Online Privacy in Mobile and Web Platforms: Risk Quantification and Obfuscation Techniques

Author:
Masood, Rahat

Publication Date:
2019

DOI:
https://doi.org/10.26190/unsworks/21642

License:
https://creativecommons.org/licenses/by-nc-nd/3.0/au/
Link to license to see what you are allowed to do with this resource.

Downloaded from http://hdl.handle.net/1959.4/64984 in https://unsworks.unsw.edu.au on 2024-05-01

Online Privacy in Mobile and Web Platforms: Risk Quantification and Obfuscation Techniques

Rahat Masood

A thesis in fulfillment of the requirements for the degree of

Doctor of Philosophy

UNSW
AUSTRALIA

School of Electrical Engineering and Telecommunications
Faculty of Engineering
The University of New South Wales

September 2019

Thesis/Dissertation Sheet

Surname/Family Name	:	**Masood**
Given Name/s	:	**Rahat**
Abbreviation for degree as give in the University calendar	:	**Ph.D.**
Faculty	:	**Faculty of Engineering**
School	:	**School of Electrical Engineering and Telecommunications (EE&T)**
Thesis Title	:	**Online Privacy in Mobile and Web Platforms: Risk Quantification and Obfuscation Techniques**

Abstract 350 words maximum: (PLEASE TYPE)

The wide-spread use of the web and mobile platforms and their high engagement in human lives pose serious threats to the privacy and confidentiality of users. It has been demonstrated in a number of research works that devices, such as desktops, mobile, and web browsers contain subtle information and measurable variation, which allow them to be fingerprinted. Moreover, behavioural tracking is another form of privacy threat that is induced by the collection and monitoring of users gestures such as touch, motion, GPS, search queries, writing pattern, and more. The success of these methods is a clear indication that obfuscation techniques to protect the privacy of individuals, in reality, are not successful if the collected data contains potentially unique combinations of attributes relating to specific individuals.

With this in view, this thesis focuses on understanding the privacy risks across the web and mobile platforms by identifying and quantifying the privacy leakages and then designing privacy preserving frameworks against identified threats. We first investigate the potential of using touch-based gestures to track mobile device users. For this purpose, we propose and develop an analytical framework that quantifies the amount of information carried by the user touch gestures. We then quantify users privacy risk in the web data using probabilistic method that incorporates all key privacy aspects, which are uniqueness, uniformity, and linkability of the web data. We also perform a large-scale study of dependency chains in the web and find that a large proportion of websites under-study load resources from suspicious third-parties that are known to mishandle user data and risk privacy leaks.

The second half of the thesis addresses the abovementioned identified privacy risks by designing and developing privacy preserving frameworks for the web and mobile platforms. We propose an on-device privacy preserving framework that minimizes privacy leakages by bringing down the risk of trackability and distinguishability of mobile users while preserving the functionality of the existing apps/services. We finally propose a privacy-aware obfuscation framework for the web data having high predicted risk. Using differentially-private noise addition, our proposed framework is resilient against adversary who has knowledge about the obfuscation mechanism, HMM probabilities and the training dataset.

Declaration relating to disposition of project thesis/dissertation

I hereby grant to the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or in part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all property rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstracts International (this is applicable to doctoral theses only).

…… …… ……

Signature Witness Signature Date

The University recognises that there may be exceptional circumstances requiring restrictions on copying or conditions on use. Requests for restriction for a period of up to 2 years must be made in writing. Requests for a longer period of restriction may be considered in exceptional circumstances and require the approval of the Dean of Graduate Research.

FOR OFFICE USE ONLY Date of completion of requirements for Award:

Originality Statement

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Signed: .....

Rahat Masood, Sydney, Australia

Date: .....

INCLUSION OF PUBLICATIONS STATEMENT

UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in their thesis in lieu of a Chapter if:

- The student contributed greater than 50% of the content in the publication and is the "primary author", ie. the student was responsible primarily for the planning, execution and preparation of the work for publication
- The student has approval to include the publication in their thesis in lieu of a Chapter from their supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis

Please indicate whether this thesis contains published material or not.

☐ *This thesis contains no publications, either published or submitted for publication (if this box is checked, you may delete all the material on page 2)*

☒ *Some of the work described in this thesis has been published and it has been documented in the relevant Chapters with acknowledgement (if this box is checked, you may delete all the material on page 2)*

☐ *This thesis has publications (either published or submitted for publication) incorporated into it in lieu of a chapter and the details are presented below*

CANDIDATE'S DECLARATION

I declare that:

- I have complied with the Thesis Examination Procedure
- where I have used a publication in lieu of a Chapter, the listed publication(s) below meet(s) the requirements to be included in the thesis.

Name	**Signature**	**Date (dd/mm/yy)**
Rahat Masood		

Postgraduate Coordinator's Declaration (to be filled in where publications are used in lieu of Chapters)

I declare that:

- the information below is accurate
- where listed publication(s) have been used in lieu of Chapter(s), their use complies with the Thesis Examination Procedure
- the minimum requirements for the format of the thesis have been met.

PGC's Name	**PGC's Signature**	**Date (dd/mm/yy)**

Copyright Statement

I hereby grant the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all proprietary rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation. I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstract International (this is applicable to doctoral theses only). I have either used no substantial portions of copyright material in my thesis or I have obtained permission to use copyright material; where permission has not been granted I have applied/will apply for a partial restriction of the digital copy of my thesis or dissertation.

Signed: .....

Rahat Masood, Sydney, Australia

Date: .....

Authenticity Statement

I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis. No emendation of content has occurred and if there are any minor variations in formatting, they are the result of the conversion to digital format.

Signed: .....

Rahat Masood, Sydney, Australia

Date: .....

Abstract

The wide-spread use of the web and mobile platforms and their high engagement in human lives pose serious threats to the privacy and confidentiality of users. Service providers use variety of technologies to collect a lot of information about their users; from basic details like name and birthday to complete history of a user searches, clicks, locations, and devices and often a very sensitive information such as health and financial records. One obvious reason to collect such information is to create personalized user experience with the goal of increasing revenue, but at the same time this information is often used to profile users for targeted advertising, perform aggregate measurement and analytics e.g. traffic statistics, or in some cases, sold to third-parties in (un)anonymized form either for research or other purposes. This plethora of collecting as much information about users, sometimes unknowingly to them, has raised serious privacy concerns in the digital world. Among many, Online Tracking is one of the glitches that may have devastating consequences on a user's private life. Research has shown that desktops and mobile devices, and associated web browsers and mobile apps contain subtle information which allow them to be "fingerprinted or tracked".

As a matter of fact, online tracking is no longer limited to traditional mechanisms of storing web browser cookies, or IP addresses. Now, advanced tracking techniques such as behavioral-based tracking is being used to more precisely target users. Behavioral-based tracking is induced by the collection and monitoring of users online activities and has the potential to track or identify the actual (physical) person. In addition, first-party websites import a range of external resources from various third-party domains that further load resources hosted on other domains. For each website, this creates a dependency chain underpinned by a form of implicit trust between the first-party and transitively connected third-parties. This inter-connectivity of web services also leads to online tracking by various unknown third-parties without user consent. Thus, the success of tracking mechanisms is a clear indication that anonymization or obfuscation techniques to protect the privacy of individuals, in reality, are not successful if the collected data contains potentially unique combinations of attributes relating to specific individuals.

With this in view, this thesis focuses on understanding the privacy risks across web and mobile platforms by identifying and quantifying the privacy leakages and then designing privacy preserving frameworks against identified leakages. We first investigate the potential of using touch-based gestures for tracking mobile device users. For this purpose, we propose and develop an analytical framework that quantifies the amount of information carried out by the user touch gestures. Our findings highlight that user touch gestures exhibit high uniqueness and can help re-identifying users with high accuracy.

We then quantify users' privacy risk in web data using probabilistic methods that incorporate three key privacy considerations: uniqueness, uniformity, and linkability of web data. Our experimental results show that the proposed quantitative method is effective in predicting privacy risks in web data. We also perform a large-scale study of dependency chains in the web and find that a large proportion of websites under-study load resources from suspicious third-parties that are known to mishandle user data and risk privacy leaks.

The second half of the thesis addresses the abovementioned identified privacy risks by designing and developing privacy preserving frameworks for the web and mobile platforms. We propose an on-device privacy preserving framework that minimizes privacy risks by bringing down the threat of tracking and identification of mobile users while preserving the functionality of the existing apps/services. We formulate our problem as time-series modeling and forecasting that overcomes the problem of handling unpredictable data and balancing privacy-utility when sensor data is highly dynamic. Rigorous experiments on datasets show that out framework limits user tracking and identification threats while maintaining a reasonable level of utility.

We finally propose a privacy-aware obfuscation framework for the web data. Using differentially-private noise addition, our proposed framework is resilient against adversaries who have knowledge about the obfuscation mechanism and the training dataset. Our experimental study conducted on two real web datasets validates the significance and efficacy of our framework. Our results indicate that some obfuscated entries offer almost no privacy risk while achieving high utility.

We conclude this thesis with two key findings. First, we observe that users can be tracked and identified through their unique behavior on the web and mobile platforms. For instance, we show that touch sensors on mobile devices and data entries on the web have an ability to uniquely identify users, leading to a threat of online tracking. Second, we realize that new privacy-preserving methods are required to handle tracking issues in an online web and mobile environment. The second half of our thesis is dedicated to the design and development of such methods.

Acknowledgements

I would like to express my sincere gratitude and appreciation to my supervisor, Professor Mohamed Ali (Dali) Kaafar for giving invaluable guidance and support throughout my PhD research work. The quality of this thesis would not be possible without him sharing his exceptional scientific knowledge and innovative insights.

I extend my sincere gratitude and appreciation to my UNSW supervisor Professor Aruna Seneviratne, for his unconditional support, guidance, and constant encouragement. I am extremely grateful for his time, valuable advices, and constructive suggestions. It was a great privilege and an honor for me to be his PhD student.

I would like to thank and acknowledge Dr. Hassan Jameel Asghar and Dr. Dinusha Vatsalan for being very supportive and helpful in providing constructive feedback on my research with their expertise and proof-reading my writing despite their busy schedule. My thanks also go to Benjamin Zi Hao Zhao, Dr. Muhammad Ikram, and Dr. Gareth Tyson for their skillful research collaborations and valuable contributions.

I am thankful to the University of New South Wales (UNSW) and Data61-CSIRO for offering me an opportunity to conduct my research studies. Both institutions are well supportive of students, and their progress, as well as their wellbeing. I am honored to be an International Postgraduate Research Scholarship (IPRS) recipient from the Australian Government Department of Innovation, Industry, Science and Research. I am sincerely thankful for the financial support provided by the IPRS and the logistics provided by UNSW, and Data61-CSIRO for my studies.

I have been fortunate to work within a group of great people at Data61-CSIRO. I therefore, extend my special thanks to all my colleagues and friends at Data61-CSIRO for keeping me motivated and encouraged during my studies. I will always remember all the enjoyable moments that we shared together.

And finally, last but not least, I would like to thank my mother, siblings, and husband for their unconditional love, care, encouragement, and understanding. They all were my strength throughout this journey.

Dedicated

to

my loving Parents

for supporting me all the way!

Contents

List of Figures

List of Tables

List of Abbreviations

ADF	Dickey–Fuller Test.
ADTree	Alternating Decision Tree.
AJAX	Asynchronous JavaScript and XML.
AOL	Americal OnLine.
API	Application Programming Interface.
APIs	Application Programming Interfaces.
AV	anti-virus.
CDF	Cumulative Distribution Function.
CDNs	Content Distribution Networks.
CSP	Content Security Policy.
CSS	Cascading Style Sheets.
DDoS	Distributed Denial of Service Attack.
DNN	Deep Neural Network.
DNS	Domain Name Service.
DNT	Do Not Track.
DSP	Digital Signal Processing.
DTW	Dynamic Time Wrapping.
EER	Equal Error Rate.
FPR	False Positive Rate.
FPRs	False Positive Rates.
GANS	Generative Adversarial Networks.

GMMs	Gaussian Mixture Models.
GPS	Global Position System.
HCI	Human Computer Interaction.
HMM	Hidden Markov Model.
HTML	Hypertext Markup Language.
HTML5	Hypertext Markup Language version 5.
HTTP	Hyper Text Transfer Protocol.
HTTPS	Hyper Text Transfer Protocol Secure.
ICMP	Internet Control Message Protocol.
IID	Independent and Identically Distributed.
IMEI	International Mobile Station Equipment Identity.
iOS	iPhone Operating System.
IP	Internet Protocol Address.
IRT	Item Response Theory.
JSON	JavaScript Object Notation.
K-NN	k-Nearest Neighbor.
LDOs	Local Shared Objects.
LP	Linear Prediction.
LPPM	Location Privacy Preserved Mechanism.
LPPMs	Location Privacy Preserved Mechanisms.
LSI	Latent Semantic Indexing.
MAE	Mean Absolute Error.
mRMR	maximum-relevancy minimum-redundancy.
MTurk	Mechanical Turk.
NISPP	Noise Injection for Search Privacy Protection.
NLTK	Natural Language Toolkit.

NSA	National Security Agency.
OQF-PIR	Optimized Query Forgery for Private Information Retrieval.
OS	Operating System.
PDS	Plausibly Deniable Search.
PEAS	private,Efficient, and Accurate Web Search.
PETS	Privacy Enhancing Technologies.
PIA	Privacy Impact Assessment.
PII	Personal Identifiable Information.
PIR	Private Information Retrieval.
PRAW	PRivacy model for the Web.
PRNU	Photo-Response Non-uniformity Noise.
PSD	Power Spectral Density.
RMS	Root Mean Square.
ROC	Receiver Operating Characteristic.
RSS	Rich Site Summary.
SDKs	Software Development Kits.
SEO	Search Engine Optimisation.
SSID	Service Set IDentifier.
STRIDE	Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privileges.
SVM	Support Vector Machine.
TBATS	Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components.
TCP	Transmission Control Protocol.
TF-IDF	Term Frequency-Inverse Document Frequency.
TMN	Track Me Not.

TPL	Tracking Protection List.
TPR	True Positive Rate.
TPRs	True Positive Rates.
UDID	Unique Device IDentifier.
URL	Uniform Resource Location.
URLs	Uniform Resource Locations.
US	United States.
VM	Virtual Machine.
VT	Virus Total.
XML	Extensible Markup Language.
XSS	Cross-Site Scripting.
ZIP	Zone Improvement Plan.

List of Publications

Journals

1. Rahat Masood, Benjamin Zi Hao Zhao, Hassan Jameel Asghar, and Mohamed Ali Kaafar. Touch and you're trapp (ck) ed: Quantifying the uniqueness of touch gestures for tracking. *Proceedings on Privacy Enhancing Technologies*, 2018(2):122–142, 2018

Conferences

1. Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kaafar, Noha Loizon, and Roya Ensafi. The chain of implicit trust: An analysis of the web third-party resources loading. In *The World Wide Web Conference*, pages 2851–2857. ACM, 2019

2. Rahat Masood, Dinusha Vatsalan, Muhammad Ikram, and Mohamed Ali Kaafar. Incognito: A method for obfuscating web data. In *Proceedings of the 2018 World Wide Web Conference*, pages 267–276. International World Wide Web Conferences Steering Committee, 2018

3. Rahat Masood, Benjamin Zi Hao Zhao, Hassan Jameel Asghar, and Moahmed Ali Kaafar. Poster: Touchtrack: How unique are your touch gestures? In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2555–2557. ACM, 2017

Under Review

1. Rahat Masood, Shlomo Berkovsky, and Mohamed Ali Kaafar. *Modern Socio-Technical Perspectives on Privacy*, chapter Tracking and Personalization. Springer, 2019

2. Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kaafar, and Noha Loizon. Measuring and analysing the chain of implicit trust: A study

of third-party resources loading. *ACM Transactions on Privacy and Security (TOPS)*, 2020

3. Rahat Masood, Dinusha Vatsalan, Hassan Jameel Asghar, and Mohamed Ali Kaafar. Privacy preserving sensory data. *Proceedings on Privacy Enhancing Technologies*, 2020

Chapter 1

Introduction

The ever changing technological landscape, high user involvement, increased societal visibility, and amalgamation of services has made privacy difficult to maintain in a digital world. Preserving user identity from being tracked is a significant challenge nowadays and has become more complex with the advancement in technologies that have an ability to cross-link data sources to infer more information. Some examples aggravating the privacy concerns include location-based tracking, face recognition, mobile sensors to identify or track people, behavioural features, interactions and gestures, and so on. Data analysis methods and the exponentially growing computational resources available for data mining tasks are another potential obstacle for maintaining privacy. For example, huge cloud-based data centres have the ability to process and compare user profiles among massive sets of records, to identify and make sense of the relevant information. As the user models and predictions become more accurate, and as the services increase their reliance on these predictions, user privacy concerns may further increase.

This thesis is an attempt to highlight privacy concerns across web and mobile platforms by identifying and quantifying the privacy leakages, and then designing privacy preserving frameworks against the identified threats. We believe that tracking-related privacy concerns, highlighted in this thesis, will take a more prominent role and will attract research works and practical industry attention alike.

1.1 Motivation

The maturity and high acceptance of digital technologies, such as web and mobile platforms, have finally made their way to user's everyday life. According to a survey, the internet user growth has reached 4.39 billion in 2019, representing an increase of 366 million (9%) since January 2018 [126]. Moreover, the ways in which people use

the internet are evolving quickly too, with mobile phones accounting for an ever-increasing share of our online activities. Now, there are 5.11 billion unique mobile users with roughly 5.5 billion smartphones in use across the world [126]. However, the high engagement of web and mobile technologies in personal lives comes with risks. Privacy is one of the pertinent issues that has only been exacerbated with the increasing use of mobile and web in our daily lives. Users have surely lost the ability to control how their data is being stored, modified, used, and exchanged between different parties. In general, online privacy refers to the right of an individual to store, display, or provision third-parties, a selective amount of information on the internet.

Recent years have seen massive privacy breaches from the tech-giant companies such as Google and Facebook. For instance, computer-science researchers at Princeton university confirmed that Google services on Android devices and iPhones store user's location data, even if users' privacy setting prevent Google from locating a user [9]. Similarly, Facebook has recently been often involved in scandals such as Cambridge Analytica data harvesting, incitement to violence in Myanmar, Russian and Iranian meddling in the US elections, and several data-exposing bugs [15]. According to research from the International Computer Science Institute, roughly 17,000 Android apps collect identifying information about a user that creates a permanent record of the activity on a user device. The collected data can be used to target users for advertising, tracking online activities, profiling, or selling data to third-parties [97]. These examples indicate that user privacy breaches have subtle and devastating effects on users personal lives.

Amongst several web and mobile privacy risks, *online user tracking and identification* happens to have direct and disturbing consequences on a user's life. Online tracking has several meanings, but one of the most valid general definition is *"following the trails and movements of someone on the internet through means such as mobile phones, desktop, and smart devices, in order to gain unique information about them for incentives such as target advertising, profiling, data exchange, etc"* [77]. It is also quite evident from the literature that mobile and web platforms contain subtle information and measurable variation which allows them to be "fingerprinted or tracked". For-instance, it was found that more than 90% of Alexa top-500 websites contain third-party tracking content [186]. Similarly, a user could be identified or tracked from motion sensors signals produced from mobile phones [57, 64, 159]. This tracking threat becomes more subtle when users are identified from anonymized datasets through inference analysis by an eavesdropper or a researcher who has access to the data. Few examples in the literature involving such

threats are the re-identification of individuals in the anonymized AOL search histories of 650,000 users [96], Netflix training data of 500,000 subscribers [165], and Massachusetts hospital discharge data [205].

Behavioral-based tracking is a form of online tracking that constructs user profiles through their gestures/actions to perform certain activities. Such gestures are collected via many media, such as touch, motion, GPS, camera, mouse, search queries, and more. As opposed to "regular" tracking mechanisms based on cookies, browser fingerprints, logins and similar, which track virtual identities or browser profiles, this type of tracking is subtle and risky. First, while regular tracking deals with virtual identities and online profiles, behaviour-based tracking has the potential to track and identify the actual (physical) person operating the device. It can track multiple users accessing the same device. Second, behaviour-based tracking has the ability to continuously track users. Third, it also leads to cross-device tracking, where the same user can be tracked on multiple devices and user data can be collated and sent to advertising companies and third-party entities to build more encompassing user profiles.

1.2 Problem Statement

While device and web browser identification is an active area of research that has been demonstrated in a number of current research works, there is still a pressing need to understand privacy leakages that could lead to user tracking on the mobile and web platforms. Illumination on these veins of research has important social, ethical, and policy implications. We therefore, focus on understanding the privacy risks involving user tracking and identification on the web and mobile platforms, and then proposing privacy preserving frameworks to combat such risks. In essence, we argue the following thesis:

Traditional tracking mechanisms have a proven ability to identify devices and browsers, and in some cases, virtual identity of a user. Behavioural-based tracking, on the other hand, has far more devastating consequences on the privacy of a user, from tracking the actual (physical) identity of a person to cross-device tracking and continuously monitoring every online movement. The concerns of behavioural-based tracking go beyond commercial purposes, and can be used to unfairly discriminate users for discounted/high prices, take advantage of vulnerable users e.g. by showing bogus cures to a medically sick patient, or leave users vulnerable to warrantless searches, identity thieves, etc. Despite these disturbing consequences, there is to our knowledge no information in the public domain to quantify how much of a privacy

problem is posed by user tracking or identification through his/her behaviour to perform certain actions on the web and mobile platforms. The user-specific behaviour include activities such as swiping, typing, tapping on the mobile touchscreens or making a web entry on search engines, social networks, forums etc. In addition, the data collected from user online activities is not just held by a single entity/organization. For-instance, websites tend to import resources from a range of third-parties that load further resources from other domains, creating a dependency chain. Not explored is the fact that these third-parties in a chain may have a tendency to gather sensitive user data without even informing a user or even a first-party website itself. There is a dire need to investigate the abovementioned privacy issues through quantification and privacy preservation methods. The existing solutions to quantify privacy and prevent tracking on web and mobile platforms have practical limitations that often preclude their developers from striking a balance between privacy and utility goals.

