cost	1,736	23	0.09	0.09
University	Min. cost	4	4	0.00	0.00
Enrolled	Candidate	86,318	7,790	1.12	6.73
Enrolled	Max. cost	10,062	1,304	0.92	0.35
Enrolled	Avg. cost	1,168	31	0.09	0.03
Enrolled	Min. cost	4	4	0.00	0.00

Table 5.3: The cost of the candidate attributes, together with the maximum, the average, and the minimum cost of a single attribute for each cost dimension.

between 10 kilobytes, 1 second, and 1 changing attribute on average, which are all equal to 10,000 points. Table 5.3 displays the cost of the candidate attributes, together with the minimum, the average, and the maximum cost of a single attribute for each cost dimension and each dataset.

Implementation of the Matching Function

The implemented *matching function* checks that the distance between the attribute values of the submitted fingerprint and the stored fingerprint is below a threshold. Appendix D provides a description of this matching function. More complex matching functions exist (e.g., based on rules and machine learning [214]). They can be integrated to the framework as long as they are monotonic¹⁴.

¹⁴A matching function is monotonic if two fingerprints that match for an attribute set C also match for any subset of C .

Baselines

We compare our method with common attribute selection methods. The entropy-based method [146, 111, 24, 91] consists into picking the attributes of the highest entropy until reaching an arbitrary number of attributes. The method based on the conditional entropy [65] consists into iteratively picking the most entropic attribute according to the attributes that are already chosen, and re-evaluating the conditional entropy of the remaining attributes at each step, until an arbitrary number of attributes is reached. Instead of limiting to a given number of attributes, we pick attributes until the obtained attribute set satisfies the sensitivity threshold. For simplification, we call *entropy* and *conditional entropy* the attribute selection methods that rely on these two metrics.

5.3.2 Attribute Selection Framework Results

In this section, we present the results obtained on the previously presented datasets by processing the Attribute Selection Framework on the instantiated attackers, and compare them with the results of the baselines. The results are obtained for the 12 cases consisting of the Cartesian product between the values of the sensitivity threshold α and those of the number of submissions β . Due to the low distinctiveness, stability, and browser population shown by the university dataset, no case among the tested cases provided solution for this dataset. For this dataset, the attacker that is given a single fingerprint submission manages to impersonate 55 among the 81 browsers, which exceeds the tested sensitivity thresholds. As for the intranet dataset, the attacker having one submission achieves to impersonate 2 browsers among the 70, which also exceeds the tested sensitivity thresholds due to the small browser population. Only the general audience and the enrolled datasets have solutions for the tested parameters. They allow us to simulate two user populations, one of a big size (30,000 users) and one of a smaller size (630 users).

Key Results

The attribute sets found by the Attribute Selection Framework (ASF) generate fingerprints that are 12 to 1,663 times smaller, 9 to 32,330 times faster to collect, and with 4 to 30 times less changing attributes between two observations, compared to the candidate attributes and on average. Compared to the attribute sets found by the baselines, the ones found by the ASF-1 generate fingerprints that are up to 97 times smaller, are collected up to 3,361 times faster, and with up to 7.2 times less changing attributes between two observations, on average. These gains come with a higher computation cost, as the ASF-1 explores more attribute sets by three orders of magnitude compared to the baselines. However, the implemented attribute sets can be updated rarely, and the usability gain is reflected on each authentication performed by each user.

Increasing the number of explored paths by the ASF to three does not significantly change the results. The attribute sets found by the ASF-3 can have a lower usability cost or a higher usability cost due to local optimum (see Section 5.3.2). We show that even when considering all of our candidate attributes, the strongest attacker that is able to submit 4 fingerprints can impersonate 63 users out of the 30,000 users of the general audience dataset, and 8 users out of the 630 users of the enrolled dataset. If this attacker is able to submit 16 fingerprints, this number increases to 152 users for the general audience dataset, and to 22 users for the enrolled dataset.

Results of the Attribute Selection Framework

Figure 5.4 displays the cost of the attribute sets found by the ASF with 1 explored path (ASF-1), the ASF with 3 explored paths (ASF-3), the entropy, and the conditional entropy, for the general audience and enrolled datasets. The costs are in points, so that 10,000 additional points increase the size of fingerprints by 10 kilobytes, their collection time by 1 second, or the number of changing attributes between observations by 1 attribute, on average. There is a solution for 9 out of the 12 cases for the general audience dataset, and the enrolled dataset shows a solution for 5 cases. The cases without a solution are discussed in Section 5.3.2.

For the general audience dataset, half of the attribute sets found using the

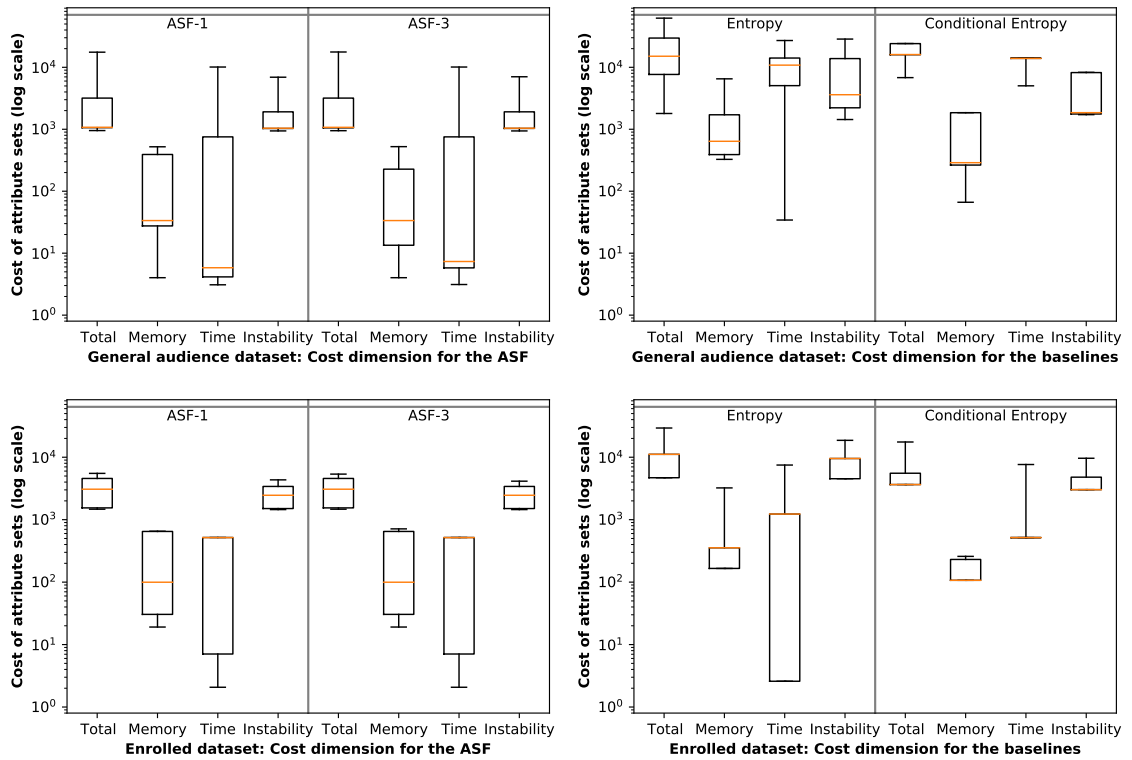


Figure 5.4: Cost of the attribute sets found by the ASF with 1 explored path (ASF-1), the ASF with 3 explored paths (ASF-3), the entropy, and the conditional entropy. The gray horizontal line is the cost when considering all the candidate attributes.

ASF-1 generate fingerprints that, on average, have a size lower than 34 bytes (less than 522 bytes for all sets), are collected in less than 0.59ms (less than 1.01 seconds for all sets), and have less than 0.02 changing attributes between two observations (less than 0.07 attributes for all sets). Compared to the candidate attributes and on average, the generated fingerprints are 12 to 1,663 times smaller, 9 to 32,330 times faster to collect, and with 4 to 30 times less changing attributes between two observations. As for the enrolled dataset, half of the attribute sets found using the ASF-1 generate fingerprints that, on average, have a size lower than 100 bytes (less than 654 bytes for all sets), are collected in less than 52ms (less than 52ms for all sets), and have less than 0.03 changing attributes between two observations (less than 0.05 attributes for all sets). Compared to the candidate attributes and on average, the generated fingerprints are 11 to 408 times smaller, 21 to 5,403 times faster to collect, and with 155 to 463 times less changing attributes between two observations.

The difference in usability cost of the attribute sets found by the ASF-3 and the ASF-1 is negligible. For the general audience dataset, the attribute sets found by the ASF-3 are as less costly as they are more costly than the attribute sets found by the ASF-1. This results in the median additional cost of each dimension being zero. The average resulting fingerprint is from 198 bytes smaller to 30 bytes larger, takes from 3ms less to collect to 0.3ms more, and has from 0.03 less changing attributes to 0.04 more. As for the enrolled dataset, only the case with a number of submissions of 4 and a sensitivity threshold of 0.015 shows difference between the solution found by the ASF-1 and by the ASF-3. The solution found by the ASF-3 is 59 bytes larger, takes 0.02ms less to collect, and has 0.02 more changing attributes, on average. Exploring more paths can counter-intuitively provide a higher usability cost, due to the local optimum problem described in Section 5.3.2. Indeed, when exploring more nodes, the followed paths can diverge as we hold more temporary solutions, which can be local optimum. The computation cost of increasing the number of explored paths is not worth the expected gain in our experimental setup.

Comparison with the Baselines

For the general audience dataset, the ASF-1 finds attribute sets that consume less resources than the baselines in all the 9 cases having a solution. The attribute sets found by the *entropy* consume more resources than the attribute sets found by the ASF-1, with a total cost from 1.8 to 14 times higher. The average generated fingerprint by the attribute sets chosen by the entropy, compared to the attribute sets chosen by the ASF-1, is from 1.6 to 97 times larger, has a collection time that is from 1.5 to 1,872 times higher, and has from 1.5 to 7.2 times more changing attributes between the consecutive fingerprints. The attribute sets found by the *conditional entropy* consume more resources than the attribute sets found by the ASF-1, with a total cost from 1.3 to 15 times higher. The average generated fingerprint by the attribute sets chosen by the conditional entropy, compared to the attribute sets chosen by the ASF-1, is from 1.5 to 16 times larger, has a collection time that is from 1.3 to 3,361 times higher, and has from 1.1 to 4.7 times more changing attributes between the consecutive fingerprints.

As for the enrolled dataset, the ASF-1 finds attribute sets that consume less resources than the baselines in all the 5 cases having a solution. The attribute sets found by the *entropy* consume more resources than the attribute sets found by the ASF-1, with a total cost from 2.4 to 5.3 times higher. The average generated fingerprint by the attribute sets chosen by the entropy, compared to the attribute sets chosen by the ASF-1, is from 0.5 to 8.6 times larger, has a collection time that is from 0.3 to 14.4 times higher, and has from 2.8 to 4.2 times more changing attributes between the consecutive fingerprints. The attribute sets found by the *conditional entropy* consume more resources than the attribute sets found by the ASF-1, with a total cost from 1.1 to 3.1 higher. The average generated fingerprint by the attribute sets chosen by the conditional entropy, compared to the attribute sets chosen by the ASF-1, is from 0.3 to 5.5 times larger, has a collection time that is from 0.9 to 249 times higher, and has from 1.1 to 2.2 times more changing attributes between the consecutive fingerprints. Although the cost of some dimensions can be lower for the solutions found by the baselines, the total cost of these solutions is always higher than for the solutions found by the ASF-1. We emphasize that the verifier can configure the cost of each dimension according to

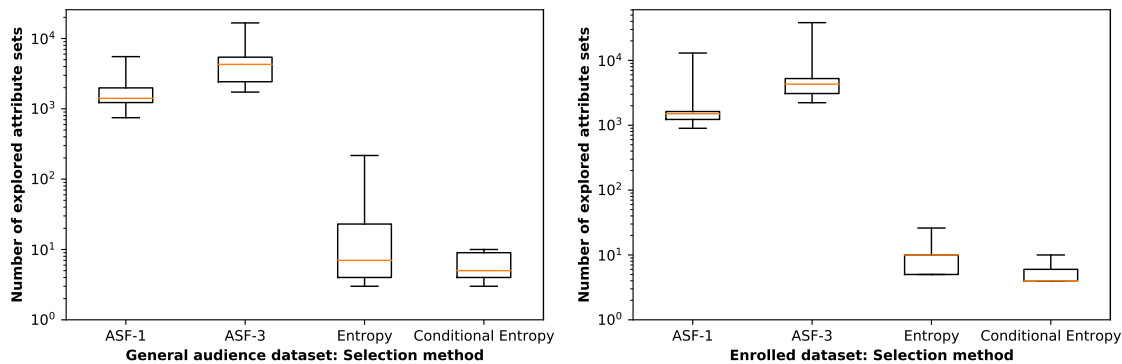


Figure 5.5: Number of explored attribute sets by the attribute selection methods.

her preferences, in order to process a trade-off between the cost dimensions.

Reasons for Sub-optimal Results

The sub-optimal solutions that are found by the Attribute Selection Framework are due to a problem of local optimum. At a given step, the most efficient attribute sets S can have supersets of higher cost than the supersets of another S' . The supersets of S are then explored, whereas the less costly supersets here would have been the supersets of S' .

Computation Cost

The attribute selection framework shows a higher computation cost. Indeed, at each stage of the exploration, the ASF explores up to $n - 1$ attribute sets¹⁵ for each temporary solution, with n being the number of candidate attributes. The ASF-1 explores more attribute sets by three orders of magnitude compared to the baselines. However, this is an upper bound as the baselines require preprocessing. Indeed, the attributes has to be sorted by their entropy or by their conditional entropy.

Figure 5.5 displays the number of attribute sets explored by the attribute selection methods when executed on the general audience and enrolled datasets. The number of explored attribute sets by the ASF-1 goes from 748 to 5,522 for

¹⁵The set of the explored attribute sets can overlap as two temporary solutions can have a common superset.

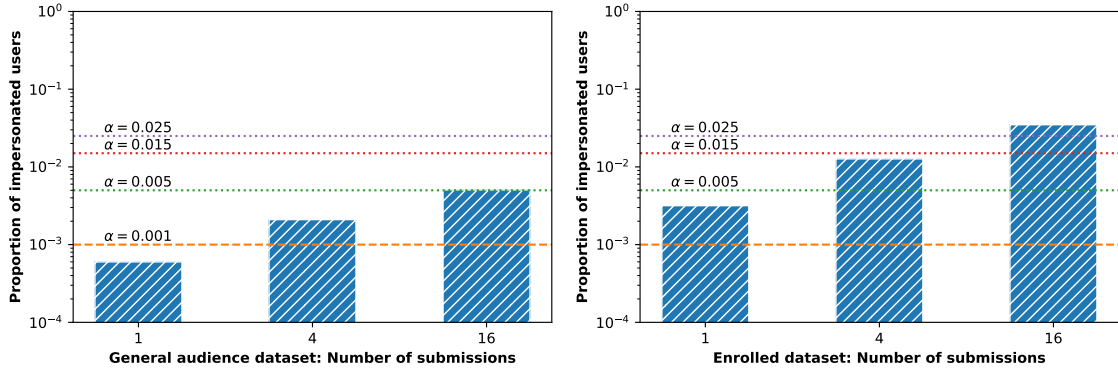


Figure 5.6: The proportion of impersonated users when considering the candidate attributes, and as a function of the number of submissions. The sensitivity thresholds α are displayed.

the general audience dataset, and from 898 to 12,984 for the enrolled dataset. The ASF-3 explores approximately 3 times more attribute sets than the ASF-1: from 1,730 to 16,659 explored attribute sets for the general audience dataset, and from 2,227 to 38,211 explored attribute sets for the enrolled dataset. The number of attribute sets explored by the entropy ranges from 3 to 217 for the general audience dataset, and from 5 to 26 for the enrolled dataset. As for the conditional entropy, it goes from 3 to 10 for the general audience dataset, and from 4 to 10 for the enrolled dataset. However, the conditional entropy method requires to sort the n attributes by their conditional entropy, which requires $\sum_{i=0}^n n - i$ steps. This difference of explored attribute sets between the entropy and the conditional entropy is explained by the latter avoiding selecting correlated attributes.

Lower Bound on the Impersonated Users

The obtained sensitivity against our instantiated attackers ranges from the minimum sensitivity when considering the candidate attributes, as displayed in Figure 5.6, to the maximum sensitivity of 1.0 when considering no attribute at all. All the possible attribute sets have their sensitivity comprised between these two extremum. For the general audience dataset, the instantiated attackers that are allowed 4 submissions are able to impersonate 63 users out of the 30,000 users, which exceeds the sensitivity threshold of 0.001. When allowed 16 submissions, the number of impersonated users increases to 152, which exceeds the sensitivity

threshold of 0.005 that corresponds to 150 users. As for the enrolled dataset, one submission is enough for the instantiated attackers to impersonate 2 users, which exceeds the sensitivity threshold of 0.001 that corresponds to no user at all. When allowed 4 submissions, the number of impersonated users increases to 8, which exceeds the sensitivity threshold of 0.005 that corresponds to 3 users. Finally, the attackers that are allowed 16 submissions achieve to impersonate 22 users, exceeding the sensitivity threshold of 0.025 that corresponds to 15 users. Hence, no solution exists for these attackers, as the number of impersonated users surpasses all the sensitivity thresholds.

Selected Attributes

In this section, we provide the list of the attributes that are selected by the attribute selection framework. Table 5.4 lists the selected attributes for the general audience dataset, and Table 5.5 lists the selected attributes for the enrolled dataset. The nomenclature used is the same as in Appendix I. We have 9 combinations of sensitivity threshold and number of submissions that show a solution for the general audience dataset, and 5 combinations for the enrolled dataset. The attribute selection framework is executed twice with two number of explored paths (1 and 3), hence we respectively have 18 and 10 cases for which it found a solution. Among the selected attributes, six are selected for both the general audience and the enrolled dataset. We discuss below the ten most selected attributes that are selected in at least four cases for one of the dataset. The three most selected attributes for the general audience dataset are also among the most selected for the enrolled dataset. They concern hardware and software components that we expect to not have a strong link together: the browser window size, the number of logical processor cores, the graphics driver, the browser version, a scheme drawn in the browser, the input and output peripherals, and the type of network connection. The attributes that compose the selected attribute sets for each case are less correlated with each other. For the general audience dataset, they have an average normalized conditional entropy of 0.165 compared to 0.067 for the pairs of candidate attributes. As for the enrolled dataset, they show an average normalized conditional entropy of 0.234 compared to 0.100 for the pairs of candidate

Selected Attribute	$k = 1$								$k = 3$					
	$\beta = 1$		$\beta = 4$		$\beta = 16$		$\beta = 1$		$\beta = 4$		$\beta = 16$			
	$\alpha = 0.001$	$\alpha = 0.005$	$\alpha = 0.015$	$\alpha = 0.025$	$\alpha = 0.005$	$\alpha = 0.015$	$\alpha = 0.025$	$\alpha = 0.001$	$\alpha = 0.005$	$\alpha = 0.015$	$\alpha = 0.025$	$\alpha = 0.005$	$\alpha = 0.015$	$\alpha = 0.025$
W.innerHeight	•	•	•	•	•	•	•	•	•	•	•	•	•	•
N.hardwareConcurrency	•	•	•	•	•	•	•	•	•	•	•	•	•	•
WG.UNMASKED_RENDERER_WEBGL	•	•	•	•	•	•	•	•	•	•	•	•	•	•
N.appVersion	•	•	•	•	•	•	•	•	•	•	•	•	•	•
HTML5 canvas inspired by AmIUnique	•	•	•	•	•	•	•	•	•	•	•	•	•	•
N.connection.type	•	•	•	•	•	•	•	•	•	•	•	•	•	•
N.plugins	•	•	•	•	•	•	•	•	•	•	•	•	•	•
[[N, W].doNotTrack, N.msDoNotTrack]	•	•	•	•	•	•	•	•	•	•	•	•	•	•
Accept-Encoding HTTP header	•	•	•	•	•	•	•	•	•	•	•	•	•	•
Height of first bounding box	•	•	•	•	•	•	•	•	•	•	•	•	•	•
Origin of a created div	•	•	•	•	•	•	•	•	•	•	•	•	•	•
W.ontouchstart support	•	•	•	•	•	•	•	•	•	•	•	•	•	•
W.[performance, console].jsHeapSizeLimit	•	•	•	•	•	•	•	•	•	•	•	•	•	•
S.width	•	•	•	•	•	•	•	•	•	•	•	•	•	•
W.openDatabase support	•	•	•	•	•	•	•	•	•	•	•	•	•	•
WM.COMBINED_TEXTURE_IMAGE_UNITS	•	•	•	•	•	•	•	•	•	•	•	•	•	•
N.platform	•	•	•	•	•	•	•	•	•	•	•	•	•	•
HTML5 canvas similar to Morellian	•	•	•	•	•	•	•	•	•	•	•	•	•	•
Accept-Language HTTP header	•	•	•	•	•	•	•	•	•	•	•	•	•	•
WM.CUBE_MAP_TEXTURE_SIZE	•	•	•	•	•	•	•	•	•	•	•	•	•	•
A.sampleRate	•	•	•	•	•	•	•	•	•	•	•	•	•	•

Table 5.4: The attributes selected by the attribute selection framework for each experimentation setup of the general audience dataset. We denote k the number of explored paths, β the number of fingerprint submissions, and α the sensitivity threshold.

Selected Attribute	$k = 1$				$k = 3$			
	$\beta = 1$		$\beta = 4$		$\beta = 1$		$\beta = 4$	
	$\alpha = 0.005$	$\alpha = 0.015$	$\alpha = 0.025$	$\alpha = 0.015$	$\alpha = 0.005$	$\alpha = 0.015$	$\alpha = 0.025$	$\alpha = 0.015$
W.innerHeight	•	•	•	•	•	•	•	•
N.mediaDevices.enumerateDevices()	•		•	•	•		•	•
N.hardwareConcurrency		•	•	•		•	•	•
WG[...].UNMASKED_RENDERER_WEBGL	•			•	•		•	•
Listing of N	•			•	•		•	•
W.Intl.v8BreakIterator().resolvedOptions()		•				•		
WM.VERTEX_UNIFORM_VECTORS	•				•			
N.deviceMemory			•			•		
WM.COMBINED_TEXTURE_IMAGE_UNITS		•				•		
Accept-Language HTTP header						•		•
W.mozInnerScreenX							•	
Any conditional HTTP headers								•
X-Network-Info HTTP header								•
Accept-Encoding HTTP header								•

Table 5.5: The attributes selected by the attribute selection framework for each experimentation setup of the enrolled dataset. We denote k the number of explored paths, β the number of fingerprint submissions, and α the sensitivity threshold.

attributes.

The `innerHeight` property of the `window` JavaScript object provides the height of the visible part of the browser window. It is selected in all the cases of both the general audience and the enrolled dataset. It is the first attribute to be selected during the exploration as it provides the highest efficiency. Indeed, its usability cost is low with, on average and respectively for the general audience and the enrolled dataset, a size of 3.02 bytes and 4.6 bytes, a collection time of 0.14ms and 0.03ms, and a change between 9.38% and 25.96% of the observations. Moreover, it is the fifth most distinctive attribute of the general audience dataset and the most distinctive for the enrolled dataset. For the general audience dataset, it shows an entropy of 8.53 bits and the most common value is only shared by 2.70% of the fingerprints. As for the enrolled dataset, it shows an entropy of 7.87 bits and the most common value is shared by 6.55% of the fingerprints.

The `hardwareConcurrency` property that is collected from the `navigator` JavaScript object provides the number of logical processor cores of the device that runs the browser. This attribute is selected in 11 cases for the general audience dataset and in 6 cases for the enrolled dataset. It shows a high efficiency mostly due to the very low usability cost. Indeed, it shows, on average and respectively for the general audience and the enrolled dataset, a size of 1 byte and 3.99 bytes, a collection time of 0.17ms and 0.19ms, and a change between 0.11% of the observations for the two datasets. However, it shows a lower distinctiveness with an entropy of 1.88 bits for the two datasets. The most common value is shared by 39.64% of the fingerprints for the general audience dataset, and by 46.05% of the fingerprints for the enrolled dataset.

The `UNMASKED_RENDERER_WEBGL` property of an initialized WebGL Context provides a textual description of the graphics driver. This attribute is selected in 8 cases for the general audience dataset, and in 4 cases for the enrolled dataset. It has, on average and respectively for the general audience and the enrolled dataset, a size of 24.51 bytes and 46.43 bytes, a collection time of 0.27ms and 0.21ms, and a change between 0.91% and 10.79% of the observations. Although the most common value is shared by 28.27% of the fingerprints of the general audience dataset, it still provides an entropy of 5.89 bits. As for the enrolled dataset, it has a similar entropy of 5.81 bits and the common value is shared by 12.70% of the fingerprints.

The `appVersion` property of the `navigator` JavaScript object provides the version of the browser. This attribute is selected in 7 cases for the general audience dataset. It shows, on average, a size of 101.76 bytes, a collection time of 0.13ms, and a change between 1.57% of the observations. Although the most common value is shared by 22.61% of the fingerprints, it still provides an entropy of 7.52 bits.

The HTML5 canvas inspired by the AmIUnique study [127] is selected in 7 cases for the general audience dataset¹⁶, mainly due to its high distinctiveness. It has, on average, a size of 63.98 bytes, a collection time of 71.17ms, and a change between 1.36% of the observations. It shows an entropy of 7.76 bits, and the most common value is shared by 7.09% of the fingerprints.

The `connection.type` property of the `navigator` JavaScript object provides the type of the network connection in use by the browser. This attribute is selected in 6 cases for the general audience dataset, mainly due to its low usability cost. It provides, on average, a size of 1.47 bytes, a collection time of 0.21ms, and a change between 0.80% of the observations. It shows a lower distinctiveness compared to the other attributes, with an entropy of 0.61 bits and the most common value being shared by 89.52% of the fingerprints.

The `enumerateDevices` function of the `MediaDevices` API provides the list of the input and output devices (e.g., microphones, headsets). This attribute is selected in 6 cases for the enrolled dataset. It provides, on average, a size of 55.53 bytes, a collection time of 43.81ms, and a change between 17.87% of the observations. It is the 11th most distinctive attribute of the enrolled dataset with an entropy of 5.27 bits and the most common value being shared by 11.19% of the fingerprints.

The list of plugins is selected in 5 cases for the general audience dataset. It provides, on average, a size of 101.71 bytes, a collection time of 2.4ms, and a change between 4.72% of the observations. The most common value is shared by 28.54% of the fingerprints, and this attribute provides an entropy of 3.82 bits.

The `doNotTrack` property can be collected from the `navigator` or the `window` JavaScript object, and was developed to allow the user to express his will to be tracked or not. Since, the working group of the W3C working on this specifica-

¹⁶The HTML5 canvas inspired by the AmIUnique study [127] is not part of the attributes of the enrolled dataset.

tion is closed¹⁷, and the feature begins to be removed from browsers (e.g., it was removed from Safari¹⁸ due to its use for fingerprinting). This attribute is selected in 4 cases for the general audience dataset, mostly due to its low usability cost. It provides, on average, a size of 6.07 bytes, a collection time of 0.16ms, and a change between 0.02% of the observations. The most common value is shared by 42.46% of the fingerprints, and this attribute provides an entropy of 1.87 bits.

The list of the properties of the `navigator` JavaScript object is selected in 4 cases for the enrolled dataset. It provides, on average, a size of 540.97 bytes, a collection time of 0.19ms, and a change between 17.87% of the observations. The most common value is shared by 19.59% of the fingerprints, and this attribute provides an entropy of 4.19 bits.

5.4 Conclusion

In this chapter, we propose FPSelect, a framework for a verifier to tailor his fingerprinting probe by picking the attribute set that limits the sensitivity against an instantiated attacker, and reduces the usability cost.

We formalize the Attribute Selection Problem that the verifier has to solve, show that it is a generalization of the Knapsack Problem, model the potential solutions as a lattice of attribute sets, and propose a greedy exploration algorithm to find a solution. To compare the attribute sets, we propose an illustrative measure of their usability cost that depends on the size, the collection time, and the instability of the generated fingerprints. We also propose an illustrative sensitivity measure that takes as parameter the number of fingerprints that the attacker is able to submit, and considers the worst case of the attacker knowing the distribution of the fingerprints among the protected users. Given these parameters, the sensitivity measure evaluates the proportion of users that are impersonated by the attacker.

We evaluate FPSelect on two real-life browser fingerprint datasets, and compare it with common attribute selection methods that rely on the entropy and the

¹⁷<https://www.w3.org/2011/tracking-protection>

¹⁸https://developer.apple.com/documentation/safari-release-notes/safari-12_1-release-notes

conditional entropy. The attribute sets found by FPSelect generate fingerprints that are 12 to 1,663 times smaller, 9 to 32,330 times faster to collect, and with 4 to 30 times less changing attributes between two observations, compared to the candidate attributes and on average. Compared to the baselines, the attribute sets found by FPSelect generate fingerprints that are up to 97 times smaller, are collected up to 3,361 times faster, and with up to 7.2 times less changing attributes between two observations, on average. These gains come with a higher computation cost, as the ASF-1 explores more attribute sets by three orders of magnitude compared to the baselines. However, the attribute sets used can be updated rarely, and the usability gain is reflected on each authentication performed by each user. Increasing the number of paths explored by the ASF from one to three does not significantly change the results in our experimental settings. The attribute sets found can show a higher usability cost due to local optimum.

Our goal to reduce the usability cost of the attributes is reached using FPSelect. We achieve a reduction of the usability cost of up to three orders of magnitude compared to the baselines, and of up to four orders of magnitude compared to the candidate attributes. Although the computation cost of FPSelect is also three orders of magnitude higher, the reduction of the usability cost is reflected on each authentication performed by each user. Among the hundreds of candidate attributes, less than ten attributes suffice to reduce the sensitivity to a threshold close to the minimum sensitivity obtained using the candidate attributes. The attributes that compose the selected attribute sets are less correlated with each other compared to the pairs of candidate attributes. They concern hardware and software components that are not expected to have a strong link together, like the browser window size and the number of logical processor cores.

Chapter 6

Conclusion

User authentication consists for a verifier to assert that a user is really who he claims to be. When this takes place on the web (i.e., the user authenticates through his web browser on the website of the verifier), we call this process web user authentication or web authentication for short. The authentication of the user has to be done through his web browser. Due to this limitation, new standards are released to extend the functionalities of web browsers. An example is the Web Authentication API [216] which gives browsers the access to external components (e.g., a security token) to process the authentication using asymmetric cryptography. Although the Web Authentication API provides the access to new authentication factors through web browsers, the constraints inherent to these factors are still present. For example, using security tokens in a company requires the technical staff to provide the security tokens to the employees, maintain the tokens (e.g., handling lost tokens), and teach the employees how to use them. Moreover, the employees has to constantly possess their security token. Moreover, if the website is publicly accessible, the number of potential visitors – and conjointly attackers – is large compared to a physical authentication process (e.g., the access to an office by electronic keys). Web authentication makes a great usage of an authentication factor already well-used before the genesis of the web: the password [151]. Indeed, passwords are easy to use (i.e., they simply have to be remembered and typed), to deploy (i.e., the verifier collects them through standard HTML5 forms, stores them as character strings, and compares them identically),

and users are familiar with them. However, passwords are fallible to many attacks: brute force, dictionary [28], credential stuffing [210], or targeted knowledge attacks [218]. Due to the many passwords that users have to remember, they tend to use identical or similar passwords across websites [44, 84]. The leak or the theft of one of their password can then impact the security of all their accounts. To cope with these flaws, multi-factor authentication arises, such that each additional factor provides an additional security barrier. However, they come at the cost of usability or deployability [27]. The users have to remember, possess, or perform additional actions, and the verifier has to deploy, maintain, and teach the users how to use the authentication factor.

In the meantime, a new web tracking technique arose in 2010: browser fingerprinting [56]. It consists into collecting properties from a web browser to build its fingerprint which can be unique. Similarly to the biometric authentication factors that can be used for both identification and authentication (e.g., facial or fingerprint recognition), browser fingerprints can also serve as an authentication factor. Most works on browser fingerprinting for authentication focus on the design of the authentication mechanism [212, 177, 79, 203, 47, 124, 184], and the large-scale empirical studies on browser fingerprints focus on their effectiveness as a web tracking tool [56, 127, 81, 178]. Few is known on the properties of browser fingerprints when used for authentication, like their effectiveness to recognize the fingerprints of a browser and distinguish those of different browsers. Moreover, the properties of browser fingerprints depend on the fingerprinted browser population and on the attributes that compose the fingerprints. The large-scale empirical studies on browser fingerprints (more than 100,000 fingerprints) consider fewer than thirty attributes [56, 127, 214, 81]. This underestimates the distinctiveness of the fingerprints (e.g., [81] reports a rate of 33.6% of unique fingerprints), as it increases the chances for browsers to share the same fingerprint. All this whereas more than a hundred attributes are accessible. The studies that consider more than fifty attributes either work on less than two thousand users [111, 178], or do not analyze the resulting fingerprints at all [64]. In this thesis, we studied the properties of browser fingerprints when used as an authentication factor, considering hundreds of fingerprinting attributes and four browser populations, including a population of nearly two million browsers.

Correlations are expected to occur between the attributes when considering as many as hundreds of them. We can reduce the implemented attributes to a subset to reduce the cost of using them. Indeed, attributes take up to several minutes to collect [152, 155, 157, 191, 193, 179], which is impractical in an online authentication context. However, removing an attribute can result in a loss of distinctiveness. Previous works only consider the well-known attributes [56, 127, 81], remove the attributes of the lowest entropy [214], iteratively pick the attribute of the highest weight (typically the entropy) until a threshold is reached [146, 111, 65, 24, 91, 208], or evaluate every possible attribute set [69]. In this thesis, we proposed the FPSelect framework to select the attributes to use such that the sensitivity against powerful attackers is bounded, and the cost of using them is close to being minimal.

6.1 Contributions

In this thesis, we aimed at exploring the adequacy of browser fingerprinting for web authentication. First, we evaluated browser fingerprints according to properties inspired of biometric authentication factors. Then, we proposed a framework to select the set of attributes to use to keep the sensitivity against attackers low and reduce the cost of using the fingerprints (e.g., the collection time).

6.1.1 Assessing the Adequacy of Browser Fingerprints for Web Authentication

In Chapter 4 we made the link between the digital fingerprints that distinguish Humans and the browser fingerprints that distinguish browsers, to evaluate the latter according to properties inspired by biometric authentication factors. We formalized and assessed these properties on four real-life browser fingerprint datasets. The properties include the distinctiveness of the fingerprints, their stability, their collection time, their size, the loss of effectiveness among browser types, the accuracy of a simple illustrative verification mechanism, and the acceptability of browser fingerprinting for web authentication. The datasets comprise a large-scale fingerprint dataset collected over a period of six months that contains 4,145,408

fingerprints composed of 216 attributes, including 9 instances of dynamic attributes.

The dataset collected from the standardized computers of the University of Rennes 1, named the university dataset, shows different properties compared to the other three datasets. Considering the time-partitioned datasets, more than 81% of the fingerprints are unique on the long-term (11% for the university dataset). Between two observations of the fingerprint of a browser, more than 81% of the attributes are expected to stay identical (68.7% for the university dataset), even when several months elapsed between the observations. The majority (95%) of the fingerprints take at most 10.5 seconds to collect, and all of them require less than 22.3 kilobytes storage space. By processing a simple verification mechanism, we achieved an equal error rate of at most 4.30% (29.42% for the university dataset). Our mobile fingerprints are less distinctive, more stable, lighter, and take more time to collect compared to our desktop fingerprints. We measured the satisfaction of 682 respondents who tested an authentication mechanism that relies on browser fingerprinting using the Net Promoter Score methodology. Among the respondents, 22% are detractors, 38% are passive, and 40% are promoters.

To better comprehend the results on the complete fingerprints, we broke down the analysis to the attribute level. From 63% to 91% of the attributes have at most 100 distinct values (all the attributes of the university dataset have at most 12 values). About the normalized entropy, the majority of the attributes show a normalized entropy comprised between 0.0 and 0.3 excluded, at the exception of the university dataset for which nearly half of the attributes show a null entropy. From 57% to 80% of the attributes are expected to stay identical between 99% of the observations. Few attributes consume high temporal or mnemonic resources. Less than 40 attributes have a median collection time above 5ms and less than 20 attributes weigh more than 100 bytes. Most of the attributes are correlated with another one as they provide less than 1 bit of conditional entropy when the value of another attribute is known. The dynamic attributes are generally more distinctive, less stable, and take more time to collect compared to the fixed attributes.

6.1.2 Attribute Selection according to a Security and Usability Trade-off

In Chapter 5 we proposed the FPSelect framework to help verifiers choose the attributes to include in their fingerprinting probe. The choice is made according to a trade-off between the sensitivity against a modeled attacker and the cost of using the browser fingerprints. We formalized the Attribute Selection Problem that the verifier has to solve, showed that it is a generalization of the Knapsack Problem, modeled the search space as a lattice of attribute sets, and proposed a greedy exploration algorithm to find a solution. We proposed an illustrative measure of the sensitivity against a modeled attacker as the proportion of the protected users that he manages to impersonate. It takes as parameter the number of fingerprints that the attacker is able to submit, and considers the strong attackers that know the fingerprint distribution among the protected users. As for the usability cost, we proposed an illustrative measure that captures the size, the collection time, and the instability of the generated fingerprints.

We evaluated FPSelect on two real-world fingerprint datasets, and compared it with the common attribute selection methods that rely on the entropy and the conditional entropy. In our experimental settings, FPSelect finds attribute sets of a lower usability cost for the same sensitivity level, compared to the baselines and the candidate attributes. Compared to the baselines and on average, the attribute sets found by FPSelect generate fingerprints that are up to 97 times smaller, are collected up to 3,361 times faster, and with up to 7.2 times less changing attributes between two observations. Compared to the candidate attributes and on average, the attribute sets found by FPSelect generate fingerprints that are 12 to 1,663 times smaller, 9 to 32,330 times faster to collect, and with 4 to 30 times less changing attributes between two observations.

We reached our goal to reduce the usability cost of the attributes using FPSelect. We achieved a reduction of the usability cost of up to three orders of magnitude compared to the baselines, and of up to four orders of magnitude compared to the candidate attributes. Although the computation cost of FPSelect is also three orders of magnitude higher, the reduction of the usability cost is reflected on each authentication performed by each user. Among the hundreds of candidate

attributes, less than ten attributes suffice to reduce the sensitivity to a threshold close to the minimum sensitivity obtained using the candidate attributes. We also remarked that the attributes that compose the selected attribute sets are less correlated with each other compared to the pairs of candidate attributes.

6.2 Future Works

6.2.1 Extension of FPSelect to Attackers having Targeted Knowledge

In Chapter 5 we proposed the FPSelect framework to help verifiers select the attribute set that has a low usability cost and keeps the sensitivity against attackers bounded. The sensitivity is measured as the proportion of impersonated users given the attacker that has the knowledge of the fingerprint distribution among the protected users. We plan to extend the FPSelect framework to support the attackers that have targeted knowledge about the web environment of users (e.g., the type of device they use) or the value of some attributes. We assume that the attacker is able to collect the fixed attributes of a victim, but is not able to obtain the dynamic attributes as the set of instructions can be changed on each fingerprinting. This extension can be performed by considering (1) that the attacker knows the fingerprint distribution of a subpopulation (e.g., the mobile browsers, the browsers that share the same value for the fixed attributes), (2) that he can link a victim to a subpopulation (e.g., he knows that a victim uses a mobile browser), and (3) that he submits the β most common fingerprints of the subpopulation to which each victim belongs. Attackers can also try to infer the value of the dynamic attributes: if they manage to obtain pairs of challenge and response, they can try to infer the response to an unseen challenge. To the best of our knowledge, no study addresses the sensitivity against such attackers. Finally, the behavior of FPSelect on other experimental setups (browser populations, attributes, measures, parameters) would be interesting to investigate.

6.2.2 Challenge-Response Mechanisms and Dynamic Attributes

The studies [124, 184] that propose to use dynamic attributes in challenge-response mechanisms only focus on the HTML5 canvas. The value of the dynamic attributes can change following changes of the web environment (e.g., a browser update), resulting in the failure of the response to the challenge. These false matches can be reduced by two ways. First, using several dynamic attributes allow the verifier to rely on several responses instead of a single one, hence increasing the robustness against the changes of dynamic attributes values. For example, if the verifier use ten dynamic attributes and observes a single failed response, he could deem the user legitimate. Retrieving the accessible dynamic attributes and the domain of their instructions is the first step to design a challenge-response mechanism that includes several dynamic attributes. Additionally to the HTML5 canvas, other attributes are dynamic: the WebGL canvas [153], the audio fingerprinting methods [179], and the installed¹ extensions [199, 205] or fonts [65]. Second, the currently known dynamic attributes are mainly media files that are processed by the browser. In our verification mechanisms, we consider these attributes as matching if their value for the two compared fingerprints are identical. More complex matching mechanisms can be designed for these attributes. For example, the response to the canvas challenge can be compared to the awaited response using image comparison algorithms.

6.2.3 Web Environment Components Impacting Attributes

In Chapter 4 we studied the properties of millions of fingerprints composed of hundreds of attributes, and observed that some attributes can be completely inferred when the value of another attribute is known. This can come from their value being tied to the same component of the web browser environment. For example, the `userAgent` and the `appVersion` JavaScript properties depend on the browser version, as well as the `Accept-Language` HTTP header and the textual representation of a specific date depend on the configured language. These

¹The attributes that infer the presence of an extension or a font can be made dynamic by inferring the presence of a different set of items on each fingerprinting.

examples of attributes that depend on the same component of the web browser environment are straightforward and easy to grasp. However, the underlying components that impact other attributes are harder to retrieve (e.g., the evaluation of mathematical functions, the HTML5 canvas [36, 127], the audio fingerprinting methods [179]). To the best of our knowledge, no study identified the underlying web environment components that impact an attribute, nor proposed methods to retrieve these components. The studies that are the closest are the works on cross-browser fingerprinting [53, 25, 38]. They seek to limit the attributes to the ones that help to recognize the underlying device and operating system independently of the browser in use. However, they do not identify which underlying components impact the attributes. Retrieving the underlying components that impact the attributes could help a verifier to link the change of the attributes to the change of the components. For example, retrieving the attributes that evolve when the browser is updated allow the verifier to consider the evolution of these attributes as less suspicious than the evolution of a random set of attributes. This way, the users would undergo the account recovery process less often, as the evolution of the attributes that are impacted by the same component would be recognized.

6.2.4 Matching Function based on Fingerprints Evolution

In Section 4.2.5 we evaluated the accuracy of a simple verification mechanism that checks whether the number of identical attributes between two fingerprints are above a threshold. In Appendix D we compared this verification mechanism with one that authorizes differences between the two attribute values by comparing them using a distance function. We observed that authorizing differences helps both the fingerprints of a browser and these of different browsers to match, which resulted in a lower accuracy compared to the simple verification mechanism. Studying the evolution patterns of fingerprints and the accuracy of a verification mechanism that recognizes these patterns would be interesting. These patterns can comprise the evolution of a single attribute (i.e., how an attribute evolves), the co-evolution of attributes (i.e., how attributes evolve together), and according to the time elapsed between the observations. Indeed, the higher the time elapsed between the observations of the fingerprint of a browser, the more numerous dif-

ferences are expected to occur. An example of evolution patterns for an attribute that would provide the version number is that it either stays identical, gets its value incremented by 1, or incremented by 2 if the browser is updated after a long time. These evolution patterns would be assigned different likelihood according to the elapsed time between the observations. For example, for a short time-span between the observations, the attribute has high chances to stay identical, has fewer chances to get its value increased by 1, and even fewer chances to get its value increased by 2. Moreover, the decrease of the value of this attribute would be considered as highly unlikely. Relying on evolution patterns could also identify the replay of an old fingerprint by an attacker. The replayed fingerprint would be an ancestor of the fingerprint in use by the victim, and the evolution would be unlikely. Fingerprint evolution patterns are double-edged as attackers can use them. A powerful attacker that knows the evolution patterns, and that possesses a dataset of old fingerprints, could make them virtually evolve to obtain up-to-date fingerprints.

6.3 Perspectives on Browser Fingerprinting for Authentication

In Chapter 4, we made the link between biometric authentication factors and browser fingerprinting. We could extend this link by adapting the works on privacy-preserving or revocable biometric authentication [182] to the authentication mechanisms that leverage browser fingerprinting. The browser fingerprinting domain also lack publicly accessible datasets that researchers could work on. This stems from the sensitive and private nature of browser fingerprints. The anonymization of sensitive data to protect the privacy of experimenters and provide enough information for researchers to study is already being studied [206, 130, 138, 131, 54]. Techniques coming from this field of research could be used to anonymize browser fingerprint datasets, like the generation of synthetic datasets [229]. This is challenging due to the several dimensions of browser fingerprints that can be altered or lost in the anonymization: the fingerprint distribution in the browser

population, the correlation between the attributes², together with the evolution and the co-evolution of the attributes through time.

²The correlation between the attributes lead to the privacy paradox [56, 162, 11] which renders a browser more distinctive when trying to fake the value of some attributes. For example, lying about the underlying operating system can result in a unique web environment (e.g., a Safari mobile browser running on Windows 10). Any anonymization technique should care about this effect which can lower the level of privacy (e.g., attributes can be inferred by others) or generate unrealistic data (e.g., a fingerprint of a highly unlikely web environment).

Appendix A

Browser Fingerprinting Attributes

In this appendix, we describe the 216 *source attributes* that are included in our probe, and the 46 *extracted attributes* that are derived from source attributes. We group these attributes into families, and provide references to related studies. Their name is sufficient to retrieve the corresponding browser property, and when needed, we provide a brief description of the method for reproducibility. We focus here on the description of the method, and provide a *complete list* of the attributes and their property in Appendix I.

A.1 JavaScript Properties

Most attributes are *properties* that are accessed through common *JavaScript objects*. The `navigator` object provides information on the browser (e.g., version), its customization (e.g., language), the underlying system (e.g., operating system), and supported functionalities (e.g., list of available codecs). The `screen` object provides information on the screen size, the orientation, the pixel density, and the available space for the web page. The `window` object provides information on the window containing the web page, like its size or the support of storage mechanisms. The `document` object gives access to the web page content, but also to a few properties. Such properties are already included in previous studies [56, 127,

81] or open source projects [67], but are usually limited to less than 20 properties.

A.2 HTTP Headers

We include up to 16 attributes that are collected from *HTTP headers*, most of which are already used in previous studies [56, 127, 81]. Among these 16 attributes, 15 consist of the value of a specific field of the HTTP headers. Among the remaining headers, we ignore a set of headers, and store the name and the value of the others into a dedicated attribute. Table A.1 presents the list of the specific HTTP headers that are stored into a dedicated attribute, and the headers that are ignored, for each experiment.

The base set of the specifically collected HTTP headers are the headers that are commonly present in an HTTP request (e.g., the `User-Agent`). They also include non-standard headers that are typically prefixed by the X character [213]. The base set of the ignored HTTP headers are the headers that we deem indistinct between browsers when considering the same fingerprinting probe (e.g., the `Host` header is about the server). These base sets are used for the general audience, the intranet, and the university experiment. After analyzing the fingerprints of the general audience dataset, we modified the fingerprinting probe to collect and ignore different sets of headers. These new sets of headers are used for the enrolled experiment only, which was the only experiment launched after these modifications of the probe. The modifications include the placement of the collected headers that lack distinctiveness (e.g., `X-ATT-DeviceId`) into the ignored headers. This lack of distinctiveness mainly comes from these headers being almost never encountered. We also noticed that non-standard headers were added by the proxy of the collection server (e.g., `X-Forwarded-Server`), and add them to the ignored headers. Finally, we ignore the headers that concern the navigation of the user (e.g., `Origin`, `Cookie`, or the `Referer` that contains the website that lead to the one that is visited), which we consider as falling outside the recognition of the environment of the browser.

Table A.1: The HTTP headers that are stored into a dedicated attribute, and the headers that are ignored, for each experiment.

	Gen. Aud., Intranet, Univ.	Enrolled
Collected	Accept	Accept
	Accept-Charset	Accept-Charset
	Accept-Encoding	Accept-Encoding
	Accept-Language	Accept-Language
	Cache-Control	Cache-Control
	Connection	User-Agent
	TE	Via
	Upgrade-Insecure-Requests	X-Network-Info
	User-Agent	X-Requested-With
	Via	X-WAP-Profile
	X-ATT-DeviceId	
	X-Network-Info	
	X-Requested-With	
	X-UIDH	
	X-WAP-Profile	
Ignored	charset	charset
	Content-Length	Connection
	Content-Type	Content-Length
	Host	Content-Type
		Cookie
		Host
		Origin
		Referer
		TE
		Upgrade-Insecure-Requests
		X-Artifactory-Override-Base-Url
		X-ATT-DeviceId
		X-Forwarded-For
		X-Forwarded-Host
		X-Forwarded-Port
	X-Forwarded-Proto	
	X-Forwarded-Server	
	X-Real-IP	
	X-UIDH	

A.3 Enumeration or Presence of Browser Components

One attribute family that provides a high diversity is the *browser components* that are installed in the browser. The presence of some components can directly be accessed (e.g., the installed plugins¹), whereas the presence of others have to be inferred (e.g., the installed fonts). The list of components that are given in this section have the components separated by a comma.

A.3.1 List Attributes

Previous studies already identified the *list of plugins* and the *list of fonts* as highly distinctive [56, 127], hence we include these two attributes in our fingerprinting script. We enumerate the list of plugins, and check the size of text boxes [65] to infer the presence of 66 fonts. Additionally, we get the *list of mime types* (i.e., the supported data format), and the *list of the speech synthesis voices*.

A.3.2 Support of Codecs

We infer the *support of video codecs* by creating a `video` element and checking if it can play a given type using the `canPlayType()` function. The *support of audio codecs* is done using the same method, but for an `audio` element. We infer the *support of streaming codecs* by calling the `isTypeSupported()` function of the `window.[WebKit, moz, ms, \emptyset]MediaSource` object, and checking the presence of both the audio and the video codecs. We apply the same method to infer the *support of recording codecs*, but on the `MediaRecorder` object instead.

A.3.3 List of Video Codecs

The 15 *video codecs* for which we infer the presence are the following: `video/mp2t; codecs="avc1.42E01E,mp4a.40.2"`, `video/mp4; codecs="avc1.42c00d"`, `video/mp4; codecs="avc1.4D401E"`, `video/mp4; codecs="mp4v.20.8"`, `video/mp4; codecs="avc1.42E01E"`, `video/mp4; codecs="avc1.42E01E, mp4a.40.2"`, `video/mp4;`

¹At the exception of the Firefox browsers that now only display the Flash plugin.

codecs="hvc1.1.L0.0", video/mp4; codecs="hev1.1.L0.0", video/ogg; codecs="theora", video/ogg; codecs="vorbis", video/webm; codecs="vp8", video/webm; codecs="vp9", application/dash+xml, application/vnd.apple.mpegURL, audio/mpeg-url.

A.3.4 List of Audio Codecs

The 9 *audio codecs* for which we infer the presence are the following: audio/wav; codecs="1", audio/mpeg, audio/mp4; codecs="mp4a.40.2", audio/mp4; codecs="ac-3", audio/mp4; codecs="ec-3", audio/ogg; codecs="vorbis", audio/ogg; codecs="opus", audio/webm; codecs="vorbis", audio/webm; codecs="opus".

A.3.5 List of Fonts

The 66 *fonts* for which we infer the presence are the following: Andale Mono; AppleGothic; Arial; Arial Black; Arial Hebrew; Arial MT; Arial Narrow; Arial Rounded MT Bold; Arial Unicode MS; Bitstream Vera Sans Mono; Book Antiqua; Bookman Old Style; Calibri; Cambria; Cambria Math; Century; Century Gothic; Century Schoolbook; Comic Sans; Comic Sans MS; Consolas; Courier; Courier New; Garamond; Geneva; Georgia; Helvetica; Helvetica Neue; Impact; Lucida Bright; Lucida Calligraphy; Lucida Console; Lucida Fax; LUCIDA GRANDE; Lucida Handwriting; Lucida Sans; Lucida Sans Typewriter; Lucida Sans Unicode; Microsoft Sans Serif; Monaco; Monotype Corsiva; MS Gothic; MS Outlook; MS PGothic; MS Reference Sans Serif; MS Sans Serif; MS Serif; MYRIAD; MYRIAD PRO; Palatino; Palatino Linotype; Segoe Print; Segoe Script; Segoe UI; Segoe UI Light; Segoe UI Semibold; Segoe UI Symbol; Tahoma; Times; Times New Roman; Times New Roman PS; Trebuchet MS; Verdana; Wingdings; Wingdings 2; Wingdings 3.

A.4 Extension Detection

The list of the installed *browser extensions* cannot be directly accessed, but their presence can be inferred by the changes brought to the web page by extensions [205],

Table A.2: Extensions detected by the changes they bring to the page content.

Extension	Page content change
Privowny	W.privownyAddedListener[EXT] is supported
UBlock	D.head has <code>display: none !important;</code> and <code>:root</code> as style
Pinterest	D.body.data-pinterest-extension-installed is supported
Grammarly	D.body.data-gr-c-s-loaded is supported
Adguard	W.AG_onLoad is supported
Evernote	Element with <code>style-1-cropbar-clipper</code> as id exists
TOTL	W.ytCinema is supported
IE Tab	W.ietab.getVersion() is supported

or by the presence of web accessible resources [199, 106]. We check the changes that are brought to the web page by the 8 extensions that are listed in Table A.2, and the availability of the web accessible resources of the 8 extensions that are listed in Table A.3.

A.4.1 Detection of Ad Blockers

We also infer the presence of an *advertisement blocker* by creating an invisible dummy advertisement, and by checking if it is removed or not. The dummy advertisement is a created division which has the id property set to "ad_ads_pub_track", the class set to "ads .html?ad= /?view=ad text-ad textAd text_ad text_ads text-ads", and the style set to "width: 1px !important; height: 1px !important; position: absolute !important; left: -1000px !important; top: -1000px !important;".

A.5 Size and Color of Web Page Elements

A.5.1 Bounding Boxes

The attributes related to the *bounding boxes* concern a `div` element to which we append a `span` element. The `div` element has his `style` property set to the values displayed in Table A.3. The `span` element contains a specifically crafted text that is provided below. The size of the bounding boxes (i.e., the width and the

Table A.3: Extensions detected by the availability of their web accessible resource. C stands for chrome, and R stands for resource.

Extension	Web accessible resource
Firebug	C://firebug/skin/firebugBig.png
YahooToolbar	R://635abd67-4fe9-1b23-4f01-e679fa7484c1/icon.png
EasyScreenshot	C://easyscreenshot/skin/icon16.png
Ghostery	R://firefox-at-ghostery-dot-com/data/ images/ghosty-16px.png
Kaspersky	R://urla-at-kaspersky-dot-com/data/icon-16.png
VideoDownloadHelper	R://b9db16a4-6edc-47ec-a1f4-b86292ed211d/data/ images/icon-18.png
GTranslate	R://aff87fa2-a58e-4edd-b852-0a20203c1e17/icon.png
Privowny	C://privowny/content/icons/ privowny_extension_logo.png

height of the rectangles of the `div` and the `span` elements) are then collected using the `getClientRects` function. The text of the `span` element is composed of Unicode characters (e.g., emojis, letters of a non-latin alphabet) that starts with `\u` and a string. This text is set to "`\ua9c0 \u2603 \u20B9 \u2604 \u269b \u2624 \u23B7 \u262c \u2651 \u269d \u0601 \u0603 \u060e \u06dd \ud8-3c \udfe1 mmmmmmmmmmlil \u102a`".

A.5.2 Width and Position of a Created `div` Element

The attribute named *width and position* of a created `div` is the properties of `width` and `transform-origin` of a newly created `div` element, obtained by calling the `getComputedStyle` function. This created `div` element is afterward used to infer the color of layout components, as described below.

A.5.3 Colors of Layout Components

The attribute *colors of layout components* is obtained by applying the color of several layout components (e.g., the scroll bar) to the created `div` element, and by getting the color back from the property `W.getComputedStyle(new_div).color`. Each of the tested component gets its color extracted this way, and aggregated

Table A.4: Properties of the `div` element measured for the attributes related to the bounding boxes.

Property	Value
<code>position</code>	<code>absolute</code>
<code>left</code>	<code>-9999px</code>
<code>textAlign</code>	<code>center</code>
<code>objectFit</code>	<code>scale-down</code>
<code>font</code>	<code>68px / 83px Helvetica, Arial, Sans-serif</code>
<code>zoom</code>	<code>66%</code>
<code>MozTransform</code>	<code>scale(0.66)</code>
<code>visibility</code>	<code>hidden</code>

in this attribute. The color of each element is afterward extracted as a single attribute. They are displayed at the end of the Table [I.1](#).

A.6 WebGL Properties

Our script collects several properties from the WebGL API. To obtain them, we create a `canvas` element and get its WebGL context by calling `getContext()` using any of the following parameters: `webgl`, `webgl2`, `moz-webgl`, `experimental-webgl`, or `experimental-webgl2`.