1.3 Proposed Solution

We corroborate our thesis within two volets of research. In the first part, we tend to understand privacy issues involving user tracking and re-identification across web and mobile platforms. We identify, quantify, and predict privacy risks that arise from user unique and repeatable behavioural activities such as swiping on a mobile phone, typing on a mobile phone, searching specific query in a search engine, etc. In addition, we explore the web ecosystem to investigate malicious third-parties in a dependency chain that could lead to privacy leaks such as tracking. The second part of the thesis aims to reduce/lower the identified privacy risks by proposing privacy-preserving frameworks for web and mobile platforms.

We further classify the first part of thesis, 'Privacy Risk Identification and Quantification', into three subparts. The first subpart investigates the privacy leakages induced by the collection and monitoring of touch gestures of mobile device users. We propose and develop an analytical framework that quantifies the amount of information carried by the user touch gestures mainly swipes, keystrokes, taps, and handwriting. Our findings highlight that user touch gestures exhibit high uniqueness and users could be correctly re-identified with high accuracy, indicating that touch-based tracking is possible. The second subpart proposes an adversarial resistant, quantitative method that predicts privacy risks of users' web data. The proposed risk prediction method is applicable to any type of web application such as social networks, search engines, blogs, product reviews etc. In the third sub-

part, we conducted a large-scale study of dependency chains in the web to find malicious/suspicious third-parties and their activities on the web.

In the second part of thesis, 'Privacy-Preserving Frameworks', we first design and develop an on-device privacy preserving framework that minimizes the private information of a mobile user before releasing to a server, whilst maintaining the intended utility of an application/service. The framework addresses two privacy risks, *trackability* and *distinguishability* by obfuscating raw data coming from various apps. Finally, we develop an adversarial resistant obfuscation mechanism to improve the privacy of web data. The proposed mechanism obfuscates high risk data entries with semantically similar lower risk data entries and is shown to be effective against adversary who has knowledge about the datasets, obfuscation mechanism and model learned risk probabilities.

1.4 Thesis Contributions

This thesis makes the following contributions:

- **Quantifying the Uniqueness of Touch Gestures for Tracking:** We first investigate the potential of using touch-based gestures for tracking the users on mobile devices, which we refer to as touch-based tracking. To the best of our knowledge, this is the first study considering the potential of touch gestures to profile users. In order to demonstrate the likelihood of touch-based tracking, we develop an analytical framework that quantifies the amount of information contained in touch gestures, at different levels of granularity i.e. individual features of gestures, samples of gestures, as well as samples of combinations of gestures. We develop a game-like app called "TouchTrack" for Android devices that specifically captures four widely used touch screen gestures: i) swipes, ii) taps, iii) keystrokes, and iv) handwriting. Through this app, users can check the uniqueness and tracking potential of his/her gestures. We gather gesture samples from 89 users, and demonstrate that touch gestures contain sufficient information to uniquely identify and track users. Additionally, we also show that returning users could be correctly re-identified with high accuracy, indicating that touch-based tracking is possible. (cf. **Chapter 4**)

- **Quantification of Privacy Risks of Web Data:** While user privacy in web has been an active area of research for the last two decades, much of these work has been done on improving the anonymization methods or privacy preserving

publishing of web data. Only limited studies have been done on evaluating and predicting privacy risk of users in the web [175, 88, 13, 23, 164]. In this thesis, we quantify and predict the privacy risk of users through their web actions by using a probabilistic model, Hidden Markov Model (HMM) that calculates probabilities of uniqueness, uniformity, and linkability learned from the training data. Uniqueness refers to user web data entries that are unique enough to reveal his identity, Uniformity refers to repetition of the same web data entry by a user that puts more confidence in revealing his identity, whereas Linkability refers to how much Personal Identifiable Information (PII) is released by a user that could link him to the corresponding data. To the best of our knowledge, no work has been done that quantifies risks by considering these three key aspects of privacy. We measure the privacy risk associated with search queries and apps reviews and the results show that our privacy prediction method is reliable enough to identify high risk web entries via three aspects. (cf. **Chapter 5**)

- **Measuring and Analyzing the Chain of Implicit Trust:** A modern web ecosystem works by loading resources from a range of third-party domains such as ad providers, tracking services, and analytics services. However, often overlooked is the fact that these third-parties load resources from other domains, forming a dependency chain. Although there has been extensive work looking at the presence of third-parties in general [79, 168, 138], little work has focused on how content is indirectly loaded and its impact on security and privacy. In this thesis, we conduct a large-scale study on the dependency chain of the Alexa's top-200K domains to find out the presence of suspicious third-party content. These suspicious or potentially malicious third-parties are known to mishandle user data and risk privacy leaks as evident from real-life incidents [96, 165, 205, 69, 35]. We then try to inspect what activities are undertaken within the suspicious third-party resource, Javascript. The activities of these scripts are diverse. For example, we find evidence of malicious search poisoning activities when JavaScript codes are loaded from third-parties. (cf. **Chapter 6**)

- **Privacy Preserving Framework for Mobile Sensor's Data:** Sensors embedded in smart devices monitor user's environment with high accuracy and provide variety of services to a device user, from finding routes, to health monitoring and handwritten words recognition. These sensors have a potential of disclosing private information about a user, that eventually leads to

user tracking or identification. To address this problem, we propose a privacy preserving framework that minimizes privacy leakages by bringing down the risk of tracking and identification of individual users while preserving the functionality of the existing apps/services. Our framework allows users to send information to the server so as to (i) keep user and apps isolated of privacy preserving mechanism, (ii) minimally affect the app's utility/accuracy, while (iii) providing privacy guarantee that they will not be identified via tracking or distinguishing from other users. We formulate our problem as time-series modeling and forecasting that overcomes the problem of handling unpredictable data and balancing privacy-utility when where sensor data is highly dynamic. The proposed framework is resilient against noise filtering attacks [215] as it adds correlated noise-series to the forecasted time-series such that the noise is indistinguishable by an adversary. Rigorous experiments on publicly available datasets show that out framework limits user tracking and identification threats while maintaining a reasonable level of utility. (cf. **Chapter 7**)

- **Incognito: A Method for Obfuscating Web Data:** Web users unintentionally leave digital traces of their personal information, interests, and intents while using the online services, such as social networks, discussion forums, blogs and knowledge sharing communities, product review sites, and search engines. Users' web data could therefore reveal private/sensitive information about them based on inference analysis by an eavesdropper or a researcher who gets access to these web data. Literature in web privacy also shows that inference attack is possible even when the datasets are anonymised (i.e. user identifiers are removed, encoded or masked). We propose a framework that obfuscates high risk web data entries with semantically similar lower risk data entries, with some utility loss. We use adversarial machine learning technique in obfuscation method to make our framework reliable against adversary attacks. The technique combines differential privacy-based noise addition with our previously introduced HMM in chapter 5. Our adversary model assumes that given a dataset, framework knowledge, and HMM based probabilities, the adversary is able to estimate the privacy risk values and could differentiate between the original and the altered data by getting all possible paths in the HHM that have higher risks. Our obfuscation framework guarantees privacy against adversarial attacks with high accuracy. (cf. **Chapter 8**)

1.5 Published and Submitted Work

This thesis has resulted in the (submissions)publications (to)in peer reviewed venues (ordered by chapter, indicated in bold):

- Rahat Masood, Shlomo Berkovsky, and Mohamed Ali Kaafar. *Modern Socio-Technical Perspectives on Privacy*, chapter Tracking and Personalization. Springer, 2019 (*under review*) (**Chapter 2 & 3**)
- Rahat Masood, Benjamin Zi Hao Zhao, Hassan Jameel Asghar, and Mohamed Ali Kaafar. Touch and you're trapp (ck) ed: Quantifying the uniqueness of touch gestures for tracking. *Proceedings on Privacy Enhancing Technologies*, 2018(2):122–142, 2018 (**Chapter 4**)
- Rahat Masood, Benjamin Zi Hao Zhao, Hassan Jameel Asghar, and Moahmed Ali Kaafar. Poster: Touchtrack: How unique are your touch gestures? In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2555–2557. ACM, 2017 (**Chapter 4**)
- Rahat Masood, Dinusha Vatsalan, Muhammad Ikram, and Mohamed Ali Kaafar. Incognito: A method for obfuscating web data. In *Proceedings of the 2018 World Wide Web Conference*, pages 267–276. International World Wide Web Conferences Steering Committee, 2018 (**Chapter 5 & 8**)
- Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kaafar, Noha Loizon, and Roya Ensafi. The chain of implicit trust: An analysis of the web third-party resources loading. In *The World Wide Web Conference*, pages 2851–2857. ACM, 2019 (**Chapter 6**)
- Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kaafar, and Noha Loizon. Measuring and analysing the chain of implicit trust: A study of third-party resources loading. *ACM Transactions on Privacy and Security (TOPS)*, 2020 (*under review*) (**Chapter 6**)
- Rahat Masood, Dinusha Vatsalan, Hassan Jameel Asghar, and Mohamed Ali Kaafar. Privacy preserving sensory data. *Proceedings on Privacy Enhancing Technologies*, 2020 (*under review*) (**Chapter 7**)

1.6 Thesis Organization

The rest of the thesis is organized as follows:

In **Chapter 2**, we define the context of this thesis by presenting the basic concepts and techniques of online tracking and also discusses its privacy implications and solutions.

In **Chapter 3**, we review the existing literature. We first discuss existing techniques to fingerprint or track mobile, web, and other platforms. We then summarize previously proposed privacy-preserving solutions for the mobile and web data.

In **Chapter 4**, we introduce a new privacy threat, touch-based tracking, which is induced by the collection of a user touch gestures on mobile devices. We present an analytical framework that quantifies the amount of information about users leaked by touch gestures and also show the accuracy of user tracking by correctly identifying the returning users.

In **Chapter 5**, we present our method to measure and predict privacy risk involving user web data entries. We also analyze privacy risk prediction results on two real web datasets and validates the significance and efficacy of our method.

In **Chapter 6**, we analyze the Alexa's top 200K domains to measure the extensiveness of dependency chains. We then further investigate suspicious third-party domains and their possible activities in the dependency chain.

In **Chapter 7**, we present our privacy preserving framework to minimize the privacy risks emanating from the mobile sensor's data. We address two threats, trackability (tracking) and distinguishability (identification) by obfuscating raw data coming from various apps. We then show the validation of the proposed framework through a series of experiments.

In **Chapter 8**, we propose an adversarial-resistant privacy preserving method that obfuscates high risk user web data entries with semantically similar data. We then show the effectiveness of our method by conducting experiments on two real web datasets. Results indicate that privacy risk are significantly reduced however,at

the cost of utility.

Finally, we conclude this thesis and frame future work in **Chapter 9**.

Chapter 2

Preliminaries/Background

This chapter summarizes the background material that contributes to the understanding of basic concepts and techniques of online tracking followed by its privacy implications and solutions. Section 2.1 overviews the types, techniques and various entities involved in online tracking. We discuss privacy implications and generic solutions to prevent online tracking in Section 2.2. We introduce a case-study of relating online tracking with personalization in Section 2.3 and show how personalization could be achieved by tracking users online.

2.1 Online Tracking

As mentioned earlier in the Section 1.1, online tracking is the ability of an adversary to follow the trails and movements of a user through means such as mobile phones, desktop, web browser etc. The subtle form of online tracking happens in a converged online environment, where users are utilizing multiple devices to access the services. In this situation, online tracking becomes more efficient, as multiple devices are linked to the same user. We refer to this as “cross-device tracking". Online tracking has various types and extensions: from detecting user interests when visiting a web page to detailed analysis of user's life, including location, social relations, health, political beliefs, etc. A combination of such information increases the chances of identifying and appropriately tracking a user online.

Web Tracking is one of the main sources of profiling that tracks users across different visits or sites. There are various design, implementation, and deployment methods that enable tracking. For instance, for an externally hosted website, a service provider can embed third-party content, which is hosted on servers tracking website visitors, or a website incorporates active content like JavaScripts snippets or libraries supplied by third-party to implement the tracking functionality. It was found that more than 90% of Alexa top-500 websites contain third-party tracking

content [186] and that 70% of the cookies recorded were third-party cookies set by just 25 third-party domains [75].

Mobile Tracking fingerprints and identifies users through the devices that are equipped with sophisticated sensors, such as microphones, GPS, accelerometers and so on, which generate highly sensitive data that can be used as unique fingerprints. These devices are always connected and being carried everywhere by their users, which not only makes them perfect targets for advertisers, but also often leads to physical tracking of users. It has been shown that a user could be identified or tracked from motion sensors signals produced from mobile phones [57, 64, 159]. Similar to web tracking, mobile devices contain various identifiers that solely or in combination lead to user tracking and profiling. For example, researchers demonstrated how the use of WiFi SSID (the Service Set IDentifier representing the WiFi network devices connect to) in its active discovery mode can lead to revealing geographical location of users [187] or to enabling the physical tracking [62]. A follow-up study has shown how to infer social relationship between mobile device owners by tracking their WiFi fingerprints [52].

In addition, many third-parties are performing cross-device or cross-app tracking that can provide a more complete view into user's behaviour. In cross-device tracking, third parties link together the devices that belong to a user [32], whereas in cross-app tracking an app identifies other apps installed on the device [4].

2.1.1 Tracking Techniques

In recent years, online tracking techniques have been extensively studied and it has been found that these techniques use information such as IP addresses, cookies, Javascript, and more for user identification purposes. In general, tracking could be performed using the following techniques:

- **Cookie** is a text stored by a user's web browser and transmitted as part of an HTTP request. Cookies are essential to manage long user sessions; however, they can also be used to uniquely identify a user's browser. While some purposes could be benign, service providers can use cookies to track users and collect their web activity. One special form of cookie is *persistent cookie*, which stores identifying information, such as user preferences for a long period of time. Similarly, *third-party cookies* are set while fetching website content like as images, frames, Javascripts, etc. *Cookie syncing* is another type, where unique identifiers are correlated with each other, so that the same user is identified in an external database. All these type of cookies raise serious

privacy concerns as they could be used by websites to track users across visits or multiple pages.

- **Javascript** codes can be loaded both from first- and third-party domains, and are widely used by ad networks, content distribution networks (CDNs), tracking services, analytics platforms, and online social networks [109]. They have the ability to track information about browsers such as cached objects, history of visited links, user-agent strings, or language preferences. In addition, they can read and write from/to a cookie database or reconstruct user identifiers. Such information enables servers and third-party domains to regularly track users using HTTP requests. The dynamic nature of Javascript also allows service providers to construct a behavioural profile of a user. For example, through Javascript event handlers, it is possible to obtain information about a user's mouse clicks and movements, scrolling, and so on [12].

- **Cache** stores the content of webpages and other information in the browser, to minimize latency and redundant network activity. This technique improves performance; however, it is possible for a server to associate a unique tracking identifier with each client requesting content for the first time. A server can then use Javascript and standardized messages to check if the content is cached or not, in order to identify a user. This technique is usually implemented for resources like images and is difficult to defend unless the cache content is cleared regularly, e.g., when closing the browser.

- **Supercookies** also known as *unique identifier headers*, inject user information into packets, which is then sent from a user device to a server. Some prominent supercookie types are *Flash Cookie* and *EverCookies*, where the former is maintained by the Adobe Flash plugin and the latter is a combination of various tracking mechanisms. *Local Shared Objects (LDOs)* is another form, supported by browser plugins, which can track users using unique identifiers. These objects are invisible to the browser and therefore it is impossible to examine their content. LDOs are retained in the browser even when the user deletes cookies and browser storage. For this reason, LDOs are used to store copies of browser cookies or other unique identifiers.

- **Stateless tracking** allows websites to track users based on information such as user agent, fonts, screen resolution, and more. Common techniques to track users using fingerprinting are as follows: (i) canvass fingerprinting detects minor differences in display hardware by reading back rendered text

from a storage area mapped to the display, (ii) font/plugin fingerprinting involves detection of fonts or plugins supported by a browser, (iii) MediaStream Fingerprinting is performed through Media Capture and Streams API that generates a unique stream identifier, (iv) WebRTC determines local IP address behind any firewall and can generate a unique tracking identifier, and (v) user agents / IP address in combination can be used to identify the user behind a browser. Although some of these techniques individually produce medium-entropy identifiers, it has been shown that a combination of these is unique enough to generate a high-entropy identifier.

Figure 2.1 shows the eco-system of an online tracking. We refer interested readers to [35], for a survey and in-depth study of online tracking mechanisms.

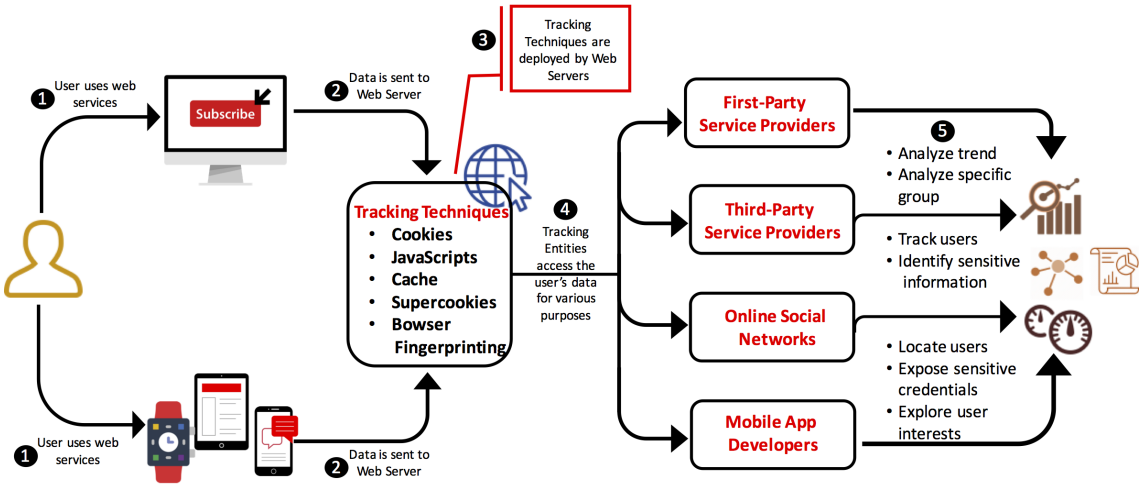


Figure 2.1: Eco-System of Online Tracking

2.1.2 Tracking Entities

The above mentioned tracking techniques can be used by various entities for various purposes. The most prominent entities are listed below:

- **First-Party Tracking** is performed by the service providers with which the user directly interacts. This entity controls the web domain a user has explicitly visited. A naive example of first-party tracking is Google, which tracks user interests via the search engine. Each time a user enters query in the search bar, Google keeps record of this and shows related links and advertisements in subsequent searches. A similar method is adopted by Facebook, where a user could be tracked via interests shown through likes, comments, and more.

- **Third-Party Tracking** is performed by the entities that track users across different services, e.g., websites. It can also be an entity that provides resources while a page is being displayed. Typical resources are the content embedded in the page or external content accessed by script running in the page. For instance, Google Analytics is a third-party entity that performs tracking to obtain statistics and send them to website publishers.

- **Online Social Networks** also track users around the web. These networks behave as publishers as well as ad networks. They allow advertisements to be displayed on their websites and at the same time, track user interest, e.g. via likes and comments, and sell them to advertisers. They also track users outside their network by planting widgets on other websites [42].

- **Mobile Devices** are equipped with multiple sensors like microphones, cameras, GPS, accelerometers, touch, and more [57, 58, 56, 157]. They also store personal information about their users: phone numbers, current location, owner's name, unique phone ID number, and so on. A combination of this information enables mobile app developers to track users without their consent and awareness.

2.1.3 Why Tracking

Online Tracking has several potential incentives. First-party can track to personalize user experience across sessions, to detect frauds, or to conform with law enforcement requiring websites to log user activities for fraud prevention and anti-laundering. However, there are cases where first-party websites voluntarily sell user identities. For example, Datalogix buy user information from companies, compile user dossiers, and then use it to target advertising [35]. Sometimes, a first-party can also act as a third-party; for instance, Facebook enforces users to provide their real names. This allows Facebook to identify user for personalizing widgets on external websites.

On the other hand, third-party tracking has a range of motivations, which can be grouped into six main reasons:

- **Advertising** is one of the most common reasons to track and identify users online. In order to sell products, gain revenues, or increase product awareness, businesses and companies build associations with ad networks. To be successful, it is however important to identify users and target the right ads on a website. For example, a user interested in buying a pair of shoes should be shown the ads related to shoes instead of other products.

- **Third-party measurement and analytics services** offer first-party websites to better understand their users. These services include demographics, content view distribution, and more. Some analytic companies follow a paid model, where an analytics company takes precautions to silo data between clients. In contrast, other companies offer a free analytics service; for example, Google Analytics tracks to obtain aggregate traffic statistics that are sent to service providers to allow them improve their content or enhance their services.

- **Social Networks** have become another way to track and identify users. These networks allow websites to offer personalized content and single sign-on services. Examples include Facebook's like and comment widget and Google's like button. These are offered for free, to increase user engagement and conduct market research. However, there are social services, such as Disqus, that exist almost exclusively in a third-party context [158]. The issues of collecting usage data and selling it for ad targeting and market research have been heavily debated recently.

- **Content providers** offer to host content such as video, maps, news, weather, stocks, and other media for embedding into websites. Youtube, for example, offers third-party widget to generate revenues through in-widget advertising. Many others, such as the Associated Press, also charge for their content.

- **Front-end services** includes Javascript libraries and APIs to speedup webpage loading and enable new page functionality, e.g., Google Feed API.

- **Hosting platforms** assist service providers to distribute their content. These platforms include blogs and content distribution networks, such as Akamai. All these services, in one way or another, help each other to track and identify users with the purpose of attracting customers, gaining profits, improving user experience, and increasing their business scope.

2.1.4 Behavioural Tracking: State-of-the-Art

Behaviour-based tracking has the ability to continuously and surreptitiously track users while they are interacting with their devices. This type of tracking is performed by data custodians, receivers, or consumers, in order to provide personalized services to their customers with the goal of increasing revenues. For instance, advertising companies take advantage of user behaviour profiles, user interests, characteristics,

or activities to display advertisements that is relevant for the user. Examples of data collected through user activities include the location of a user at a certain time, user touchscreen interaction, duration of the calls, dialed numbers, mouse clicks, and so on.

The ubiquity of smart devices and the fact that most data from touch and motion sensors can be extracted by any web service makes behavioural-based tracking a serious privacy threat. This not only represents a valuable source of information for analytics and ad services, but also for app developers who can (mis)use the information to track individuals on a single device or across devices. However, not all use cases of behavioural-based tracking are negative. It can also be beneficial to users and service providers alike. Some argue the benefits of behaviour-based tracking as a way of receiving useful information, e.g. relevant ads or health monitoring. For instance, monitoring a phone's motion might reveal changes in gait, which could be indicators of ailments or depression. Similarly, a child using their parent's smartphone can automatically have parental control enabled. Behavioural-based tracking could also bring commercial benefits to the user (e.g. displaying discounts and sales on the product of interest to the user).

Nevertheless, behaviour-based tracking is still perceived as a threat to privacy, mainly because of the continuous surveillance of users' online and physical activities. Imagine a user walking down a street who would like to know about interested places in the vicinity that matches his profile, but at the same time, all of his activities are under continuous surveillance.

2.2 Privacy Implications of Tracking

Although online tracking has been performed for a number of reasons that bring tremendous value, it also raises serious privacy concerns having subtle and devastating effects. Researchers, civil organizations, and policy-makers have identified several ways tracking can cause privacy leaks.

- **Global surveillance**, performed by government for security reasons or by companies for commercial benefits, is one such risk. Such a surveillance is not only a threat to privacy but there may be chances that collected information is distorted and leads to incorrect decisions. The potential dangers would be error, abuse, and lack of transparency and accountability. Bujlow et al. [35] explains how NSA is doing global surveillance through logins, cookies, Google PREFIDs, or DoubleClick cookies.

- **Profiling**, performed by service providers to personalize content for users, is another risk. A news site may display news matching user's previous items, a merchant may propose products based on user's previous shopping, or a search engine may refine results based on user's previous queries. Thus, content and service personalization is a source of information leakage. Often, such a profiling may seriously impact users. For example, it was shown that a person discovered his teenage daughter's pregnant when she received advertisement of baby food. The teenager was profiled as pregnant based on her shopping behaviour [69]. Similarly, Gmail was shown to use words from the sent and received emails to target ads. The emails were scanned without a user's explicit permission and used to identify the themes and trends for ad targeting [35].

- **Anonymized public data** is required in several business applications and research studies, to improve the provided services by utilizing the available information and rich user data. However, studies have shown that users could be identified even from anonymized datasets through inference analysis by an eavesdropper. A few examples involving such threats are the re-identification of users in the anonymized AOL search histories, Netflix training data that was attacked, and Massachusetts hospital discharge data [96, 165, 205]. In addition, eavesdroppers can violate privacy of users by tracking their activities, thereby inferring their personal profiles. Therefore, a user's privacy is at risk when their data can be distinguised from other users and linked to the user.

- **Personalized Search**, which offers the benefit of presenting information that the user would like to see based on their queries, is another reason to track. However, it has been shown in [155] that even anonymized search queries could lead to identification of users and their interests. Web measurements and analytics used to enhance user experience not only lead to virtual user tracking, but in some cases can disclose the user's physical identity.

- Lastly, tracking was found to be the reason for **price discrimination** based on geographical location, affluence of the user, and the referrer. Examples include credit card interest rates, hotel bookings, and insurance coverage. A case appeared in 2009 when Kevin Johnson reported to have his credit limit in American Express lowered to $3800 from $10800 after he shopped online at Walmart. American Express claimed that it was due to the fact that many other Walmart customers have problems with paying the credit back [53]. Bujlow et al. [35] provides a detailed overview of how such implications occur.

2.2.1 Privacy Preserving Solutions

There has been a continuous effort by researchers, businesses, and non-profit organizations to provide efficient solutions to overcome user privacy/tracking issues. Some of these efforts resulted in legal frameworks and policies, as well as in technological products. These initiatives include:

- **Blocking Tools:** A number of browser tools and plugins have been developed to protect users from tracking. These tools performed various functionalities such as detecting or blocking list of third-party trackers, informing users how much information is revealed to trackers, allowing only executable content from trusted domains to run, detecting flash cookies and deleting them, and more. Some noteworthy blocking tools are NoScript, BetterPrivacy, Ghostery, Do Not Track Plus, AdBlock Plus, and PrivacyBucket. A detailed analysis on these tools has been provided in [76]. There is also a Tracking Protection List (TPL) approach that contains web addresses of misbehaving tracking sites published by various organizations. Other ways to protect information include tools like private browsing modes of major browsers and anonymity networks.

- **Privacy-by-Design:** This is deemed to be an essential step towards better privacy protection, as it is based on the idea that privacy requirements should be taken into account while designing a system. As any process, privacy by design should have well-defined objectives, methodologies, and evaluation metrics. The privacy objectives of the system could be defined by conducting a preliminary Privacy Impact Assessment (PIA) or a privacy risk analysis. A range of methods for security risk analysis has been defined, but a few of them are dedicated to privacy risk analysis. The methods that could be adapted for privacy analysis are STRIDE or EBOIS. A few behavioural advertising systems, like Adnostic, PrivAd and RePriv, consider privacy as a design requirement. The main objective of these systems is to limit tracking, while still serving behavioural advertisements. For instance, PrivAd includes a trusted third-party that anonymizes clients and prevents ad network from identifying them [84]. In Adnostic, the browser continuously updates user profiles [209], allowing the ad network to offer several ads to the browser, where the browser picks the most relevant to the profile ad.

- **Do Not Track (DNT):** Major browsers implement the DNT (Do Not Track) methodology to show websites that are forbidden from tracking. DNT is a technology and policy proposal that enables users to opt out of tracking by

(all) websites they do not visit, including analytics services, advertising networks, and social platforms [71]. Technically, the implementation of DNT is simple; a browser sends a DNT header in every HTTP request to websites the user's wish to opt out of tracking. This includes web pages and all the objects/scripts embedded within a page. However, it is up to the discretion of an advertiser to respect user preferences. There are several ways to validate if an advertiser is following the instructions, ranging from self-regulation via the Network Advertising Initiative, to supervised self-regulation or co-regulation, to direct regulation.

- **Consent-based Mechanisms** are also used to inform users and/or prevent them from being tracked [77]. However, one key issue with consent-based mechanisms is that the entity that informs users is often not the only entity to track users. For instance, third-party trackers also collects and share information about users, which the first-party may be not be aware of. In this situation, either the service provider should ask all third-parties to declare the purposes of their data collection, or the third-parties should inform users of tracking before asking for a consent. Therefore, there could be two layers of information which must be transparent for user when third-parties are called. The first should explain why third-parties are called and which services rely on these third-parties and the second should explain how the third-parties process users data. A "tag manager" is also a technical implementation of the cookie consent that could block third-party scripts if consent has not been obtained.

- **Privacy-Preserving Methods:** Several privacy enhancing technologies (PETs) methods have been proposed to preserve the privacy of a user such as anonymization, identity management systems, privacy proxies, encryption mechanisms, differential privacy, and more. For example, GooPIR [67] and PRivAcy model for the Web (PRAW) [196] are standalone applications, where the former adds noise to Google queries and the latter generates fake queries in different topics of interest of the user. A study by Chen et al. [47] investigated the effectiveness of different obfuscation strategies and policies for online social networks and proposed a novel obfuscation strategy not requiring knowledge about the classifier. Salman et al. [189] and Li et al. [141] proposed methodologies that prevent inference attacks by distorting data before making it publicly available. Raval et al. [183] proposed utility-aware obfuscation framework that limits the risk of disclosing sensitive information from sensors data. Another work proposed a privacy-preserved mechanism for user location

data while reducing utility loss [30]. Similarly, Shokri et al. [26] tried to protect user trajectories by generating fake privacy-preserved location traces. A framework that automatically selects location privacy-preserving mechanism based on user requirement of privacy and utility has been proposed in [41]. In another recent paper [54], Das proposed an obfuscation scheme [58] to defeat fingerprinting based on motion sensors.

- **User Agents** can also prevent tracking by providing users with relevant choices. Most user agents include functionalities that allow users to examine cookies associated with a domain or a web page, showing expiry duration, their contents and the associated host domain [77]. Such information can be presented as user agent settings through a user interface to get a valid consent mechanism from the user. This has already been implemented by a browser extension that uses DNT Consent API to take consent from user before sending or receiving any data from the browser. Similarly, Content-Security-Policy API (CSP) is another tracking prevention tool that prevents cross-site scripting, click-jacking and other code injection attacks. CSP provides a standard method for first-party services to declare specific type of content that user agents should be allowed to load on that website – covered types are JavaScript, CSS, HTML frames, web workers, fonts, images, embeddable objects etc. If any of these content types are provided in the source list within the CSP header, then user agent will load only that content type in a browser and block rest of the types. In this way user agents can be told to block iframes from being loaded when they have not been explicitly allowed by the site designers or which refuse to respect the provided CSP. In general, user agents can prevent tracking at various granularity levels. This includes (i) items the user wants to block or take consent, like list of websites, tracking companies, (ii) locations of blockage iii) types of data, or iv) purpose of data.

- **Opt-Out:** Some tracking companies allow users to set opt-out cookies. If implemented properly, this option disables user tracking. However, opt-out cookies are not considered reliable, as they are not supported by all ad networks and are easy to interpret by those wishing to track users. Moreover, they have a limited lifetime, so they must be periodically renewed. These cookies are lost when the user cleans the cookies from their web browser.

2.3 Personalization and Tracking: A Case Study

Personalized technologies are deployed nowadays by virtually every website and mobile app. These technologies facilitate the "*provision of content and services tailored to individuals based on knowledge about their preferences and behaviour*" [19]. While personalized services started two decades ago with use cases like web content filtering and eCommerce recommendations, they have spread since to applications like music, tourism, eHealth, and many more [34].

2.3.1 Purpose of Personalization

Naturally, the tailoring of services offered by personalization can benefit both the service provider and the end user. For the former, it allows to uplevel the quality of the service, as it gets adjusted to the needs and preferences of the user. This can lead to improved performance, such as increased revenues, higher click-through rate, or returning customers. Likewise, the users also benefit from the personalization, as the overall user experience is improved. For example, personalization can shorten the discovery of a desired content or reduce the costs of buying a product.

Many algorithmic approaches for personalization have been developed, evaluated, and deployed. Some of them rely on statistical correlations of past user behavior [169], while others capitalize on extensive domain knowledge [60]. Regardless of the underlying personalization algorithm, a necessary precondition for personalized services is the availability of reliable and up-to-date representation of the user, i.e., their interests, preferences, and needs, as encapsulated by the user model [19].

The user models typically reflect the goals and domain of the personalized service. For example, an email filtering plugin should be able to distinguish between genuine senders and spammers, while a movie recommender should know what movie genres are liked and disliked by the user. Thus, no one-size-fits-all representation of the user model can be conceived and the target data is learned implicitly from observable user interactions with the system and other users.

2.3.2 Personalization via Online Tracking

As mentioned earlier, Internet users are being increasingly tracked and their personal data are extensively used in exchange for services. In the current era, when people use real identities to communicate on the web, maintaining privacy has became a complicated challenge. Service providers are using a variety of personal information to personalize the content. The privacy challenge becomes more important with the

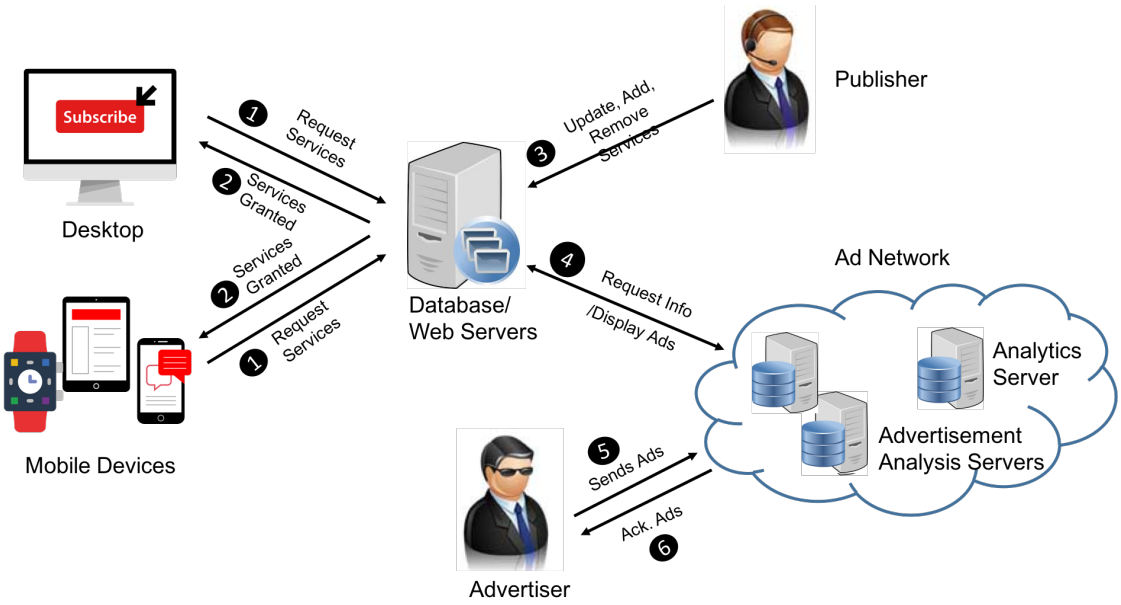


Figure 2.2: Eco-System of Advertisement Network

dissemination of smart phones and devices offering new possibilities for personalization. On the other hand, personalization algorithms and technologies are steadily improving, making behavioral profiling more powerful, yet raising a multitude of privacy challenges.

To understand the personalization system, Figure 2.2 shows an exemplary working diagram of an advertisement network system. In an advertisement network system, there are three main entities: the publisher, the advertiser, and the ad network. The publisher is an entity that owns a website or service; the advertiser is an entity that wants to advertise product to users; and the ad network actually collects advertisements from an advertiser and displays them on a publisher's website. If a user clicks on an advertisement, the ad network collects money from an advertiser and pays part of it to the publisher. It is thus important for the ad network to generate accurate and complete profiles of users, in order to increase the click chances and maximize the revenues. These three entities also exist in a mobile environment, where a mobile app acts as a publisher, while the roles of an advertiser and ad networks remain unchanged. It could be argued that, compared to desktop devices, mobile devices pose serious threats to privacy as many apps record user sensitive information like locations, movements, gestures, and more. Such an information is more helpful to ad network companies for generating more accurate user profile with the cost of privacy leaks for users.

2.3.3 Relationship

Personalization is hard to achieve without some loss of privacy, since users personal information is needed by a service provider to tailor or customize services. On the contrary, it has been argued that users are willing to share their personal interests or information in exchange for apparent benefits of using personalized products or services [45, 18]. To build trust, service providers ensure anonymity of their customers for the services usage, and in some cases, the anonymity is guaranteed for a lifetime. However, research shows that linking anonymized data to other databases with personally identifiable information allows (re)identification of a user [155]. Therefore, privacy risks are not just limited to a particular service provider, rather these risks are pervasive concerns where personal information provided by users to different services could be linked together to track/identify them ubiquitously.

Toch et al. [208] links privacy to three different personalization categories which are social, behaviour, and mobile web.

- In a **social-based personalization**, providing privacy is a major concern because of three main reasons: (i) users are willing to reveal more information, (ii) social networks compromise not only a single user's privacy but also their friend privacy, and (iii) social networks can reveal potentially embarrassing information. An example here is a case of 2008, where 8% of US companies fired 1000 workers because of information released on social networks [172]. Facebook Beacon advertising program [199] and Instant Personalization [125] are two major cases that appeared in 2007 and 2010 and faced major criticism from users, media, and lawsuits. Similarly, social search, personalized recommendation, and targeted advertising are the most widely used forms of social personalization [173].

- **Behavioural-based personalization** poses several privacy risks, such as unsolicited marketing. Another risk involves linking behavioural profiles to server-side user accounts, so that advertiseers can target users across different devices. Cross-system personalization is yet another, risk where personalized systems can use information from other systems to track users. For-example a car rental company's personalization system can exploit user's GPS location data to track their places of interests or driving patterns.

- **Mobile-based personalization** has increased with the spread of smartphones and phone sensors. With this, the ability of service providers to continuously track users has also grown. Sensor data has been used in various ways

for personalization. One way is the improvement of search results, such that search results displayed on a smartphone are tuned according to the user's location, highlighting nearby venues and services. Similarly, installation of various apps on mobile phones conveys user interests, helping app developers to show targeted ads. Ullah et al. [210] performed a measurement study of in-app advertisement and showed that GoogleAdMob has a higher proportion of targeted than generic ads.

Friedman et al. [86] discussed the risks associated with recommender systems. Authors mentioned that privacy breaches are either due to direct data access or due to data sharing with third-parties. In both cases, effects of privacy breaches can be significant, such as exposure of sensitive information, re-identification of anonymized data, leaks through shared device, or service inference by the recommender.

2.3.4 Balancing Personalization and Privacy

It is reasonable to expect that if user information is collected and treated fairly, they would be more inclined to share their personal data with service providers and use the personalized services. However, striking the balance between privacy and personalization is quite a challenge. In this section we discuss the technological measures that could be taken to minimize tracking via personalization.

- **Pseudonymous personalization** is a basic yet common approach to hide true user identity. It allows to use the same pseudonym across different sessions and to create or maintain more than one pseudonym. This helps users to separate different aspects of their online activity and control which service provider can access which persona [10, 99]. However, anonymity is difficult to maintain when payments or non-electronic services are involved. It has also been shown that hiding explicit identities like usernames and emails, is not sufficient to prevent tracking. There are cases where users have been identified through their anonymized data [155].

- **Client-side personalization** is another way to prevent online tracking. This type of privacy preservation implies data storage and subsequent personalization processes to take place on client-side [87]. Since data collection and processing occur at the client side rather than the server side, users may perceive more control over their data and lower privacy risks. However, the challenge with this approach is that existing personalization algorithms need to be redesigned to fit the client-side model [211].

- **Distribution, Aggregation or Privacy Preserving Techniques** have been proposed as the means to maintain user privacy in recommender systems. One strategy is to distribute user data across a set of machines; however, this solution aggravates personalization based on data of other users [37]. Another strategy is to use encrypted aggregation of user data [191]. As mentioned earlier, privacy preserving approaches like obfuscation or perturbation is another way to hide sensitive information while achieving a reasonable utility [20, 85].

- **User controls and feedback** is another way to preserve privacy in personalized systems. Kay et al. [123, 124] suggested adding scrutability to user modeling and personalized systems. The term 'scrutability' signifies the ability of users to understand and control what goes into their user models, what parts from their models are available to various services, and how the model is managed and maintained. This allows users to restrict service providers in accessing their sensitive data. However, achieving such a level of balance is currently challenging due to poor user understanding of these notions.

2.4 Conclusion

This chapter studies the background material to understand the research problem of this thesis by outlining the concept of online tracking specific to mobile and web technologies, as well as the more advanced behavioral tracking. Privacy implications of online tracking, highlighted by organizations and researchers, are also illustrated. Following this, the chapter ties the streams of personalization and tracking together and discusses various aspects of their relationships, including the currently deployed tracking methods for personalization. Lastly, this chapter discusses the ways to balance between personalization benefits and privacy concerns. This includes the state-of-the-art practices, current challenges, and practical recommendation for system developer willing to strike this balance. In the next chapter, we thoroughly discuss the literature work in the area of mobile and web privacy, particularly focusing on tracking issues and privacy preserving solutions.

Chapter 3

Related Work

In this chapter, we conduct our literature survey in two main directions. First, we consider the related work on tracking or fingerprinting mobile and web platforms. We then discuss studies focusing on third-party domains and the possibility of web tracking. Secondly, we present the existing literature on privacy-preserving solutions for the web and mobile data. We conclude this chapter by highlighting the limitations of existing literature and briefly discuss how this thesis sets out to bridge the existing gaps through the proposed contributions.

3.1 Existing Online Tracking Techniques

Fingerprinting techniques have been numerously studied by the research community. Perhaps, the pioneering work in the threat of tracking dates back to Sweeney, who showed for the first time that coarse-grained information such as birthday, gender, and ZIP code can uniquely identify a person [206]. This work was followed by several studies that provided measurement insights into web and device tracking. The success of such methods is a clear indication that anonymization techniques to protect the privacy of individuals may fail if the collected data contains unique combinations of attributes relating to specific individuals. In this section, we present the existing online tracking technologies and categorize them based on the tracking medium: web browser, mobile phones, or other devices.

3.1.1 Web Browser-based Tracking

In the past decade, several studies measured and analyzed web tracking. Krishnarmurthy et al. [130] provided an early insight into web tracking, followed by continual increase in third-party tracking techniques. Eckersley [70] quantified the uniqueness of web browsers based on user agent and/or the browser configuration (plugins,

fonts, cookies, screen resolution) and showed that 90% of browsers can be uniquely identified by user agent, cookies, time zone, plugins, and fonts. The algorithm was able to detect returning browsers, even if some features changed over time.

Following this, Yen et al. [223] quantified the amount of information revealed by host identifiers including IP addresses, cookies, and user login IDs. Authors used month-long datasets of a web-mail service and a search engine for the analyses. Further, they discussed the implications of cookie-churn on privacy and security, along with the utilization of host fingerprinting for improving security. An extended approach by Boda et al. [28] showed that cross-browser fingerprinting could achieve high uniqueness if enough data was collected by the operating system.