The property `MAX_TEXTURE_MAX_ANISOTROPY_EXT` is obtained from one of the properties `[WEBKIT_EXT_, MOZ_EXT_, EXT_]texture_filter_anisotropic`. To get the attributes prefixed by `UNMASKED`, we first get an identifier named `id` from the unmasked property of the `getExtension('WEBGL_debug_renderer_info')` object, and then get the actual value by calling `getParameter(id)`. Finally, to get the `COMPRESSED_TEXTURE_FORMATS` property, we have to load the `[WEBKIT_]WEBGL_compressed_texture_s3tc` extension first.

A.7 WebRTC Fingerprinting

We include a WebRTC fingerprinting method similar to the method proposed by Takasu et al. [\[207\]](#). The method consists into getting information about the Session



Figure A.1: HTML5 canvas inspired by the AmIUnique [127] study in PNG format.



Figure A.2: HTML5 canvas similar to the Morellian [124] study in PNG format.

Description Protocol of a generated WebRTC connection. Due to the variability of this information, we create two different connections and hold only the values that are identical between these two. As this method leaks internal IP addresses, we hash them directly on the client.

A.8 HTML5 Canvases

We dedicate the Section 4.3.7 to a focus on dynamic attributes, and provide here examples of the two HTML5 canvases that are inspired by previous studies. Our script includes a canvas inspired by the AmIUnique study [127] in both PNG and JPEG formats (Figure A.1 displays an example of the PNG version), and an enhanced version of it similar to the Morellian study [124] in PNG format (Figure A.2 displays an example).

A.9 Audio Fingerprinting

We dedicate the Section 4.3.7 to a focus on dynamic attributes, and provide here the concrete implementation and the scheme of the Audio Nodes network used in the audio fingerprinting methods. The audio fingerprinting methods are inspired by the methods described by Englehardt et al. [57], and are designed to form complex networks of `AudioNode` objects to have more chances to induce fingerprintable behaviors.

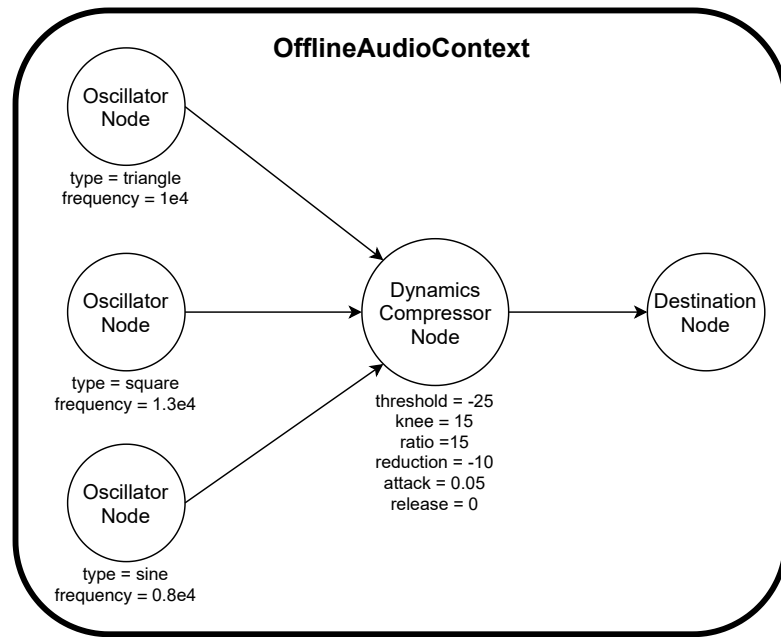


Figure A.3: Architecture of the network of `AudioNode` objects for the simple audio fingerprinting method.

A.9.1 Simple Process

The *simple process* consists of three `OscillatorNode` objects that generate a periodic wave, connected to a single `DynamicsCompressorNode`, and finishing to a `AudioDestinationNode`. The architecture of the network of `AudioNode` objects for the simple process is depicted in Figure A.3, together with the parameters set for each node. The `OscillatorNode` objects are started one after the other, and overlap at some time. The sequence of events is the following: (1) the triangle oscillator node is started at $t = 0$ seconds, (2) the square oscillator is started at $t = 0.10$ seconds, (3) the triangle oscillator is stopped $t = 0.20$ seconds, together with the start of the sine oscillator node, (4) the square oscillator is stopped at $t = 0.25$ seconds. When the rendering of the audio context is done, the `complete` event is triggered, and gives access to a `renderedBuffer` that contains the audio data encoded as a buffer of 32 bits floating-point numbers. The *audio fp simple* (AFS) attribute is an integer computed as the sum of the values of the `renderedBuffer`, that are first casted to absolute integers.

A.9.2 Advanced Process

The *advanced process* consists of four `OscillatorNode` objects, two `BiquadFilterNode` objects, two `PannerNode` objects, one `DynamicsCompressorNode` object, one `AnalyserNode` object, and one `AudioDestinationNode` object. The architecture of the networks of `AudioNode` objects for the simple process is depicted in Figure A.4, together with the parameters set for each node. The `OscillatorNode` objects are started one after the other, and overlap at some time. The sequence of events is the following: (1) the triangle oscillator node and the sine oscillator node with a frequency of 280 are started at $t = 0$ seconds, (2) the square oscillator is started at $t = 0.05$ seconds, (3) the triangle oscillator is stopped $t = 0.10$ seconds, (4) the sine oscillator with a frequency of 170 is stopped at $t = 0.15$ seconds, (5) the square oscillator is stopped at $t = 0.20$ seconds. When the rendering of the audio context is done, the `complete` event is triggered, and gives access to a `renderedBuffer` that contains the audio data encoded as a buffer of 32 bits floating-point numbers. The *audio fp advanced* (AFA) attribute is an integer computed as the sum of the values of the `renderedBuffer`, that are first casted to absolute integers. The *audio fp advanced frequency data* (AFA-FD) attribute is the sum of the frequency data obtained through the `getFloatFrequencyData` function of the `Analyser Node`.

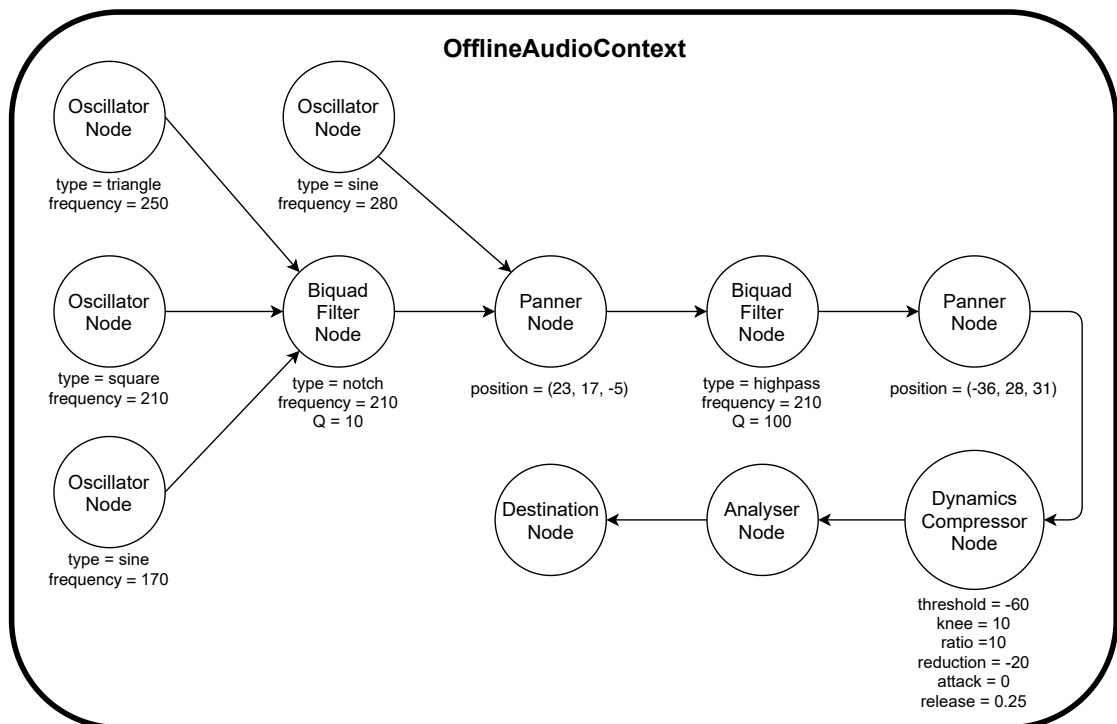


Figure A.4: Architecture of the network of `AudioNode` objects for the simple audio fingerprinting method.

Appendix B

Updates of the Fingerprinting Probe

As web technologies evolve, new attributes become accessible (e.g., these brought by new functionalities) and others become deprecated (e.g., the Battery Status API removed from Firefox [21]). The fingerprinting probe was updated through time to include new accessible attributes, and to remove the attributes that threw a lot of errors, that lack distinctiveness, or that are highly correlated to others. In this appendix, we focus on the JavaScript attributes, and refer the reader to Appendix A for the changes applied to the collected HTTP headers. As the fingerprinting probe was updated following the analysis of the general audience dataset, the provided numbers are related to it (e.g., the number of browsers).

B.1 For the Intranet Experiment

The fingerprinting probe was modified between the general audience and the intranet experiment to exclude 6 attributes, include 8 new attributes, and fix an instability factor in our custom canvas. The base fingerprinting probe is the probe used during the general audience experiment, that embark 216 attributes divided into 200 JavaScript properties and 16 attributes collected from the HTTP headers.

B.1.1 Attributes Removed

We remove the WebRTC fingerprinting method [207, 66] due to the many errors that were encountered. When collecting this attribute, 41.90% of the browsers did not support one of the functions used, 38.30% of them had a variable used that was undefined, and 10.44% of them threw an exception. We also remove the three audio fingerprinting attributes [179] due to errors. When collecting these attributes, 27.95% of the browsers did not support one of the functions used, and more than 14.16% of them had a variable used that was undefined. These errors can be explained by the high proportion of Internet Explorer browsers into the general audience dataset, and these two APIs being unsupported by these browsers at the time of the experiment. Indeed, Internet Explorer never supported the WebAudio API¹, nor the WebRTC API [37], and the Microsoft browser that was designed to replace Internet Explorer, namely Microsoft Edge, only started to fully support the WebRTC API on January 15, 2020. Although this explains the high amount of errors, these attributes still need to be reworked as they are among the less stable and the most time-consuming attributes. Indeed, the WebRTC fingerprinting method has a sameness rate of 0.765 for an average collection time of 1.56 seconds, and the audio fingerprinting methods have a sameness rate lower than 0.958 for an average collection time higher than 2.38 seconds.

B.1.2 Attributes Added

We add `baseLatency`² to the properties collected from the initialized `AudioContext`. It provides the latency between the reception of audio data by the `AudioDestinationNode` and its playing by the audio system of the device, in seconds. We add `deviceMemory`³ to the properties collected from the `navigator` object. It provides the approximation of the amount of ram of a device in GiB, and is limited to a set values that were reduced for privacy reasons⁴. We add the collection of the media input and output devices (e.g., microphone, camera, speakers) through the

¹<https://caniuse.com/#feat=audio-api>

²<https://developer.mozilla.org/en-US/docs/Web/API/AudioContext/baseLatency>

³<https://developer.mozilla.org/en-US/docs/Web/API/Navigator/deviceMemory>

⁴<https://github.com/w3c/device-memory/commit/6317a7379a43830460178744188dbd1451816ccd>

`enumerateDevices`⁵ function of the `navigator.mediaDevices` object. We add the collection of the `FaceDetector` and the `BarcodeDetector` properties from the `window` object. This allows us to detect the support of these interfaces that belong to the Shape Detection API⁶, which provide shape detection functionalities (e.g., barcode, face). Finally, we add `downlinkMax`, `effectiveType`, and `rtt` to the properties extracted from the `navigator.connection` variable. These attributes concern the network connection currently in use.

B.1.3 Instability Fix of our Custom HTML5 canvas

During the intranet experiment, we encountered a high instability in our custom canvas, notably on Safari browsers. We identified the instruction that generated this instability. It is the call to the `transform`⁷ instruction that downscales and tilts the whole canvas image. As a result, we remove this instruction from the instructions that generate our custom canvas, which are not anymore downscaled nor tilted.

B.2 For the Enrolled Experiment

The fingerprinting probe was modified between the launches of the intranet and of the enrolled experiment to exclude 2 attributes, include 6 new attributes, and to fix an instability of the HTML5 canvases generated on Safari browsers.

B.2.1 Attributes Removed

As the enrolled experiment consisted into experimenters testing an authentication platform that relies on passwords and browser fingerprinting, we sought to reduce the collection time of the fingerprints. Some time-consuming and bogus attributes were already removed for the intranet experiment. Moreover, we have several canvases whereas they typically concern the same browser components (e.g., the

⁵<https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/enumerateDevices>

⁶<https://wicg.github.io/shape-detection-api>

⁷<https://html.spec.whatwg.org/multipage/canvas.html#dom-context-2d-transform>

graphical card, the canvas API) and are correlated. As a result, we only held our custom HTML5 canvas and removed the two canvases inspired by previous studies, namely AmIUnique [127] and Morellian [124]. This choice was motivated by our custom canvas being the most distinctive among the three remaining HTML5 canvases, and which knowledge allows to the most efficiently infer the value of the others. For numbers about these two aspects, we refer the reader to Section 4.3.7.

B.2.2 Attributes Added

We add an attribute that infers the keyboard layout by getting the actual character on the keyboard in use given what is at this place on the common US keyboard. This is done by getting the `KeyboardLayoutMap`⁸ from the `navigator.keyboard` object, and retrieving the actual keys that are at the position of the letters `qwerty` on a US keyboard. Although this method is not supported by the majority of the browsers, it is supported by Chrome⁹ which is the most common browser both worldwide [50] and in France [49]. We also add `onLine`¹⁰ to the properties collected from the `navigator` object, which informs whether the browser is connected to a network or not. Finally, we add four attributes related to the state of the battery of the device. They are collected from a `BatteryManager` interface¹¹ that is obtained from the `navigator.getBattery()` promise. They are the properties `charging`, `chargingTime`, `dischargingTime`, and `level`.

B.2.3 Instability Fix of our Custom HTML5 canvas

During the enrolled experiment, we encountered another instability of our custom canvas, notably on Safari browsers. We identified the instruction that generated this instability. It is the call to the `shadowBlur` and to the `shadowColor` instructions¹² that generates the red background gradient. As a result, we remove this instruction from the instructions that generate our custom canvas, which do not anymore have this background gradient.

⁸<https://developer.mozilla.org/en-US/docs/Web/API/KeyboardLayoutMap>

⁹<https://caniuse.com/#search=getLayoutMap>

¹⁰<https://html.spec.whatwg.org/#navigator.online>

¹¹<https://w3c.github.io/battery>

¹²<https://html.spec.whatwg.org/multipage/canvas.html#shadows>

B.3 For the University Experiment

Before launching the university experiment, we sought to reduce the amount of collected attributes to respect data minimization. To do so without losing distinctiveness, we remove 22 attributes that can be completely inferred by another one. Examples are the `numberOfInputs` and the `numberOfOutputs` of an initialized `AudioAnalyser` that always share the same value, and the `product` property of the `navigator` object that is completely inferred when knowing the value of the `vendor` property.

Appendix C

Browser Classification Keywords

In this appendix, we provide the keywords used to infer the browser family, the operating system, and whether the browser is a robot (i.e., whether it is an automated tool and not a genuine visitor). We match these keywords on the `userAgent` JavaScript property, that is first set to lower case. We manually compiled the keywords, by searching for meaningful keywords inside the collected `userAgent` values.

C.1 Robot Keywords

We check that the `userAgent` does not contain the keywords, nor is set to the exact values, that are listed in Table C.1.

C.2 Device Type

To infer the device type of a browser, we match keywords sequentially with the `userAgent` of the browser. The set of keywords can overlap between two device types (e.g., the `userAgent` of touchpad browsers often contain keywords of mobile browsers, like `mobile` for example). Due to this overlapping problem, we verify that the `userAgent` of the browser contains the keyword of the device type, and does not contain the keyword of some other device types. Table C.2 lists the keywords that we leverage to infer each device type.

Table C.1: The keywords and the exact `userAgent` values that we consider as indicating a robot browser. The long values are cut at a blank space and displayed with indentations.

Blacklisted keyword	Blacklisted value
googlebot	mozilla/4.0 (compatible; msie 7.0; windows nt 6.1;
evaliant	trident/7.0; slcc2; .net clr 2.0.50727;
bot.html	.net clr 3.5.30729; .net clr 3.0.30729;
voilabot	media center pc 6.0; .net4.0c; .net4.0e)
google web preview	mozilla/5.0 (x11; linux x86_64) applewebkit/537.36
spider	(khtml, like gecko) chrome/52.0.2743.116
bingpreview	safari/537.36
	mozilla/5.0 (windows nt 6.3; rv:36.0) gecko/20100101
	firefox/36.0
	mozilla/5.0 (macintosh; intel mac os x 10.10; rv:38.0)
	gecko/20100101 firefox/38.0

The *mobile devices* are smartphones, and do not include touchpads. We check that their `userAgent` contain a `mobile` keyword, and no `touchpad` nor `miscellaneous` keywords. To infer that a device is a *touchpad*, we check that their `userAgent` contain a `touchpad` keyword, and no `miscellaneous` keywords. The *miscellaneous devices* are game consoles and smart TVs. We simply check that their `userAgent` contain a `miscellaneous` keyword. Finally, to infer that a device is a *desktop* computer, we check that their `userAgent` does not contain any of the `mobile`, `touchpad`, or `miscellaneous` keywords. In the table, we omit a `miscellaneous` keyword due to its size, which is "`opera/9.80 (linux i686; u; fr) presto/2.10.287 version/12.00 ; sc/ihd92 stb`".

C.3 Browser and Operating System Families

Table C.3 lists the keywords that we leverage to infer the family of a browser, and Table C.4 lists the keywords that we leverage to infer the operating system family of a browser. As a keyword can be in the `userAgent` of two different families, we check the keywords sequentially in the order presented in the tables, and classify a device in the first family having a keyword that matches.

Table C.2: The keywords that we consider as indicating each device type.

Mobile	Touchpad	Miscellaneous
phone	ipad	wii
mobile	tablet	playstation
android	terra pad	smart-tv
iphone	tab	smarttv
blackberry		googletv
wpdesktop		opera tv
		appletv
		nintendo
		xbox

Table C.3: The keywords that we consider as indicating each browser family.

Browser Family	Keywords
Firefox	Firefox
Edge	Edge
Internet Explorer	MSIE, Trident/7.0
Samsung Browser	SamsungBrowser
Chrome	Chrome
Safari	Safari

Table C.4: The keywords that we consider as indicating each operating system family.

OS Family	Keywords
Windows Phone	Windows Phone
Android	Android
iOS	iPad, iPhone (but not Mac OS X)
Mac OS X	Mac OS X (but not iPad, nor iPhone)
Windows 10	Windows NT 10.0
Windows 8.1	Windows NT 6.3
Windows 7	Windows NT 6.1
Ubuntu	Ubuntu
Other Windows	Windows NT, Windows 7, Windows 98, Windows 95, Windows CE, Windows Mobile
Other Linux	Linux, CrOS, NetBSD, FreeBSD, OpenBSD, Fedora, Mint

Appendix D

Advanced Verification Mechanism

Section 4.1.4 describes a simple verification mechanism that checks that the number of identical attributes between the presented and the stored fingerprint is below a threshold. In this appendix, we present the results obtained using an *advanced verification mechanism* that incorporates matching functions that authorize limited changes between the attribute values of the presented and of the stored fingerprint. The methodology to obtain the datasets is the same as the one described in Section 4.2.5. The same-browser comparisons are the same for the two verification mechanisms, but the different-browsers comparisons are re-sampled randomly.

Figure D.1 displays the distribution of the matching attributes between the same-browser comparisons and the different-browsers comparisons for the four datasets. Figure D.2 displays the false match rate (FMR), which is the proportion of the same-browser comparisons that are classified as different-browsers comparisons, and the false non-match rate (FNMR), which is the inverse. The displayed results are the average for each number of matching attributes among the month samples.

D.1 Attributes Matching

The *advanced verification mechanism* leverages matching functions for the comparison of the attributes. It counts the attributes that match between the two compared fingerprints, given the matching functions, and considers the evolution

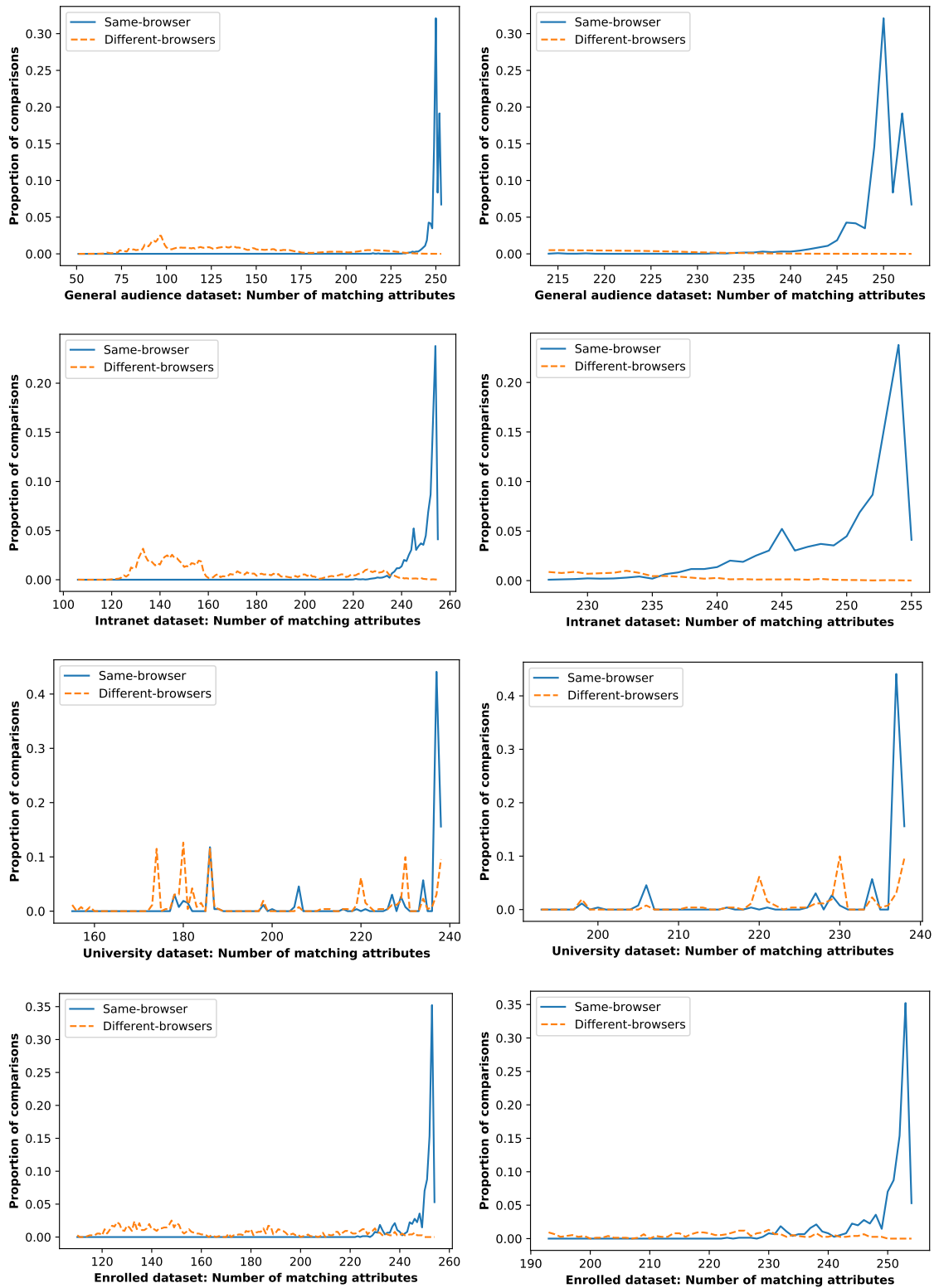


Figure D.1: The number of matching attributes between the same-browser comparisons and the different-browsers comparisons for the four datasets. The figure on the left starts from the lowest observed value, and the figure on the right starts from the value for which 0.5% of the same-browser comparisons are below (20% for the university dataset).

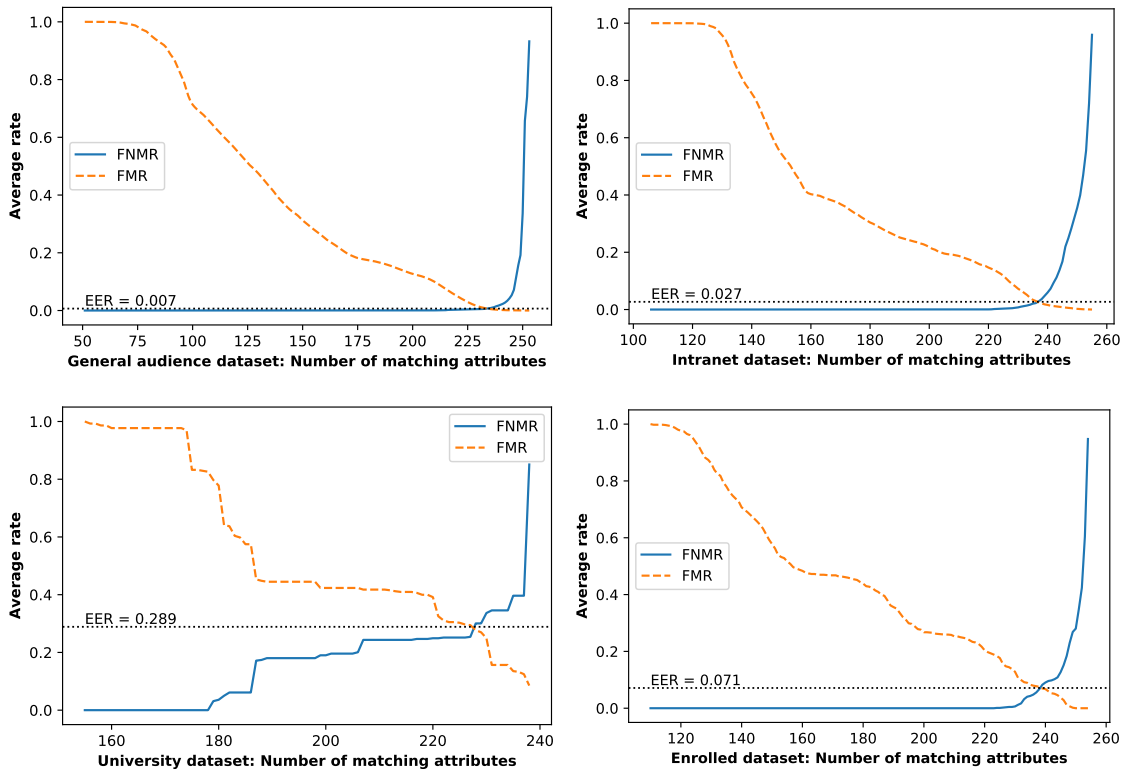


Figure D.2: False match rate (FMR) and false non-match rate (FNMR) given the required number of matching attributes, averaged among the month samples for the four datasets.

legitimate if this number is above a threshold. More formally, we seek to compare the stored fingerprint f to the presented fingerprint g . To do so, we compare the values $f[a]$ and $g[a]$ of the attribute a for the fingerprints f and g , using the matching function \approx^a . The *matching function* \approx^a of the attribute a verifies that the distance between $f[a]$ and $g[a]$ is below a threshold θ^a . Finally, we deem that g is a legitimate evolution of f , if the total number of matching attributes between f and g is above a threshold Θ .

Similarly to previous studies [56, 105, 214], we consider a distance measure that depends on the type of the attribute. The minimum edit distance [102] is used for the textual attributes, the Jaccard distance [225] is used for the set attributes, the absolute difference is used for the numerical attributes, and the reverse of the Kronecker delta (i.e., $1 - \delta(x, y)$) is used for the categorical attributes. The *distance thresholds* of each attribute is obtained by training a Support Vector Machines [87] model on the two classes of each month sample, and extract the threshold from the resulting hyperplane. At the exception of the dynamic attributes that are required to be identical (i.e., the distance threshold is null), as they would contribute to a challenge-response mechanism [124, 184].

D.2 Comparison to the Simple Verification Mechanism

The matching functions of the advanced verification mechanism leads to more matching attributes than identical attributes between two fingerprints, on average and for our four datasets. However, the increase is more significant for the different-browsers comparisons than the same-browser comparisons. The different-browsers comparisons have an average number of matching attributes that is between 105.44% and 107.75% higher than the average number of identical attributes. The same-browser comparisons show a lower increase, which is comprised between 100.27% and 102.24%. As both sets of comparisons have a higher number of matching attributes, the separation of the two sets is not facilitated by the matching functions. Compared to the simple verification mechanism, the advanced mechanism shows an equal error rate that is increased by 0.05 points for the general

audience dataset, by 0.51 points for the intranet dataset, and by 2.83 points for the enrolled dataset. Only the university dataset shows an equal error rate that is decreased by 0.55 points when considering the advanced mechanism. This can be explained by the improvement in the recognition of the fingerprints coming from the same browser that is brought by the matching functions. Indeed, the fingerprints of the university dataset have a lower stability compared to the other datasets.

D.3 Accuracy for the General Audience Dataset

Each of the two sets of comparisons for the general audience dataset contains 3,467,289 comparisons over the six months of the experiment. Table D.1 compares the results of the simple verification mechanism that leverages the identical attributes, and the advanced verification mechanism that leverages the matching attributes, for the general audience dataset. Considering the matching functions only increases the average number of matching attributes for the same-browser comparisons by 0.81 attributes, whereas the increase is greater for the different-browsers comparisons at 7.18 attributes. The equal error rate is slightly higher for the advanced verification mechanism than for the simple one, respectively at 0.66% against 0.61%.

D.4 Accuracy for the Intranet Dataset

The set of the same-browser comparisons for the intranet dataset is composed of 9,667 comparisons, and the set of the different-browsers comparisons contains 9,264 comparisons. Table D.2 compares the results of the simple verification mechanism that leverages the identical attributes, and the advanced verification mechanism that leverages the matching attributes, for the intranet dataset. Considering the matching functions only increases the average number of matching attributes for the same-browser comparisons by 0.67 attributes, whereas the increase is greater for the different-browsers comparisons at 12 attributes. The equal error rate is slightly higher for the advanced verification mechanism than for the

Table D.1: Comparison between the simple verification mechanism that uses identical attributes and the advanced verification mechanism that uses matching attributes for the general audience dataset.

Result	Simple	Advanced
Number of attributes	253	253
Same-browser: identical/matching attributes	[72; 252]	[81; 253]
Same-browser: avg. identical/matching attributes	248.64	249.45
Same-browser: standard deviation	3.91	3.69
Different-browsers: identical/matching attributes	[34; 253]	[51; 253]
Different-browsers: avg. identical/matching attributes	127.41	134.59
Different-browsers: standard deviation	44.06	43.25
Equal error rate	0.61%	0.66%
Threshold of identical/matching attributes	232	234

simple one, respectively at 2.70% against 2.19%.

D.5 Accuracy for the University Dataset

The set of the same-browser comparisons for the university dataset is composed of 263 comparisons, and the set of the different-browsers comparisons contains 261 comparisons. Table D.3 compares the results of the simple verification mechanism that leverages the identical attributes, and the advanced verification mechanism that leverages the matching attributes, for the enrolled dataset. Considering the matching functions only increases the average number of matching attributes for the same-browser comparisons by 4.90 attributes, whereas the increase is greater for the different-browsers comparisons at 13.14 attributes. The equal error rate is higher for the advanced verification mechanism than for the simple one, respectively at 28.87% against 29.42%.

Table D.2: Comparison between the simple verification mechanism that uses identical attributes and the advanced verification mechanism that uses matching attributes for the intranet dataset.

Result	Simple	Advanced
Number of attributes	255	255
Same-browser: identical/matching attributes	[110; 254]	[166; 255]
Same-browser: avg. identical/matching attributes	248.87	249.54
Same-browser: standard deviation	6.23	5.62
Different-browsers: identical/matching attributes	[46; 254]	[106; 255]
Different-browsers: avg. identical/matching attributes	154.87	166.88
Different-browsers: standard deviation	37.35	34.73
Equal error rate	2.19%	2.70%
Threshold of identical/matching attributes	233	237

Table D.3: Comparison between the simple verification mechanism that uses identical attributes and the advanced verification mechanism that uses matching attributes for the university dataset.

Result	Simple	Advanced
Number of attributes	238	238
Same-browser: identical/matching attributes	[157; 237]	[178; 238]
Same-browser: avg. identical/matching attributes	218.46	223.36
Same-browser: standard deviation	28.63	21.55
Different-browsers: identical/matching attributes	[139; 238]	[155; 238]
Different-browsers: avg. identical/matching attributes	188.05	201.19
Different-browsers: standard deviation	32.43	25.67
Equal error rate	29.42%	28.87%
Threshold of identical/matching attributes	217	228

Table D.4: Comparison between the simple verification mechanism that uses identical attributes and the advanced verification mechanism that uses matching attributes for the enrolled dataset.

Result	Simple	Advanced
Number of attributes	254	254
Same-browser: identical/matching attributes	[214; 253]	[223; 254]
Same-browser: avg. identical/matching attributes	248.18	249.14
Same-browser: standard deviation	6.67	6.14
Different-browsers: identical/matching attributes	[71; 254]	[110; 249]
Different-browsers: avg. identical/matching attributes	160.23	168.94
Different-browsers: standard deviation	38.47	38.52
Equal error rate	4.30%	7.13%
Threshold of identical/matching attributes	232	238

D.6 Accuracy for the Enrolled Dataset

Each of the two sets of comparisons for the enrolled dataset contains 755 comparisons over the first six months of the experiment¹. Table D.4 compares the results of the simple verification mechanism that leverages the identical attributes, and the advanced verification mechanism that leverages the matching attributes, for the enrolled dataset. Considering the matching functions only increases the average number of matching attributes for the same-browser comparisons by 0.96 attributes, whereas the increase is greater for the different-browsers comparisons at 8.71 attributes. The equal error rate is higher for the advanced verification mechanism than for the simple one, respectively at 7.13% against 4.30%.

¹The enrolled experiment was processed over seven months, and the samples have a size of two months. Due to this sampling, the seventh month is not considered.

Appendix E

Acceptability Survey

You can find below the questions of the survey published by *Le Lab* on its experimentation platform, and the brief description of the authentication mechanism that integrates browser fingerprints as an authentication factor.

E.1 Questions and Possible Answers

As the users of the experimentation platform are French people, the questions were asked in French, but we provide here an English translation. The questions marked with an asterisk (*) are open-ended questions for which the participant was provided a text field for her to write her answer. Questions 1 to 6 concern general information about the participant. Questions 7 to 11 concern the use of and the opinion of the participant about the current authentication methods. Questions 12 to 14 concern the opinion of the participant about the description of the authentication mechanism to test that leverages browser fingerprints. Questions 15 to 16 concern the opinion of the participant about the actual test of the authentication mechanism.

1. Genre
 - Male
 - Female
2. Age

- Less than 25 years old
- 25-34 years old
- 35-44 years old
- 45-54 years old
- 55 years old and more

3. Household composition

- 1 person
- 2 persons
- 3 persons
- 4 persons
- more than 5 persons

4. Socio-professional categories

- Higher managerial and professional occupations
- Employees
- Retirees
- Intermediate professions
- Craftsmen, tradesmen, and entrepreneurs
- Other categories

5. When it comes to new technologies, do you consider yourself more of ...?

- An expert
- A confirmed user
- A normal user

6. Which of the following equipment do you personally use?

- Smartphone

- Computer
 - Touchpad
 - Home phone
 - Mobile phone (other than smartphones)
7. Which authentication method(s) do you know? And which one(s) have you already used?
- Login/password
 - Fingerprint
 - SMS code
 - Facial recognition
 - Mobile Connect
 - NFC contactless card
 - Security key (like Yubikey)
 - Iris recognition
 - Other(s)
8. Overall, are you satisfied with the authentication methods currently available to you for accessing online services or applications?
- A score on a scale from 0 to 10, with 0 being the greatest dissatisfaction and 10 the greatest satisfaction
9. For what reason(s) are you not completely satisfied?*
10. Have you ever experienced difficulties with an authentication solution?*
11. What do you do to memorize your password(s)?
- I use a password manager software (e.g., Keepass, 1password, etc.)
 - I rely on my web browser to store my passwords
 - I always use the same password to remember it easily

- I save my passwords in a file or a notepad
 - I do nothing in particular
 - Other(s)
12. Overall, how interested are you in this authentication solution¹?
- Not at all interested
 - Somewhat interested
 - Pretty interested
 - Very interested
13. What do you like about this authentication solution? What do you dislike about this authentication solution?*
14. From that description, would you say that this authentication solution is...?
- Easy to use
 - Enhances security
 - Trustworthy
 - Innovative
 - Nothing to report
15. Did you register your fingerprint? Overall, are you satisfied with the experience you were offered?
- A score on a scale from 0 to 10, with 0 being the greatest dissatisfaction and 10 the greatest satisfaction
16. For what reason(s)?*

¹The participant was provided the description of the authentication method that relies on browser fingerprints, which is provided above.

E.2 Description of the Test Authentication Mechanism

The brief description of the test authentication mechanism is presented to the participants before they answer to Question 12. As the participants are French, the description is in French, and you can find an English translation below. The authentication mechanism has two authentication modules related to fingerprinting : one that authenticates through browser fingerprints via a browser, and one that authenticates through device fingerprints via an Android application. In this thesis, we focus on browser fingerprinting, hence the device fingerprinting part is omitted.

Solution [name of the solution] improves the security of your identity on websites by supplementing your login/password with an authentication of your equipment.

Compatible with most Android browsers and mobile applications, it requires no installation of third-party software, nor any particular actions from you.

Its operation relies on browser fingerprinting technology. Based on technical characteristics that we collect from your browser or your mobile phone (model, screen size, configured language...), we create an almost unique digital fingerprint of your equipment. To identify your equipment, we compare the digital fingerprint collected with those associated with the equipments that you have previously registered.

Appendix F

Test Authentication Mechanism

In this appendix, we describe the working of the test authentication mechanism that leverages browser fingerprinting as an authentication factor. The mechanism was integrated into a web platform, which allowed users to create an account and test the mechanism. As the web platform was developed as a proof of concept, the users connected could solely access their account information, and no real web application was accessible. We integrate a challenge-response mechanism by the HTML5 canvas which instructions vary on each fingerprinting. We communicated about the web platform on social media and to colleagues. Moreover, the respondents of the acceptability survey sent by *Le Lab* on the enrolled experiment were invited to test the web platform. The authentication mechanism was implemented in the Keycloak¹ open source identity and access management framework.

F.1 Enrollment

The enrollment is the initial step during which the user creates her account on the web platform. During this step, the user provides her email address, a password, and her browser fingerprint is collected. As we integrate a challenge-response mechanism, the instructions used to collect the HTML5 canvas are randomized. During this step, we store both the instructions and the hash of the generated HTML5 canvas. An email that contains a unique link is sent to her email address

¹<https://www.keycloak.org>

to verify her email address.

F.2 Authentication

The authentication is the main functionality for the users to test. The user accesses the login page of the web platform, identifies with her email address, and first authenticates using her password. If her password matches, her browser fingerprint is collected and verified. Through the life of the web platform, we tested several verification mechanisms that include a mechanism similar to the simple verification mechanism described in Section 4.2.5, and verification mechanisms that leverage machine learning. During this step, we collect two HTML5 canvases, one to authenticate the user, and one to use for the next authentication if the user is authenticated. The instructions used to generate the first canvas are retrieved from the previous fingerprinting, and the generated canvas is matched against the previously stored canvas. If both the canvas and the other attributes match, the second canvas and the instructions used to generate it replace the previous information. These will be used to generate and verify the canvas on the next authentication. After that, the user is deemed legitimate and is given access to the account.

F.3 Account Recovery and Multi-browser Support

F.3.1 Password Recovery

The user is limited to a number of attempts on a time range when authenticating using her password. When this limit is reached, the access to the account is silently blocked for several minutes. During this period, the user is only displayed the message saying that the credentials are incorrect. Moreover, whenever the password submitted is wrong, the user is warned by an email which indicates the time and the IP address used for the attempt. She is also asked to contact an administrator of the platform if she is not the one who made the attempt. If the

user has forgotten her password, a button allows her to provide her email address to recover it. An email that contains a one-time password is sent to this email address. After the user has entered this one-time password, she is given the ability to enter a new password for her account.

F.3.2 Browser Fingerprint Recovery and Multi-browser

After the user has successfully authenticated by her password, she reaches the step during which her browser fingerprint is verified. During this step, it happens that her browser is unrecognized, due to updates to the web environment of the browser, or due to the use of an unknown browser. In this case, the user can access the fingerprint update step by verifying her email address similarly to the password recovery process. After the user has verified her email address, the fingerprint of her current browser is collected, and she is given two choice. She can either use this fingerprint as the update of a registered browser, or she can specify that her current browser is a new one. In the first case, the fingerprint of the registered browser is then updated to the collected fingerprint, and in the second case a new browser is registered with the collected fingerprint. In any case, the HTML5 canvas is generated using a new set of instructions, which are stored together with the newly generated canvas. The user is given the ability to register up to five browsers, and to name each browser to recognize them when she updates the fingerprint of one of them.

Appendix G

Demonstrations of Measures

Monotony

In this appendix, we provide the demonstration of the monotony of the illustrative sensitivity and usability cost measures.

G.1 Monotony of the Illustrative Sensitivity

Theorem 2. *Considering the same limit on the number of submissions β , the mapping from users to their browser fingerprint \mathcal{M} , the probability mass function p , and the set of matching functions Φ . For any couple of attribute sets C_k and C_l so that $C_k \subset C_l$, we have $s(C_k) \geq s(C_l)$ when measuring the sensitivity using Algorithm 2.*

Proof Sketch. We focus on a fingerprint $f \in F_{C_l}$ when considering the attribute set C_l , and the dictionary V used to attack f . We project f to the attribute set C_k to obtain the fingerprint $g \in F_{C_k}$. This fingerprint g can be attacked using the dictionary W composed of the fingerprints of V projected to C_k . As the matching function works in an attribute basis, if a fingerprint $h \in V$ matches with f , we have $f[a] \approx^a h[a] : \forall a \in C_l$ that is true. As C_k is a subset of C_l , we also have $f[a] \approx^a h[a] : \forall a \in C_k$ that is true. By applying this projection to each fingerprint and the associated dictionary of attack fingerprints, the fingerprints spoofed when considering C_l are also spoofed when considering C_k . The sensitivity

when considering C_k is therefore at least equal to that when considering C_l .

When projecting the fingerprints of the attack dictionaries to the attribute set C_k , some of them can come to the same fingerprint. In which case, more fingerprints can be added to these dictionaries until reaching the submission limit β . This can lead to more spoofed fingerprints and impersonated users. The sensitivity when considering C_k can be higher than when considering C_l .

For any attribute sets C_k and C_l so that $C_k \subset C_l$, we then have $s(C_k) \geq s(C_l)$ when measuring the sensitivity using Algorithm 2. \square

G.2 Monotony of the Illustrative Usability Cost

Theorem 3. *For a given fingerprint dataset D , and attribute sets C_k and C_l so that $C_k \subset C_l$, we have $\text{cost}(C_k, D) < \text{cost}(C_l, D)$.*

Proof. We consider C_i and C_j two attribute sets, so that they differ by the attribute a_d such that $C_j = C_i \cup \{a_d\}$. We consider the fingerprint dataset D from which the measures are obtained.

The memory cost of C_j is strictly greater than the memory cost of C_i . As C_j and C_i differ by the attribute a_d , we have

$$\text{mem}(C_j, D) = \text{mem}(C_i, D) + \frac{\sum_{(f,b,t) \in D} \text{size}(f[a_d])}{\text{card}(D)} \quad (\text{G.1})$$

The size of the attribute a_d is strictly positive, hence we have the inequality $\text{mem}(C_j, D) > \text{mem}(C_i, D)$.

The temporal cost of C_j is greater than or equal to the temporal cost of C_i , as adding an attribute cannot reduce the collection time. The attribute a_d can be sequential or asynchronous, and can take identical or more time than the current longest attribute of C_i . Below, we explore all these cases. (1) If a_d is asynchronous, we have the following cases: (a) a_d takes less time than the longest asynchronous attribute a_l , then the collection time is either that of a_l or the total of the sequential attributes, so a_d does not influence the collection time and $\text{time}(C_j, D) = \text{time}(C_i, D)$, (b) a_d takes more time than the longest asynchronous attribute, but less or equal to the total of the sequential attributes, then the

maximum is the total of the sequential attributes and $\text{time}(C_j, D) = \text{time}(C_i, D)$, (c) a_d takes more time than both the longest asynchronous attribute and the total of the sequential attributes, then the collection time is that of a_d , and $\text{time}(C_j, D) > \text{time}(C_i, D)$. (2) If a_d is sequential, we have the following cases: (a) a_d increases the total collection time of the sequential attributes, but the total stays below that of the longest asynchronous attribute, then we have the equality $\text{time}(C_j, D) = \text{time}(C_i, D)$, (b) a_d increases the total collection time of the sequential attributes, which is then higher than that of the longest asynchronous attribute, then $\text{time}(C_j, D) > \text{time}(C_i, D)$ ¹. These are all the possible cases, hence we have $\text{time}(C_j, D) \geq \text{time}(C_i, D)$.

The instability cost of C_j is greater than or equal to that of C_i , as either a_d is completely stable and $\text{ins}(C_j, D) = \text{ins}(C_i, D)$, otherwise a_d is unstable and $\text{ins}(C_j, D) > \text{ins}(C_i, D)$. We then have $\text{ins}(C_j, D) \geq \text{ins}(C_i, D)$.

As the cost weight vector γ is composed of strictly positive real numbers, the cost of C_j is therefore strictly higher than the cost of C_i due to the memory cost. Recursively, it holds for any C_k and C_l so that $C_k \subset C_l$, hence the cost is monotonic. For any fingerprint dataset D , and attribute sets C_k and C_l so that $C_k \subset C_l$, we have $\text{cost}(C_k, D) < \text{cost}(C_l, D)$. \square

¹Either the total collection time of the sequential attributes of C_i does not exceed that of its longest asynchronous attribute, and adding a_d results in the total collection time of the sequential attributes surpassing that of the longest asynchronous attribute. In this case, $\text{time}(C_j, D) > \text{time}(C_i, D)$. Either the total collection time of the sequential attributes of C_i exceeds that of its longest asynchronous attribute. As a_d increases the total collection time of the sequential attributes, we have $\text{time}(C_j, D) > \text{time}(C_i, D)$.

Appendix H

Explanations to the Experimenters

In this appendix, we provide the messages that we displayed to the users to explain our experiments.

H.1 Intranet Experiment

Figure H.1 displays the banner asking for the consent of the user before collecting her browser fingerprint. As the content of the banner is in French, here is its translation:

By clicking on "Accept", you authorize the [confidential] team to collect your browser fingerprint. For more information, visit our article on [the intranet website].

The translation of the article of the intranet website that explains the intranet experiment is shown below:

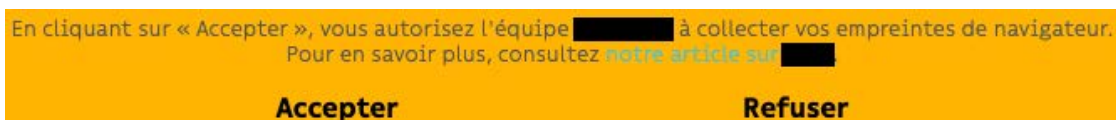


Figure H.1: Consent banner of the intranet experiment

Help us improve our browser fingerprinting algorithms

Browser fingerprinting is a new transparent authentication method based on the hardware and software features of your workstation. All this information is collected from your web browser and allows us to create a virtually unique fingerprint.

Although our previous work has shown a high degree of uniqueness of fingerprints, we are confronted with the evolution of these fingerprints over time. This is why we have worked on the implementation of machine learning algorithms capable of anticipating this evolution. In order to validate their correct operation, we are setting up a campaign to collect fingerprints from the [intranet website] homepage.

How does it work?

You will see a banner on the [intranet website] home page. By clicking on the "accept" link, you authorize us to collect the characteristics of your workstation (OS, screen size, list of fonts, list of plug-ins, ...). Once you give your consent, a script on the [intranet website] homepage will transparently collect the characteristics necessary for the creation of the fingerprint. We will also place a cookie on your computer in order to follow the evolution of your fingerprint. Of course, we do not collect your IDs, IP addresses, or browsing history.

And if I do not agree

It would be a shame not to help us, but we respect your choice. In this case, please click on the "decline" link. If you change your mind, simply delete the cookie from your browser. The next time you log on to the [intranet website] home page, you will see the collection request banner again.

I want to know more

If you are curious, feel free to contact me [email address of the project manager].

H.2 University Experiment

Below we display a translation of the web page shown to the users of the computers of the university when asking their consent to participate in the experiment. At the bottom of this message, the question *I authorize the collection of my browser fingerprint* is displayed to the user with a two radio buttons showing the two options: *yes* or *no*. After the user has made a choice, she needs to validate it by clicking on a button displaying the message: *Validate my choice*.

Help us improve our browser fingerprint authentication mechanisms

The Institute of Research and Technology b<>com in association with IRISA and the University of Rennes 1 is developing new user authentication methods based on browser fingerprinting technology.

What is browser fingerprinting?

The browser fingerprinting is a technique that consists in collecting a set of technical characteristics about a browser and aggregating them into a fingerprint.

Why we collect your fingerprints

As part of our research, we wish to study the characteristics of browser fingerprints, including their diversity and evolution over time.

What we collect (or not)

For our experiment, we collect the following personal data:

- The date and time of collection.
- The identifier of your browser.
- Browser features, including:
 - The User-Agent HTTP header contains information about the browser and operating system used and their version.
 - The screen resolution and the window size.

- The time zone and the language configured in the browser.
- An image (canvas) generated by the browser that contains variations depending on various software and hardware components (the operating system, the web browser, the graphical card and its driver, and the browser settings, among others).

Storage conditions

All the data we collect is kept confidential on the local IRT b<>com servers. Data related to the experiment are kept for 3 years from the start of the experiment, and are used only for research purposes by the IRT b<>com. For legal reasons, we are obliged to keep the technical connection data for a period of 12 months after registration.

Your Rights on the Data

You have the right to access, rectify and delete your data. To do so, please contact us by email at the following address [email address of the project].

What if I do not want to participate in the experiment?

It would be a shame not to help us, but we respect your choice. We do not collect your browser fingerprint without your explicit consent. If you change your mind, just change your choice in the extension management menu.

To know more about it

For more information, you can visit this web page. You can also contact us by email at the following address [email address of the project].

Consent and duration of the experiment

By clicking on the "Accept" button, you authorize the IRT b<>com to collect your browser fingerprint on a regular basis for the duration of the experiment, i.e., from 06/11/2019 to 10/01/2020. Your consent is the legal basis for our collection of your personal data within the framework of the experimentation, you can withdraw it at any time through the extension management menu of your browser.

H.3 Enrolled Experiment

The enrolled experiment took place on a public website¹. Below, we provide a translation of the explanation of the experiment that the experimenters can access on an informational page² and a translation of the terms and conditions of use³. The terms and conditions of use was redacted by the legal department of the Institute of Research and Technology b<>com in collaboration with the research team. Additionally to this informational page, the users were informed of the collection of their personal data during both the enrollment and the authentication process. Their rights regarding their personal data were also recalled during the enrollment.

H.3.1 Informational page

About the experiment

Who are we?

We are a research team from the Trust & Security laboratory of the Institute of Research and Technology b<>com, located in Rennes. With the help of Orange, INRIA and IMT Atlantique, we are working on new authentication methods to connect to Internet services.

What is the purpose of our experiment?

We seek to strengthen password security by adding a supplementary authentication element, without introducing any constraint or manipulation for users.

After an initial laboratory study phase, we want to test in real conditions the operation of two new technologies based on the fingerprinting of browsers and mobile terminals.

¹<https://demo.barbacan.irt-b-com.org>

²<https://demo.barbacan.irt-b-com.org/about.html>

³<https://demo.barbacan.irt-b-com.org/cgu.html>

We wish to verify both their proper functioning over time, but also their ease of use in regular use.

How can you help us?

By registering on our experimental authentication platform, you will be able to test our two authentication methods from your computer or Android mobile.

In order to verify that these authentication methods remain effective over time, we would like you to log in to our platform on a regular basis.

At any time, you will be able to share your comments with us through our forum.

How long does the experiment last?

The experiment takes place from October 1, 2019 to February 26, 2020. At the end of the experiment, the accounts created for the experiment will be deleted.

Where can I find the terms and conditions of use?

To this address: <https://demo.barbacan.irt-b-com.org/cgu.html>.

Where to find the mobile application?

The mobile application will soon be available on the Google Play store.

What is browser fingerprinting?

Browser fingerprinting is a technique that collects technical information about a web browser to create a digital fingerprint.

Although this technique is already used for marketing purposes, our work concerns its use for authentication.

What is device fingerprinting?

Device fingerprinting is an adaptation of browser fingerprinting to the world of mobile devices. It is presented as a library integrated into a mobile application.

The technique consists in collecting data directly from the operating system of the mobile device. This data is used to create an almost unique digital fingerprint.

How can you contact us?

You can leave us a message on the forum or send us an email to terminal.authentication(at)b-com.com.

About b<>com

As a technology provider for companies that seek to accelerate their competitiveness through digital, b<>com serves the cultural, creative, digital infrastructure, defense, industry 4.0 and health industries.

Its laboratories mix talents from multiple disciplines and cultures in the fields of artificial intelligence, immersive video and audio, content protection, 5G networks, the Internet of Things and cognitive technologies...

Coming from the industrial and academic worlds, its researchers and engineers work on its campus of Rennes and its sites in Paris, Brest and Lannion.

Thanks to its advanced engineering team and its own scientific means, b<>com offers its customers ingredients and solutions that make the difference.

H.3.2 Terms and conditions of use

General conditions of use of the application

PREAMBLE

The FONDATION B-COM develops new user authentication solutions based on features of computer communication devices such as computers or smartphones.

To this end, it has developed authentication technologies based on:

- Either on the browser used by the user when he consults the Internet with

his device;

- or on the characteristics specific to each device.

As the FONDATION B-COM seeks to experiment these solutions on a real scale, it is offering a group of experimenters from the general public to contribute to this test phase free of charge.

The general conditions governing this test phase are as follows.

DEFINITIONS

In this document, words and expressions that begin with a capital letter, whether used in the singular or plural, masculine or feminine, have the meanings set out below:

- Test Application means the Terminal Authentication application dedicated to the experimentation phase and through which the User accesses his personal interface.
- Browser Fingerprinting refers to the algorithmic solution developed by FONDATION B-COM which consists in collecting and analyzing a set of characteristic data related to the Browser used by the User during his Internet consultations (browser, operating system, web page resolution, list of plug-ins, etc), in order to authenticate the latter.
- GTU means the present General Terms of Use.
- Device Fingerprinting refers to the algorithmic solution developed by FONDATION B-COM which consists in collecting and analyzing the characteristic data of the User's Equipment (screen size, installed software, operating system, etc.) to authenticate the latter.
- Equipment refers to the User's electronic communication equipment such as Smartphones, tablets, computers or similar devices that enable the User to consult the Internet with a Browser or to use a mobile application.

- b<>com designates the Fondation de Coopération Scientifique b<>com (FONDATION B-COM) whose headquarters location is ZAC DES CHAMPS BLANCS, 1219, AVENUE CHAMPS BLANCS, 35510 CESSON-SÉVIGNÉ (FR), organizer of the tests and responsible for processing the personal data collected during these tests.
- Browser refers to any software that allows, among other things, to display websites on a screen, to carry out searches via search engines and to download files in any format, such as Microsoft Edge, Firefox, Google Chrome, Opera, Safari, etc.
- Site means the website/web portal (<https://demo.barbacan.irt-b-com.org>) or the Test Application that the User will be invited to use to test the Solutions.
- Solutions refers to the Browser Fingerprinting and Device Fingerprinting solutions developed by FONDATION B-COM that collect and analyze a set of data specific to the User's Browser (Browser Fingerprinting) or Device (Device Fingerprinting) in order to authenticate the User.
- Third party means any person, natural or legal person, outside the GTU.
- User designates any person who is willing to participate in the test phase of the Solutions free of charge.

SUBJECT

The purpose of these GTU is to set the terms and conditions for testing the Solutions by Users.

The User expressly acknowledges having read and accepted without reservation these GTU, which prevail over any other document.

These GTU may be subject to subsequent modifications, the applicable version being that in force on the date of first connection to the Site by the User.

COMMITMENTS

The User explicitly acknowledges:

- that he has voluntarily and without constraint applied for this phase of experimentation during a call organized by b<>com or one of its partners or service providers;
- to have had perfect information concerning the Solutions, in particular that a certain amount of technical data from his Browser and/or his Equipment is collected by the Solutions in order to initiate authentication procedures.

PREREQUISITE

For any use of the Solutions, the User shall ensure that it has up-to-date and compatible Equipment as well as access to telecommunication networks.

The User must be over 18 years of age.

Access to the experimentation phase requires the opening of a User account via the Test Application.

A User account is personal and associated with a designated natural person, so that the User is prohibited from sharing his or her identifiers, including with potential employees in his or her professional capacity.

The use and preservation of login credentials is the sole responsibility of the User.

Thus, any loss or involuntary disclosure of elements likely to allow a Third Party to take cognizance of the User's identifiers must be immediately reported in writing to b<>com.