Olejnik et al. [170] performed a large-scale analysis of web browsing histories to track users. Their results were shown to detect 97% of browsers by inspecting only four web pages in the browser history. Akin to this, Laperdrix et al. [137] explored browser fingerprints validity by collecting more than 100K fingerprints composed of 17 attributes. Their results showed that HTML5 and Canvas API offer highly distinguishable features. Similarly, Fifield et al. [80] proposed a fingerprint technique based on the measurement of on-screen dimensions of font glyphs.

A crawler-based measurement study of online tracking at 1M websites was reported by Englehardt in [72]. The analysis was based on stateful (cookies) and stateless (fingerprinting) tracking, effect of browser privacy tools, and data exchange between different sites (cookie syncing). The authors developed an open source privacy measurement tool, which simplifies data collection for privacy studies on a scale of millions of websites. Similarly, Libert [146] studied the effect of third-party HTTP request on the top 1M websites and showed that Google can track across 80% of websites through third-party domains. It has been shown that 80-90% of browsers can be uniquely identified. Besides HTTP cookies, other entities such as Flash cookies, WebGL, and HTML5 were also used as a tracking medium [186, 161].

It is important to mention that a number of side channel and timing attacks have been launched on web browsers to leak the browser histories and cache information [218, 224]. These attacks can de-anonymize users in social networks, uncover user data, or reveal data to service providers or ad networks. Two different studies, [48] and [178], showed that usernames and online social profiles can uniquely identify user profiles and link users across different social platforms. In these works, fingerprinting was based on device configuration, device settings, and device hardware. Table 3.1 summarizes popular web-based tracking techniques.

Table 3.1: Summary of Existing Web-Based Online Tracking Techniques

Techniques Applied	Firefox Plugin and Extensions, Host Tracking Graph, Font metric-based fingerprinting, open-source web privacy measurement tool (OpenWPM), Markov Chain, mitmproxy
Tracking Metrics	Anonymity Sets, Entropy, Information Gain, Correlation, Normalized Shannon's entropy, Conditional Entropy, Information Surprisal
Datasets	Alexa Websites, Hotmail, Bing, Operating Systems, Facebook profiles, twitter profiles, Google, eBay, LDAP directory
Attributes	Cookies, JavaScripts, Uniquely Identified URLs, User Agent, HTTP ACCEPT headers, Cookies enabled?,Screen resolution, Timezone, Browser plugins, plugin versions and MIME types, System fonts, Partial supercookie test, IP address, ser login ID, locality, short user ID, timestamp, OS, basic fonts, all fonts, Content encoding, Content language, List of plugins, Use of local/session storage, color depth, List of HTTP headers, Platform, Do Not Track, Canvas, WebGL Vendor, WebGL Renderer, Use of an ad blocker, Unicode code points, tracker-owned cookies, site-owned cookie, HTML5 Local Storage, Flash LSOs, battery level, charge/discharge time readouts, gender, age, relationship status, Interested in, current city and current country, ToDataURL, fillText, strokeText, URL of the caller script, navigator, screen object
Accuracy	Precision, recall, false Positives, true Positives, Uniqueness, EnergyFull, Voltage

3.1.1.1 Online Tracking from Third-Parties on the Web

There has been a wealth of research into the utilization and exploitation of third-parties [109, 168, 138, 108, 203, 202]. Falahrastegar *et al.* [79] inspected the use of third-parties across top Alexa websites, exploring how third-party operators differ based on region. Nikiforakis *et al.* [168] demonstrated in 2012 that large proportions of websites rely on JavaScript libraries hosted on ill-maintained external web servers, making JavaScript exploits trivial. Lauinger *et al.* [138] led a further study, classifying sensitive libraries and the vulnerabilities caused by them. Similarly, Ikram et al. [109] proposed a machine-learning based tracking detection method that detects tracking JavaScript programs on the web. Their method used one-class machine learning classifier (SVM) that utilizes similarities between tracking JavaScript programs based on syntactic and semantic features. One key aspect of their work is the ability of a classifier to discover previously unseen tracking JavaScript programs. Further, Hozinger *et al.* [100] found 61 JavaScript exploits and empirically defined three main attack vectors. Gomer *et al.* [90] analysed users' exposure to tracking in the context of search, showing that 99.5% of users are tracked by popular trackers

within 30 clicks.

Bashir *et al.* [14] studied websites' resource inclusion trees and analyzed retargeted ads using crowdsourcing. This allowed them to identify and classify ad domains, as well as predominant cookie matching partners in the ad exchange environment. Our study is far broader, and sheds light on dependency chains across many different types of websites rather than simply inspecting advertisements. More related is Kumar *et al.* [132], who characterized websites' resource dependencies on third-party services. They found that dependency chains are widespread. This means, for example, that 55% of websites are prevented from fully migrating to HTTPS by their dependencies. Their focus was not, however, on identifying suspicious or malicious activities.

3.1.2 Mobile-based Tracking

Mobile device fingerprinting, on the other hand, is a recent but emerging concern. In general, the aforementioned techniques for browser fingerprinting can also be used for mobile tracking. However, studies revealed that mobile browsers do not have such distinguishable features as plugins, and fonts [70]. Thus, several studies proposed alternative methods to fingerprint mobile devices. These techniques utilize different physical characteristics of a mobile device, e.g., camera, sensors, microphones, and speakers. For instance, a study by Kurtz [134] focused on device configurations. Dey et al. [64] used the vibration motor to develop accelerometer fingerprints, and then applied machine learning to extract features.

A study by Das et al. [57] proposed a fingerprint mechanism to uniquely identify smartphones based on motion sensors (accelerometer and gyroscope) and inaudible audio stimulation, along with a mechanism to obfuscate the fingerprints by calibrating sensors. Noise-based sensor fingerprinting for mobile devices has also been discussed in [55, 56, 228], which focused on acoustic components such as speakers, microphones, or cameras. These techniques require access to microphone which needs a separate permission. Bojinov [29] utilized the noisy nature of hardware sensors such as accelerometer and microphones. Similarly, images taken by mobile phone camera were investigated to derive a noise pattern that is considered to be different in each device sensor [51, 148].

A work in [157] focused on mobile users identification and tracking based on touch-based gestures. A number of studies have also focused on privacy-preserved online behavior targeting for various purposes, including advertising [84, 209]. Another work by Spooren et al. [200] analyzed 59 mobile device fingerprints and concluded

that "the fingerprints taken from mobile devices are far from unique". However, they did not consider canvas test for fingerprinting. A study by Simon et al. [198] presented a new side-channel attack against smartphone keyboards that support gesture typing. They identified returning users with 97% accuracy using a set of 35 sentences and the system also correctly predicted sentences.

A number of studies have focused on identifying mobile user traits and characteristics using the information provided by mobile SDKs to third party-apps, such as the running apps, device model, operating system, and so on [195]. Kurtz [134] showed that mobile devices can be tracked through personalized configurations (installed apps, top 50 songs, device, Wifi name, etc) without involving hardware identifiers such as Unique Device Identifier (UDID), International Mobile Station Equipment Identity (IMEI), and others. Although these approaches affect user privacy, they are not directly related to our objective of identifying user based on behavioural biometrics. A work by Zhao et al. [227] showed the existence of a diverse set of mobile users using clustering and feature ranking. Their results identified 382 categories of users based on their app usage patterns. Table 3.2 summarizes popular mobile tracking techniques.

3.1.3 Other Tracking Techniques

Device or host fingerprinting tracks users or devices based on device properties, such as hardware or configuration information. One of the prominent works on remote device fingerprinting was presented by Kohno [129] and proposed a method to measure device clock skew using ICMP and TCP traffic.

Some works also deal with remote fingerprinting based on wireless traffic; for example, radiometric analysis of 802.11 transmitters [174], signal phase identification of Bluetooth transmitters [92], or timing analysis of 802.11 probe request frames [63]. For example, [174] utilized manufacturing defects in hardware to identify the device and, by association, the end user. Many efforts on tracking wireless devices focused on other hardware characteristics, such as radio frequency and drivers [166, 3]. While these techniques can also be used to identify smartphones, on the other hand, these calculations are resource intensive and require user cooperation. In addition, identifiers such as network names and IP addresses also help in host fingerprinting [174].

Table 3.2: Summary of Existing Mobile-Based Online Tracking Techniques

Data Collection Methodology	Online Mobile Data Collection App at Playstore, Public Available Datasets Collection, Static Data Collection Test Environment (all test users and test phones have same data collection environment), Specific Test Mobile Data collection, Public Available Websites for Data Collection
Tracking Metrics	Jaccard Similarity Coefficients, Machine Learning Classifiers (Support Vector Machine (SVM), Naive-Bayes classifier, Multiclass Decision Tree, k-Nearest Neighbor (k-NN), Quadratic Discriminant Analysis classifier, Bagged Decision Trees, Gaussian mixture models (GMMs)), Euclidean Distance, Cosine Distance, Entropy, L2 Distance, Maximum-Likelihood Estimation, Correlation, Relative Information Gain, Information Gain, Recurrent Neural Network, K-Means Clustering
Datasets	Data from Mobile Phones (Motorola Droid, Apple, Samsung, Nokia, Black Berry, LG, HTC, Nexus, Smart Chips) - Personalized Phone Settings (e.g. installed apps, settings), Motion Sensors Data (Accelerometer, Gyroscope, Magnetometer), Touch-Sensors Data, Speaker/Microphones Audio Data, Ambient light Data, GPS, Camera)
Attributes	IMEI, Wifi Mac Address, Unique Device Identifier (UDID), WiFi MAC Address, Closed Captioning Enabled, Guided Access Enabled, In-app Purchases Allowed, Inverted Colors Enabled, Mono Audio Enabled, Twitter Set-up, VoiceOver Enabled, VoIP Allowed, Jailbreak, Carrier Name, Internet Connection Type, Current ISP, Current Public IP, Device Country, Device Language, Device Model, Device Name, iOS Version, Installed Apps (Icon Cache), Installed Apps (URL Schemes), Installed Keyboards, Top 50 Songs, WiFi SSID, Calendar Names, Contacts, Photo Album Titles, Reminder List Names, Twitter Account Name, IMSI, Frequency –based Features (Spectral Centroid, Spectral Spread, Spectral Skewness, Spectral Kurtosis, Spectral Entropy, Spectral Flatness, Spectral Brightness, Spectral Rolloff, Spectral Roughness, Spectral Irregularity, Spectral RMS, Low-Energy-Rate, Spectral Flux, Spectral Attack Time, Spectral Attack Slop), Time-based Features (Mean, Std-Dev, Average Deviation, Skewness, Kurtosis, RMS Amplitude, Lowest Value, Highest Value, Percentiles, Min, Max, ZCR , Non-negative Count), Audio Features (Spectral Flatness, MFCCs, Chromagram, Tonal Centroid), Photo-response Non-uniformity Noise (PRNU), Touch-based Features (Raw Area, Pressure, X and Y Coordiantes, Derived Statistical Features such as Max, Min, Average, Percentile)
Accuracy	Fingerprint Stability, Discrimination Accuracy, Re-identification Accuracy, Precision, Recall, Average Precision, Average Recall, Accuracy, ROC, F-Score, True Negative, False Negative, Confusion Matrix, False Acceptance

3.2 Privacy Preserving Solutions for Web and Mobile Data

In this section, we discuss the extant privacy preserving techniques or mechanisms proposed for the web and mobile data. It is to be noted that all the below-mentioned solutions offer protection against virtual tracking/identification of users. For instance, TrackMeNot (TMN) [103] and PRivAcy model for the Web (PRAW) [196] prevent web browser tracking. Similarly, solutions proposed for mobile environments are only confined to protect device rather than the actual user. In order to prevent actual user being tracked, it is first necessary to identify the patterns or trends that are common in user behaviour and then propose a solution that does not lead to user tracking whilst maintaining utility of the functionality.

3.2.1 Privacy Preserving Solutions for Web Data

To counter the privacy risks in user web search histories, several privacy preserving methods have been proposed which can be categorized based on their deployment, the system-centirc, network-centric, and user-centric. In a system-centric approach, several privacy preserving search engines have been developed that allow users to obtain the results of their searches without revealing their search queries or their search activities to the search engine [39, 89, 95, 135]. In a network centric-solution an anonymous communication channel is used to hide the users' identities in order to make users' comments or queries non-linkable [66, 185]. However, features extracted from the users' web browsers allow linkability. Therefore, in a user-centric approach, the real queries or other web data are perturbed to reduce the linkability [70].

Most of the work in web search privacy is based on interleaving fake queries as noise to the queries of the user. Such noise addition techniques send fake queries either as inidividual search queries or modified user queries with fake keywords, such that the privacy of user's search queries is preserved. TrackMeNot (TMN) [103] is proposed as a Firefox plugin to randomly issue dummy queries from predefined Rich Site Summary (RSS) feeds. GooPIR is a standalone application for noise addition to Google queries [68], which modifies the user queries by adding dummy keywords, and then the search results are re-ranked locally based on the original user queries. PRivAcy model for the Web (PRAW) [196] is another technique, which continuously generates fake queries in different topics of interest of the user. This is done by generating user profiles from user queries and corresponding responses and thus the fake queries added will be in the general area of interest of the user in order to make

the distinction between real and fake queries difficult. Bloom cookies is yet another privacy preserving approach that encodes users' profiles in a compact space with privacy preserving Bloom filters to allow personalisation without being able to track users [160].

Murugesan et al. [163, 164] presented an obfuscation technique of plausibly deniable search (PDS), where a user's original query is combined with a set of unrelated queries, sharing the similar characteristics as the original query. It supports a notion of plausible deniability such that the probability of issuing an actual query is similar to the probability of issuing one of the (k to support k-plausible deniability) generated cover queries. The paper used Latent Semantic Indexing (LSI) based approach to generate fake queries and used DMOZ to show effectiveness of the approach. The approach emphasized that subsequent search queries are important in revealing user information; however they consider it as a future work of the PDS system.

Monedero et al. [184] formalized a model for an Optimized Query Forgery for Private Information Retrieval (OQF-PIR) that obfuscates user profiles and thus minimizes the privacy risk. The authors obfuscated user profile by making them up to average population profile and calculated Kullback-Leiber divergence between average population profile and the actual user profile. Similar to [184], Ye at al. [222] also proposed an obfuscation method that first finds an optimal number of dummy query distribution among finite set of categories and then injects these dummy queries as a noise into original user queries. Their approach is called Noise Injection for Search Privacy Protection (NISPP). In order to measure search privacy breach, NISPP utilized mutual information as a metric between user original queries and the diluted queries. Their results indicated that adding query noise reduced the privacy breaches to much extent.

Roca et al. [40] presented a protocol that distorted user profiles to the web search engine by enabling third parties to share queries with each other. Their results showed an affordable query delays and overhead in terms of computational and communications cost, whilst providing privacy benefits to the users. Al-Rfou et al. [6] analyzed TMN dummy queries using clustering algorithm and measured similarity between dummy queries with the set of recently issued queries by the user. Their analysis did not take into account any obfuscation mechanism rather it just computed similarity between queries. A drawback could be that fake queries having similarity ratio equal to original query fall into same cluster.

Cerqueus et al. [179] followed a hybrid approach by proposing PEAS (private, Efficient, and Accurate Web Search) system that provided an efficient unlinkability protocol to hide user identities (i.e. unlinking queries from the user), and an ac-

curate indistinguishability (unidentification) protocol to hide the difference between original and fake queries. The system showed promising results by decreasing 81.9% the number of queries linked to original user and retrieving upto 95.3% of accurate results through search engines.

A work by Xu et al. [221] demonstrated the ability of balancing between users' privacy and search quality. The algorithm first organized user personal information into hierarchical user profile, and ranked general terms to higher level than specific terms. The hierarchical profile enabled users to select the portion of their private information which should be exposed to server using a threshold. An additional privacy metric was estimated that indicated the amount of information revealed with the threshold value. The experimental results showed that user profiles are useful in improving search quality while exposing a small portion of private information about users.

Few studies have been conducted on obfuscation methods for other web data, such as social networks. Weinsberg et al. [219] studied the impact of obfuscation on the utility of recommendation systems with different classifiers. Salman et al. [189] and Li et al. [141] proposed methodologies to prevent inference attacks against published data by distorting data before making it publicly available while providing utility guarantee. A study by Chen et al. [47] investigated the effectiveness of different obfuscation strategies and policies for online social networks and proposed a novel obfuscation strategy based on the χ^2 feature selection metric without requiring knowledge about the classifier used by an adversary. The proposed strategy is able to significantly reduce the inference accuracy as validated by a set of experiments on Facebook dataset.

On the other hand, only limited works have considered quantifying privacy in web data. Peddinti et al. [175] evaluated the privacy guarantees offered by the TMN based on machine learning classifiers such as Logistic (Regression), Alternating Decision Trees (ADTree), and Random Forest. The study conducted on the AOL search logs and queries generated by the TMN software demonstrated that a search engine, equipped with a short history of user search queries, can bypass the obfuscation techniques with an average accuracy of 48.88%, and an average misclassification rate of 0.02% respectively.

Gervais et al. [88] also evaluated the query obfuscation techniques such as TMN and fake query generation, by learning the linkability between users' original and fake queries via machine learning algorithms. The paper proposed a quantitative framework that models attacks against web query obfuscation mechanisms and measures the privacy of user web search behaviour. The framework models user web

search behaviour, obfuscation mechanism, adversary knowledge about the obfuscation mechanism and the users' history of web searches, linkage function that captures and splits the fake and real queries, and a privacy metric to evaluate privacy at the query level and the semantic level.

Balsa et al. [13] performed qualitative analysis on six existing obfuscation techniques by investigating their privacy characteristics. The study provides insights into the deficiencies of existing solutions however, it did not analyze and compare the techniques quantitatively. The study also discussed a set of features that are required to avoid pitfalls of previous techniques while designing obfuscation based private web search techniques. Another study by Chow et al. [49] proposed two features that could be used to differentiate TMN dummy queries from real user queries.

A work by Biega et al. [23] studied quantifying privacy risk in web data by manually developing rules for sensitive key-value pairs and performing probabilistic calculation of the rules based on user's search history. Rule-based approaches are time-consuming as well as non-reliable for real-time risk prediction. A ranking-based Information Retrieval-centric approach to privacy risk evaluation in online communities is proposed by Biega et al. [24]. This approach uses ranking as a means of modelling a rational adversary who targets the most afflicted users. In [147], a framework for computing privacy scores of users in online social networks was proposed based on the sensitivity and visibility of a set of profile items. The proposed approach adapts the Item Response Theory (IRT) to calculate visibility and sensitivity of a set of profile items, such as real name, email, relationship status, and mobile phone number.

Murugesan et al. [164] studied inference attacks on web data and defined two qualitative metrics to measure the accuracy of web search obfuscation mechanisms against inference attacks, identifiability and linkability. Identifiability is referred to as determining a person who issued a query whereas linkability is referred to as associating a person to the query. Linkability in combination with identifiability is a serious privacy concern which not only determines a user but also identifies the interests and personal information of the user. In order to identify a user, the most naive method is to use parameters such as IP addresses, HTTP cookies, and client-side tools (for example, operating system and user name). Other methods are profiling users based on their queries, click-through, user preferences (for example, language and settings), and rich-client side (browsing history).

3.2.2 Privacy Preserving Solutions for Mobile Data

We find a number of techniques that formulate obfuscation problem as a min-max optimization and solve using adversarial networks. Raval et al. [183] proposed utility-aware obfuscation framework, OLYMPUS, that limits the risk of disclosing user sensitive information from sensors data such as images. The framework was based on generative adversarial networks (GANS), with a game between obfuscator and an attacker. Thus, given a training dataset and an access to target application (for utility), OLYMPUS learns an obfuscation mechanism that minimizes both privacy and utility losses. The framework was tested on different benchmark datasets and a real world handwriting recognition mobile application. Results indicate that OLYMPUS successfully protect user information without compromising much of application accuracy. However, the framework needs the data of other users to train the obfuscation mechanism, (which is assumed to be gathered from publically available datasets), and is applicable only to apps that are using machine learning classifier to output utility classification score. Moreover, their framework is not generalized as the Deep Neural Network (DNN) needs to be tuned for different types of applications. In addition, the framework is user-dependent where an input from a user is required to specify privacy and utility labels. Their threat model trusts a user/developer to hook up an application with a framework.

Malekzadeh et al. [152, 151, 150] proposed approaches to protect sensory data using autoencoders. However, their schemes only work when public and private data are clearly mentioned. Akin to [183], these approaches suffer from the same drawback i.e. tuning different parameters against each data type. In addition, their work is tested only on a specific data i.e. motion sensors, and did not discuss its applicability for different mobile data types. Shokri [197] proposed a methodology for designing optimal user-centric obfuscation mechanisms against adaptive inference attacks. The mechanism is based on Stackelberg game that minimizes utility losses under specified privacy constraints. This approach works well on certain data types such as location trajectories that are discretized however, it is not clear how this scheme works for continues data such as images, touch sensors, GPS locations etc.

A number of other adversarial approaches have also been proposed, e.g. [7] uses GANS to obscure the difference between the real and synthesized datasets. However, this approach does not offer privacy guarantees on the generated synthetic data. Moreover, this approach works best for the batch and offline processing of the data. Similarly, [104] also proposed utility aware privacy framework based on GANS. The purpose of this framework is also to generate synthetic data with

privacy and utility guarantees. AttriGuard [121] protects data from an attacker that attempts to infer private attributes. This adversarial scheme is based on the principle of transferability as it assumes that defeating a particular attacker should defeat other similar attackers as well. Akin to other literature, these approaches are only applicable for batch processing of the data and are not suitable for real-time sensor data types.