The User also undertakes to immediately change his password as soon as possible.

ACCESS TO THE EXPERIMENTATION PHASE

The User receives by email an electronic invitation containing a login link to the Site and a temporary one-time password.

At the first connection to the Site, the User is identified by the origin of the invitation link and the temporary password that will have been assigned to him,

or by the Orange partner of b<>com for Orange users. After acceptance of these GTU and personalization of his password by the User, an imprint of the data of his Browser and/or the Equipment with which he has connected is taken.

This fingerprinting consists of collecting numerous technical characteristics of the Equipment or information relating to the Browser used.

A cookie (i.e. a small file saved on the User's Equipment that enables b<>com to recognize it at each connection) is also installed in order to calculate the performance of the Solutions.

For each subsequent connection, the User's authentication is carried out via the personalized password and/or the analysis of the fingerprint of the Browser or the Equipment.

At each connection, an update of the fingerprint is performed. The User will be systematically informed of the update of the fingerprint by a concomitant display on the screen of the Equipment.

During any authentication, the User may refuse the scan by clicking on the "cancel" button. This procedure cancels the authentication procedure in progress.

If authentication fails, the User may request that the fingerprint be reset or added, via a temporary reset link sent to him/her upon first request.

The User can delete the enrolled fingerprints at any time during the test via the user interface of the Account Management Application.

It is reminded that the cost of the mobile connection to access and use the Test Application is the responsibility of the User.

DATA COLLECTED

Nature of the data

The data collected will be processed under the responsibility of the Fondation de Coopération Scientifique b<>com whose address is:

1219 avenue des Champs Blancs,
ZAC Les Champs Blancs,
35510 CESSON-SEVIGNE
(SIREN 751 468 943)

The data collected for the experimentation phase are of a personal and technical nature.

The "Personal Data" that can directly identify the User such as:

- Name,
- First name,
- Email address.

The "Technical Data" are the technical characteristics of the Equipment or Browsers making up the fingerprints operated by the Solutions which, independently of each other, do not make it possible to identify the User directly, but whose combination makes it possible to identify the User of an Equipment or a Browser with a high success rate.

These Technical Data are:

For the Browser Fingerprinting:

- Hardware features such as screen size, graphics card, etc.
- Software features such as operating system, browser, installed plugins, fonts, etc.
- Customization features such as default language, time zone, etc.

For the Device Fingerprinting:

- Hardware features such as screen size, graphics card, etc.
- Software features such as operating system, browser, etc.
- Customization features such as default language, time zone, etc.
- Accessory data such as list of installed applications and their installation date, list of paired wifi networks and Bluetooth devices.

This Technical Data is collected and stored in a "pseudonymized" form, i.e. not allowing it to be attributed to a User unless the pseudonymization key is available.

However, the User is informed that the fingerprints deteriorate if they are not updated, as the characteristics that constitute the fingerprints of the Browser and/or the Equipment change.

For the purposes of User authentication when logging into his User account, b<>com also keeps the password associated with the account as well as an Orange identifier for Users who have applied for the tests via the Orange partner of b<>com.

All data relating to a User is kept confidential and internal to b<>com.

Nature of the data

The processing is based on the consent of the User whose data is collected. Consent is collected at the time of the first connection to the Application. The User is informed that he may withdraw his consent at any time.

Technical Data is collected for research purposes and to improve the Solutions. Fingerprints are the statistical data collected during the experimentation phase. Personal Data is collected in order to organize the tests and then to facilitate any requests for the exercise of their rights by Users. All Data will be kept for a period of three (3) years in a pseudonymized form.

Access and user rights

The User can view the list of registered Equipment at any time through the Account Management Application.

The exercise of his rights of access, modification, rectification, limitation, portability and/or deletion of his Personal Data is carried out through the user interface of the Account Management Application or by contacting the administrator b<>com at the email address [personal-data\[at\]b-com.com](mailto:personal-data[at]b-com.com).

b<>com does not associate any sub-contractor to the processing of the Users'

data nor will it transfer these data outside the European Union. b<>com may however be led to transmit to its partner Orange certain data of the Users (in particular name and surname) in the hypothesis of the participation of these Users coming from the Orange pool of experimenters for the general public within the framework of a reward program organized by Orange, and in order to allow their reward.

If the User considers that his rights relating to the data are not respected by b<>com, he can address a complaint to the CNIL (Commission Nationale de L'informatique et des Libertés – cnil.fr) or any other competent control authority.

Technical connection data

The technical data of connection of each User to the servers of b<>com are collected and are kept by b<>com for 1 year from their registration in a non-nominative form and decorrelated from the data of the experiment. These data are kept for legal security reasons, in order to be able to identify a User in case of illicit activity.

INTERRUPTION / SUSPENSION

As the experimentation is carried out by b<>com for scientific purposes, b<>com does not guarantee the permanence, continuity and quality of authentication by the Solutions. However, as this is an experimental phase, it is not bound to Users by any obligation of any kind whatsoever.

b<>com reserves the right to interrupt the operation of the Solutions in order to update them or for any question related to security or maintenance.

b<>com may decide unilaterally and without prior notice to discontinue the experimentation phase for any reason whatsoever. The User may inform b<>com of any difficulty encountered during the use of the Solutions at the email address in article 6.c.

RESPONSIBILITY

The User agrees to use the Solutions solely for the purpose for which they are intended.

The User acknowledges that b<>com is under no particular obligation whatsoever towards the User, the experimentation phase being undertaken graciously by the User without expectation of any commercial consideration of any kind whatsoever from b<>com.

In particular, b<>com is not liable for any malfunction of the Solutions, the User may at any time withdraw from the experimentation phase and delete all of his Personal and Technical Data.

b<>com cannot be held responsible in any way:

- Failure by the User or a Third Party to comply with the requirements of b<>com;
- Error of manipulation of the Solutions by the User or a Third Party and its consequences;
- Any intervention on the User's Equipment;
- Damages having a cause external to the Solutions or arising from a case of force majeure;
- Contamination by viruses of the User's Data and/or tools and/or software and/or Equipment, the protection of which is the responsibility of the User;
- Malicious intrusions by third parties on the User Account or Data piracy, despite the reasonable security measures in place;
- In the event of damage resulting from the loss, alteration or any fraudulent use of the Solutions, the accidental transmission of viruses or other harmful elements, the attitude or behavior of a Third Party, the non-conclusion of an operation;

- Damage that could be suffered by the Equipment connected to the Solutions, which is under the full responsibility of the User;
- Possible misuse of User identifiers, and more generally any information of a sensitive nature for the User.

In any case, the User shall refrain from damaging the reputation of b<>com and in particular from denigrating the Solutions, on any medium, including the Internet and social networks.

The User is solely responsible for the damages that his agents and/or his Equipment could cause to Third Parties and in particular for the consequences of a bad use of the Solutions. The responsibility of b<>com is also excluded in the event of fault or negligence of the User and/or its agents, in particular in the event of incorrect or incomplete information transmission or use of the Solutions not in accordance with the instructions and, where applicable, the documentation provided. The User guarantees b<>com against any recourse that the latter may have to suffer as a result of the use of the Solutions made by the User.

INTELLECTUAL PROPERTY

The User acknowledges that the Solutions and the Site, in particular the texts, photographs, illustrations, videos, software, content, databases, APIs, sounds, graphics, logos, or any other information or support presented by b<>com or made available to the User as part of the experimentation phase are protected by copyright, trademark and patent law and any other intellectual property right.

b<>com grants the User a non-exclusive right to use the Solutions as well as any documentation attached thereto, in the form of a license and not a sale, for the entire duration of the experimentation phase.

The present license does not confer to the User any intellectual property right on the Solutions or any copy thereof, which remain the full and exclusive property of b<>com.

Any use, reproduction, extraction not expressly authorized by b<>com under the present terms is illicit, in accordance with article L.122-6 of the Intellectual Property Code.

In particular, the User is not authorized to:

- Represent, distribute, market the elements of the Solutions, whether free of charge or for a fee, without prior written authorization from b<>com;
- Use the Solution in any way whatsoever for the purpose of designing, producing, distributing or marketing a similar, equivalent or substitute service;
- Make the Solutions available, in any way whatsoever, for the benefit of any Third Party;
- Modify, translate, reproduce, dismantle, disassemble, or derive in any way whatsoever the code of the Solutions or the accompanying documentation;
- The User therefore formally refrains from intervening or having a Third Party intervene on the Solutions. Moreover, in the case of hosted Solutions, no backup copies are permitted.

INDEPENDENCE OF CLAUSES

If any provision(s) of these GTU is (are) declared invalid pursuant to a law, a regulation, or as a result of a court decision, the other provision(s) will retain their full force and effect.

ABSENCE OF WAIVER

The fact that b<>com does not take advantage of the User's failure to comply with its obligations shall not be interpreted as a waiver of the right to invoke any subsequent failure to comply with the same or other obligations, or as a waiver of the right to ensure the application of said obligations.

APPLICABLE LAW - COMPETENT COURT

These GTU are subject to French law.

The User and b<>com will make every effort to resolve amicably any disputes that may arise relating to the validity, interpretation or execution of the present Terms and Conditions.

In the absence of an amicable agreement, the most diligent party may bring the dispute before the competent French jurisdiction.

Appendix I

Attributes List and Properties

In this chapter, we provide the list of our fingerprinting attributes together with their properties. We display the initial attributes used for the general audience dataset, and the attributes that were added for each following dataset. The exhaustive list of the attributes of the four attributes together with their properties is available on the website related to this thesis¹. Table I.1 lists the initial attributes of the general audience dataset, together with their number of distinct values observed during the experiment (Values), their normalized entropy (N. Ent.), their minimum normalized conditional entropy (MNCE), their sameness rate (% Same), their median size (Size), and their median collection time (Time). Table I.2 presents these properties for the attributes that were added for the intranet experiment, and Table I.3 displays the attributes added for the enrolled experiment.

To stay concise, we replace the name of common JavaScript objects or of API calls by abbreviations. We denote D the JavaScript `document` object, M the `Math` object, N the `navigator` object, S the `screen` object, and W the `window` object. Additionally, we denote A an initialized `Audio Context`, AA an initialized `AudioAnalyser`, and AD the `A.destination` property. Finally, we denote WG an initialized `WebGL Context`, WM the `WG.MAX_` prefix, and WI the `WG.IMPLEMENTATION_` prefix.

Due to the diversity of JavaScript engines, some properties are accessible

¹<https://project.inria.fr/fpauth/attributes>

through different names, regularly prefixed by `moz` for Firefox or `ms` for Internet Explorer. We use square brackets to easily denote these cases, and consider that `A. [B, C]` means that the property is accessed through `A.B` or `A.C`. If there is only one element inside these brackets, this one is optional. We denote `[...]` a part that is omitted but described in the corresponding attribute family description.

Table I.1: The attributes of the general audience dataset, together with their distinct values, their normalized entropy, their minimum normalized conditional entropy (MNCE), their stability, their median size, and their median collection time. The attributes from which we derive the extracted ones are ignored for the MNCE.

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
N.userAgent	38,863	0.394	0.046	0.978	115	0.000
Listing of N	1,660	0.207	0.009	0.989	502	0.001
Listing of screen	82	0.129	0.000	0.999	209	0.000
N.language	228	0.066	0.006	0.999	2	0.000
N.languages	1,448	0.094	0.010	0.998	17	0.000
N.userLanguage	124	0.036	0.001	1.000	1	0.000
N.systemLanguage	115	0.037	0.001	1.000	1	0.000
N.browserLanguage	52	0.036	0.000	1.000	1	0.000
N.platform	32	0.068	0.000	1.000	5	0.000
N.appName	5	0.003	0.000	1.000	8	0.000
N.appVersion	37,310	0.342	0.000	0.984	107	0.000
N.appMinorVersion	10	0.035	0.000	1.000	1	0.000
N.product	2	0.003	0.000	1.000	5	0.000
N.productSub	10	0.067	0.000	1.000	8	0.000
N.vendor	21	0.064	0.000	1.000	1	0.000
N.vendorSub	2	0.035	0.000	1.000	1	0.000
N.cookieEnabled	1	0.000	0.000	1.000	4	0.000
N.cpuClass	6	0.039	0.000	1.000	1	0.000
N.oscpu	60	0.071	0.000	1.000	1	0.000
N.hardwareConcurrency	28	0.086	0.025	0.999	1	0.000
N.buildID	1,351	0.076	0.010	0.989	1	0.000
[N.security, D.security[Policy]]	30	0.038	0.001	1.000	7	0.000
N.permissions	3	0.045	0.000	1.000	1	0.000
W.Notification.permission	5	0.043	0.001	0.999	7	0.000
W.Notification.maxActions	3	0.041	0.000	1.000	1	0.000

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
N. msM, m axTouchPoints	42	0.098	0.004	0.999	3	0.000
D. createEvent("TouchEvent") support	3	0.032	0.000	1.000	1	0.000
W. ontouchstart support	3	0.032	0.000	1.000	1	0.000
N. javaEnabled()	4	0.045	0.008	0.997	1	0.000
N. taintEnabled()	3	0.045	0.000	1.000	1	0.000
[[N, W] .doNotTrack, N. msDoNotTrack]	11	0.085	0.012	1.000	6	0.000
N. connection support	3	0.022	0.000	1.000	1	0.000
N. connection.type	12	0.028	0.003	0.992	1	0.000
N. connection.downlink	91	0.032	0.003	0.995	1	0.000
N. mozC, c onnection.bandwidth	6	0.023	0.000	1.000	3	0.000
N. mediaDevices support	3	0.044	0.000	0.999	1	0.000
N. mediaDevices.getSupportedConstraints()	12	0.090	0.000	0.997	144	0.000
W. Intl.Collator().resolvedOptions()	311	0.096	0.000	0.999	115	0.005
W. Intl.DateFormat().resolvedOptions()	1,849	0.154	0.011	0.996	111	0.003
W. Intl.NumberFormat().resolvedOptions()	260	0.070	0.000	0.999	138	0.001
W. Intl.v8BreakIterator().resolvedOptions()	75	0.046	0.000	0.999	1	0.000
N. getGamepads()	18	0.090	0.000	0.998	1	0.001
W. InstallTrigger.enabled()	4	0.037	0.000	1.000	1	0.000
W. InstallTrigger.updateEnabled()	4	0.037	0.000	1.000	1	0.000
N. msManipulationViewsEnabled	5	0.052	0.000	1.000	3	0.000
N. msP, p ointerEnabled	9	0.051	0.000	1.000	3	0.000
D. msCapLockWarningOff	3	0.039	0.000	1.000	1	0.000
D. msCSSOMElementFloatMetrics	4	0.039	0.000	1.000	1	0.000
N. msW, w ebdriver	6	0.048	0.001	1.000	3	0.000
W. Debug.debuggerEnabled	5	0.042	0.000	0.989	1	0.000
W. Debug.setNonUserCodeExceptions	4	0.042	0.000	0.989	1	0.000

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
new Date(2016, 1, 1).getTimezoneOffset()	60	0.008	0.001	0.999	2	0.000
Different Timezone at 01/01 and 06/01	3	0.005	0.002	0.999	1	0.000
S.width	1,280	0.192	0.005	0.987	4	0.000
S.height	1,016	0.188	0.015	0.987	3	0.000
W.screenX	3,071	0.125	0.047	0.925	1	0.000
W.screenY	1,181	0.126	0.049	0.925	1	0.000
S.availWidth	1,746	0.202	0.016	0.985	4	0.000
S.availHeight	1,353	0.268	0.058	0.984	3	0.000
S.availTop	460	0.060	0.003	1.000	1	0.000
S.availLeft	372	0.048	0.005	0.999	1	0.000
S.(pixelDepth, colorDepth)	14	0.031	0.001	1.000	5	0.000
S.deviceXDPI	249	0.073	0.000	0.993	1	0.000
S.deviceYDPI	249	0.073	0.000	0.993	1	0.000
S.systemXDPI	75	0.053	0.000	0.999	1	0.000
S.systemYDPI	75	0.053	0.000	0.999	1	0.000
S.logicalXDPI	6	0.039	0.000	1.000	1	0.000
S.logicalYDPI	6	0.039	0.000	1.000	1	0.000
W.innerWidth	2,572	0.263	0.023	0.957	4	0.000
W.innerHeight	2,297	0.388	0.181	0.906	3	0.000
W.outerWidth	2,481	0.293	0.074	0.909	4	0.000
W.outerHeight	4,046	0.327	0.117	0.872	3	0.000
W.devicePixelRatio	2,035	0.103	0.026	0.992	1	0.000
W.mozInnerScreenX	3,682	0.065	0.011	0.991	1	0.000
W.mozInnerScreenY	3,170	0.102	0.020	0.991	1	0.000
W.offscreenBuffering	4	0.067	0.000	1.000	4	0.000
S.orientation	3	0.044	0.000	1.000	1	0.000

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
S.[orientation.type, [moz, ms]Orientation]	26	0.107	0.003	0.994	21	0.000
S.orientation.angle	7	0.050	0.002	0.996	1	0.000
W.localStorage support	4	0.001	0.001	1.000	1	0.001
W.sessionStorage support	4	0.000	0.000	1.000	1	0.000
W.indexedDB support	3	0.002	0.000	1.000	1	0.000
W.openDatabase support	3	0.045	0.000	1.000	1	0.000
W.caches support	3	0.045	0.000	1.000	1	0.000
M.tan(-1e300)	15	0.087	0.002	1.000	19	0.000
M.tan(3.14159265359 * 0.3333 * 1e300)	12	0.085	0.000	0.999	18	0.000
M.acos(0.0000000000000001)	4	0.058	0.000	1.000	18	0.000
M.acosh(1.00000000000001)	7	0.071	0.000	1.000	24	0.000
M.asinh(0.00001)	6	0.071	0.000	1.000	23	0.000
M.asinh(1e300)	6	0.058	0.000	1.000	17	0.000
M.atan(2)	3	0.018	0.000	1.000	18	0.000
M.atan2(0.01, 1000)	3	0.045	0.000	1.000	23	0.000
M.atanh(0.0001)	5	0.070	0.000	1.000	22	0.000
M.cosh(15)	8	0.058	0.000	1.000	18	0.000
M.exp(-1e2)	8	0.028	0.000	1.000	21	0.000
M.exp(1e2)	9	0.028	0.000	1.000	22	0.000
M.LOG2E	3	0.039	0.000	1.000	18	0.000
M.LOG10E	3	0.000	0.000	1.000	18	0.000
M.E	2	0.000	0.000	1.000	17	0.000
M.LN10	3	0.000	0.000	1.000	17	0.000
D.defaultCharset	71	0.075	0.001	0.999	1	0.000
Width and height of fallback font text	2,347	0.199	nan	0.998	11	0.100
W.[performance, console].jsHeapSizeLimit	24	0.083	0.002	0.991	3	0.000

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
W.menubar.visible	5	0.035	0.000	1.000	4	0.000
W.isSecureContext	4	0.045	0.000	0.999	5	0.000
S.fontSmoothingEnabled	4	0.042	0.001	1.000	1	0.000
new Date(0)	1,846	0.118	0.004	0.998	82	0.004
new Date("0001-1-1")	2,107	0.150	0.010	0.999	60	0.002
new Date(0) then setFullYear(0)	2,376	0.136	0.013	0.998	61	0.001
Detection of an ad blocker	19	0.002	0.001	0.999	1	2.157
Firebug resource detection	3	0.037	0.000	1.000	1	0.055
YahooToolbar resource detection	3	0.037	0.000	1.000	1	0.056
EasyScreenshot resource detection	3	0.037	0.000	1.000	1	0.056
Ghostery resource detection	3	0.037	0.000	1.000	1	0.057
Kaspersky resource detection	3	0.037	0.000	1.000	1	0.057
VideoDownloadHelper resource detection	3	0.038	0.001	0.998	1	0.057
GTranslate resource detection	3	0.037	0.000	1.000	1	0.059
Privowny resource detection	2	0.037	0.000	1.000	1	0.059
Privowny page content change	3	0.000	0.000	1.000	3	2.170
UBlock page content change	4	0.000	0.000	1.000	1	2.183
Pinterest page content change	10	0.001	0.001	0.999	1	2.153
Grammarly page content change	3	0.000	0.000	1.000	1	2.184
Adguard page content change	3	0.000	0.000	1.000	1	2.165
Evernote page content change	3	0.000	0.000	1.000	1	2.181
TOTL page content change	3	0.000	0.000	1.000	1	2.181
IE Tab page content change	11	0.000	0.000	1.000	1	2.156
WebRTC fingerprinting	671,254	0.294	0.144	0.765	1	0.764
WG.SHADING_LANGUAGE_VERSION	23	0.103	0.000	0.996	18	0.001
WG.VERSION	247	0.123	0.008	0.995	10	0.000

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
WG.VENDOR	11	0.080	0.000	0.997	7	0.000
WG.RENDERER	14	0.089	0.000	0.996	12	0.000
WG.ALIASED_POINT_SIZE_RANGE	42	0.129	0.006	0.996	5	0.000
WG.ALIASED_LINE_WIDTH_RANGE	30	0.077	0.003	0.996	3	0.000
WM.VIEWPORT_DIMS	13	0.107	0.009	0.995	11	0.000
WG.SUBPIXEL_BITS	9	0.039	0.001	0.997	1	0.000
WG.SAMPLE_BUFFERS	5	0.039	0.000	0.996	1	0.000
WG.SAMPLES	9	0.075	0.001	0.992	1	0.000
WG.COMPRESSED_TEXTURE_FORMATS	3	0.034	0.000	0.998	23	0.000
WM.VERTEX_UNIFORM_ATTRIBUTES	18	0.118	0.006	0.996	3	0.000
WM.COMBINED_TEXTURE_IMAGE_UNITS	19	0.079	0.003	0.996	2	0.000
WM.FRAGMENT_UNIFORM_ATTRIBUTES	18	0.109	0.004	0.996	3	0.000
WM.CUBE_MAP_TEXTURE_SIZE	11	0.084	0.008	0.995	5	0.000
WG.STENCIL_VALUE_MASK	8	0.050	0.000	0.996	10	0.000
WG.STENCIL_WRITEMASK	7	0.050	0.000	0.996	10	0.000
WG.STENCIL_BACK_VALUE_MASK	8	0.050	0.000	0.996	10	0.000
WG.STENCIL_BACK_WRITEMASK	7	0.050	0.000	0.996	10	0.000
WM.TEXTURE_SIZE	10	0.081	0.009	0.995	5	0.000
WG.DEPTH_BITS	7	0.047	0.000	0.996	2	0.000
WM.VARYING_ATTRIBUTES	19	0.121	0.009	0.996	2	0.000
WI.COLOR_READ_FORMAT	7	0.073	0.003	0.994	4	0.000
WM.RENDERBUFFER_SIZE	11	0.080	0.003	0.995	5	0.000
WG.STENCIL_BITS	5	0.016	0.000	0.997	1	0.000
WM.TEXTURE_IMAGE_UNITS	7	0.033	0.000	0.997	2	0.000
WM.VERTEX_ATTRIBS	8	0.017	0.000	0.997	2	0.000
WM.VERTEX_TEXTURE_IMAGE_UNITS	9	0.057	0.001	0.996	2	0.000

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
WL.COLOR_READ_TYPE	6	0.041	0.000	0.996	4	0.000
WM.TEXTURE_MAX_ANISOTROPY_EXT	9	0.029	0.000	0.997	2	0.001
WG.getContextAttributes()	54	0.114	0.009	0.995	138	0.000
WG.getSupportedExtensions()	535	0.209	0.027	0.990	401	0.008
WG.[...].UNMASKED_VENDOR_WEBGL	27	0.115	0.000	0.995	9	0.000
WG.[...].UNMASKED_RENDERER_WEBGL	3,786	0.268	0.073	0.991	20	0.000
WebGL precision format	25	0.071	0.001	0.996	114	0.001
Our designed WebGL canvas	1,158	0.263	0.023	0.990	64	0.041
Width and position of a created div	17,832	0.324	nan	0.940	18	0.086
Colors of layout components	7,707	0.153	nan	0.986	492	0.090
Size of bounding boxes of a created div	16,396	0.369	nan	0.470	31	0.195
Presence of fonts	17,960	0.305	0.110	0.996	198	0.456
Support of video codecs	84	0.114	0.001	0.999	78	0.002
Support of audio codecs	52	0.128	0.002	0.999	61	0.001
Support of streaming codecs	50	0.132	0.010	0.999	133	0.002
Support of recording codecs	7	0.069	0.000	0.999	140	0.001
W.speechSynthesis.getVoices()	3,967	0.204	0.034	0.945	250	0.550
N.plugins	314,518	0.394	0.100	0.950	134	0.001
N.mimeTypes	174,876	0.311	0.017	0.982	112	0.000
A.state	5	0.082	0.000	0.999	7	0.000
A.sampleRate	16	0.070	0.019	0.997	5	0.000
AD.channelCount	5	0.036	0.000	1.000	1	0.000
AD.channelCountMode	4	0.036	0.000	1.000	8	0.000
AD.channelInterpretation	4	0.036	0.000	1.000	8	0.000
AD.maxChannelCount	20	0.058	0.003	1.000	1	0.000
AD.numberOfInputs	3	0.035	0.000	1.000	1	0.000

Attribute	Values	N. Ent.	MINCE	% Same	Size	Time
AD.numberOfOutputs	3	0.035	0.000	1.000	1	0.000
AA.channelCount	5	0.067	0.000	1.000	1	0.001
AA.channelCountMode	5	0.037	0.000	1.000	3	0.000
AA.channelInterpretation	4	0.036	0.000	1.000	8	0.000
AA.numberOfInputs	4	0.036	0.000	1.000	1	0.000
AA.numberOfOutputs	4	0.036	0.000	1.000	1	0.000
AA.fftSize	3	0.035	0.000	1.000	4	0.000
AA.frequencyBinCount	3	0.035	0.000	1.000	4	0.000
AA.maxDecibels	3	0.035	0.000	1.000	3	0.000
AA.minDecibels	3	0.035	0.000	1.000	4	0.000
AA.smoothingTimeConstant	4	0.046	0.000	0.998	3	0.000
Audio fp simple	337	0.153	0.004	0.958	18	1.403
Audio fp advanced	561	0.147	0.001	0.953	17	1.636
Audio fp advanced frequency data	546	0.161	0.011	0.950	17	1.639
Our designed HTML5 canvas (PNG)	269,874	0.420	0.021	0.922	64	0.260
Our designed HTML5 canvas (JPEG)	205,005	0.399	0.001	0.936	64	0.265
HTML5 canvas inspired by AmIUnique (PNG)	8,948	0.353	0.002	0.986	64	0.031
HTML5 canvas inspired by AmIUnique (JPEG)	6,514	0.312	0.001	0.989	64	0.039
HTML5 canvas similar to Morellian (PNG)	41,845	0.385	0.034	0.947	64	0.037
Accept HTTP header	26	0.028	0.000	0.997	3	0.000
Accept-Encoding HTTP header	30	0.019	0.002	1.000	13	0.000
Accept-Language HTTP header	2,833	0.124	0.022	0.999	35	0.000
User-Agent HTTP header	20,961	0.350	0.002	0.978	108	0.000
Accept-Charset HTTP header	18	0.002	0.000	1.000	1	0.000
Cache-Control HTTP header	47	0.055	0.023	0.706	1	0.000
Connection HTTP header	2	0.000	0.000	1.000	5	0.000

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
TE HTTP header	2	0.000	0.000	1.000	1	0.000
Upgrade-Insecure-Requests HTTP header	2	0.000	0.000	1.000	1	0.000
X-WAP-Profile HTTP header	4	0.000	0.000	1.000	1	0.000
X-Requested-With HTTP header	151	0.004	0.000	1.000	1	0.000
X-ATT-DeviceId HTTP header	1	0.000	0.000	1.000	1	0.000
X-UIDH HTTP header	1	0.000	0.000	1.000	1	0.000
X-Network-Info HTTP header	4	0.000	0.000	1.000	1	0.000
Via HTTP header	4,272	0.007	0.003	0.999	1	0.000
Any conditional HTTP headers	5,394	0.095	0.042	0.899	192	0.000
Number of bounding boxes	15	0.062	0.008	0.998	1	0.195
Number of plugins	54	0.147	0.000	0.984	1	0.001
Number of WebGL extensions	28	0.176	0.000	0.991	2	0.008
Width and height of first bounding box	12,937	0.350	nan	0.486	30	0.195
Width and height of second bounding box	1,332	0.103	nan	0.941	1	0.195
Width and height of third bounding box	772	0.076	nan	0.965	1	0.195
List of widths of bounding boxes	6,690	0.299	nan	0.986	16	0.195
List of heights of bounding boxes	2,222	0.264	nan	0.474	14	0.195
Width of first bounding box	4,418	0.281	0.038	0.987	14	0.195
Height of first bounding box	1,848	0.246	0.070	0.490	14	0.195
Width of second bounding box	471	0.085	0.002	0.998	1	0.195
Height of second bounding box	398	0.088	0.007	0.941	1	0.195
Width of third bounding box	224	0.060	0.004	0.999	1	0.195
Height of third bounding box	343	0.064	0.000	0.966	1	0.195
Width of a created div	15,473	0.316	0.007	0.940	6	0.086
Origin of a created div	16,375	0.316	0.008	0.942	11	0.086
Width of fallback font text	1,029	0.184	0.024	0.998	5	0.100

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
Height of fallback font text	1,159	0.164	0.010	0.998	5	0.100
Color of ActiveBorder element	702	0.078	0.005	1.000	18	0.090
Color of ActiveCaption element	475	0.073	0.002	1.000	18	0.090
Color of AppWorkspace element	321	0.067	0.001	1.000	18	0.090
Color of Background element	2,917	0.074	0.017	1.000	16	0.090
Color of ButtonFace element	297	0.079	0.004	1.000	18	0.090
Color of ButtonHighlight element	264	0.058	0.000	1.000	18	0.090
Color of ButtonShadow element	343	0.076	0.001	1.000	18	0.090
Color of ButtonText element	104	0.004	0.000	1.000	12	0.090
Color of CaptionText element	123	0.014	0.000	1.000	12	0.090
Color of GrayText element	333	0.071	0.004	1.000	18	0.090
Color of Highlight element	1,088	0.097	0.016	0.987	17	0.090
Color of HighlightText element	89	0.049	0.001	1.000	18	0.090
Color of InactiveBorder element	334	0.060	0.000	1.000	18	0.090
Color of InactiveCaption element	441	0.062	0.001	1.000	18	0.090
Color of InactiveCaptionText element	265	0.088	0.006	0.999	15	0.090
Color of InfoBackground element	239	0.057	0.000	1.000	18	0.090
Color of InfoText element	96	0.003	0.000	1.000	12	0.090
Color of Menu element	376	0.087	0.004	1.000	18	0.090
Color of MenuText element	124	0.020	0.001	1.000	12	0.090
Color of Scroll-bar element	275	0.072	0.000	1.000	18	0.090
Color of ThreeDDarkShadow element	75	0.071	0.001	1.000	18	0.090
Color of ThreeDFace element	297	0.062	0.000	1.000	18	0.090
Color of ThreeDHighlight element	261	0.048	0.000	1.000	18	0.090
Color of ThreeDLightShadow element	280	0.074	0.001	1.000	18	0.090
Color of ThreeDShadow element	339	0.075	0.000	1.000	18	0.090

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
Color of Window element	329	0.019	0.001	1.000	18	0.090
Color of WindowFrame element	140	0.069	0.000	1.000	18	0.090
Color of WindowText element	107	0.004	0.000	1.000	12	0.090

Table I.2: The additional attributes of the intranet dataset, together with their distinct values, their normalized entropy, their minimum normalized conditional entropy (MNCE), their stability, their median size, and their median collection time.