Das et al. [57] studied the feasibility of conducting sensor fingerprinting on mobile phones and also discussed countermeasures based on calibration and noise addition. They showed the effect of obfuscation sensor data on fingerprinting and utility. In another paper, Das et. al [58] proposed an obfuscation scheme to defeat motion sensors based fingerprinting. The results indicate that accuracy of fingerprinting device is reduced with no significant impact on the usability of motion sensors. In one of his recent work [54], Das et. al studied sensor APIs and the scripts that use such APIs in fingerprinting. He also evaluated the efficacy of current countermeasures in blocking such scripts. Erdogdu [74] proposed a privacy-utility aware framework for time series data using information theoretic approach. However, their scheme relies on previous data entries of a user to generate an obfuscated data at time t. In addition, the scheme does not clearly describe obfuscation steps and how their framework behaves with different sensor data types.

Regarding location-based privacy, a recent paper proposed a privacy-preserved mechanism for user location data while reducing utility loss [30]. Apart from focusing only on location data, this approach is only tested for a specific utility type i.e. recommended places. Similarly, Shokri et al. [26] tried to protect user trajectories by generating fake yet semantically real privacy-preserved location traces. The scheme was also tested against location inference attacks. However, this scheme considered a threat model where the whole batch of synthetic data is to be released in public and did not consider real-time privacy-preserving of a location data. A framework, PULP, has been proposed [41] that automatically selects location privacy-preserved mechanism (LPPM) based on user requirement of privacy and utility. The framework is tested on two LPPMs, GEO-I and Promesse [8, 182]. PULP also required offline profiling of LPPMs and investigated on specific privacy and utility metrics. Similarly, Bilogrevic et al. [25] proposed a methodology to design automatic personalized location privacy protection mechanisms. They studied the motivation of the user in sharing location information, and then predict the utility implications of a privacy-protection mechanism. Similar location privacy preserving mechanisms have been proposed in [98, 139].

3.3 Conclusion

In this chapter, first, we discuss the existing web and mobile tracking techniques. We observe that the existing literature is confined to fingerprinting virtual identities. For-instance, the browser fingerprinting techniques utilized characteristics such as plug-ins, fonts, caches, histories etc. to uniquely identify a browser. Likewise, keystroke dynamics-based tracking focuses either on keyboard-equipped devices or keystroke based gestures only. To the best of our knowledge, we do not find any tracking technique that utilizes user behavioural activities, such as swiping or writing on mobile devices or searching on the web, to reveal user identity. There is a significant need to investigate various behavioural-based tracking methods on the web and mobile platforms with the purpose to identify stealth-mode privacy leakages that can even track the physical identity of a person.

We also discuss the existing studies focusing on third-party web tracking and found that little work has been devoted to investigate the impact of indirect (implicit) loading of the web resources from third-parties. Moreover, these prior studies ignore the presence of dependency chains (cf. **Chapter 6**) and treat all third-parties as "equal", regardless of where they are loaded in the dependency chain. We also do not find any work in literature that analyzes the role of dependency chains in loading suspicious third-party content.

Next, we discuss the existing literature on privacy preserving mechanisms for web data. However, none of these works focused on measuring or predicting privacy risks at run-time when a user is actively participating in online web activities. The existing web privacy quantification methods are not comprehensive and generic to be applicable for different web data types for-instance, search queries, blogs, product reviews, and social network comments. In addition, no work has handled the user high-risk data entries at run-time i.e. predicting and then obfuscating the high risk data entries. Though, adversarial machine learning has been an active area of research, there is still a need to investigate this emerging technology in the context of web data obfuscation.

Finally, we provide a literature on mobile sensor data obfuscation mechanisms. There has been a continuous effort by researchers to provide efficient solutions to overcome user identifiability issues emanating from mobile sensors data. However, some of these solutions are not considered suitable for continuously released of mobile sensory data whereas, some approaches are limited only to motion sensors, such as gyroscope and accelerometer, as they are based on calibration errors, i.e. offset and gain. Calibration errors do not hold true for other mobile data types such as GPS or

touch sensors. Few recent studies proposed deep neural network (DNN) based on-device data obfuscation. However, these mechanisms are either application specific i.e. developers must hookup the obfuscation mechanism with their apps to balance privacy and utility, or user-interactive i.e. user needs to train the learning mechanism and provide privacy and utility labels for classification. Hence, these limitations raise a concern of trusting application developers and users to appropriately implement obfuscation mechanism. In addition, few solutions need the data of other users to train the obfuscation mechanism, (which is assumed to be gathered from publically available datasets). Finally, these mechanisms focus on a single privacy aspect i.e. user distinguishability/identification and did not discuss the privacy notion of trackability across different sessions.

Chapter 4

Quantifying the Uniqueness of Touch Gestures for Tracking

In this chapter, we introduce a new privacy threat, 'touch-based tracking', which is induced by the collection and monitoring of touch gestures of mobile device users. We demonstrate the likelihood of touch-based tracking by focusing on touch gestures widely used to interact with touch devices such as swipes and taps. Our objective is to quantify and measure the information carried by touch-based gestures which may lead to tracking users. For this purpose, we develop an information theoretic method that measures the amount of information about users leaked by gestures when modelled as feature vectors. Our methodology allows us to evaluate the information leaked by individual features of gestures, samples of gestures, as well as samples of combinations of gestures. Through our purpose-built app, called TouchTrack, we gather gesture samples from 89 users, and demonstrate that touch gestures contain sufficient information to uniquely identify and track users.

In Section 4.1, we highlight the privacy problem of tracking users through touch gestures and briefly discuss our contributions. Section 4.2 covers our methodology of collecting data using TouchTrack app, and then presents the descriptive statistics about our dataset. Section 4.3 outlines our proposed probabilistic analytical framework in detail. In Section 4.4, we discuss the results on the amount of information conveyed by the users of our dataset for different touch gestures and their combinations. We summarize the results in Section 4.5. Finally, we conclude in Section 4.6.

4.1 Motivation

Touch gestures such as swipes, taps and keystrokes, are common modes of interaction with smart touchscreen-enabled devices, e.g., smartphones, smart watches and smart glasses. Major platforms including Android OS, iOS, watchOS and Android Wear provide a variety of APIs to help app developers detect gestures aiming to enhance the quality of experience of apps. Access to these APIs allows apps to collect raw gesture data from different sensors available on the smart device. The fine-grained nature of this data means that there is potential of learning more about users than is perhaps necessary for the proper functioning of an app. Indeed one area where touch gestures have been exploited is continuous authentication through which users are authenticated by profiling their touch behaviour [27, 83].

In this chapter, we argue and verify that touch gestures constitute a privacy threat as they enable a new form of tracking of individuals, which we refer to as "touch-based tracking," which is the ability to continuously and surreptitiously track and distinguish users via their touch behaviour while they are interacting with their devices. As compared to "regular" tracking mechanisms, e.g., based on cookies, browser fingerprints, browser user agents, logins and IP addresses, several factors make touch-based tracking potentially riskier. First, while regular tracking tracks virtual identities such as online profiles [70, 171, 137], touch-based tracking has the potential to track and identify the actual (physical) person operating the device. It can distinguish and track multiple users accessing the same device. Second, touch-based tracking possesses the capability to *continuously* track users. Third, it also leads to cross-device tracking where the same user can potentially be traced on multiple mobile devices. Cross-device tracking introduces additional privacy and security risks, where user data can be collated and sent to advertising companies and third party entities to build user profiles based on their activities on smartphones, tablets, smart watches and various IoT devices. However, demonstrating this type of tracking requires a more generalized approach, e.g. to validate the stability of features across devices, which we leave as future work.

Not all use cases of touch-based tracking are negative. It can also be beneficial to users and service providers alike. For instance, the identification of multiple users using the same device may help in providing content more suitable for each of them. A child using his/her parent's smartphone can automatically have parental control enabled. Touch-based tracking could also bring commercial benefits to the user (e.g. displaying discounts and sales on the product of interest to the user). The reader might notice a link between touch-based tracking and touch-based continuous

authentication. There are major differences in the two notions. The latter verifies a claimed identity based on prior knowledge of the identity and former tracks users even without the knowledge of any disclosed identity.

The ubiquity of the touchscreen devices and the fact that most if not all data from touch events and/or other sensors can be extracted by any mobile app without requesting special permission makes touch-based tracking a serious privacy threat for users. This not only represents a valuable new source of information for analytics, tracking, and ad services but also for app developers who can (mis)use the information to track individuals on a single device or across multiple devices. The objective of this chapter is to quantify the amount of information carried by user's touch gestures and hence to evaluate their tracking capabilities. Our main contributions are summarised as follows.

- We investigate the potential of using touch-based gestures for tracking, which we call touch-based tracking. We quantify the amount of information contained in these gestures which could lead to user tracking. To the best of our knowledge, this is the first study considering the potential of touch gestures to profile users. Our work complements research on other forms of tracking such as through web browsers, host devices, and online social profiles by fingerprinting browser features, device configurations, and user attributes [70, 171, 178, 48, 55, 223, 129, 134, 170].

- We develop an **analytical framework** that measures the amount of identifying information (in bits and relative mutual information) contained in touch gestures, represented as feature vectors, at different levels of granularity. At the finest level, our framework quantifies the information carried by individual features, e.g., pressure on screen and area covered by the gesture. At the second level, our framework quantifies the information carried by a gesture sample, e.g., a single swipe. At the third level, our framework calculates the amount of information carried by multiple samples of the same gesture, e.g., a collection of swipes. Lastly, we measure the information carried by a collection of samples from multiple gestures, e.g., swipes and taps. We apply our framework on four widely used touch screen gestures: i) swipes, ii) taps, iii) keystrokes, and iv) handwriting, and four sub-categories of swipe: i) up swipe, ii) down swipe, iii) left swipe, and iv) right swipe. The framework is generic enough to apply to any behavioural biometric modality which can be expressed as feature vectors.

- We **develop and deploy a game-like app** called "TouchTrack" for Android powered devices. It consists of three well known open source games: 2048 (for swipes),[1] Lexica (for taps),[2] Logo Maniac (for keystrokes),[3] and one custom built app for handwriting. These games were selected to capture touch gestures in a natural way. Through our TouchTrack app the user can check the uniqueness and tracking potential of his/her gestures.

- Using our TouchTrack app, we carry out a user study comprised of 89 participants and gathered a total of 40,600 samples of touch gestures. For each touch gesture, we **identify features that contain high amount of identifying information** using the *maximum-relevancy minimum-redundancy* (mRMR) algorithm [177]. The algorithm attempts to constrain features to a subset which are mutually dissimilar to each other, but similar to the classification variable, which in our case was the set of users. We give details in Section 4.4.2. We find that the most revealing features were the 80th percentile of area from left swipes, the 20th percentile of area and the 50th percentile of pressure from downward swipes which yielded 56.1%, 55.50% and 46.13% of information, respectively.

- With the same dataset, we **measure the amount of information contained in samples from the same gesture and from multiple gestures**.[4] We find that 50 features in a single handwriting sample contribute 68.71% of information about users, which increases to 73.7% with multiple samples. We further identify that two or three different gestures combined together reveal more information about users. For instance swipes, handwriting, and keystrokes carry 98.5%, while handwriting, taps, and swipes disclose 95.1% of information. Among users who performed all the four gestures, our framework revealed 98.89% of information about users.

- Finally, we also **validate our framework in terms of correctly identifying a returning user**. This is important since the same user might have two different samples from the same gesture that could be mutually dissimilar (thus showing high uniqueness) but will not result in identifying the user. We measure the true positive and false positive rates (TPR and FPR) of our method. We define TPR as the rate at which a unique user (in our set of

[1] https://github.com/gabrielecirulli/2048
[2] https://github.com/lexica/lexica
[3] https://github.com/Luze26/LogoManiac
[4] We use the relative mutual information as our metric for identifying information. For details, see Section 4.3.

users) is identified as the *correct* user given a test sample (or set of samples). Likewise, FPR is defined as the rate at which the wrong user is identified as the target user or a set of more than one users is identified as the set of possible users given a test sample (or set of samples). We observe that with multiple samples, swipes and handwriting show a TPR of 90.0% and 91.0%, respectively. For a combination of gestures we find that swipes and handwriting combined together had a TPR of 93.75%. In terms of FPR, we find that swipes, handwriting, and keystrokes taken together had an FPR of only 0.8%.

4.2 Data Collection

To illustrate the potential of touch-based tracking, we developed and launched a purpose-built app named *TouchTrack* to capture gesture samples. We first give an overview of the TouchTrack app, followed by our data collection approach. We then briefly describe some statistics about our dataset.

4.2.1 Selection of Gestures

Our selection of gestures was based on how frequently they are performed by users of smartphones or other touchscreen devices. We narrowed our selection to swipes (including left, right, upward and downwards swipes, and the group of four taken together), taps, keystrokes and handwriting. Swipes and taps are by far the most frequent gestures on smartphone apps. Keystrokes are also frequently used for typing text messages or entering web addresses. Unlike tap, which could be performed at any point on the touch screen, a keystroke is restricted to tapping on the mobile keyboard. We therefore separated the two. Writing on the touchscreen using fingers is an important alternative input method on a smartphone. We did not include other less frequent gestures such as pinching (for zooming in or out).

4.2.2 The TouchTrack App

The purpose of TouchTrack is to collect gesture samples as raw readings from the touch sensors, send them to our server, and finally inform the users about the uniqueness of their gestures by displaying the results computed via our framework at the server. To keep the user interested in using our app, we decided to design it like a game. The app is made up of four games, three of them are based on popular open-source games and a fourth game was purposely developed by us. We selected

these four games so as to capture the user gestures in a most natural way. We briefly describe each game in Appendix A.1 along with the screenshots of the games.

When a new user uses TouchTrack, he/she is required to sign up using a unique username. The username together with the device ID is hashed and then stored in our database. This is done to ensure that gesture samples from different users are kept separate to establish the ground truth. This also helps us to identify returning users and devices. For matters of ease and privacy, we did not require the user to enter a password with the username. Even though our app sends data in HTTPs, we did not want to collect any additional sensitive data, such as passwords, from users. Once the user has played one or more games, the uniqueness of the corresponding gestures are computed through our quantitative framework (described in Section 4.3) and are shown to the user in both visual and tabular forms. Screen shots of results are shown in Figure A.2 of Appendix A.1. For user convenience, our app starts showing results after a single gesture. However, to get more accurate results, the user is encouraged to perform more gestures. We would like to remark that our results may still not be reflective of user touch behaviour in the real world, as displaying uniqueness results might encourage the user to change touch behaviour to avoid privacy leakage. Probable change in user behaviour due to feedback has been acknowledged before in the case of browser-based tracking [70]. We have made the TouchTrack app available on Google Play Store.

4.2.3 The Raw Dataset

To collect gesture samples, we invited participants through two means: via emails to targeted participants and via social networking platforms. At first we uploaded a closed testing version of TouchTrack on Google Play Store and invited colleagues and acquaintances via email to install our app. A total of 25 participants responded to the first phase. In the second phase, we published the first version on Google Play Store and received a further 56 responses through our personal and social contacts. We received 8 responses from the users who installed our app without invitation. We also included them in our analysis. The app was available on Google Play Store for two months for data collection purposes. Once the data is collected, we start our analysis and results interpretation. The data collected from the 89 users served two purposes: to identify features most effective in fingerprinting users and to train our analytical framework to evaluate the uniqueness of gestures.

Table 4.1 shows the list of raw touch features gathered from the touch sensors of the devices used by the users across all gestures. By default, these features can be

Table 4.1: Raw Features

Raw Touch Features	X-Coordinate, Y-Coordinate, Finger Pressure, Finger Area, Screen Orientation, Finger Orientation, Stroke Time, X-Tilt , Y-Tilt

obtained from Android APIs without requiring any security permission. We used the `MotionEvent` Android API to detect and collect touch data. We did not use motion features for our analysis because we observed that certain motion sensors such as accelerometer, gyroscope and rotation vector did not produce any raw data in many phones, and returned a null value to the Android API.

4.2.4 Ethics Consideration

Prior to data collection, we underwent and obtained an ethics approval from the ethics board of our organization, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia. The board accesses all types of human related research, both within Australia and overseas and confirms if the data collection strategy complies with the national Statement on Ethical Conduct on Human Research and any relevant state and national legislation [167]. The users were informed about the purpose of TouchTrack and what data is being collected through the Participants Information Sheet (PIS) and privacy policy available within TouchTrack app. Appendix C contains a detailed overview of both sheets.

Throughout data collection, we did not attempt to obtain the real identities of the participants via, for instance, a linkage study. The data collected was not released publicly. No identifying information other than the user selected username and device ID was stored at our server side. Moreover, only the hash of the username and device ID were stored. The only other information stored at the server were the raw data from gestures from each username-device ID pair. A privacy disclaimer was displayed as soon as the app was launched by a participant providing the above details. The participant was allowed to opt-out. We informed users that their personal information will remain anonymous and took their consent beforehand.

4.2.5 Data Statistics

Table 4.2 shows the summary statistics of our data collection. The numbers are broken down into number of users, samples of gestures and features associated with each gesture. We collected a total of 40,600 gesture samples. Among these samples,

Table 4.2: Touch Gesture Data Statistics

Gesture	Number of Users	Number of Samples	Number of Features
Swipes	81	16611	229
Up Swipes	78	3568	229
Down Swipes	71	4781	229
Left Swipes	63	4252	229
Right Swipes	65	4010	229
Taps	89	16225	7
Keystrokes	49	6473	8
Handwriting	36	1291	241
All Gestures:	30	25186	
Total:	89	40600	

swipe and tap gestures had the most number of samples. There were a total of 89 users who downloaded and used our app; however, only 30 users used all four games and hence provided samples for all gestures. Our app was installed on 49 different smart phone models, with *Google Nexus 5* being used by 11 users and *Samsung Galaxy S7 Edge* by 8 users. Nine of the 11 users of Google Nexus 5 used our test smartphone to record gesture samples as they did not have an Android powered device. We could distinguish between users of the same device via their hashed user ID.

4.3 Methodology for Computing the Uniqueness of User Gestures

In this section we describe our methodology behind computing uniqueness of gestures. We begin with an overview, followed by notations and then the methodology in detail.

4.3.1 Overview

Recall that the purpose of calculating uniqueness is to demonstrate touch-based tracking. For this, we need to show (a) the uniqueness of gestures, (b) similarity of gestures from the same user. To do this, we first obtain gesture samples, i.e., series of raw data values captured from the sensors of the smart device from a set of users. Once these samples are collected we extract a set of salient features, thus representing each gesture by a feature vector. The set of selected features is the

topic of Section 4.4.1. For now we assume that each gesture is associated with a fixed set of features. Once we have populated our dataset with an initial list of users and gesture samples as instances of feature vectors, we then proceed to find the uniqueness of the gestures at different levels. At the smallest level, we assess the uniqueness of single features, by checking how many users exhibit a given 'test' feature value among the total users in the dataset. At the next level we assess the uniqueness of a feature vector, i.e., a gesture sample, by checking how many users in the dataset are likely to exhibit a given test feature vector. Likewise, we do this for a collection of samples from the same gesture, and finally for the collection of samples from a set of different gestures. In what follows, we define an abstract representation of our dataset, and how we compute the uniqueness of gestures at multiple levels using this abstract dataset.

4.3.2 Background and Notations

We denote the set of users by $\mathbb{U}$, the set of gestures by $\mathbb{G}$, and the feature space by $\mathbb{F}$. We denote our dataset by D which is modelled as a multiset of rows. The columns of D are indexed by a $u \in \mathbb{U}$, followed by a $g \in \mathbb{G}$, a feature vector $\mathbf{f} \in \mathbb{F}$, and finally by an average feature vector $\overline{\mathbf{f}} \in \overline{\mathbb{F}}$. The average feature vector $\overline{\mathbf{f}}$ is the average of all feature vectors $\mathbf{f}$ under a gesture $g \in \mathbb{G}$ and a user $u \in \mathbb{U}$. The ith feature under $\mathbb{F}$ is denoted by $\mathbb{F}_i$. The dataset is illustrated in Table 4.3. We define a random variable U that takes on values from $\mathbb{U}$, a random variable G that takes on values of subsets of $\mathbb{G}$, and a random variable F_g that takes on values of subsets of feature vectors from the gesture g. When considering only a single gesture g, we shall drop the subscript and denote the random variable as F. We use the notation $\{a\}$ to indicate a set of cardinality more than 1, whose generic element is denoted by a. For instance, $\{\mathbf{f}\}$ is a set of two or more feature vectors. The random variable F can take feature values as well. Abusing notation, we will denote this by $F = f_i$, where f_i is the value of the ith feature. A predicate on a row from D denoted

$$(\mathbb{U} = u, \mathbb{G} = g, \mathbb{F}_1 = f_1, \mathbb{F}_2 = f_2, \ldots, \overline{\mathbb{F}} = \overline{\mathbf{f}}),$$

is the conjunction of clauses $(\mathbb{U} = u)$, $(\mathbb{G} = g)$, and so on. The predicate evaluates to 1 if a row satisfies each clause, and 0 otherwise. We can have possibly empty clauses. When considering a feature vector, we may simply use $\mathbb{F}$ to represent the conjunction of its constituent clauses. For example, the predicate

$$(\mathbb{U} = \text{Alice}, \mathbb{G} = \text{Swipe}, \mathbb{F} = (0.012, 0.567, *, *, \ldots, *)),$$

Table 4.3: Structure of the Dataset D.