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
A.baselatency	11	0.174	0.009	0.989	5	0.000
N.connection.downlinkMax	3	0.076	0.000	1.000	5	0.000
N.connection.effectiveType	5	0.080	0.000	0.990	5	0.000
N.connection.rtt	20	0.158	0.053	0.742	5	0.000
N.deviceMemory	5	0.080	0.001	1.000	5	0.000
W.FaceDetector	2	0.000	0.000	1.000	6	0.000
W.BarcodeDetector	3	0.014	0.000	0.998	6	0.000
N.mediaDevices.enumerateDevices()	74	0.322	0.053	0.875	39	0.004

Table I.3: The additional attributes of the enrolled dataset, together with their distinct values, their normalized entropy, their minimum normalized conditional entropy (MNCE), their stability, their median size, and their median collection time.

Attribute	Values	N. Ent.	MNCE	% Same	Size	Time
N.keyboard.getLayoutMap() on QWERTY characters	5	0.131	0.003	0.999	7	0.001
N.onLine	1	0.000	0.000	1.000	8	0.000
N.getBattery().charging	5	0.147	0.012	0.967	4	0.000
N.getBattery().chargingTime	2	0.001	0.000	1.000	5	0.000
N.getBattery().dischargingTime	2	0.001	0.000	1.000	5	0.000
N.getBattery().level	2	0.001	0.000	1.000	5	0.000

Bibliography

- [1] E. Abgrall, Y. L. Traon, M. Monperrus, S. Gombault, M. Heiderich, and A. Ribault. *XSS-FP: Browser Fingerprinting using HTML Parser Quirks*. Technical Report. University of Luxembourg, Nov. 19, 2012. URL: <https://hal.inria.fr/hal-00753926> (cit. on pp. 32, 38).
- [2] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan, and C. Diaz. “The Web Never Forgets: Persistent Tracking Mechanisms in the Wild”. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2014, pp. 674–689. ISBN: 978-1-4503-2957-6. DOI: [10.1145/2660267.2660347](https://doi.org/10.1145/2660267.2660347) (cit. on pp. 2, 41, 150).
- [3] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens, and B. Preneel. “FPDetective: Dusting the Web for Fingerprinters”. In: *ACM SIGSAC Conference on Computer & Communications Security (CCS)*. 2013, pp. 1129–1140. ISBN: 978-1-4503-2477-9. DOI: [10.1145/2508859.2516674](https://doi.org/10.1145/2508859.2516674) (cit. on p. 2).
- [4] M. G. C. Acar. “Online Tracking Technologies and Web Privacy”. thesis. KU Leuven, May 2017. URL: <https://lirias.kuleuven.be/retrieve/454271> (cit. on pp. 5, 158).
- [5] Adform Fraud Protection. *How Adform Discovered HyphBot*. Nov. 22, 2017. URL: https://site.adform.com/media/85132/hyphbot_whitepaper_.pdf.
- [6] F. Alaca. “Strengthening Password-Based Web Authentication Through Multiple Supplementary Mechanisms”. thesis. Carleton University, 2018. DOI: [10.22215/etd/2018-12840](https://doi.org/10.22215/etd/2018-12840). URL: <https://doi.org/10.22215/etd/2018-12840>.
- [7] F. Alaca, A. Abdou, and P. C. van Oorschot. “Comparative Analysis and Framework Evaluating Mimicry-Resistant and Invisible Web Authentication Schemes”. In: *IEEE Transactions on Dependable and Secure Computing (TDSC)* (Apr. 11, 2019), pp. 1–1. ISSN: 1545-5971. DOI: [10.1109/TDSC.2019.2908907](https://doi.org/10.1109/TDSC.2019.2908907) (cit. on pp. 22, 24, 28, 45).

- [8] F. Alaca and P. C. van Oorschot. “Device Fingerprinting for Augmenting Web Authentication: Classification and Analysis of Methods”. In: *Annual Conference on Computer Security Applications (ACSAC)*. Oct. 2016, pp. 289–301. ISBN: 978-1-4503-4771-6. DOI: [10.1145/2991079.2991091](https://doi.org/10.1145/2991079.2991091) (cit. on pp. 37, 45).
- [9] I. Alexa Internet. *Top Sites in France - Alexa*. accessed 2020-06-30. URL: <https://www.alexa.com/topsites/countries/FR>.
- [10] M. Ali, Z. Shaikh, M. Khan, and T. Tariq. “User Profiling Through Browser Finger Printing”. In: *International Conference on Recent Advances in Computer Systems (RACS)*. Nov. 30, 2015. ISBN: 978-94-6252-146-9. DOI: [10.2991/racs-15.2016.23](https://doi.org/10.2991/racs-15.2016.23).
- [11] V. Andalibi, F. Christophe, and T. Mikkonen. “Analysis of Paradoxes in Fingerprint Countermeasures”. In: *Conference of Open Innovations Association (FRUCT)*. Nov. 6, 2017. ISBN: 978-952-68653-2-4 (cit. on p. 202).
- [12] N. Andriamilanto, T. Allard, and G. Le Guelvouit. “Guess Who? Large-Scale Data-Centric Study of the Adequacy of Browser Fingerprints for Web Authentication”. In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. Ed. by L. Barolli, A. Poniszewska-Maranda, and H. Park. 2021, pp. 161–172. ISBN: 978-3-030-50399-4. DOI: [10.1007/978-3-030-50399-4_16](https://doi.org/10.1007/978-3-030-50399-4_16). URL: <https://hal.archives-ouvertes.fr/hal-02611624/document> (cit. on p. 8).
- [13] N. Andriamilanto, T. Allard, and G. Le Guelvouit. “FPSselect: Low-Cost Browser Fingerprints for Mitigating Dictionary Attacks against Web Authentication Mechanisms”. In: *Annual Computer Security Applications Conference (ACSAC)*. 2020. ISBN: 978-1-4503-8858-0/20/12. DOI: [10.1145/3427228.3427297](https://doi.org/10.1145/3427228.3427297). URL: <https://hal.archives-ouvertes.fr/hal-02965948/document> (cit. on p. 8).
- [14] N. Andriamilanto, T. Allard, G. Le Guelvouit, and A. Garel. “A Large-scale Empirical Analysis of Browser Fingerprints Properties for Web Authentication”. In: *arXiv:2006.09511 [cs]* (June 16, 2020). under reviews. URL: <https://arxiv.org/abs/2006.09511>.
- [15] N. Andriamilanto, G. Le Guelvouit, and T. Allard. “Browser Fingerprinting for Web Authentication, A Large-scale Empirical Analysis”. In: *Rendez-Vous de la Recherche et de l’Enseignement de la Sécurité des Systèmes d’Information (RESSI)*. national conference without proceedings. 2019 (cit. on p. 8).
- [16] *Antidetect*. accessed 2020-06-30. URL: <https://antidetect.org> (cit. on p. 163).

- [17] H. Archive. *Loading Speed*. accessed 2020-06-30. URL: <https://httparchive.org/reports/loading-speed#01>.
- [18] T. H. Archive. *Median Loading Time of Web Pages*. accessed 2020-06-16. 2020. URL: <https://httparchive.org/reports/loading-speed%5C#01> (cit. on p. 101).
- [19] M. Ashouri, H. Asadian, and C. Hammer. “Large-Scale Analysis of Sophisticated Web Browser Fingerprinting Scripts”. June 2018. URL: <https://hal.archives-ouvertes.fr/hal-01811691> (cit. on p. 162).
- [20] G. Avoine, M. A. Bingöl, I. Boureau, S. apkun, G. Hancke, S. Karda, C. H. Kim, C. Lauradoux, B. Martin, J. Munilla, A. Peinado, K. B. Rasmussen, D. Singelee, A. Tchamkerten, R. Trujillo-Rasua, and S. Vaudenay. “Security of Distance-bounding: A Survey”. In: *ACM Computing Surveys (CSUR)* 51.5 (Jan. 23, 2019), pp. 1–33. ISSN: 0360-0300, 1557-7341. DOI: [10.1145/3264628](https://doi.org/10.1145/3264628) (cit. on p. 28).
- [21] *Battery Status API has been removed*. accessed 2020-06-30. 2016. URL: <https://www.fxsitecompat.dev/en-CA/docs/2016/battery-status-api-has-been-removed> (cit. on pp. 37, 215).
- [22] P. Baumann, S. Katzenbeisser, M. Stopczynski, and E. Tews. “Disguised Chromium Browser: Robust Browser, Flash and Canvas Fingerprinting Protection”. In: *ACM Workshop on Privacy in the Electronic Society (WPES)*. 2016, pp. 37–46. ISBN: 978-1-4503-4569-9. DOI: [10.1145/2994620.2994621](https://doi.org/10.1145/2994620.2994621) (cit. on pp. 41, 56, 163).
- [23] B. Bilgi and B. Tugrul. “A Shoulder-Surfing Resistant Graphical Authentication Method”. In: *International Conference on Artificial Intelligence and Data Processing (IDAP)*. Sept. 2018, pp. 1–4. DOI: [10.1109/IDAP.2018.8620934](https://doi.org/10.1109/IDAP.2018.8620934) (cit. on pp. 21, 29).
- [24] C. Blakemore, J. Redol, and M. Correia. “Fingerprinting for Web Applications: From Devices to Related Groups”. In: *IEEE Trustcom/BigDataSE/ISPA*. Aug. 2016, pp. 144–151. DOI: [10.1109/TrustCom.2016.0057](https://doi.org/10.1109/TrustCom.2016.0057) (cit. on pp. 4, 5, 46, 158, 178, 195).
- [25] K. Boda, Á. M. Földes, G. G. Gulyás, and S. Imre. “User Tracking on the Web via Cross-browser Fingerprinting”. In: *Nordic Conference on Information Security Technology for Applications (NordSec)*. 2012, pp. 31–46. ISBN: 978-3-642-29614-7. DOI: [10.1007/978-3-642-29615-4_4](https://doi.org/10.1007/978-3-642-29615-4_4) (cit. on pp. 2, 200).
- [26] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh. “Mobile Device Identification via Sensor Fingerprinting”. In: *arXiv:1408.1416 [cs]* (Aug. 6, 2014). URL: <https://arxiv.org/abs/1408.1416> (cit. on p. 33).

- [27] J. Bonneau, C. Herley, P. C. v. Oorschot, and F. Stajano. “The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes”. In: *IEEE Symposium on Security and Privacy (S&P)*. May 2012, pp. 553–567. DOI: [10.1109/SP.2012.44](https://doi.org/10.1109/SP.2012.44) (cit. on pp. [16](#), [24](#), [27](#), [45](#), [176](#), [194](#)).
- [28] J. Bonneau. “The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords”. In: *IEEE Symposium on Security and Privacy (S&P)*. May 2012, pp. 538–552. DOI: [10.1109/SP.2012.49](https://doi.org/10.1109/SP.2012.49) (cit. on pp. [1](#), [2](#), [5](#), [44](#), [158](#), [161](#), [163](#), [176](#), [194](#)).
- [29] J. Bonneau, E. Bursztein, I. Caron, R. Jackson, and M. Williamson. “Secrets, Lies, and Account Recovery: Lessons from the Use of Personal Knowledge Questions at Google”. In: *International Conference on World Wide Web (WWW)*. May 18, 2015, pp. 141–150. ISBN: 978-1-4503-3469-3. DOI: [10.1145/2736277.2741691](https://doi.org/10.1145/2736277.2741691) (cit. on pp. [21](#), [28](#)).
- [30] J. Bonneau, C. Herley, P. C. van Oorschot, and F. Stajano. “Passwords and the Evolution of Imperfect Authentication”. In: *Communications of the ACM* 58.7 (June 2015), pp. 78–87. ISSN: 0001-0782. DOI: [10.1145/2699390](https://doi.org/10.1145/2699390) (cit. on p. [2](#)).
- [31] R. Broenink. “Using Browser Properties for Fingerprinting Purposes”. In: *Twente Student Conference on IT*. 2012.
- [32] J. Brooke. “SUS: A Retrospective”. In: *Journal of Usability Studies* 8.2 (Feb. 1, 2013), pp. 29–40. ISSN: 1931-3357 (cit. on p. [24](#)).
- [33] S. Brostoff and M. A. Sasse. “Are Passfaces More Usable Than Passwords? A Field Trial Investigation”. In: *People and Computers XIV Usability or Else!* Ed. by S. McDonald, Y. Waern, and G. Cockton. 2000, pp. 405–424. ISBN: 978-1-4471-0515-2. DOI: [10.1007/978-1-4471-0515-2_27](https://doi.org/10.1007/978-1-4471-0515-2_27) (cit. on p. [23](#)).
- [34] *Browser Fingerprinting: 93% of all user configurations are unique* | Henning Tillmann. accessed 2020-06-30. URL: <https://www.henning-tillmann.de/en/2014/05/browser-fingerprinting-93-of-all-user-configurations-are-unique> (cit. on p. [162](#)).
- [35] T. Bujlow, V. Carela-Español, J. Solé-Pareta, and P. Barlet-Ros. “A Survey on Web Tracking: Mechanisms, Implications, and Defenses”. In: *Proceedings of the IEEE* 105.8 (Aug. 2017), pp. 1476–1510. ISSN: 1558-2256. DOI: [10.1109/JPROC.2016.2637878](https://doi.org/10.1109/JPROC.2016.2637878).

- [36] E. Bursztein, A. Malyshev, T. Pietraszek, and K. Thomas. “Picasso: Lightweight Device Class Fingerprinting for Web Clients”. In: *Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM)*. Oct. 24, 2016, pp. 93–102. ISBN: 978-1-4503-4564-4. DOI: [10.1145/2994459.2994467](https://doi.org/10.1145/2994459.2994467) (cit. on pp. [2](#), [4](#), [5](#), [29](#), [37](#), [39](#), [48](#), [55](#), [56](#), [149](#), [157](#), [158](#), [200](#)).
- [37] *Can I use... Support tables for HTML5, CSS3, etc - WebRTC Peer-to-peer connections*. accessed 2020-06-30. URL: <https://caniuse.com/#search=webrtc> (cit. on pp. [134](#), [216](#)).
- [38] Y. Cao, S. Li, and E. Wijmans. “(Cross-)Browser Fingerprinting via OS and Hardware Level Features”. In: *Network and Distributed System Security Symposium (NDSS)*. Jan. 1, 2017. DOI: [10.14722/ndss.2017.23152](https://doi.org/10.14722/ndss.2017.23152) (cit. on pp. [56](#), [123](#), [200](#)).
- [39] W. Clarkson, T. Weyrich, A. Finkelstein, N. Heninger, J. A. Halderman, and E. W. Felten. “Fingerprinting Blank Paper Using Commodity Scanners”. In: *IEEE Symposium on Security and Privacy (S&P)*. May 17, 2009, pp. 301–314. ISBN: 978-0-7695-3633-0. DOI: [10.1109/SP.2009.7](https://doi.org/10.1109/SP.2009.7) (cit. on p. [31](#)).
- [40] R. Cohen and L. Katzir. “The Generalized Maximum Coverage Problem”. In: *Information Processing Letters* 108.1 (Sept. 15, 2008), pp. 15–22. ISSN: 0020-0190. DOI: [10.1016/j.ipl.2008.03.017](https://doi.org/10.1016/j.ipl.2008.03.017).
- [41] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. 3rd ed. 2009. ISBN: 978-0-262-03384-8.
- [42] *Cybercrime market selling full digital fingerprints of over 60,000 users*. accessed 2020-06-30. 2019. URL: <https://www.zdnet.com/article/cybercrime-market-selling-full-digital-fingerprints-of-over-60000-users>.
- [43] A. Das, G. Acar, N. Borisov, and A. Pradeep. “The Web’s Sixth Sense: A Study of Scripts Accessing Smartphone Sensors”. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2018, pp. 1515–1532. ISBN: 978-1-4503-5693-0. DOI: [10.1145/3243734.3243860](https://doi.org/10.1145/3243734.3243860) (cit. on pp. [41](#), [45](#)).
- [44] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang. “The Tangled Web of Password Reuse”. In: *Network and Distributed System Security Symposium (NDSS)*. 2014, pp. 23–26. ISBN: 1-891562-35-5. DOI: [10.14722/ndss.2014.23357](https://doi.org/10.14722/ndss.2014.23357) (cit. on pp. [2](#), [79](#), [194](#)).
- [45] R. Data. *The Screen Object*. accessed 2019-10-01. URL: https://www.w3schools.com/jsref/obj_screen.asp (cit. on p. [30](#)).

- [46] DataReportal. *Digital 2020 Global Digital Overview*. Jan. 30, 2020. URL: <https://www.slideshare.net/DataReportal/digital-2020-global-digital-overview-january-2020-v01-226017535> (cit. on p. 30).
- [47] N. I. Daud, G. R. Haron, and S. S. S. Othman. “Adaptive Authentication: Implementing Random Canvas Fingerprinting as User Attributes Factor”. In: *IEEE Symposium on Computer Applications Industrial Electronics (ISCAIE)*. Apr. 2017, pp. 152–156. DOI: [10.1109/ISCAIE.2017.8074968](https://doi.org/10.1109/ISCAIE.2017.8074968) (cit. on pp. 48, 194).
- [48] E. Dauterman, H. Corrigan-Gibbs, D. Mazières, D. Boneh, and D. Rizzo. “True2F: Backdoor-Resistant Authentication Tokens”. In: *IEEE Symposium on Security and Privacy (S&P)*. May 2019, pp. 398–416. ISBN: 978-1-5386-6660-9. DOI: [10.1109/SP.2019.00048](https://doi.org/10.1109/SP.2019.00048) (cit. on p. 21).
- [49] *Desktop Browser Market Share France*. accessed 2020-06-30. 2020. URL: <https://gs.statcounter.com/browser-market-share/desktop/france/2020> (cit. on pp. 61, 218).
- [50] *Desktop Browser Market Share Worldwide | StatCounter Global Stats*. accessed 2020-06-30. 2020. URL: <https://gs.statcounter.com/browser-market-share/desktop/worldwide/2020> (cit. on pp. 61, 218).
- [51] Distil Networks. *Bad Bot Report: The Bot Arms Race Continues*. Distil Networks, 2019. URL: <https://www.bluecubesecurity.com/wp-content/uploads/bad-bot-report-2019LR.pdf> (cit. on p. 39).
- [52] E. K. Donald. “The Computer as Master Mind”. In: *Journal of Recreational Mathematics* 9.1 (1976), pp. 1–6.
- [53] V. X. Duong. “A Proposal of a Cross-Browser User Tracking Method with Browser Fingerprint”. Master Thesis. Keio University, 2011. URL: <https://www.sfc.wide.ad.jp/thesis/2011/files/nobita-publish-thesis.pdf> (cit. on p. 200).
- [54] C. Dwork. “Differential Privacy: A Survey of Results”. In: *Theory and Applications of Models of Computation*. Ed. by M. Agrawal, D. Du, Z. Duan, and A. Li. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–19. ISBN: 978-3-540-79228-4 (cit. on p. 201).
- [55] S. Eberz, K. B. Rasmussen, V. Lenders, and I. Martinovic. “Evaluating Behavioral Biometrics for Continuous Authentication: Challenges and Metrics”. In: *ACM Asia Conference on Computer and Communications Security (AsiaCCS)*. 2017, pp. 386–399. ISBN: 978-1-4503-4944-4. DOI: [10.1145/3052973.3053032](https://doi.org/10.1145/3052973.3053032) (cit. on pp. 25, 82).

- [56] P. Eckersley. “How Unique is Your Web Browser?” In: *International Conference on Privacy Enhancing Technologies (PETS)*. 2010, pp. 1–18. ISBN: 978-3-642-14526-1. DOI: [10.1007/978-3-642-14527-8_1](https://doi.org/10.1007/978-3-642-14527-8_1) (cit. on pp. [2–5](#), [31–34](#), [36–38](#), [41](#), [54](#), [58](#), [61](#), [66](#), [68](#), [69](#), [81](#), [85](#), [120–123](#), [158](#), [194](#), [195](#), [202–204](#), [206](#), [228](#)).
- [57] S. Englehardt and A. Narayanan. “Online Tracking: A 1-million-site Measurement and Analysis”. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Oct. 24, 2016, pp. 1388–1401. ISBN: 978-1-4503-4139-4. DOI: [10.1145/2976749.2978313](https://doi.org/10.1145/2976749.2978313) (cit. on pp. [2](#), [36](#), [38](#), [41](#), [56](#), [149](#), [211](#)).
- [58] *European Directive 2002/58/EC*. accessed 2020-06-30. URL: <https://data.europa.eu/eli/dir/2002/58/2009-12-19> (cit. on pp. [61](#), [62](#)).
- [59] *European Directive 2009/136/EC*. accessed 2020-06-30. URL: <https://data.europa.eu/eli/dir/2009/136/2009-12-19> (cit. on pp. [61](#), [62](#)).
- [60] Experian. *Experian plc - Illegal web trade of personal information soars to record highs*. accessed 2020-06-30. 2014. URL: <https://www.experianplc.com/media/news/2014/illegal-web-trade-of-personal-information-soars-to-record-highs> (cit. on p. [18](#)).
- [61] A. FaizKhademi, M. Zulkernine, and K. Weldemariam. “FPGuard: Detection and Prevention of Browser Fingerprinting”. In: *IFIP Annual Conference on Data and Applications Security and Privacy (DBSec)*. Ed. by P. Samarati. 2015, pp. 293–308. ISBN: 978-3-319-20810-7. DOI: [10.1007/978-3-319-20810-7_21](https://doi.org/10.1007/978-3-319-20810-7_21) (cit. on p. [2](#)).
- [62] N. M. Al-Fannah. “One Leak will sink a Ship: WebRTC IP Address Leaks”. In: *International Carnahan Conference on Security Technology (ICCST)*. Oct. 2017, pp. 1–5. DOI: [10.1109/CCST.2017.8167801](https://doi.org/10.1109/CCST.2017.8167801).
- [63] N. M. Al-Fannah and W. Li. “Not All Browsers are Created Equal: Comparing Web Browser Fingerprintability”. In: *International Workshop on Security (IWSEC)*. Ed. by S. Obana and K. Chida. 2017, pp. 105–120. ISBN: 978-3-319-64200-0. DOI: [10.1007/978-3-319-64200-0_7](https://doi.org/10.1007/978-3-319-64200-0_7) (cit. on pp. [56](#), [60](#)).
- [64] N. M. Al-Fannah, W. Li, and C. J. Mitchell. “Beyond Cookie Monster Amnesia: Real World Persistent Online Tracking”. In: *Information Security*. Ed. by L. Chen, M. Manulis, and S. Schneider. 2018, pp. 481–501. ISBN: 978-3-319-99136-8. DOI: [10.1007/978-3-319-99136-8_26](https://doi.org/10.1007/978-3-319-99136-8_26) (cit. on pp. [2](#), [4](#), [194](#)).

- [65] D. Fifield and S. Egelman. “Fingerprinting Web Users Through Font Metrics”. In: *Financial Cryptography and Data Security (FC)*. Ed. by R. Böhme and T. Okamoto. 2015, pp. 107–124. ISBN: 978-3-662-47854-7. DOI: [10.1007/978-3-662-47854-7_7](https://doi.org/10.1007/978-3-662-47854-7_7) (cit. on pp. [2](#), [4](#), [5](#), [29](#), [34](#), [41](#), [46](#), [54](#), [123](#), [134](#), [158](#), [178](#), [195](#), [199](#), [206](#)).
- [66] D. Fifield and M. G. Epner. “Fingerprintability of WebRTC”. In: *arXiv:1605.08805 [cs]* (May 27, 2016). URL: <https://arxiv.org/abs/1605.08805> (cit. on pp. [2](#), [33](#), [41](#), [216](#)).
- [67] *fingerprints/fingerprints2: Modern & flexible browser fingerprinting library*. accessed 2020-06-30. URL: <https://github.com/fingerprints/fingerprints2> (cit. on pp. [54](#), [204](#)).
- [68] U. Fiore, A. Castiglione, A. D. Santis, and F. Palmieri. “Countering Browser Fingerprinting Techniques: Constructing a Fake Profile with Google Chrome”. In: *International Conference on Network-Based Information Systems (NBIS)*. Sept. 2014, pp. 355–360. DOI: [10.1109/NBIS.2014.102](https://doi.org/10.1109/NBIS.2014.102) (cit. on p. [2](#)).
- [69] E. Flood and J. Karlsson. “Browser Fingerprinting”. Master Thesis. University of Gothenburg, May 2012. URL: <https://hdl.handle.net/20.500.12380/163728> (cit. on pp. [5](#), [47](#), [158](#), [195](#)).
- [70] M. Focus. *Device Fingerprinting for Low Friction Authentication*. White Paper. accessed 2020-06-30. 2019. URL: <https://www.microfocus.com/media/white-paper/device-fingerprinting-for-low-friction-authentication-wp.pdf> (cit. on p. [3](#)).
- [71] E. F. Foundation. *Do Not Track*. accessed 2019-10-01. URL: <https://www.eff.org/issues/do-not-track> (cit. on p. [31](#)).
- [72] E. F. Foundation. *Panopticklick*. accessed 2019-10-01. URL: <https://panopticklick.eff.org> (cit. on pp. [2](#), [31](#), [41](#)).
- [73] A. Francillon, B. Danev, and S. Capkun. “Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars”. In: *Network and Distributed System Security Symposium (NDSS)*. 2011. URL: <https://www.ndss-symposium.org/wp-content/uploads/2017/09/franc.pdf> (cit. on p. [28](#)).
- [74] *FraudFox | The most advance antidetect tools for privately browsing online*. accessed 2020-06-30. URL: <https://fraudfox.net>.
- [75] M. Gamassi, M. Lazzaroni, M. Misino, V. Piuri, D. Sana, and F. Scotti. “Quality Assessment of Biometric Systems: A Comprehensive Perspective based on Accuracy and Performance Measurement”. In: *IEEE Transactions on Instrumentation and Measurement* 54.4 (Aug. 2005), pp. 1489–1496. ISSN: 1557-9662. DOI: [10.1109/TIM.2005.851087](https://doi.org/10.1109/TIM.2005.851087) (cit. on pp. [6](#), [25](#), [82](#)).