$\mathbb{U}$	$\mathbb{G}$	$\mathbb{F}$			$\overline{\mathbb{F}}$
		$\mathbb{F}_1$	$\mathbb{F}_2$	$\cdots$	
Alice	Swipe	0.012	0.567	$\cdots$	$(0.021, 0.770, \cdots)$
		0.019	0.599	$\cdots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
Bob	Tap	0.023	0.608	$\cdots$	$(0.010, 0.660, \cdots)$
		0.024	0.499	$\cdots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

evaluates to 1 on the first row of Table 4.3, where '$*$' indicates that the corresponding feature values are not part of the predicate. A fuzzy predicate is a function that evaluates to 1, if the feature vector of a row is similar to the feature vector specified in the clauses according to a similarity metric. Fuzzy predicates are distinguished from predicates by replacing either the equality involving $\mathbb{F}$ or $\overline{\mathbb{F}}$ by $\approx$. For instance, the following is a fuzzy predicate

$$(\mathbb{U} = \text{Alice}, \mathbb{G} = \text{Swipe}, \mathbb{F} \approx (0.012, 0.567, \ldots, 0.314)).$$

We denote by $\#(\cdot)$ the number of rows in D satisfying the predicate (or fuzzy predicate). The entropy of the random variable U is defined as

$$\begin{aligned} H(U) &= -\sum_{u \in \mathbb{U}} \Pr(U = u) \log_2 \Pr(U = u) \\ &= -\sum_{u \in \mathbb{U}} \frac{1}{|\mathbb{U}|} \log_2 \frac{1}{|\mathbb{U}|} = \log_2 |\mathbb{U}|. \end{aligned}$$

This is the minimum number of bits of information required to distinguish each user in $\mathbb{U}$. The mutual information or information gain between U and a realization a of the random variable A is defined as

$$I(U; A = a) = H(U) - H(U \mid A = a),$$

where $H(U \mid A = a)$ is the conditional entropy given as

$$H(U \mid A = a) = -\sum_{u \in \mathbb{U}} \Pr(U = u \mid A = a) \times \log_2 \Pr(U = u \mid A = a). \tag{4.1}$$

Finally, the relative mutual information between a realization a of the random vari-

able A is defined as

$$I_R(U; A = a) = \frac{I(U; A = a)}{H(U)} = 1 - \frac{H(U \mid A = a)}{H(U)}. \tag{4.2}$$

The above measures the uniqueness of a realization of a random variable A through relative mutual information. To assess the uniqueness of all possible values the random variable A can take, we make use of the conditional entropy

$$H(U \mid A) = -\sum_{a \in A} \Pr(A = a) H(U \mid A = a) \tag{4.3}$$

Here, $\Pr(A = a)$ is calculated from the probability distribution of the random variable A. From this, the relative mutual information of the random variable A is

$$I_R(U; A) = 1 - \frac{H(U \mid A)}{H(U)}. \tag{4.4}$$

Note that while mutual information should suffice as a measure to assess uniqueness, our choice of relative mutual information is to account for the different number of users for different gestures, thus enabling us to compare results across gestures on the same scale. In what follows, we shall assess the uniqueness based on different realizations of the random variables G and F. This will be done by first calculating the conditional probabilities in Eq. 4.1 which are determined by predicates or fuzzy predicates, then computing the conditional entropy in Eq. 4.1, which then directly allows us to compute the relative mutual information in Eq. 4.2. A given realization is considered highly unique if the relative mutual information is close to 1. We will use percentages to represent the value of relative mutual information in the range $[0, 1]$ in the natural way. To assess the uniqueness of the random variables G and F in its entirety, we will make use of the relative mutual information defined by Eq. 4.4.

4.3.3 Measuring Uniqueness

We measure uniqueness based on a single feature value from a gesture sample, a single feature vector (i.e., a gesture sample), a set of feature vectors (i.e., a set of gesture samples), and finally a set of feature vectors corresponding to a set of gesture samples from multiple gestures. To measure uniqueness based on a single continuous feature, we first bin its values within discrete bins and then calculate the probability of a user producing the feature value within a bin. In contrast, to evaluate uniqueness of features vector(s), we do not bin the features, and instead

rely on fuzzy predicates. Our procedure to bin continuous features (for evaluating uniqueness of single features) is described below.

4.3.3.1 Binning Feature Values:

If a feature is continuous, then the probability that its corresponding random variable F has the value f is 0. In an actual implementation, continuous features are replaced by their floating point analogues. Still, the probability that the random variable exhibits the exact value f is negligibly small. This will result in our uniqueness based measure returning every feature value as unique (even from the same user). We therefore, bin features that are either continuous or have a large domain. Let σ denote standard deviation. Fix a gesture $g \in G$, let $n = \#(G = g)$ denote the number of samples of the feature. We use Scott's formula [193] to obtain the optimum bin size Δf as

$$\Delta f = \frac{3.49\sigma(F)}{n}.$$

Given this bin width the total number of bins are then

$$\left\lceil \frac{\max F - \min F}{\Delta f} \right\rceil.$$

Given a feature value f, its bin is calculated as

$$b = \left\lceil \frac{f - \min F}{\Delta f} \right\rceil$$

The feature value f is then converted to the feature value

$$\hat{f} = b\Delta f + \min F$$

The value $\hat{f}$ is then stored in the dataset D instead of f.

4.3.3.2 Uniqueness based on a Feature value

Given a feature value f_i corresponding to the ith feature of a gesture $g \in \mathbb{G}$, the uniqueness of the value is computed as follows. We first calculate the probability that a $u \in \mathbb{U}$ is likely to have produced this feature value. This probability is calculated by Eq 4.5.

$$\Pr(U = u \mid G = g, F = f_i) = \frac{\#(\mathbb{U} = u, \mathbb{G} = g, \mathbb{F}_i = f_i)}{\#(\mathbb{G} = g, \mathbb{F}_i = f_i)} \tag{4.5}$$

The conditional entropy in U given the feature value f_i of a sample of the gesture g is given by plugging the above conditional probability in Eq. 4.1 to obtain $H(U \mid G = g, F = f_i)$, from which the relative mutual information $I(U; G = g, F = f_i)$ can be obtained from Eq. 4.2.

Example 1. Suppose our dataset has $|\mathbb{U}| = 128$ users, giving us $H(U) = \log_2 |\mathbb{U}| = 7$ bits. Suppose now we are looking at the swipe gesture, and we are interested in the first feature having value $f_1 = 0.012$. Further suppose that out of the 128 users, only Alice and Bob have exhibited this value in the dataset, with Alice having exhibited it twice (corresponding to two different samples of swipe), and Bob only once. We have $\#(\mathbb{G} = \text{Swipe}, \mathbb{F}_1 = 0.012) = 3$, $\#(\mathbb{U} = \text{Alice}, \mathbb{G} = \text{Swipe}, \mathbb{F}_1 = 0.012) = 2$, and $\#(\mathbb{U} = \text{Bob}, \mathbb{G} = \text{Swipe}, \mathbb{F}_1 = 0.012) = 1$. Then $\Pr(U = \text{Alice} \mid G = \text{Swipe}, F = 0.012) = \frac{2}{3}$ and $\Pr(U = \text{Bob} \mid G = \text{Swipe}, F = 0.012) = \frac{1}{3}$. From this we get $H(U \mid G = \text{Swipe}, F = 0.012) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} \approx 0.92$ And finally, $I_R(U; G = \text{Swipe}, F = 0.012) = 0.8688$. We say that the feature value $f_1 = 0.012$ for the swipe gesture reveals 87% of information. □

To assess the uniqueness of the ith feature (and not just one particular feature value) we calculate the probability that the random variable F corresponding to the ith feature takes on the feature value f_i as

$$\Pr(F = f_i \mid G = g) = \frac{\#(\mathbb{G} = g, \mathbb{F}_i = f_i)}{\sum_{f \in F} \#(\mathbb{G} = g, \mathbb{F}_i = f)}. \tag{4.6}$$

That is we count all instances of the feature value f_i and divide it by the sum of all instances of feature values f in the range of F. By plugging this value and the result of conditional entropy of feature values in Eq. 4.3, we obtain the conditional entropy pertaining to F, from which we can compute the relative mutual information $I(U; F)$ from Eq. 4.4.

4.3.3.3 Uniqueness based on a Gesture Sample

To measure uniqueness of a gesture sample, we use the entire feature vector $\mathbf{f}$ corresponding to the gesture g, and check against all feature vectors of the user u. Due to high dimensionality of the feature vector, it is unlikely that any two feature vectors from the same user will be exactly the same. We therefore use the fuzzy predicate in this case, which relies on a similarity metric. We postpone our choice of the similarity metric to Section 4.3.4. The conditional probability is calculated

as

$$\Pr(U = u \mid G = g, F = \mathbf{f}) = \frac{\#(\mathbb{U} = u, \mathbb{G} = g, \mathbb{F} \approx \mathbf{f})}{\#(\mathbb{G} = g, \mathbb{F} \approx \mathbf{f})}, \tag{4.7}$$

From this probability we can then compute the conditional entropy and relative mutual information as before. Due to space limit we omit how the relative mutual information for the entire gesture is computed, which is similar to the case of the relative mutual information for a feature.

4.3.3.4 Uniqueness based on a Set of Gesture Samples

If we are given a set of feature vectors $\{\mathbf{f}\}$ from a gesture g, we first obtain the average vector $\overline{\mathbf{f}}$ from $\{\mathbf{f}\}$. Then, we compare this average vector against the average vector under $\overline{F}$ of the user $u \in U$ (for the same gesture). Given this, the probability that the set of gesture samples is from the user $u \in U$ is

$$\Pr(U = u \mid G = g, F = \{\mathbf{f}\}) = \frac{\#(\mathbb{U} = u, \mathbb{G} = g, \overline{\mathbb{F}} \approx \overline{\mathbf{f}})}{\#(\mathbb{G} = g, \overline{\mathbb{F}} \approx \overline{\mathbf{f}})} \tag{4.8}$$

Notice the use of fuzzy predicates. Given this probability, the conditional entropy and relative mutual information can be computed as before.

4.3.3.5 Uniqueness based on Multiple Gestures

Given a subset of gestures $\{g\}$ and their corresponding sets of feature vectors $\{\mathbf{f}_g\}$, we first obtain an average feature vector for each gesture, denoted $\overline{\mathbf{f}}_g$, and then count the number of rows in D that satisfy the product of the fuzzy predicates generated by the average feature vectors of the gestures involved. More specifically, the probability of the collection belonging to a user $u \in \mathbb{U}$ is calculated as

$$\Pr(U = u \mid G = \{g\}, \{F_g = \{\mathbf{f}_g\}\}) = \frac{\prod_g(\mathbb{U} = u, \mathbb{G} = g, \overline{\mathbb{F}} \approx \overline{\mathbf{f}}_g)}{\sum_{u' \in U} \left(\prod_g(\mathbb{U} = u', \mathbb{G} = g, \overline{\mathbb{F}} \approx \overline{\mathbf{f}}_g)\right)} \tag{4.9}$$

The symbol $\prod$ stands for product. In this case the product is over all gestures in $\{g\}$. For instance, if we have $\{\text{Swipe}, \text{Tap}\}$ as two gestures, then we are essentially checking if the product predicate

$$(\mathbb{G} = \text{Swipe}, \overline{\mathbb{F}} \approx \overline{\mathbf{f}}_{\text{Swipe}}) \times (\mathbb{G} = \text{Tap}, \overline{\mathbb{F}} \approx \overline{\mathbf{f}}_{\text{Tap}})$$

evaluates to 1, which is only possible if both the fuzzy predicates evaluate to 1 for a given user in D. We divide this by summing the same product predicate for all users in D. The conditional entropy and relative mutual information can be computed by plugging in the above conditional probability.

4.3.4 Calculating Fuzzy Predicates

To demonstrate how fuzzy predicates are evaluated, we use a generic feature vector $\mathbf{f}$ belonging to a gesture $g \in \mathbb{G}$. This is tested against a feature vector $\mathbf{f}'$ belonging to a row in D under $\mathbf{F}$ or $\overline{\mathbf{F}}$ (in case of the latter, we have an average feature vector). As mentioned before, evaluation of the fuzzy predicate is tied to a similarity metric. We chose the cosine similarity metric. Assume the length of $\mathbf{f}$ is m, then the cosine of the angle between $\mathbf{f}$ and $\mathbf{f}'$ is defined as

$$\cos(\mathbf{f}, \mathbf{f}') = \frac{\sum_{i=1}^{m} f_i f_i'}{\sqrt{\sum_{i=1}^{m} f_i^2}\sqrt{\sum_{i=1}^{m} f_i'^2}}, \tag{4.10}$$

which ranges between -1 and 1, the latter indicating complete similarity. Together with a threshold $\tau \in [-1, 1]$, the cosine similarity metric is then

$$s_{\cos}(\mathbf{f}, \mathbf{f}', \tau) = \begin{cases} 1, & \text{if} \cos(\mathbf{f}, \mathbf{f}') \geq \tau \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

Example 2. Given a row $(u', g', \mathbf{f}', \cdot)$ of the dataset D (ignoring the last column under $\overline{\mathbf{F}}$), the fuzzy predicate $(\mathbb{U} = u, \mathbb{G} = g, \mathbb{F} \approx \mathbf{f})$ evaluates to 1 if $u' = u$, $g' = g$ and $s_{\cos}(\mathbf{f}, \mathbf{f}', \tau) = 1$. □

The threshold τ is set by balancing the true positive rate (TPR) and the false positive rate (FPR), i.e., the value that returns the best equal error rate (EER). Details on this appear in Section 4.4.5.

4.4 Results

In this section, we present and discuss the results of applying our framework to show the uniqueness of gestures. Our goal is to show that touch gestures can be used to track users. For this, we need to show (a) that they are highly unique and (b) their ability to identify returning users. We first identify a set of features for each gesture. Then we rank the features in terms of their distinguishing capacity and finallyapply our methodology on the selected features to show uniqueness results.

4.4.1 Feature Identification and Extraction

From the raw features described in Table 4.1 (cf. Section 4.2.3), we derived more features to capture information such as averages, standard deviations, minimums and maximums. These derived features are called extracted features. A gesture sample generates a sequence of raw data points. The length of this sequence depends on the duration of the gesture and the *sampling rate* (usually around a millisecond), which is normally different across devices. This means that the sequences corresponding to two samples from the same gesture may not be of the same length. We therefore performed spline polynomial interpolation [59] to ensure the same number of data points (length of sequence) across samples from the same gesture. Since the sequences from different gestures are expected to be of different lengths, we did a separate interpolation for each gesture.

We identified a set of most commonly used features in literature (on gesture based authentication). We extracted 229 features for swipes, 7 for taps, 8 for keystrokes, and 241 for handwriting. Out of these, only 7 features are common across all gesture categories. These features are Inter-Stroke Time, Stroke Duration, Start X, Start Pressure, Start Y, Start Area, and Mid-Stroke Finger Orientation. Table A.2 in Appendix A.3 shows the list of these features. A few of the extracted features are Median of First 5 Acceleration Points, 80-percentile of pairwise X-Tilt, Std. Dev. of Pairwise Change of Area-Position, Direct End to End Direction, End to End X Distance, Median of Last 3 Velocities Points, 20-percentile pairwise Pressure etc.

4.4.2 Feature Subset Selection (FSS)

As a first step, we were interested in finding the uniqueness of gestures as a function of increasing number of features. To do this, we needed a ranking of features in terms of their distinguishing capacity.We use the maximum-relevance-minimal-redundancy (mRMR) algorithm that attempts to constrain features to a subset which are mutually as dissimilar to each other as possible, but as similar to the classification variable as possible [177]. In our case, the classification variable is the set $\mathbb{U}$ of users in our dataset. Given a set of m features $F_1, \ldots, F_m$ to find the highest rank feature, the mRMR algorithm finds the feature F_i that maximizes $I(U; F_i)$, where U takes on values from $\mathbb{U}$, and minimizes $I(F_i; F_j)$, where $j \neq i$. The mutual information $I(F_i; F_j)$ is calculated by using the joint distribution of features F_i and F_j through our dataset. Likewise, for more than one feature, the algorithm returns the subset of features that maximize, respectively minimize, the cumulative mutual information. A more detailed description of the algorithm is given below.

4.4.2.1 The mRMR Algorithm and Results

We intend to select features which have a potential to uniquely identify users based on touch gestures. In order to select the most distinguishing and non-redundant features from the given list, mRMR defines the subset of features that maximizes mutual information with the class label $I(F;C)$ and minimizes the information between $I(f_i;f_j)$[5] To apply mRMR, one must convert the features to discrete variables. We discretize features using bin's approach and use scott's rule to get equally spaced bins.

We input a list of features $S_{\text{tot}} = \{f_1, f_2, ...f_k\}$ of dimension k and a gesture g to mRMR algorithm, and receive a selected list of features F_{sel} with dimension m as an output, where $m \leq k$, and $F \subseteq S$. The subset F should produce high uniqueness and better classification accuracy compared to feature set S. The algorithm starts with an empty list and iteratively adds one feature at a time by keeping high relevancy and minimum redundancy. The relevancy is determined by measuring the mutual information of a feature with the class label $I(f_i;C)$ while redundancy is measured between features $I(f_i;f_j)$. The algorithm terminates when all features in S are exhausted. At the end, we obtain list of features F ranked merit-wise along with their MRMR values. In our case, we picked the features which were giving high relevancy, low redundancy, and where the improvement in mRMR values was not significant. The mRMR algorithm for finding the best subset of m features using forward selection strategy is formalized below.

Figure 4.1 shows resulting mRMR values corresponding to features of swipe, its sub-types, and handwriting. It is clear that mRMR values become a lot consistent after a certain range of features e.g. 170. Moreover, we observe a sharp decline in mRMR values after a set of 50 features which indicates that features have low relevancy to the class but high dependency to other features after a certain range. Also, note that x-axis starts with the second feature instead of first; the reason is that mRMR algorithm selects the first feature based only on the maximum relevancy between a feature and a class, as mentioned in Steps 4 to 7. While this could be a drawback of mRMR, we, however, validate selected features by applying our framework and also by using classification metrics.

[5]mRMR follows filter-based approach with entropy and information gain being an inherent part of feature selection.

[6]Incrementally select the jth feature from the remaining set $S - F_{sel}$ that maximizes $\varnothing(.)$ using the eq. $\mathbf{max}_{\mathbf{f_j \epsilon S - F_{sel}}}[I(f_{sel}, c) - \beta * I(f_{sel}, f_j)]$

Input: Set of all features $S_{\text{tot}} = \{f_1, f_2, ...f_k\}$, a gesture $\mathbf{g} \in \mathbb{G}$, and a Class $\mathbf{c}$.

Initialize $F_{\text{sel}} \leftarrow \varnothing$, $\beta \leftarrow \frac{1.0}{|S_{\text{tot}}|}$.
for $i = 1$ *to* $|S_{\text{tot}}|$ **do**
  Compute feature relevancy $I(f_i; c)$ using $I(S, c) = I(\{f_i, i = 1, 2, ...k\}; c)$
  Select feature which has the highest value i.e. $\max(I(S; c))$
if $F_{sel} == \varnothing$ **then**
  Append $F_{\text{sel}} \leftarrow F_{\text{sel}} + f_{max}$
  Set $f_{sel} \leftarrow \max(I(S; c))$
if $F_{sel} == |S_{\text{tot}}|$ **then**
  break
for $j = 1$ *to* $|S_{\text{tot}}|$ **do**
  if $f_j \notin F_{sel}$ **then**
    Compute $I(f_{sel}; f_j)$.
    Combine relevancy and redundancy as $\varnothing(f_j) = I(f_{sel}; c)$ - $\beta * I(f_{sel}; f_j)$.
    if $f_j \geq f_{j-1}$ **then**
      $f_{sel} \leftarrow f_j$ [6]
  Append $F_{\text{sel}} \leftarrow F_{\text{sel}} + f_{sel}$
Return F_{sel}

Algorithm 1: mRMR Feature Selection Algorithm

4.4.3 Effect of Number of Features on Uniqueness

In order to determine uniqueness of gestures as a function of features, we used sets of top i features from each gesture according to their mRMR rank, where i was incremented in discrete steps until m (the total number of features). We then evaluated their relative mutual information using our framework for the uniqueness of a single gesture sample (cf. Section 4.3.3.3) and multiple samples from the same gesture (cf. Section 4.3.3.4).

To do this we first partitioned the data from each user-gesture pair into two random but mutually exclusive sets. The first set had 80% of the gesture samples,

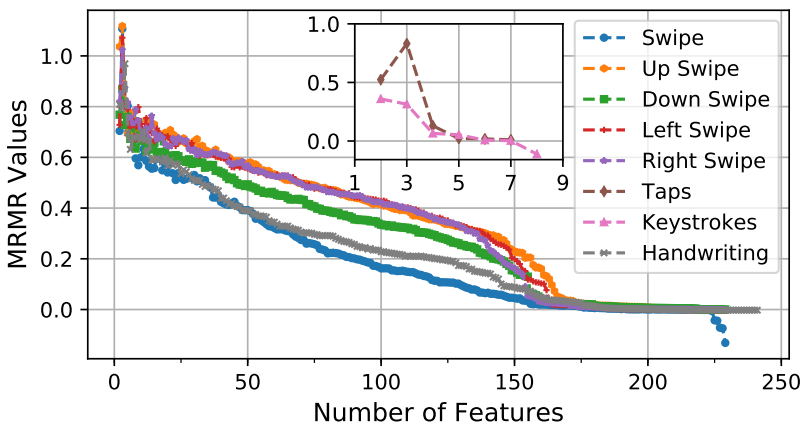


Figure 4.1: MRMR Results of all Swipes Types and Handwriting

and the second had the remaining 20% of the samples. The larger set was labelled as our dataset D, and samples from the 20% set were used for "testing." We shall call this approach the 80-20 approach throughout the rest of this chapter. We call the 20% set, the testing set. Thus, to evaluate our methodology, we select a sample from the testing set (fixing a user and a gesture), and then check against the dataset D.

Now to check the effect of an increasing number of features on the uniqueness of gestures, we selected top i features from the mRMR ranking for incremental values of i and then used the above mentioned 80-20 partitioning. We used an exhaustive approach, i.e., for testing single samples, we selected each sample in the the testing sets of all users, and then calculated the relative mutual information using Eq. 4.4. For testing a set of gesture samples, we used the entire 20% testing set as one set of gesture samples, and subsequently computed the relative mutual information. In our methodology, the relative mutual information for both these categories requires evaluating the fuzzy predicate, which in turn is determined by the threshold τ of the cosine similarity metric. For these results we set a universal threshold of $\tau = 0.8$, since we wanted to check the effect of mutual information keeping everything else fixed.