- [76] E. Gasperowicz. *OffscreenCanvas Speed up Your Canvas Operations with a Web Worker*. accessed 2020-06-30. 2018. URL: <https://developers.google.com/web/updates/2018/08/offscreen-canvas>.
- [77] *General Data Protection Regulation (GDPR) Official Legal Text*. accessed 2020-06-30. URL: <https://gdpr-info.eu> (cit. on pp. 64, 66).
- [78] Gennaro. *Microsoft Silverlight: what happened exactly? Still available?* accessed 2020-06-30. 2018. URL: <https://windowsreport.com/microsoft-silverlight>.
- [79] T. Goethem, W. Scheepers, D. Preuveneers, and W. Joosen. “Accelerometer-Based Device Fingerprinting for Multi-factor Mobile Authentication”. In: *International Symposium on Engineering Secure Software and Systems (ES-SoS)*. 2016, pp. 106–121. ISBN: 978-3-319-30805-0. DOI: [10.1007/978-3-319-30806-7_7](https://doi.org/10.1007/978-3-319-30806-7_7) (cit. on pp. 3, 33, 48, 81, 194).
- [80] M. Golla, T. Schnitzler, and M. Dürmuth. “Will Any Password Do? Exploring Rate-Limiting on the Web”. In: *USENIX Symposium on Usable Privacy and Security (SOUPS)*. Aug. 12, 2018 (cit. on pp. 172, 176).
- [81] A. Gómez-Boix, P. Laperdrix, and B. Baudry. “Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale”. In: *The Web Conference (TheWebConf)*. Apr. 2018. DOI: [10.1145/3178876.3186097](https://doi.org/10.1145/3178876.3186097) (cit. on pp. 3–5, 41, 58, 60, 66, 69, 81, 85, 88, 92, 115, 120–123, 149, 153, 158, 194, 195, 204).
- [82] G. G. Gulyas, D. F. Some, N. Bielova, and C. Castelluccia. “To Extend or not to Extend: On the Uniqueness of Browser Extensions and Web Logins”. In: *Workshop on Privacy in the Electronic Society (WPES)*. Jan. 15, 2018, pp. 14–27. ISBN: 978-1-4503-5989-4. DOI: [10.1145/3267323.3268959](https://doi.org/10.1145/3267323.3268959).
- [83] G. G. Gulyás, G. Acs, and C. Castelluccia. “Near-Optimal Fingerprinting with Constraints”. In: *Proceedings on Privacy Enhancing Technologies* 2016.4 (Oct. 1, 2016), pp. 470–487. DOI: [10.1515/popets-2016-0051](https://doi.org/10.1515/popets-2016-0051) (cit. on pp. 4, 29, 34, 37, 47, 123, 157).
- [84] W. Han, Z. Li, M. Ni, G. Gu, and W. Xu. “Shadow Attacks Based on Password Reuses: A Quantitative Empirical Analysis”. In: *IEEE Transactions on Dependable and Secure Computing (TDSC)* 15.2 (Mar. 2018), pp. 309–320. ISSN: 2160-9209. DOI: [10.1109/TDSC.2016.2568187](https://doi.org/10.1109/TDSC.2016.2568187) (cit. on pp. 79, 194).
- [85] *Have I Been Pwned: Pwned Passwords*. accessed 2020-06-30. URL: <https://haveibeenpwned.com/Passwords>.

- [86] E. Hayashi, R. Dhamija, N. Christin, and A. Perrig. “Use your Illusion: Secure Authentication Usable Anywhere”. In: *Symposium on Usable Privacy and Security (SOUPS)*. 2008, pp. 35–45. ISBN: 978-1-60558-276-4. DOI: [10.1145/1408664.1408670](https://doi.org/10.1145/1408664.1408670) (cit. on pp. 29, 176).
- [87] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. “Support Vector Machines”. In: *IEEE Intelligent Systems and their Applications* 13.4 (July 1998), pp. 18–28. ISSN: 2374-9423. DOI: [10.1109/5254.708428](https://doi.org/10.1109/5254.708428) (cit. on p. 228).
- [88] D. S. Hochbaum, ed. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., 1997. ISBN: 978-0-534-94968-6.
- [89] J. Hofmann. *Improved Security and Privacy Indicators in Firefox 70*. accessed 2020-06-30. 2019. URL: <https://blog.mozilla.org/security/2019/10/15/improved-security-and-privacy-indicators-in-firefox-70> (cit. on p. 20).
- [90] *Home - MobileConnect*. accessed 2020-06-30. URL: <https://mobileconnect.io> (cit. on p. 74).
- [91] P. Hraka. “Browser Fingerprinting”. Master Thesis. Bratislava: Comenius University, 2018. URL: <https://virpo.sk/browser-fingerprinting-hraska-diploma-thesis.pdf> (cit. on pp. 5, 46, 158, 178, 195).
- [92] *HTML Standard*. accessed 2020-06-30. URL: <https://html.spec.whatwg.org/multipage/workers.html>.
- [93] *HTTPS encryption on the web*. accessed 2020-06-30. URL: <https://transparencyreport.google.com/https/overview>.
- [94] T. Hunt. *Troy Hunt: 86% of Passwords are Terrible (and Other Statistics)*. accessed 2020-06-30. 2018. URL: <https://www.troyhunt.com/86-of-passwords-are-terrible-and-other-statistics> (cit. on p. 2).
- [95] T. Hunt. *Troy Hunt: Extended Validation Certificates are (Really, Really) Dead*. accessed 2020-06-30. 2019. URL: <https://www.troyhunt.com/extended-validation-certificates-are-really-really-dead> (cit. on p. 20).
- [96] T. Hupperich, D. Maiorca, M. Kühner, T. Holz, and G. Giacinto. “On the Robustness of Mobile Device Fingerprinting: Can Mobile Users Escape Modern Web-Tracking Mechanisms?” In: *Annual Computer Security Applications Conference (ACSAC)*. 2015, pp. 191–200. ISBN: 978-1-4503-3682-6. DOI: [10.1145/2818000.2818032](https://doi.org/10.1145/2818000.2818032) (cit. on p. 2).

- [97] T. Hupperich, D. Tatang, N. Wilkop, and T. Holz. “An Empirical Study on Online Price Differentiation”. In: *ACM Conference on Data and Application Security and Privacy (CODASPY)*. 2018, pp. 76–83. ISBN: 978-1-4503-5632-9. DOI: [10.1145/3176258.3176338](https://doi.org/10.1145/3176258.3176338) (cit. on pp. 40, 123).
- [98] U. o. S. C. Information Sciences Institute. *RFC 793 - Transmission Control Protocol*. accessed 2019-10-01. 1981. URL: <https://tools.ietf.org/html/rfc793> (cit. on p. 32).
- [99] A. International. *When Best Practice Isn't Good Enough: Large Campaigns of Phishing Attacks in Middle East and North Africa Target Privacy-Conscious Users*. accessed 2020-06-30. 2018. URL: <https://www.amnesty.org/en/latest/research/2018/12/when-best-practice-is-not-good-enough> (cit. on p. 28).
- [100] Z. Jia, X. Cui, Q. Liu, X. Wang, and C. Liu. “Micro-Honeypot: Using Browser Fingerprinting to Track Attackers”. In: *IEEE Third International Conference on Data Science in Cyberspace (DSC)*. June 2018, pp. 197–204. DOI: [10.1109/DSC.2018.00036](https://doi.org/10.1109/DSC.2018.00036) (cit. on p. 39).
- [101] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. 2nd ed. Pearson, 2009. ISBN: 978-0-13-187321-6.
- [102] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. 2nd ed. Pearson, 2009, pp. 23–27. ISBN: 978-0-13-187321-6 (cit. on p. 228).
- [103] D. Jurafsky and J. H. Martin. *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2009, pp. 325–326. ISBN: 0131873210 (cit. on pp. 7, 159, 168).
- [104] Kaing, Hong, Risher, Michael, and Schulte, Brian. “User Tracking: Persistent Cookies and Browser Fingerprinting”. Master Thesis. George Mason University, Apr. 2013. URL: https://cs.gmu.edu/~yhwang1/INFS612/2013_Spring/Projects/Final/2013_Spring_PGN_5_final_report.pdf.
- [105] N.-h. KANG, M.-z. CHEN, Y.-y. FENG, W.-n. LIN, C.-b. LIU, and G.-y. LI. “Zero-Permission Mobile Device Identification Based on the Similarity of Browser Fingerprints”. In: *International Conference on Computer Science and Technology (CST)*. July 31, 2017. ISBN: 978-1-60595-461-5. DOI: [10.12783/dtcse/cst2017/12531](https://doi.org/10.12783/dtcse/cst2017/12531) (cit. on p. 228).
- [106] S. Karami, P. Ilia, K. Solomos, and J. Polakis. “Carnus: Exploring the Privacy Threats of Browser Extension Fingerprinting”. In: *Network and Distributed System Security Symposium (NDSS)*. 2020. DOI: [10.14722/ndss.2020.24383](https://doi.org/10.14722/ndss.2020.24383) (cit. on pp. 31, 41, 157, 208).

- [107] T. L. Keiningham, B. Cooil, T. W. Andreassen, and L. Aksoy. “A Longitudinal Examination of Net Promoter and Firm Revenue Growth”. In: *Journal of Marketing* (Oct. 2, 2018). DOI: [10.1509/jmkg.71.3.039](https://doi.org/10.1509/jmkg.71.3.039) (cit. on pp. [74](#), [112](#)).
- [108] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Berlin Heidelberg: Springer-Verlag, 2004. ISBN: 978-3-540-40286-2. URL: <https://www.springer.com/gp/book/9783540402862> (visited on 10/18/2019) (cit. on p. [165](#)).
- [109] A. Kerckhoffs. “La Cryptographie Militaire”. In: *Journal des Sciences Militaires* (Jan. 1883), pp. 5–38.
- [110] Z. Kfir and A. Wool. “Picking Virtual Pockets using Relay Attacks on Contactless Smartcard”. In: *International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM)*. Sept. 2005, pp. 47–58. DOI: [10.1109/SECURECOMM.2005.32](https://doi.org/10.1109/SECURECOMM.2005.32) (cit. on p. [28](#)).
- [111] A. F. Khademi, M. Zulkernine, and K. Weldemariam. “An Empirical Evaluation of Web-Based Fingerprinting”. In: *IEEE Software* 32.4 (July 2015), pp. 46–52. ISSN: 0740-7459. DOI: [10.1109/MS.2015.77](https://doi.org/10.1109/MS.2015.77) (cit. on pp. [4](#), [5](#), [46](#), [158](#), [178](#), [194](#), [195](#)).
- [112] H. Khan, A. Atwater, and U. Hengartner. “A Comparative Evaluation of Implicit Authentication Schemes”. In: *International Workshop on Recent Advances in Intrusion Detection (RAID)*. Ed. by A. Stavrou, H. Bos, and G. Portokalidis. 2014, pp. 255–275. ISBN: 978-3-319-11379-1. DOI: [10.1007/978-3-319-11379-1_13](https://doi.org/10.1007/978-3-319-11379-1_13) (cit. on p. [22](#)).
- [113] S. Khuller, A. Moss, and J. Naor. “The Budgeted Maximum Coverage Problem”. In: *Information Processing Letters* 70.1 (Apr. 16, 1999), pp. 39–45. ISSN: 0020-0190. DOI: [10.1016/S0020-0190\(99\)00031-9](https://doi.org/10.1016/S0020-0190(99)00031-9) (cit. on p. [47](#)).
- [114] A. Kobusinska, J. Brzezinski, and K. Pawulczuk. “Device Fingerprinting: Analysis of Chosen Fingerprinting Methods”. In: *International Conference on Internet of Things, Big Data and Security (IoTBDs)*. Jan. 2017, pp. 167–177. ISBN: 978-989-758-245-5. DOI: [10.5220/0006375701670177](https://doi.org/10.5220/0006375701670177) (cit. on p. [123](#)).
- [115] T. Kohno, A. Broido, and K. C. Claffy. “Remote Physical Device Fingerprinting”. In: *IEEE Transactions on Dependable and Secure Computing* 2.2 (Apr. 2005), pp. 93–108. ISSN: 1941-0018. DOI: [10.1109/TDSC.2005.26](https://doi.org/10.1109/TDSC.2005.26) (cit. on p. [32](#)).

- [116] D. Kora and D. Simi. “Fishbone Model and Universal Authentication Framework for Evaluation of Multifactor Authentication in Mobile Environment”. In: *Computers & Security* 85 (Aug. 1, 2019), pp. 313–332. ISSN: 0167-4048. DOI: [10.1016/j.cose.2019.05.011](https://doi.org/10.1016/j.cose.2019.05.011) (cit. on p. 21).
- [117] B. Krebs. *Half of all Phishing Sites Now Have the Padlock - Krebs on Security*. accessed 2020-06-30. 2018. URL: <https://krebsonsecurity.com/2018/11/half-of-all-phishing-sites-now-have-the-padlock> (cit. on p. 20).
- [118] P. Krishnasamy, S. Belongie, and D. Kriegman. “Wet Fingerprint Recognition: Challenges and Opportunities”. In: *International Joint Conference on Biometrics (IJCB)*. Oct. 2011, pp. 1–7. DOI: [10.1109/IJCB.2011.6117594](https://doi.org/10.1109/IJCB.2011.6117594) (cit. on pp. 13, 25).
- [119] A. Kumar and A. Passi. “Comparison and Combination of Iris Matchers for Reliable Personal Authentication”. In: *Pattern Recognition* 43.3 (Mar. 1, 2010), pp. 1016–1026. ISSN: 0031-3203. DOI: [10.1016/j.patcog.2009.08.016](https://doi.org/10.1016/j.patcog.2009.08.016) (cit. on p. 21).
- [120] C. Kuo and N. Yoshiura. “A Browser Application for Keyword Recommendation Based on User Web Search”. In: *Computational Science and Its Applications (ICCSA)*. Ed. by O. Gervasi, B. Murgante, S. Misra, E. Stankova, C. M. Torre, A. M. A. Rocha, D. Tanar, B. O. Apduhan, E. Tarantino, and Y. Ryu. 2018, pp. 147–162. ISBN: 978-3-319-95168-3. DOI: [10.1007/978-3-319-95168-3_10](https://doi.org/10.1007/978-3-319-95168-3_10).
- [121] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. Freiling. “Fingerprinting Mobile Devices Using Personalized Configurations”. In: *Proceedings on Privacy Enhancing Technologies* 2016.1 (2016). ISSN: 2299-0984. DOI: <https://doi.org/10.1515/popets-2015-0027> (cit. on pp. 31, 88).
- [122] F. Lambert. *Tesla gets stolen with keyfob hack on camera in seconds here’s how to prevent it*. accessed 2020-06-30. 2019. URL: <https://electrek.co/2019/08/22/tesla-stolen-keyfob-hack-camera-how-to-prevent-it> (cit. on p. 28).
- [123] P. Laperdrix. “Browser Fingerprinting: Exploring Device Diversity to Augment Authentication and Build Client-Side Countermeasures”. thesis. Université Bretagne Loire, Oct. 3, 2017. URL: <https://tel.archives-ouvertes.fr/tel-01729126>.
- [124] P. Laperdrix, G. Avoine, B. Baudry, and N. Nikiforakis. “Morellian Analysis for Browsers: Making Web Authentication Stronger With Canvas Fingerprinting”. In: *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*. June 2019, pp. 43–66. ISBN: 978-3-030-22038-9. DOI:

- [10.1007/978-3-030-22038-9_3](#) (cit. on pp. 3–5, 27, 37, 45, 49, 55, 65, 81, 83, 124, 125, 134, 135, 149, 157, 158, 194, 199, 211, 218, 228).
- [125] P. Laperdrix, B. Baudry, and V. Mishra. “FPRandom: Randomizing Core Browser Objects to Break Advanced Device Fingerprinting Techniques”. In: *International Symposium on Engineering Secure Software and Systems (ES-SoS)*. Ed. by E. Bodden, M. Payer, and E. Athanasopoulos. 2017, pp. 97–114. ISBN: 978-3-319-62105-0. DOI: [10.1007/978-3-319-62105-0_7](#) (cit. on pp. 2, 41).
- [126] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine. “Browser Fingerprinting: A Survey”. In: *ACM Transactions on the Web* 14.2 (Apr. 9, 2020), 8:1–8:33. ISSN: 1559-1131. DOI: [10.1145/3386040](#).
- [127] P. Laperdrix, W. Rudametkin, and B. Baudry. “Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints”. In: *IEEE Symposium on Security and Privacy (S&P)*. May 2016, pp. 878–894. DOI: [10.1109/SP.2016.57](#) (cit. on pp. 2–5, 33, 35, 36, 38, 41, 55, 56, 58, 60, 61, 66–69, 81, 85, 88, 92, 115, 120–125, 134, 135, 149, 153, 158, 189, 194, 195, 200, 203, 204, 206, 211, 218).
- [128] P. Laperdrix, W. Rudametkin, and B. Baudry. “Mitigating Browser Fingerprint Tracking: Multi-level Reconfiguration and Diversification”. In: *IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. May 2015, pp. 98–108. DOI: [10.1109/SEAMS.2015.18](#) (cit. on pp. 2, 41, 163).
- [129] H. Le, F. Fallace, and P. Barlet-Ros. “Towards Accurate Detection of Obfuscated Web Tracking”. In: *IEEE International Workshop on Measurement and Networking (M&N)*. Sept. 2017, pp. 1–6. DOI: [10.1109/IWMN.2017.8078365](#) (cit. on p. 149).
- [130] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. “Incognito: Efficient Full-Domain K-Anonymity.” In: *ACM SIGMOD International Conference on Management of Data (SIGMOD)*. Vol. 2005. 2005, pp. 49–60. ISBN: 978-1-59593-060-6. DOI: [10.1145/1066157.1066164](#) (cit. on p. 201).
- [131] N. Li, T. Li, and S. Venkatasubramanian. “t-Closeness: Privacy Beyond k-Anonymity and l-Diversity”. In: *IEEE International Conference on Data Engineering (ICDE)*. May 15, 2007, pp. 106–115. ISBN: 978-1-4244-0803-0. DOI: [10.1109/ICDE.2007.367856](#) (cit. on p. 201).
- [132] Z. Li, A. S. Rathore, C. Song, S. Wei, Y. Wang, and W. Xu. “PrinTracker: Fingerprinting 3D Printers using Commodity Scanners”. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Jan. 15,

- 2018, pp. 1306–1323. ISBN: 978-1-4503-5693-0. DOI: [10.1145/3243734.3243735](https://doi.org/10.1145/3243734.3243735) (cit. on p. 31).
- [133] X. Liu, Q. Liu, X. Wang, and Z. Jia. “Fingerprinting Web Browser for Tracing Anonymous Web Attackers”. In: *IEEE International Conference on Data Science in Cyberspace (DSC)*. 2016, pp. 222–229. DOI: [10.1109/DSC.2016.78](https://doi.org/10.1109/DSC.2016.78) (cit. on p. 39).
- [134] LogMeIn. *The 3rd Annual Global Password Security Report*. White Paper. 2019. URL: <https://lp-cdn.lastpass.com/lporcamedia/document-library/lastpass/pdf/en/LMI0828a-IAM-LastPass-State-of-the-Password-Report.pdf> (cit. on p. 79).
- [135] LogMeIn. *The Password Exposé*. White Paper. 2017. URL: <https://lp-cdn.lastpass.com/lporcamedia/document-library/lastpass/pdf/en/LastPass-Enterprise-The-Password-Expose-Ebook-v2.pdf> (cit. on p. 79).
- [136] B. Lu, X. Zhang, Z. Ling, Y. Zhang, and Z. Lin. “A Measurement Study of Authentication Rate-Limiting Mechanisms of Modern Websites”. In: *Annual Computer Security Applications Conference (ACSAC)*. Dec. 3, 2018, pp. 89–100. ISBN: 978-1-4503-6569-7. DOI: [10.1145/3274694.3274714](https://doi.org/10.1145/3274694.3274714) (cit. on p. 17).
- [137] S. Luangmaneerote, E. Zaluska, and L. Carr. “Survey of Existing Fingerprint Countermeasures”. In: *International Conference on Information Society (i-Society)*. Oct. 2016, pp. 137–141. DOI: [10.1109/i-Society.2016.7854198](https://doi.org/10.1109/i-Society.2016.7854198).
- [138] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. “L-diversity: Privacy Beyond K-anonymity”. In: *ACM Trans. Knowl. Discov. Data* 1.1 (Mar. 2007). ISSN: 1556-4681. DOI: [10.1145/1217299.1217302](https://doi.org/10.1145/1217299.1217302) (cit. on p. 201).
- [139] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition*. 2nd ed. 2003. ISBN: 978-1-84882-253-5. DOI: [10.1007/978-1-84882-254-2](https://doi.org/10.1007/978-1-84882-254-2) (cit. on pp. 6, 21, 24, 31, 82, 83).
- [140] A. J. Mansfield and J. L. Wayman. *Best Practices in Testing and Reporting Performance of Biometric Devices*. Centre for Mathematics and Scientific Computing, National Physical Laboratory, 2002 (cit. on p. 25).
- [141] F. Marcantoni, M. Diamantaris, S. Ioannidis, and J. Polakis. “A Large-scale Study on the Risks of the HTML5 WebAPI for Mobile Sensor-based Attacks”. In: *The Web Conference (TheWebConf)*. 2019, pp. 3063–3071. ISBN: 978-1-4503-6674-8. DOI: [10.1145/3308558.3313539](https://doi.org/10.1145/3308558.3313539) (cit. on p. 33).

- [142] P. Markert, M. Golla, E. Stobert, and M. Dürmuth. “Work in Progress: A Comparative Long-Term Study of Fallback Authentication”. In: *Network and Distributed System Security Symposium (NDSS)*. 2020. URL: <https://www.ndss-symposium.org/ndss-paper/auto-draft-30/> (cit. on pp. 45, 108).
- [143] P. Marks. “Dark Web’s Doppelgängers Aim to Dupe Antifraud Systems”. In: *Communications of the ACM* 63.2 (Jan. 22, 2020), pp. 16–18. ISSN: 0001-0782. DOI: [10.1145/3374878](https://doi.org/10.1145/3374878). URL: <https://doi.org/10.1145/3374878> (cit. on p. 162).
- [144] T. Matsumoto, H. Matsumoto, K. Yamada, and S. Hoshino. “Impact of Artificial “gummy” Fingers on Fingerprint Systems”. In: *Optical Security and Counterfeit Deterrence Techniques*. Vol. 4677. Apr. 19, 2002, pp. 275–289. DOI: [10.1117/12.462719](https://doi.org/10.1117/12.462719) (cit. on p. 28).
- [145] J. R. Mayer. ““Any person a pamphleteer” Internet Anonymity in the Age of Web 2.0”. Master Thesis. Faculty of the Woodrow Wilson School of Public and International Affairs, Apr. 7, 2009. URL: <https://arks.princeton.edu/ark:/88435/dsp01nc580n467> (cit. on pp. 2, 31, 123).
- [146] J. P. F. C. R. Mendes. “noPhish Anti-phishing System using Browser Fingerprinting”. Master Thesis. Coimbra, Portugal: Universidade de Coimbra, 2011. URL: <https://estagios.dei.uc.pt/cursos/mei/relatorios-de-estagio/?id=279> (cit. on pp. 5, 46, 158, 178, 195).
- [147] W. Meng, D. S. Wong, S. Furnell, and J. Zhou. “Surveying the Development of Biometric User Authentication on Mobile Phones”. In: *IEEE Communications Surveys Tutorials (COMST)* 17.3 (2015), pp. 1268–1293. ISSN: 1553-877X. DOI: [10.1109/COMST.2014.2386915](https://doi.org/10.1109/COMST.2014.2386915) (cit. on p. 25).
- [148] M. Mihajlov, B. J. Blazic, and S. Josimovski. “Quantifying Usability and Security in Authentication”. In: *IEEE Annual Computer Software and Applications Conference (COMPSAC)*. July 2011, pp. 626–629. DOI: [10.1109/COMPSAC.2011.87](https://doi.org/10.1109/COMPSAC.2011.87).
- [149] G. Milka. “Anatomy of Account Takeover”. In: *Enigma*. 2018. URL: <https://www.usenix.org/node/208154> (cit. on pp. 2, 21, 23).
- [150] *Mobile Connect Authenticate - Simple and secure user authentication on aglobal scale*. accessed 2020-06-30. URL: <https://mobileconnect.io/wp-content/uploads/2019/02/MC-Authenticate-functional-data-sheet-FINAL-1.pdf> (cit. on p. 74).
- [151] R. Morris and K. Thompson. “Password Security: A Case History”. In: *Commun. ACM* 22.11 (Nov. 1979), pp. 594–597. ISSN: 0001-0782. DOI: [10.1145/359168.359172](https://doi.org/10.1145/359168.359172) (cit. on p. 193).

- [152] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham. “Fingerprinting Information in JavaScript Implementations”. In: *Proceedings of W2SP*. Ed. by H. Wang. Vol. 2. May 2011 (cit. on pp. 4, 158, 195).
- [153] K. Mowery and H. Shacham. “Pixel perfect: Fingerprinting canvas in HTML5”. In: *Proceedings of Web 2.0 Security & Privacy (W2SP)* (2012), pp. 1–12. URL: <https://www.ieee-security.org/TC/W2SP/2012/papers/w2sp12-final4.pdf> (cit. on pp. 2, 4, 5, 29, 33, 35, 41, 48, 55, 56, 158, 174, 199).
- [154] Mozilla. *Service Worker API - Web APIs | MDN*. accessed 2020-06-30. URL: https://developer.mozilla.org/en-US/docs/Web/API/Service%5C_Worker%5C_API.
- [155] M. Mulazzani, P. Reschl, M. Huber, M. Leithner, S. Schrittwieser, E. Weippl, and F. Wien. “Fast and Reliable Browser Identification with Javascript Engine Fingerprinting”. In: *Web 2.0 Workshop on Security and Privacy (W2SP)*. Vol. 5. 2013 (cit. on pp. 4, 158, 195).
- [156] *Multilogin - Replace Multiple Computers With Virtual Browser Profiles - Multilogin*. accessed 2020-06-30. URL: <https://multilogin.com> (cit. on p. 163).
- [157] G. Nakibly, G. Shelef, and S. Yudilevich. “Hardware Fingerprinting Using HTML5”. In: *arXiv:1503.01408 [cs.CR]* (2015). URL: <https://arxiv.org/abs/1503.01408> (cit. on pp. 4, 158, 195).
- [158] A. Narayanan and V. Shmatikov. “Fast Dictionary Attacks on Passwords using Time-space Tradeoff”. In: *ACM Conference on Computer and Communications Security (CCS)*. Nov. 7, 2005, pp. 364–372. ISBN: 978-1-59593-226-6. DOI: 10.1145/1102120.1102168. URL: <https://doi.org/10.1145/1102120.1102168>.
- [159] N. Neverova, C. Wolf, G. Lacey, L. Fridman, D. Chandra, B. Barbello, and G. Taylor. “Learning Human Identity From Motion Patterns”. In: *IEEE Access* 4 (2016), pp. 1810–1820. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2016.2557846 (cit. on p. 21).
- [160] R. S. Nickerson. “Confirmation Bias: A Ubiquitous Phenomenon in Many Guises”. In: *Review of General Psychology* 2.2 (June 1, 1998), pp. 175–220. ISSN: 1089-2680. DOI: 10.1037/1089-2680.2.2.175 (cit. on p. 114).
- [161] K. Niinuma, U. Park, and A. K. Jain. “Soft Biometric Traits for Continuous User Authentication”. In: *IEEE Transactions on Information Forensics and Security (TIFS)* 5.4 (Dec. 2010), pp. 771–780. ISSN: 1556-6021. DOI: 10.1109/TIFS.2010.2075927 (cit. on p. 21).

- [162] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. “Cookieless Monster: Exploring the Ecosystem of Web-Based Device Fingerprinting”. In: *IEEE Symposium on Security and Privacy (S&P)*. May 2013, pp. 541–555. DOI: [10.1109/SP.2013.43](https://doi.org/10.1109/SP.2013.43) (cit. on p. 202).
- [163] N. Nikiforakis, W. Joosen, and B. Livshits. “PriVaricator: Deceiving Fingerprinters with Little White Lies”. In: *International Conference on World Wide Web (WWW)*. May 18, 2015, pp. 820–830. ISBN: 978-1-4503-3469-3. DOI: [10.1145/2736277.2741090](https://doi.org/10.1145/2736277.2741090) (cit. on p. 41).
- [164] L. O’Gorman. “Comparing Passwords, Tokens, and Biometrics for User Authentication”. In: *Proceedings of the IEEE* 91.12 (Dec. 2003), pp. 2021–2040. ISSN: 1558-2256. DOI: [10.1109/JPROC.2003.819611](https://doi.org/10.1109/JPROC.2003.819611) (cit. on pp. 25, 27).
- [165] P. Oechslin. “Making a Faster Cryptanalytic Time-Memory Trade-Off”. In: *Annual International Cryptology Conference (CRYPTO)*. Ed. by D. Boneh. 2003, pp. 617–630. ISBN: 978-3-540-45146-4. DOI: [10.1007/978-3-540-45146-4_36](https://doi.org/10.1007/978-3-540-45146-4_36) (cit. on p. 17).
- [166] L. Olejnik, G. Acar, C. Castelluccia, and C. Diaz. “The Leaking Battery”. In: *International Workshop on Data Privacy Management, and Security Assurance (DPM)*. 2016, pp. 254–263. ISBN: 978-3-319-29882-5. DOI: [10.1007/978-3-319-29883-2_18](https://doi.org/10.1007/978-3-319-29883-2_18) (cit. on pp. 33, 37).
- [167] L. Olejnik, C. Castelluccia, and A. Janc. “Why Johnny Can’t Browse in Peace: On the Uniqueness of Web Browsing History Patterns”. In: *Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs)*. July 2012. URL: <https://petsymposium.org/2012/papers/hotpets12-4-johnny.pdf>.
- [168] L. Olejnik, S. Englehardt, and A. Narayanan. “Battery Status Not Included: Assessing Privacy in Web Standards”. In: *International Workshop on Privacy Engineering (IWPE)*. 2017. URL: http://ceur-ws.org/Vol-1873/IWPE17_paper_18.pdf.
- [169] J. Oliver. “Fingerprinting the Mobile Web”. Master Thesis. London, UK: Imperial College London, June 18, 2018. URL: https://www.doc.ic.ac.uk/~livshits/papers/theses/john_oliver.pdf.
- [170] S. Pankanti, S. Prabhakar, and A. Jain. “On the Individuality of Fingerprints”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 24.8 (Aug. 2002), pp. 1010–1025. ISSN: 1939-3539. DOI: [10.1109/TPAMI.2002.1023799](https://doi.org/10.1109/TPAMI.2002.1023799) (cit. on p. 82).

- [171] P. Papadopoulos, P. Ilija, M. Polychronakis, E. P. Markatos, S. Ioannidis, and G. Vasiliadis. “Master of Web Puppets: Abusing Web Browsers for Persistent and Stealthy Computation”. In: *Network and Distributed System Security Symposium (NDSS)*. Feb. 2019. DOI: [10.14722/ndss.2019.23070](https://doi.org/10.14722/ndss.2019.23070) (cit. on p. 162).
- [172] S. Pearman, J. Thomas, P. E. Naeini, H. Habib, L. Bauer, N. Christin, L. F. Cranor, S. Egelman, and A. Forget. “Let’s Go in for a Closer Look: Observing Passwords in Their Natural Habitat”. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Oct. 30, 2017, pp. 295–310. ISBN: 978-1-4503-4946-8. DOI: [10.1145/3133956.3133973](https://doi.org/10.1145/3133956.3133973) (cit. on pp. 44, 79).
- [173] T. Petsas, G. Tsirantonakis, E. Athanasopoulos, and S. Ioannidis. “Two-factor Authentication: Is the World Ready? Quantifying 2FA Adoption”. In: *European Workshop on System Security (EuroSec)*. Apr. 21, 2015, pp. 1–7. ISBN: 978-1-4503-3479-2. DOI: [10.1145/2751323.2751327](https://doi.org/10.1145/2751323.2751327) (cit. on pp. 2, 21, 23).
- [174] O. I. Platform. *OpenAM - Open Access Manager & Open Identity Platform*. accessed 2019-10-01. URL: <https://www.openidentityplatform.org/openam> (cit. on p. 43).
- [175] J. Postel. *RFC 792 - Internet Control Message Protocol*. accessed 2019-10-01. 1981. URL: <https://tools.ietf.org/html/rfc792> (cit. on p. 32).
- [176] S. Prabhakar, S. Pankanti, and A. Jain. “Biometric Recognition: Security and Privacy Concerns”. In: *IEEE Security Privacy* 1.2 (Mar. 2003), pp. 33–42. ISSN: 1558-4046. DOI: [10.1109/MSECP.2003.1193209](https://doi.org/10.1109/MSECP.2003.1193209) (cit. on p. 25).
- [177] D. Preuveneers and W. Joosen. “SmartAuth: Dynamic Context Fingerprinting for Continuous User Authentication”. In: *Annual ACM Symposium on Applied Computing (SAC)*. 2015, pp. 2185–2191. ISBN: 978-1-4503-3196-8. DOI: [10.1145/2695664.2695908](https://doi.org/10.1145/2695664.2695908) (cit. on pp. 3, 42–44, 81, 194).
- [178] G. Pugliese, C. Riess, F. Gassmann, and Z. Benenson. “Long-Term Observation on Browser Fingerprinting: Users Trackability and Perspective”. In: *Proceedings on Privacy Enhancing Technologies* 2020.2 (Apr. 1, 2020), pp. 558–577. DOI: [10.2478/popets-2020-0041](https://doi.org/10.2478/popets-2020-0041) (cit. on pp. 3, 4, 37, 42, 46, 58, 61, 66, 81, 194).
- [179] J. S. Queiroz and E. L. Feitosa. “A Web Browser Fingerprinting Method Based on the Web Audio API”. In: *The Computer Journal* (Jan. 22, 2019). DOI: [10.1093/comjnl/bxy146](https://doi.org/10.1093/comjnl/bxy146) (cit. on pp. 2, 4, 5, 33, 36, 41, 48, 55, 56, 158, 174, 195, 199, 200, 216).

- [180] N. Quermann, M. Harbach, and M. Dürmuth. “The State of User Authentication in the Wild”. In: *USENIX Symposium on Usable Privacy and Security (SOUPS)*. 2018. URL: <https://wayworkshop.org/2018/papers/way2018-quermann.pdf> (cit. on pp. 15, 16, 23, 108).
- [181] A. Rabkin. “Personal Knowledge Questions for Fallback Authentication: Security Questions in the Era of Facebook”. In: *USENIX Symposium on Usable Privacy and Security (SOUPS)*. July 23, 2008, pp. 13–23. ISBN: 978-1-60558-276-4. DOI: [10.1145/1408664.1408667](https://doi.org/10.1145/1408664.1408667) (cit. on pp. 21, 23).
- [182] N. K. Ratha, J. H. Connell, and R. M. Bolle. “Enhancing Security and Privacy in Biometrics-based Authentication Systems”. In: *IBM Systems Journal* 40.3 (2001), pp. 614–634. ISSN: 0018-8670. DOI: [10.1147/sj.403.0614](https://doi.org/10.1147/sj.403.0614) (cit. on p. 201).
- [183] V. Rizzo, S. Traverso, and M. Mellia. “Unveiling Web Fingerprinting in the Wild Via Code Mining and Machine Learning”. In: *Privacy Enhancing Technologies (PETS)* 2021.1 (Jan. 1, 2021), pp. 43–63. DOI: <https://doi.org/10.2478/popets-2021-0004>. URL: <https://content.sciendo.com/view/journals/popets/2021/1/article-p43.xml>.
- [184] F. Rochet, K. Efthymiadis, F. Koeune, and O. Pereira. “SWAT: Seamless Web Authentication Technology”. In: *The Web Conference (TheWebConf)*. May 2019, pp. 1579–1589. ISBN: 978-1-4503-6674-8. DOI: [10.1145/3308558.3313637](https://doi.org/10.1145/3308558.3313637) (cit. on pp. 3, 5, 49, 55, 81, 158, 194, 199, 228).
- [185] M. Roland, J. Langer, and J. Scharinger. “Applying Relay Attacks to Google Wallet”. In: *International Workshop on Near Field Communication (NFC)*. 2013, pp. 1–6. DOI: [10.1109/NFC.2013.6482441](https://doi.org/10.1109/NFC.2013.6482441) (cit. on p. 28).
- [186] J. F. R. Roy T. Fielding. *RFC 7231 - Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. accessed 2020-06-30. 2014. URL: <https://tools.ietf.org/html/rfc7231#section-5.5.3> (cit. on pp. 29, 88, 162, 173).
- [187] J. F. R. Roy T. Fielding. *RFC 7231 - Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. accessed 2019-10-01. 2014. URL: <https://tools.ietf.org/html/rfc7231> (cit. on pp. 30, 32).
- [188] S. Ruoti, B. Roberts, and K. Seamons. “Authentication Melee: A Usability Analysis of Seven Web Authentication Systems”. In: *International Conference on World Wide Web (WWW)*. 2015, pp. 916–926. ISBN: 978-1-4503-3469-3. DOI: [10.1145/2736277.2741683](https://doi.org/10.1145/2736277.2741683) (cit. on p. 24).

- [189] N. Sae-Bae, K. Ahmed, K. Isbister, and N. Memon. “Biometric-rich Gestures: A Novel Approach to Authentication on Multi-touch Devices”. In: *Conference on Human Factors in Computing Systems (SIGCHI)*. May 5, 2012, pp. 977–986. ISBN: 978-1-4503-1015-4. DOI: [10.1145/2207676.2208543](https://doi.org/10.1145/2207676.2208543) (cit. on p. 21).
- [190] B. Safaei, A. M. Monazzah, M. Bafroei, and A. Ejlali. “Reliability Side-Effects in Internet of Things Application Layer Protocols”. In: Dec. 2017. DOI: [10.1109/ICSRS.2017.8272822](https://doi.org/10.1109/ICSRS.2017.8272822).
- [191] T. Saito, K. Yasuda, T. Ishikawa, R. Hosoi, K. Takahashi, Y. Chen, and M. Zalasiski. “Estimating CPU Features by Browser Fingerprinting”. In: *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. July 2016, pp. 587–592. DOI: [10.1109/IMIS.2016.108](https://doi.org/10.1109/IMIS.2016.108) (cit. on pp. 4, 33, 158, 195).
- [192] T. Saito, T. Noda, R. Hosoya, K. Tanabe, and Y. Saito. “On Estimating Platforms of Web User with JavaScript Math Object”. In: *Advances in Network-Based Information Systems (NBIS)*. Ed. by L. Barolli, N. Kryvinska, T. Enokido, and M. Takizawa. 2019, pp. 407–418. ISBN: 978-3-319-98530-5. DOI: [10.1007/978-3-319-98530-5_34](https://doi.org/10.1007/978-3-319-98530-5_34) (cit. on p. 33).
- [193] T. Saito, K. Yasuda, K. Tanabe, and K. Takahashi. “Web Browser Tampering: Inspecting CPU Features from Side-Channel Information”. In: *International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*. Nov. 8, 2017, pp. 392–403. ISBN: 978-3-319-69810-6 978-3-319-69811-3. DOI: [10.1007/978-3-319-69811-3_36](https://doi.org/10.1007/978-3-319-69811-3_36) (cit. on pp. 4, 33, 158, 195).
- [194] R. Schutt and C. O’Neil. *Doing data science: Straight talk from the frontline*. O’Reilly, 2014, pp. 181–182 (cit. on pp. 7, 159, 168).
- [195] M. Schwarz, F. Lackner, and D. Gruss. “JavaScript Template Attacks: Automatically Inferring Host Information for Targeted Exploits”. In: *Network and Distributed System Security Symposium (NDSS)*. 2019. DOI: [10.14722/ndss.2019.23155](https://doi.org/10.14722/ndss.2019.23155) (cit. on p. 123).
- [196] SecureAuth. *Device / Browser Fingerprinting - Heuristic-based Authentication*. accessed 2020-06-30. URL: <https://docs.secureauth.com/pages/viewpage.action?pageId=33063454> (cit. on p. 3).
- [197] A. Serwadda and V. V. Phoha. “Examining a Large Keystroke Biometrics Dataset for Statistical-Attack Openings”. In: *ACM Transactions on Information and System Security* 16.2 (Sept. 1, 2013), 8:1–8:30. ISSN: 1094-9224. DOI: [10.1145/2516960](https://doi.org/10.1145/2516960) (cit. on p. 28).

- [198] R. Shay, S. Komanduri, A. L. Durity, P. (Huh, M. L. Mazurek, S. M. Segreti, B. Ur, L. Bauer, N. Christin, and L. F. Cranor. “Designing Password Policies for Strength and Usability”. In: *ACM Transactions on Information and System Security* 18.4 (May 6, 2016), 13:1–13:34. ISSN: 1094-9224. DOI: [10.1145/2891411](https://doi.org/10.1145/2891411) (cit. on p. 18).
- [199] A. Sjösten, S. Van Acker, and A. Sabelfeld. “Discovering Browser Extensions via Web Accessible Resources”. In: *ACM Conference on Data and Application Security and Privacy (CODASPY)*. 2017, pp. 329–336. ISBN: 978-1-4503-4523-1. DOI: [10.1145/3029806.3029820](https://doi.org/10.1145/3029806.3029820) (cit. on pp. 2, 4, 29, 34, 37, 54, 133, 135–137, 157, 199, 208).
- [200] *Smarter Security with Device Fingerprints*. accessed 2020-06-30. 2015. URL: <https://identityrocks.blogspot.com/2015/08/smarter-security-with-device.html>.
- [201] M. Smith, C. Disselkoen, S. Narayan, F. Brown, and D. Stefan. “Browser History re:visited”. In: *USENIX Workshop on Offensive Technologies (WOOT)*. Aug. 2018. URL: <https://www.usenix.org/conference/woot18/presentation/smith> (cit. on p. 37).
- [202] J. Solano, L. Camacho, A. Correa, C. Deiro, J. Vargas, and M. Ochoa. “Risk-Based Static Authentication in Web Applications with Behavioral Biometrics and Session Context Analytics”. In: *Applied Cryptography and Network Security Workshops (ACNS)*. Ed. by J. Zhou, R. Deng, Z. Li, S. Majumdar, W. Meng, L. Wang, and K. Zhang. 2019, pp. 3–23. ISBN: 978-3-030-29729-9. DOI: [10.1007/978-3-030-29729-9_1](https://doi.org/10.1007/978-3-030-29729-9_1) (cit. on pp. 21, 22, 42, 44, 163).
- [203] J. Spooren, D. Preuveneers, and W. Joosen. “Leveraging Battery Usage from Mobile Devices for Active Authentication”. In: *Mobile Information Systems* (2017). DOI: [10.1155/2017/1367064](https://doi.org/10.1155/2017/1367064) (cit. on pp. 3, 42–44, 81, 194).
- [204] J. Spooren, D. Preuveneers, and W. Joosen. “Mobile Device Fingerprinting Considered Harmful for Risk-based Authentication”. In: *European Workshop on System Security (EuroSec)*. 2015, 6:1–6:6. ISBN: 978-1-4503-3479-2. DOI: [10.1145/2751323.2751329](https://doi.org/10.1145/2751323.2751329) (cit. on pp. 44, 66, 88, 92, 115, 153).
- [205] O. Starov and N. Nikiforakis. “XHOUND: Quantifying the Fingerprintability of Browser Extensions”. In: *IEEE Symposium on Security & Privacy (S&P)*. May 24, 2017, pp. 941–956. DOI: [10.1109/SP.2017.18](https://doi.org/10.1109/SP.2017.18) (cit. on pp. 2, 29, 34, 37, 41, 54, 133, 135–137, 199, 207).

- [206] L. Sweeney. “k-anonymity: A Model for Protecting Privacy”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.5 (Oct. 1, 2002), pp. 557–570. ISSN: 0218-4885. DOI: [10.1142/S0218488502001648](https://doi.org/10.1142/S0218488502001648) (cit. on p. 201).
- [207] K. Takasu, T. Saito, T. Yamada, and T. Ishikawa. “A Survey of Hardware Features in Modern Browsers: 2015 Edition”. In: *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. July 2015, pp. 520–524. DOI: [10.1109/IMIS.2015.72](https://doi.org/10.1109/IMIS.2015.72) (cit. on pp. 33, 128, 210, 216).
- [208] K. Tanabe, R. Hosoya, and T. Saito. “Combining Features in Browser Fingerprinting”. In: *Advances on Broadband and Wireless Computing, Communication and Applications (BWCCA)*. Ed. by L. Barolli, F.-Y. Leu, T. Enokido, and H.-C. Chen. 2018, pp. 671–681. ISBN: 978-3-030-02613-4. DOI: [10.1007/978-3-030-02613-4_60](https://doi.org/10.1007/978-3-030-02613-4_60) (cit. on pp. 5, 47, 158, 195).
- [209] A. C. Team. *Flash & The Future of Interactive Content*. accessed 2020-06-30. 2017. URL: <https://blog.adobe.com/en/publish/2017/07/25/adobe-flash-update.html> (cit. on pp. 33, 123, 139, 140).
- [210] K. Thomas, F. Li, A. Zand, J. Barrett, J. Ranieri, L. Invernizzi, Y. Markov, O. Comanescu, V. Eranti, A. Moscicki, D. Margolis, V. Paxson, and E. Bursztein. “Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials”. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Oct. 30, 2017, pp. 1421–1434. ISBN: 978-1-4503-4946-8. DOI: [10.1145/3133956.3134067](https://doi.org/10.1145/3133956.3134067) (cit. on pp. 2, 20, 79, 162, 194).
- [211] C. F. Torres, H. Jonker, and S. Mauw. “FP-Block: Usable Web Privacy by Controlling Browser Fingerprinting”. In: *European Symposium on Research in Computer Security (ESORICS)*. Sept. 21, 2015, pp. 3–19. ISBN: 978-3-319-24176-0 978-3-319-24177-7. DOI: [10.1007/978-3-319-24177-7_1](https://doi.org/10.1007/978-3-319-24177-7_1) (cit. on p. 2).
- [212] T. Unger, M. Mulazzani, D. Frühwirth, M. Huber, S. Schrittwieser, and E. Weippl. “SHPF: Enhancing HTTP(S) Session Security with Browser Fingerprinting”. In: *International Conference on Availability, Reliability and Security (ARES)*. Sept. 2013, pp. 255–261. DOI: [10.1109/ARES.2013.33](https://doi.org/10.1109/ARES.2013.33) (cit. on pp. 3, 42, 81, 194).
- [213] N. Vallina-Rodriguez, S. Sundaresan, C. Kreibich, and V. Paxson. “Header Enrichment or ISP Enrichment?: Emerging Privacy Threats in Mobile Networks”. In: *ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Network Function Virtualization (HotMiddlebox)*. 2015, pp. 25–30. ISBN: 978-1-4503-3540-9. DOI: [10.1145/2785989.2786002](https://doi.org/10.1145/2785989.2786002) (cit. on p. 204).

- [214] A. Vastel, P. Laperdrix, W. Rudametkin, and R. Rouvoy. “FP-STALKER: Tracking Browser Fingerprint Evolutions”. In: *IEEE Symposium on Security and Privacy (S&P)*. May 21, 2018, pp. 728–741. DOI: [10.1109/sp.2018.00008](https://doi.org/10.1109/sp.2018.00008) (cit. on pp. [3–5](#), [37](#), [41](#), [46](#), [56](#), [81](#), [86](#), [123](#), [158](#), [177](#), [194](#), [195](#), [228](#)).
- [215] A. Vastel, W. Rudametkin, R. Rouvoy, and X. Blanc. “FP-Crawlers: Studying the Resilience of Browser Fingerprinting to Block Crawlers”. In: *Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb)*. 2020 (cit. on pp. [39](#), [123](#)).
- [216] W3C. *Web Authentication: An API for accessing Public Key Credentials - Level 2*. accessed 2020-06-30. 2020. URL: <https://www.w3.org/TR/2020/WD-webauthn-2-20200730> (cit. on p. [193](#)).
- [217] K. S. Walia, S. Shenoy, and Y. Cheng. “An Empirical Analysis on the Usability and Security of Passwords”. In: *International Conference on Information Reuse and Integration for Data Science (IRI)*. Aug. 11, 2020. DOI: [10.1109/IRI49571.2020.00009](https://doi.org/10.1109/IRI49571.2020.00009) (cit. on p. [44](#)).
- [218] C. Wang, S. T. Jan, H. Hu, D. Bossart, and G. Wang. “The Next Domino to Fall: Empirical Analysis of User Passwords across Online Services”. In: *ACM Conference on Data and Application Security and Privacy (CODASPY)*. Mar. 13, 2018, pp. 196–203. ISBN: 978-1-4503-5632-9. DOI: [10.1145/3176258.3176332](https://doi.org/10.1145/3176258.3176332) (cit. on pp. [18](#), [194](#)).
- [219] D. Wang, D. He, P. Wang, and C.-H. Chu. “Anonymous Two-factor Authentication in Distributed Systems: Certain Goals are Beyond Attainment”. In: *IEEE Transactions on Dependable and Secure Computing* 12.4 (2014), pp. 428–442 (cit. on p. [27](#)).
- [220] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang. “Targeted Online Password Guessing: An Underestimated Threat”. In: *ACM SIGSAC Conference on Computer and Communications Security (CCS)*. Oct. 24, 2016, pp. 1242–1254. ISBN: 978-1-4503-4139-4. DOI: [10.1145/2976749.2978339](https://doi.org/10.1145/2976749.2978339) (cit. on pp. [16](#), [17](#), [19](#), [28](#), [79](#), [163](#), [176](#)).
- [221] *WebGL*. accessed 2020-06-30. URL: <https://get.webgl.org> (cit. on pp. [1](#), [35](#), [56](#)).
- [222] *WebRTC*. accessed 2020-06-30. URL: <https://webrtc.org> (cit. on p. [1](#)).
- [223] M. Weir, S. Aggarwal, B. d. Medeiros, and B. Glodek. “Password Cracking Using Probabilistic Context-Free Grammars”. In: *IEEE Symposium on Security and Privacy (S&P)*. May 2009, pp. 391–405. DOI: [10.1109/SP.2009.8](https://doi.org/10.1109/SP.2009.8) (cit. on pp. [16](#), [17](#)).

- [224] S. Wiefeling, L. Lo Iacono, and M. Dürmuth. “Is This Really You? An Empirical Study on Risk-Based Authentication Applied in the Wild”. In: *IFIP International Conference on ICT Systems Security and Privacy Protection (SEC)*. 2019, pp. 134–148. ISBN: 978-3-030-22312-0. DOI: [10.1007/978-3-030-22312-0_10](https://doi.org/10.1007/978-3-030-22312-0_10) (cit. on pp. 22, 23).
- [225] W. Wu, J. Wu, Y. Wang, Z. Ling, and M. Yang. “Efficient Fingerprinting-Based Android Device Identification with Zero-Permission Identifiers”. In: *IEEE Access* 4 (Nov. 8, 2016), pp. 8073–8083. ISSN: 2169-3536. DOI: [10.1109/ACCESS.2016.2626395](https://doi.org/10.1109/ACCESS.2016.2626395) (cit. on p. 228).
- [226] Z. Yang, R. Zhao, and C. Yue. “Effective Mobile Web User Fingerprinting via Motion Sensors”. In: *IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 1398–1405. DOI: [10.1109/TrustCom/BigDataSE.2018.00194](https://doi.org/10.1109/TrustCom/BigDataSE.2018.00194) (cit. on p. 33).
- [227] T.-F. Yen, Y. Xie, F. Yu, R. (Yu, and M. Abadi. “Host Fingerprinting and Tracking on the Web: Privacy and Security Implications”. In: *Network and Distributed System Security Symposium (NDSS)*. Feb. 2012. URL: https://www.ndss-symposium.org/wp-content/uploads/2017/09/11_3.pdf.
- [228] T.-F. Yen, Y. Xie, F. Yu, R. (Yu, and M. Abadi. “Host Fingerprinting and Tracking on the Web: Privacy and Security Implications”. In: *Network and Distributed System Security Symposium (NDSS)*. Feb. 2012. URL: https://www.ndss-symposium.org/wp-content/uploads/2017/09/11_3.pdf (cit. on p. 2).
- [229] J. Yoon, L. N. Drumright, and M. van der Schaar. “Anonymization Through Data Synthesis Using Generative Adversarial Networks (ADS-GAN)”. In: *IEEE Journal of Biomedical and Health Informatics* 24.8 (Aug. 2020), pp. 2378–2388. ISSN: 2168-2208. DOI: [10.1109/JBHI.2020.2980262](https://doi.org/10.1109/JBHI.2020.2980262) (cit. on p. 201).
- [230] J. Yu and E. Chan-Tin. “Identifying Webrowsers in Encrypted Communications”. In: *Workshop on Privacy in the Electronic Society (WPES)*. 2014, pp. 135–138. ISBN: 978-1-4503-3148-7. DOI: [10.1145/2665943.2665968](https://doi.org/10.1145/2665943.2665968).
- [231] V. Zorkadis and P. Donos. “On Biometrics-based Authentication and Identification from a Privacy-protection Perspective: Deriving Privacy-enhancing Requirements”. In: *Information Management & Computer Security* 12.1 (Jan. 1, 2004), pp. 125–137. ISSN: 0968-5227. DOI: [10.1108/09685220410518883](https://doi.org/10.1108/09685220410518883) (cit. on pp. 6, 25, 82).

List of Author's Publications

- [12] N. Andriamilanto, T. Allard, and G. Le Guelvouit. “Guess Who? Large-Scale Data-Centric Study of the Adequacy of Browser Fingerprints for Web Authentication”. In: *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*. Ed. by L. Barolli, A. Poniszewska-Maranda, and H. Park. 2021, pp. 161–172. ISBN: 978-3-030-50399-4. DOI: [10.1007/978-3-030-50399-4_16](https://doi.org/10.1007/978-3-030-50399-4_16). URL: <https://hal.archives-ouvertes.fr/hal-02611624/document> (cit. on p. 8).
- [13] N. Andriamilanto, T. Allard, and G. Le Guelvouit. “FPSelect: Low-Cost Browser Fingerprints for Mitigating Dictionary Attacks against Web Authentication Mechanisms”. In: *Annual Computer Security Applications Conference (ACSAC)*. 2020. ISBN: 978-1-4503-8858-0/20/12. DOI: [10.1145/3427228.3427297](https://doi.org/10.1145/3427228.3427297). URL: <https://hal.archives-ouvertes.fr/hal-02965948/document> (cit. on p. 8).
- [15] N. Andriamilanto, G. Le Guelvouit, and T. Allard. “Browser Fingerprinting for Web Authentication, A Large-scale Empirical Analysis”. In: *Rendez-Vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information (RESSI)*. national conference without proceedings. 2019 (cit. on p. 8).

Titre : Authentification Forte par Prise d'Empreinte de Navigateur

Mots-clés : prise d'empreinte de navigateur, authentification web, authentification multi-facteur

Résumé : L'authentification web consiste à vérifier que le visiteur d'un site web est bien le détenteur d'un compte. Pour ce faire, plusieurs informations peuvent servir de preuve de détention, dont les empreintes de navigateur. Celles-ci sont des propriétés collectées à partir d'un navigateur permettant d'en constituer une empreinte potentiellement unique. Au travers de cette thèse, nous proposons deux contributions :

1. Nous étudions l'adéquation des empreintes de navigateur pour de l'authentification. Nous faisons le lien entre les empreintes digitales et celles des navigateurs afin d'évaluer ces dernières selon des propriétés d'informations biométriques. Nous basons notre étude sur l'analyse de quatre jeux de don-

nées d'empreintes de navigateur, dont un comprenant presque deux millions de navigateurs.

2. Nous proposons FPSelect, un outil de sélection d'attributs tels qu'ils satisfassent un niveau de sécurité et réduisent les contraintes d'utilisation. Le niveau de sécurité est mesuré selon la proportion d'utilisateurs usurpés étant donné les attributs utilisés, une population de navigateurs, et un attaquant modélisé. Les contraintes sur l'utilisation sont mesurées selon le temps de collecte des empreintes, leur taille, et leur instabilité. Nous comparons les résultats de FPSelect avec des méthodes usuelles de sélection d'attributs sur deux jeux de données.

Title: Leveraging Browser Fingerprinting for Web Authentication

Keywords: browser fingerprinting, web authentication, multi-factor authentication

Abstract: Web authentication is the verification that a visitor claiming an account legitimately owns this account. Several authentication factors were proposed such that each one provides a supplementary security barrier. Browser fingerprints notably came out as a promising candidate. They are the aggregation of properties collected from a web browser, which compose a potentially unique fingerprint. In this thesis, we provide two contributions:

1. We investigate the adequacy of browser fingerprints for web authentication. We make the link between the digital fingerprints that distinguish browsers, and the biological fingerprints that distinguish Humans, to evaluate browser fingerprints according to properties in-

spired by biometric authentication factors. We assess these properties on four real-life browser fingerprint datasets, which include one of nearly two million browsers.

2. We propose FPSelect, an attribute selection framework to find the attribute set that satisfies a security requirement and reduces the usability cost. The security is measured as the proportion of impersonated users given a fingerprinting probe, a user population, and a modeled attacker. The usability is quantified by the collection time of browser fingerprints, their size, and their instability. We compare our framework with common baselines using on two real-life fingerprint datasets.