The outcome of this analysis is depicted in Table 4.4. We note that for all gestures, the relative mutual information increases with increasing number of features. Also, the uniqueness of a set of gesture samples is generally higher than single samples, and in all cases surpasses the uniqueness of single samples as we increase the number of features. The uniqueness of multiple swipe samples is the highest, with 92.01% (highlighted green with *), followed by handwriting (85.93%) and downward swipes (77.52%). On the other hand, samples of taps and keystrokes exhibit least uniqueness carrying 34.73% and 41.02% of information. This may also be due to the low number of features identified for these gestures. We observe that given a single gesture sample, handwriting provides 79.49% (highlighted green with *) of information about the user and a keystroke gives the least amount of information i.e. 28.76%.

The above analysis does not take into account the true positive rate (TPR) of the uniqueness measurement. Given a test sample from a user $u \in \mathbb{U}$, we mark it as a true positive if the corresponding predicate only evaluates to 1 on the user u. Otherwise, we mark it as a false positive. Note that this means that if the predicate evaluates to 1 for more than one user, then we consider it as a false positive even if it evaluated to 1 on the user u. The TPR and FPR are then evaluated over all possible test samples. Table 4.5 shows the TPRs and FPRs for different sets

Table 4.4: Relative Mutual Information for a varying set of features. Green cells with * indicate highest relative mutual information for a gesture sample and a set of gesture samples. Blue highlighted rows indicate our final selection of features.

Gesture	# of Features	Rel. Mutual Information of		Gesture	# of Features	Rel. Mutual Information of	
		Gesture Sample	Set of Gesture Samples			Gesture Sample	Set of Gesture Samples
Swipe (Expected IS: 6.32 bits)	15	43.23%	40.09%	Left Swipe (Expected IS: 5.97 bits)	15	47.71%	52.89%
	20	45.53%	41.91%		20	50.59%	55.59%
	25	46.09%	45.40%		25	50.71%	56.85%
	30	48.18%	45.60%		30	52.38%	60.03%
	50	57.79%	63.39%		50	53.96%	68.66%
	75	61.39%	75.34%		75	57.96%	70.62%
	100	61.87%	83.28%		100	59.82%	71.11%
	150	62.88%	88.50%		150	62.64%	71.55%
	200	63.13%	91.23%		200	65.52%	74.23%
	229	64.10%	92.01%*		229	65.77%	74.68%
Up Swipe (Expected IS: 6.28 bits)	15	45.93%	43.30%	Right Swipe (Expected IS:6.02 bits)	15	48.29%	53.71%
	20	48.05%	46.39%		20	50.03%	54.14%
	25	48.26%	46.59%		25	50.44%	56.00%
	30	48.56%	47.43%		30	51.24%	56.19%
	50	49.02%	50.23%		50	52.27%	57.48%
	75	55.09%	63.81%		75	55.59%	62.62%
	100	58.68%	68.79%		100	56.58%	65.35%
	150	58.74%	69.22%		150	57.11%	65.88%
	200	61.53%	71.94%		200	59.88%	67.48%
	229	61.68%	73.11%		229	60.12%	67.65%
Down Swipe (Expected IS: 6.14 bits)	15	48.46%	46.85%	Handwriting (Expected IS:5.16 bits)	15	47.06%	52.24%
	20	51.44%	49.89%		20	49.35%	52.78%
	25	51.53%	51.17%		25	52.93%	55.94%
	30	51.60%	51.43%		30	55.57%	58.99%
	50	52.22%	54.58%		50	68.71%	73.72%
	75	58.33%	67.51%		75	72.09%	77.19%
	100	60.55%	70.05%		100	74.75%	79.34%
	150	62.33%	71.35%		150	78.08%	83.47%
	200	65.25%	75.59%		200	78.16%	85.36%
	229	65.51%	77.52%		241	79.49%*	85.93%
Keystroke (Expected IS:5.61 bits)	1	23.92%	17.83%	Tap Expected IS:6.47 bits)	1	24.80%	15.17%
	2	26.29%	20.00%		2	26.25%	25.23%
	3	26.62%	29.97%		3	27.85%	26.94%
	4	26.86%	30.49%		4	28.75%	33.25%
	5	26.86%	30.49%		5	29.48%	34.25%
	6	27.25%	36.70%		6	29.55%	34.44%
	7	27.34%	37.63%		7	29.58%	34.73%
	8	28.76%	41.02%				

of features corresponding to a single sample and multiple samples from a gesture. We found that TPR decreases with the increase in number of features. The most probable reason for this continual decrease is the variations in a user's own touch behaviour as the dimension of the feature space increases.

With only 15 features, the TPR is 89% or above for all gestures (with multiple samples). In terms of FPR rates, we see that keystrokes and taps have relatively high FPR rates (43% and 37%, respectively, for multiple samples). The FPR decreases as we increase the number of features. We selected first 50 features for handwriting, 50 for swipes (all four types), 8 for keystrokes, and 7 for taps (highlighted blue in Tables 4.4 & 4.5) for further analysis, as these presented a good balance between TPR and FPR. We also performed 10-fold cross-validation and splits such as 30-70, 40-60

Table 4.5: TPR and FPR of Gesture for a varying number of features. Green cells with * indicate highest TPR and low FPR for a gesture sample and a set of gesture samples. Blue highlighted rows indicates our final selection of features.

Gesture	# of	Gesture Sample		Set of Gesture Samples		Gesture	# of	Gesture Sample		Set of Gesture Samples	
	Features	TPR	FPR	TPR	FPR		Features	TPR	FPR	TPR	FPR
Swipe	15	0.67	0.15	0.92	0.20	Left Swipe	15	0.55	0.07	0.91	0.14
	20	0.67	0.14	0.94	0.20		20	0.53	0.07	0.90	0.12
	25	0.62	0.11	0.94	0.17		25	0.51	0.06	0.86	0.12
	30	0.57	0.11	0.92	0.16		30	0.46	0.04	0.86	0.11
	50	0.23	0.02	0.89	0.07		50	0.34	0.02	0.83	0.07
	75	0.14	0.009	0.89	0.03		75	0.33	0.02	0.81	0.07
	100	0.11	0.007	0.89	0.02		100	0.31	0.02	0.77	0.07
	150	0.10	0.007	0.84	0.01		150	0.30	0.02	0.78	0.07
	200	0.10	0.007	0.76	0.009		200	0.29	0.02	0.77	0.06
	229	0.10	0.006*	0.76	0.009*		229	0.29	0.02	0.75	0.06
Up Swipe	15	0.56	0.10	0.89	0.16	Right Swipe	15	0.55	0.08	0.93	0.13
	20	0.54	0.10	0.89	0.14		20	0.54	0.06	0.88	0.12
	25	0.52	0.09	0.88	0.14		25	0.53	0.06	0.86	0.12
	30	0.52	0.09	0.88	0.14		30	0.51	0.06	0.86	0.11
	50	0.51	0.08	0.85	0.13		50	0.51	0.05	0.85	0.10
	75	0.40	0.05	0.82	0.07		75	0.41	0.03	0.85	0.09
	100	0.37	0.05	0.79	0.05		100	0.41	0.03	0.85	0.08
	150	0.37	0.04	0.79	0.05		150	0.41	0.03	0.85	0.08
	200	0.35	0.04	0.77	0.05		200	0.39	0.03	0.83	0.08
	229	0.34	0.04	0.74	0.04		229	0.39*	0.03	0.83	0.08
Down Swipe	15	0.57	0.11	0.92	0.17	Handwriting	15	0.67	0.11	0.97	0.16
	20	0.53	0.09	0.90	0.14		20	0.64	0.10	0.94	0.16
	25	0.53	0.08	0.88	0.14		25	0.47	0.05	0.92	0.14
	30	0.53	0.08	0.87	0.14		30	0.47	0.04	0.89	0.13
	50	0.49	0.08	0.85	0.12		50	0.29	0.01	0.88	0.06
	75	0.40	0.04	0.85	0.07		75	0.24	0.01	0.82	0.05
	100	0.38	0.04	0.84	0.06		100	0.21	0.009	0.79	0.05
	150	0.34	0.03	0.84	0.05		150	0.15	0.006*	0.73	0.04
	200	0.32	0.03	0.82	0.04		200	0.14	0.006*	0.64	0.03
	229	0.32	0.03	0.77	0.04		241	0.13	0.006*	0.61	0.03
Keystroke	1	0.46	0.23	0.96	0.43	Tap	1	0.54	0.26	0.95	0.37
	2	0.44	0.21	0.96	0.40		2	0.53	0.22	0.94	0.35
	3	0.43	0.18	0.95	0.37		3	0.53	0.21	0.94	0.34
	4	0.41	0.17	0.93	0.36		4	0.32	0.10	0.85	0.27
	5	0.40	0.17	0.93	0.36		5	0.31	0.10	0.85	0.27
	6	0.22	0.09	0.83	0.32		6	0.31	0.10	0.85	0.26
	7	0.22	0.09	0.83	0.32		7	0.31	0.10	0.85	0.26
	8	0.18	0.08	0.79	0.27						

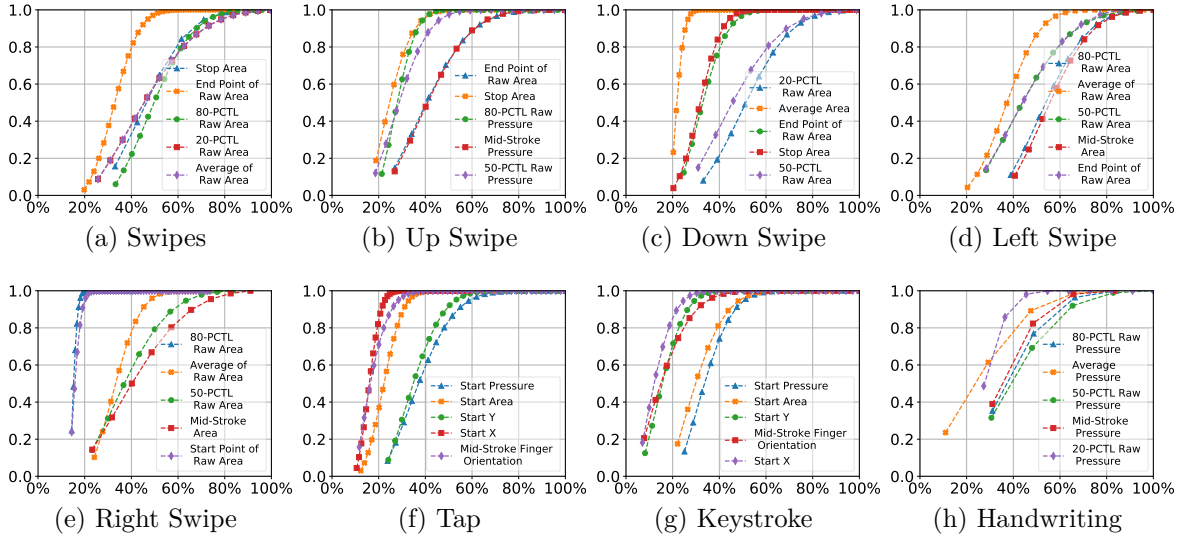
using Weka to evaluate the relative mutual information, TPR and FPR of a single gesture sample and multiple samples from the same gesture. The results gathered from these approaches were similar to 20-80 approach. We are thus mentioning results from 20-80 approach only.

4.4.4 Uniqueness of Individual Features

Before assessing the uniqueness of features we binned any continuous features or features with a large domain. See Section 4.3.3.1 for details. To assess the uniqueness of features, we again divided our dataset using the aforementioned 80-20 partition. Then, we exhaustively computed the relative mutual information defined in Eq. 4.4

as a measure of uniqueness for each feature value in the testing sets of all users.

We found that *80-percentile of area* in left swipe reveals 56.10% of information about a user, followed by *20-percentile of area* in down swipe 55.50%. Similarly, *50-percentile of pressure* yields 46.13% of information from down swipe. Among features which are shared among all gestures,*start area* contains 52.5% of information, followed by *start pressure* yielding 45.4% of information. On the other extreme, *inter-stroke time* for a keystroke reveals minimum amount of user information, i.e,. 7%. We observe no trend (in terms of dependency) among features,except that relative information decreases in decending order of the features.



(a) Swipes (b) Up Swipe (c) Down Swipe (d) Left Swipe

(e) Right Swipe (f) Tap (g) Keystroke (h) Handwriting

Figure 4.2: Cumulative Distribution Function (CDF) of Features. Y-axes represents fraction of the participant population and X-axes are Relative Mutual Information in percentage. The graph shows that Swipe, Left and Down Swipe reveals more than 50% of information for half of the population, respectively.

We also computed the cumulative distribution function (CDF) of the relative mutual information through Eq. 4.2 for a given feature. We present the CDF of top five features of every gesture in figure 4.2. It is evident that features corresponding to different statistics of *area* and *pressure*, e.g., average, percentiles etc., reveal the most information about a user, i.e., more than 40% of information for half of the users in our database. As before, we notice that all types of swipes and handwriting reveal the most information about users, and taps and keystrokes have relatively less information leakage about users.

4.4.5 Uniqueness of a Gesture Sample

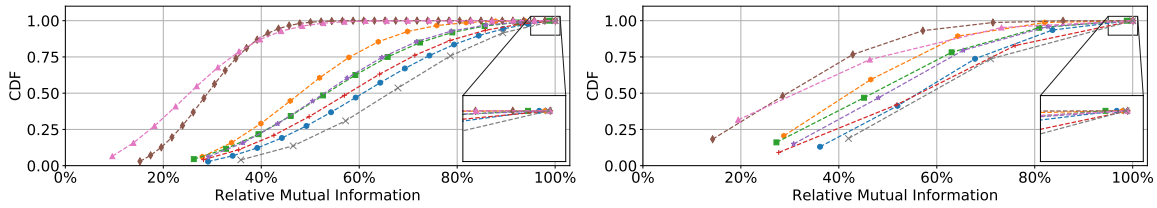
Recall from Section 4.3.3.3 that for gesture sample, we need to calculate the fuzzy predicate using the cosine similarity metric (cf. Section 4.3.4). Once we have fixed the set of features, and hence fixed the feature space, we need to find the threshold τ of the cosine similarity metric that balances uniqueness of gesture samples and correctly (and uniquely) identifying a returning user. To do this, we once again split the data into an 80-20 partition, and then evaluated the equal error rate (EER) (i.e., the rate at which 1 – TPR equals FPR) by varying the threshold. Table 4.6 shows the threshold that gave the lowest EER against each gesture. We can see that our methodology correctly re-identifies a returning user up to 81% (19% EER) of the time if given a handwriting sample. The worst performance is a TPR of 61% (39% EER) when a sample of keystroke is provided.

After fixing the threshold, we computed the uniqueness through our relative mutual information metric, i.e., Eq. 4.3. The results showed that a handwriting sample reveals the highest amount of information (68.71%), followed by swipes (57.77%). The four types of swipes, i.e., left, up, down, and right swipes, yield 53.9%, 52.2%, 52.2%, and 48.5% of information, respectively. However, taps and keystroke reveal only 29.5% and 26.2% of information, respectively.

Figure 4.3a shows the CDF of a gesture sample calculated for each of the gesture (through the relative mutual information metric of Eq. 4.2). We observe a considerable difference in the range of information revealed by different gestures, with handwriting exposing more than 60% of information for half of the users in the database. Following this, the swipes also show high uniqueness, revealing 30% to 65% of information about 75% of users. This suggests that handwriting and swipes carry highly identifiable information about users. The ROC of a gesture sample for all gesture types is shown in figure 4.4a.

Table 4.6: Thresholds of the cosine similarity metric for a gesture sample. τ = Threshold, EER = Equal Error Rate.

Gesture	τ	EER	Gesture	τ	EER
Swipe	0.38	22%	Up Swipe	0.55	27%
Down Swipe	0.52	28%	Left Swipe	0.48	22%
Right Swipe	0.58	22%	Handwriting	0.40	19%
Tap	0.29	35%	Keystroke	0.13	39%



(a) CDF of a Gesture Sample. Relative Information of Respective Categories are: -●- Swipe: 57.7%, -⬡- Up Swipe: 48.5%, -■- Down Swipe: 52.2%, -+- Left Swipe: 53.9%, -⋆- Right Swipe: 53.3%, -◆- Tap: 29.5%, -▲- Keystroke: 26.2%, -×- Handwriting: 68.7%

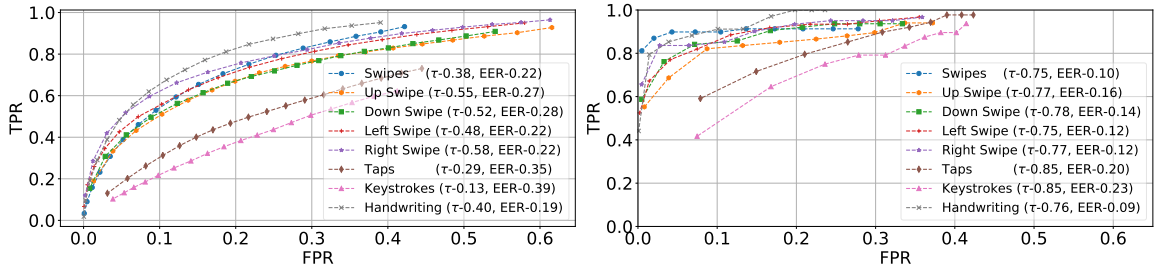
(b) CDF of Set of Gesture Samples. Relative Information of Respective Categories are: -●- Swipes: 63.3%, -⬡- Up Swipes: 50.23%, -■- Down Swipes: 54.5%, -+- Left Swipes: 68.6%, -⋆- Right Swipes: 57.4%, -◆- Taps: 34.7%, -▲- Keystrokes: 41.0%, -×- Handwriting: 73.7%

Figure 4.3: Cumulative Distribution Function (CDF) of Gesture Sample(s).

4.4.6 Uniqueness of a Set of Gesture Samples

We now consider the amount of information revealed by multiple samples of each gesture. We computed a different threshold of the cosine similarity metric for this category, and chose the one which resulted in the best EER. Table 4.7 shows the threshold and the corresponding EER values. Comparing this table to Table 4.6, we see that the rate of re-identifying a returning user is higher reaching up to 91% (9% EER) for handwriting. This means that combining a few samples of the same gesture may allow for more accurate tracking.

Based on the threshold values obtained, we then apply the cosine similarity metric on the dataset and calculate relative mutual information through Eq. 4.4. Once again handwriting reveals 73.7% of information, followed by left swipe which yields 68.6% of information of user gestures. In accordance with previous results, taps and keystrokes reveal minimum amount of information about users, i.e., 34.71% and 41.0%, respectively. Looking at the CDF of relative mutual information in Figure 4.3b, we can observe that swipes, its subtypes, and handwriting consistently perform



(a) ROC of a Gesture Sample

(b) ROC of Set of Gesture Samples

Figure 4.4: ROC of Gesture Sample(s).

Table 4.7: Thresholds of the cosine similarity metric for a set of gesture samples. τ = Threshold, EER = Equal Error Rate.

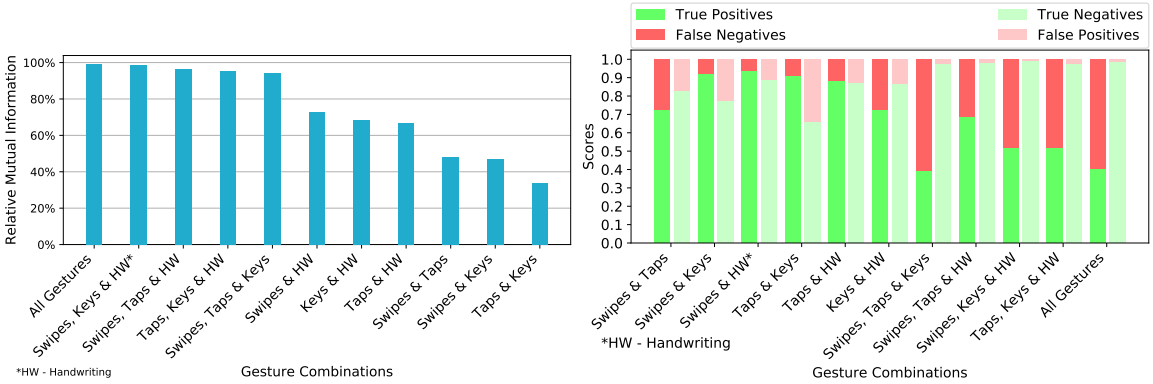
Gesture	τ	EER	Gesture	τ	EER
Swipe	0.75	10%	Up Swipe	0.77	16%
Down Swipe	0.78	14%	Left Swipe	0.75	12%
Right Swipe	0.77	12%	Handwriting	0.76	09%
Tap	0.85	20%	Keystroke	0.85	23%

better in revealing information than taps and keystrokes. As discussed earlier, the less information from keystrokes and taps can be due to the less number of features identified for these gestures. The ROC of a set of gesture samples for all gesture types is shown in figure 4.4b.

4.4.7 Uniqueness of Gesture Categories Combination

Next we consider multiple gestures in different combinations and measure their uniqueness through our methodology outlined in Section 4.3.3.5. Figure 4.5a shows the quantification results for different gesture combinations. We found that a combination of all gestures reveal a maximum of 98.89% of information about users, followed by the combination of swipes, handwriting & keystrokes that yield 98.5% of information. In contrast, the combination of taps and keystroke reveals minimum information, i.e., 33.5%. We would like to emphasise here that information revealed by the combination of various gestures is dependent on the number of users who had performed all gestures in the combination. This number was different for different gesture combinations. For example, the total number of users who performed taps was 89, whereas only 49 users submitted keystroke samples (cf. Table 4.2). Furthermore, the total number of users who had performed both taps and keystrokes were 45. This is one reason for our choice of the relative mutual information metric (as opposed to simple mutual information) which "normalises" the mutual information.

We also tested these gesture combinations in terms of re-identifying returning users. The thresholds for the cosine similarity metric for each gesture were as reported in the previous section (Table 4.7). Figure 4.5b shows the TPR and FPR of the different combinations of gestures. Since the threshold for the cosine similarity metric for each gesture was already set, the figure does not report EER as TPR and FPR are not balanced. For this reason, we also show the true negative and false negative rates. We see that as we increase the number of gestures in our combina-



(a) Uniqueness of Combination of Gestures (b) TPR and FPR for a Combination of Gestures

Figure 4.5: Results of Combining Gestures

tion, the FPR drastically decreases, but so does the TPR. For instance, all gestures together yield 0.99% FPR but also a low TPR (just above 40%). The lowest FPR was recorded by the combination of swipes, handwriting and keystrokes (0.85%). The main reason for a big drop in TPR as compared to the rate of single gestures, is mainly due to the rather strict metric of only labelling a given combination as being from a user if the predicate for each gesture evaluates to 1 (cf. Section 4.3.3.4). This can be changed by using, for instance, a majority rule.

We also investigate the impact of different users using the same device. We present our results in Appendix A.2.

4.5 Discussion

Our results reveal some important findings which we enlist below.

1. Multiple samples of a gesture taken together reveal more accurate information about a user than a single sample of a gesture. This means that tracking based on collecting larger number of user samples is likely to be more accurate. Having said that a single gesture sample or a single feature of a gesture also reveal enough information (upto 68% and 56%, respectively) so as significantly narrow down the set of possible users for tracking.

2. Swipes and handwriting carry more information as compared to taps and keystrokes. This is largely due to the rich set of information that can be derived as features from swipes and handwriting. In contrast, taps and keystrokes are simpler gestures from which only a few characteristic features can be derived.

3. Features based on the area and pressure of the finger performing the gesture are the most informative. This shows that there is significant inter-user variation in the area covered and the pressured exerted on the screen.

4. Overall, we show that tracking using touch gestures is highly feasible and accurate. This is demonstrated by the fact that we can correctly identify a returning user with a true positive rate of up to 91% with the handwriting samples. Though, TPR is a critical metric for an authentication system that evaluates the system's ability to correctly identify users. However, in case of tracking, a near 100% TPR is less critical, e.g., it may be acceptable for an advertising company to correctly track 90% of users and display related ads, whilst showing unrelated ads to 10%. This is indeed the case with our scheme, where 9% of users will be incorrectly classified (receive irrelevant ads). Still, our methodology would correctly classify 91% of the users to display relevant ads. In future, we plan to expand the framework to improve the classification results, and hence minimise the FPR.

5. Our data collection procedure did not impose any condition on how users needed to interact with their smartphones such as sitting, standing, and walking postures. Our results are still able to show high uniqueness and accurate re-identification of returning users.

Touch-based Tracking vs. Continuous/Implicit Authentication: The reader might confuse the notion of touch-based tracking with touch-based continuous or implicit authentication. Even though the main goal of this chapter is to quantify the amount of information carried by touch gestures and hence to evaluate the tracking capabilities using touch based features, in the following we would like to clarify some major differences between the two notions. Conceptually, the goal of authentication is to verify a claimed identity which assumes prior knowledge of the identity. The aim of touch-based tracking is to track users with or without the knowledge of any disclosed identity. Here we highlight further technical differences.

1. A typical continuous authentication scheme involves a classifier which is trained on the data of a target user.[7] This training data is gathered during a preceding registration phase. The classifier knows which training model to target for classification given a claimed identity. Touch-based tracking on the other hand is supposed to function without knowing the identity of the current user. This implies no registration phase and therefore the absence of

[7] Or a set of users using the same device, in which case each user has a separate training model. See for instance [220].

training models for target users. Thus, classification methods used for continuous authentication are not readily applicable for touch-based tracking.

2. Continuous authentication schemes require a certain number of samples before an authentication decision can be made with confidence. This is less of a stringent requirement on touch-based tracking, and tracking may proceed with even a single observation. The probable user set may be large, but it does not hinder tracking. Therefore, classification methods used in continuous authentication schemes are too restricted for use in touch-based tracking.

3. As a corollary to above, high classification rate, i.e., near 100% TPR and low FPR, is critical for the success of a continuous authentication scheme. In the case of tracking, a high TPR or misclassification rate is less critical, e.g., it may be acceptable for an advertising company to correctly track 90% of users and display related ads, whilst showing unrelated ads to 10%.

4. A final point is around measuring uniqueness. The goal of touch-based tracking is to illustrate how tracking is probable. This involves measuring how touch gestures, for instance, convey unique information. This needs to be measured at all levels: from single features to a collection of samples from multiple gestures. Our goal is to demonstrate how different granularity of information contained in gestures contribute to uniqueness and subsequent tracking of users. Some of this information does not lead to (successful) continuous authentication, e.g., uniqueness of single features. On the other hand, for touch-based tracking, this information is useful as it can be used in conjunction with other information (not necessarily from gestures) to more accurately track users.

In light of the above, we argue that touch-based tracking requires a different methodology from continuous authentication systems (to assess uniqueness of touch-based gestures).

4.6 Conclusion

In this chapter, we argue and verify that touch-based gestures on touchscreen devices enable the threat of a form of persistent and ubiquitous tracking which we call *touch-based tracking*. We proposed and developed an analytical framework that quantifies the amount of information carried by the user touch gestures mainly

swipes, keystrokes, taps, and handwriting. We quantify uniqueness at four different levels from a feature value to the combinations of gestures altogether. In addition, we also developed an android app, called TouchTrack, that collects users gesture data and provides real-time results about the uniqueness of their gestures. Our findings highlight that user touch gestures exhibit high uniqueness. Additionally, we also showed that returning users could be correctly re-identified with high accuracy, indicating that touch-based tracking is possible. In the next chapter we investigate the potential of tracking and identifying users through his/her online activities on the web. We demonstrate that privacy risk to track a user is high even when the data is anonymized.

Chapter 5

Quantification of Privacy Risks of Web Data

In the last chapter, we analyzed the potential of tracking users through their touch gestures on mobile devices. In this chapter, we demonstrate the likelihood of distinguishing users through their activities on the web. Users leave a trail of their personal data, interests, and intents while surfing or sharing information on the web. Web data could therefore reveal some private/sensitive information about users based on inference analysis. The identification of a user, through inference attack, is still possible even if the user sensitive data is encoded or removed. Despite having serious consequences, privacy risks concerning user behaviour on web platforms have not been investigated in the literature comprehensively. In this chapter, we use probabilistic methods to quantify privacy risks of web data that incorporates the three key privacy aspects, which are uniqueness, uniformity, and linkability of web data. Our results indicate that privacy risks to identify users are very high if a user enters 10 sensitive web entries. We measure the privacy risk associated with search queries and apps reviews and the results show that our risk quantification method is reliable enough to predict high risk web entries.

We organize this chapter as follows. In Section 5.1, we highlight privacy issues concerning user web data entries and briefly discuss about our contribution in this chapter. Section 5.2 presents the methodology that we propose for quantifying privacy risk of web users' data. In Section 5.3, we first present our datasets (Section 5.3.1), then experimental results (Section 5.3.2), and finally discussion summary (Section 5.3.3). In Section 5.4, we conclude our work.

5.1 Motivation

The wide-spread use of the web to search or share information online introduces various privacy and confidentiality threats. One such most persistent threat is users' identification and tracking via their web behavioral data [44, 201, 96]. Users unintentionally leave digital traces of their personal information, interests, and intents while using the online services, such as social networks, discussion forums, product reviews sites, and search engines, which could reveal sensitive information about them. The threat becomes more subtle when users are identified from anonymized datasets through inference analysis by an eavesdropper or a researcher who has access to the data.

To this end, we provide answer to a key question: *(1) What are the key features of web data privacy; and how to quantify privacy risks by considering these features?*. To answer this, we propose a quantitative method that predicts privacy risks of users' web data. The proposed method is later used to minimize privacy risks through obfuscation mechanism (detailed in Chapter 8 of this thesis.

Definition 5.1.1 (**Privacy Risk in Web Data**). We define privacy risk in (anonymized) web data as a risk of identifying users and thereby learning their sensitive/private information through *(1)* **uniqueness** (distinguishability) of the sequences of a user's web actions from other users' web actions, *(2)* **uniformity** (non-diversity) of the user in his web data, and *(3)* **linkability** of the user using his personal identifiable information (PII)[1] available in data.

A user's privacy is at a high risk when his web data is distinguishable from other users, has non-diversity in own data or actions, and is linkable to an individual with high confidence based on the user's PII. For example, if a user searches or comments regarding a certain disease, drug, pregnancy, or terrorism, the web history of the user could compromise privacy if the user's data is distinguishable, uniform for the user, and linkable to an individual based on PII available in previous search history. The main contributions of this chapter are as follows:

- We quantify users' privacy risk in web data using probabilistic methods, the Hidden Markov Model (HMM) that calculates probabilities of uniqueness, uniformity, and linkability learned from training data. The model is generic (applicable) to any web data, such as posts, shares, tweets, search queries, reviews, and clicks. Further, the model is dynamic in that the learned probabilities are

[1] Users often share or search for PII on the web including names, contact details, address/location details of people, and ego-surfing [23]).

updated with new data. To the best of our knowledge, no work has been done that allows such generic, comprehensive, and dynamic risk prediction in web data.

- We contribute a new large web dataset in the domain of online app reviews. We implemented a Google Play crawler that collects apps identifiers and apps meta-data by following a breadth-first-search approach. We retrieved 1,018,656 apps in a 4-week period of December 2016 and collected 16,335,480 reviews from 11,196,960 unique users. We will publish our dataset online for future research.

- We conduct an extensive empirical study using two real web datasets, the AOL dataset and our new app reviews dataset[2]. Our results indicate that privacy risk increases with sharing more data on the web. For the AOL dataset, we found that an average privacy risk reaches 100% when a user enters 10 queries. For app reviews dataset, we found that average privacy risk associated with just 1 sensitive review is 80.5%, which increases to 87.5% with 7 reviews.

5.2 The Methodology for Quantifying Privacy Risks of Web Data

In this section, we describe how users' privacy risk in web data can be predicted and measured using probabilistic methods. We begin with an overview, followed by the risk quantification.

5.2.1 Overview

Our aim is to develop a method to predict privacy risk of web data that comprehensively includes all key aspects of privacy and then obfuscate the high risk Web data using probabilistic methods. An overview of our proposed method is shown in Figure 5.1. The threat model we consider is the inference attack by a researcher or an eavesdropper who has access to anonymized (i.e., user identifiers are removed or encoded) web data as well as knowledge about our probabilistic model. The proposed method is generic and can be applicable to various applications of web data, such as online social networks, product reviews, forums, and professional networks.

[2]We contribute a new large web dataset in the domain of online app reviews by implementing a Google Play crawler that collects apps identifiers and apps meta-data.

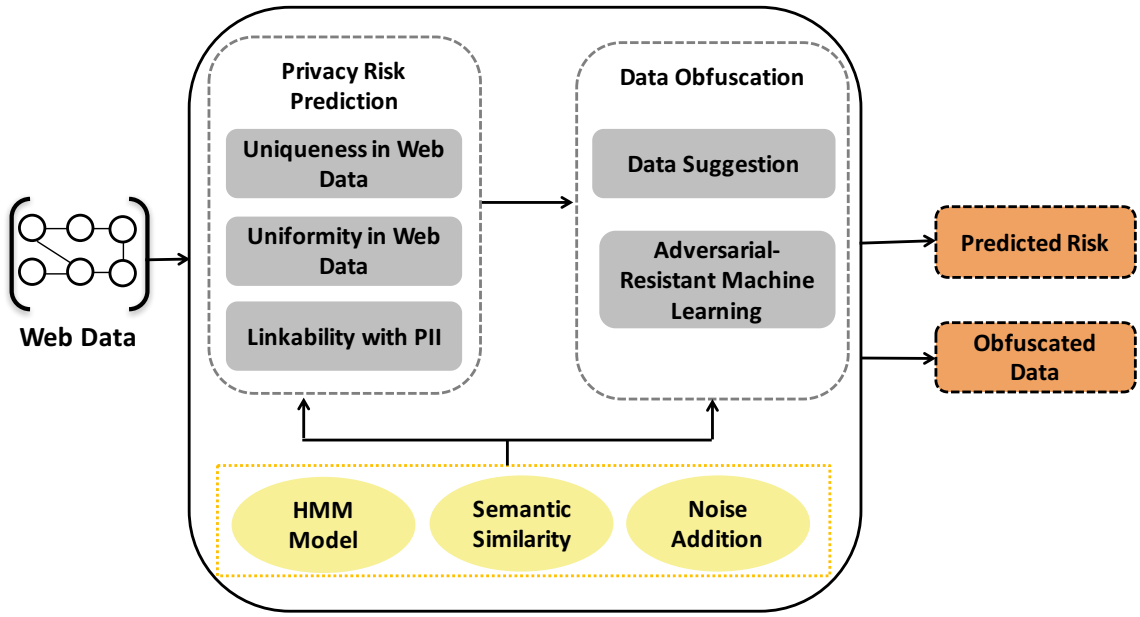


Figure 5.1: Overview of our Privacy-Aware Obfuscation Method for Web Data

The privacy risk (see Definition 5.1.1) of a user in the web data is determined by three key aspects: *(1)* uniqueness of the data, *(2)* uniformity of the user's data, and the *(3)* linkability of data to the user based on personal identifiable information (PII) available in the web data. The probability of uniqueness or distinguishability of a certain data or a sequence of data is measured as the non-likelihood of it by a user being similar to web data of other users such that it is unique or distinguished to reveal the user's identity. For example, if a user data contains 'Smith' it is less likely to be identifiable as it is a very common name in Australia, while data containing 'Dijith' (which is a less common name) is more likely to be identifiable (and therefore not anonymized). Similarly, if a user data contains a less common topic (e.g., a specific drug) it is more likely to be re-identified and the probability of distinguishability and linkability becomes even higher when the user's previous data contain personal information such as names and locations.

The probability of uniformness of a user based on the user's previous data (i.e., history) is measured as the likelihood the user has entered the data (and thereby interested in the data). The more the user has entered a certain data in previous history, the more confidence of the inference that the user is interested in this data. The joint probability of uniqueness and uniformity measures the probability of identifiability of the user in his web actions (i.e., inverse of privacy gain for the user). The probability of linkability of a user's data to an individual is based on how much PII available in the user's data. PII could reveal personal identity of a user and

therefore allows linking the corresponding data to the user. The overall privacy risk is measured as the joint probability of identifiability (uniqueness and uniformity) and linkability probabilities.

The probability of inference from a sequence of web data is often conditional probability on previous data and therefore the risk of inference becomes higher along with the user's sequence of web data (i.e., the probability of privacy preservation becomes lower with the sequence of user's data). The reason behind this intuition is that a user learns or reveals more with the sequence of web actions and therefore the data become more refined or specified to a certain topic enabling the web data sequence to being highly linkable (less anonymized) to an individual. Therefore, the inference probability becomes higher and the following web data/action by the user might be at an even higher risk of disclosure.

5.2.2 Risk Prediction

The aim of our risk prediction module is to measure users' risk of their web data being distinguishable, uniform and linkable as probabilities in a hidden Markov model (HMM). A user is represented by u_i and a data entered at a time t is represented by X_t. We train the HMM model using previous web data in order to predict a user's privacy risk of his web data entered at the current time being. HMM is a probabilistic model for representing probability distributions over sequences of observations. They are used in speech recognition systems, computational molecular biology applications, computer vision applications, and other applications of artificial intelligence and pattern recognition [101]. Assume a sequence of events (web data entered by a user) over time t as $X_1, X_2, \cdots, X_T$. These events satisfy the (first-order) Markov property, i.e., the current event X_t is independent of all the events prior to X_{t-1}. Each of these events X_t outputs observations Y_t which also satisfy the Markov property, i.e., X_t and Y_t are independent of the events and observations at all other time indices. These Markov properties state that the joint distribution of a sequence of events and their observations can be factored as:

$$p(X_{1:T}, Y_{1:T}) = p(X_1)P(Y_1|X_1)\prod_{t=2}^{T} p(X_t|X_{t-1})p(Y_t|X_t). \tag{5.1}$$

A web data entered by a user becomes a node and the probabilities of uniqueness, uniformity, and linkability are modelled in the HMM. Figure 5.2 and Figure 5.3 show examples of HMM trained for search queries related to PII and a sensitive topic cancer. The three probabilities modelled are:

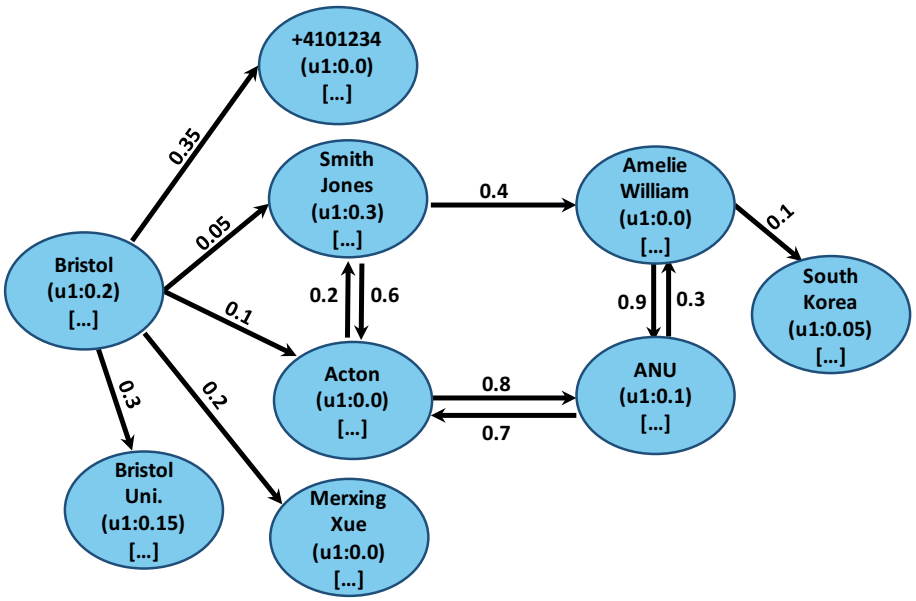


Figure 5.2: An Example of HMM model for PII topic in Web Search Data. Nodes are queries containing PII and edges between nodes represent the transition (conditional) probabilities. Each node contains observation probabilities for different users (in this example these probabilities are shown only for user $u1$).

1. Uniqueness is modelled as transition probabilities in the HMM. Transition probabilities are conditional probabilities of a data by all users given previous data sequence by all users. This is required to calculate the indistinguishability or non-uniqueness of a user's data from other users' data. The risk of a data being distinguishable depends on the previous data. The reason is that the information gain from a data becomes higher if the previous data in the same topic are considered. Nodes in the HMM include data at a time (X_t) related to personal identifiable information topic, and/or a private/sensitive topic (such as cancer, drugs, and pregnancy). Edges contain the transition probabilities between nodes ($p(X_t|X_{t-1})$). These transition probabilities are weighted by their confidence in terms of how many transitions have occurred, which is $w_T = 1/count(X_t|X_{t-1})$. For calculating the privacy risk of a user with his web data, the weighted transition probabilities are considered, i.e., $w_T \times p(X_t|X_{t-1})$.

2. Uniformity is modelled as observation probabilities in the HMM. Observation probabilities are probabilities of the data found in previous web data by different users (u_i) including the user whose risk is to be predicted (if available). Each node contains a set of observations with observation probabilities. We model these observation probabilities as different users' probabilities of the

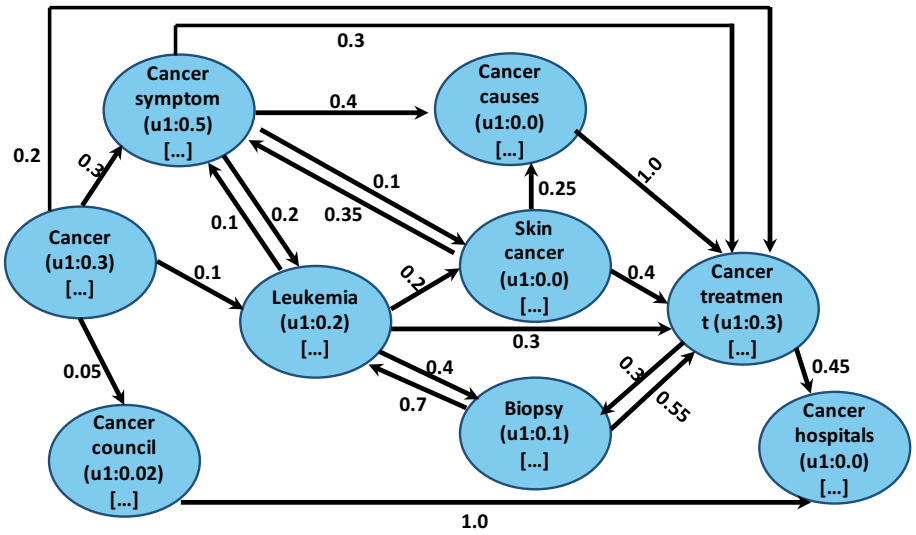


Figure 5.3: An Example of HMM model for Cancer Topic in Web Search Data. Nodes are cancer related queries and edges between nodes represent the transition (conditional) probabilities. Each node contains observation probabilities for different users (in this example these probabilities are shown only for user $u1$).

given data, X_t, found in previous data ($p(u_i|X_t)$). This is required to incorporate the non-uniformity aspect of a user as the frequency of the data entered by the user. The more a user has entered a specific data the more confidence (and therefore higher risk) in the inference that the user is interested in this data. Again these probabilities are weighted by $w_O = 1/count(u_i|X_t)$ and then inversed (as more uniform a user is higher the privacy risk is and therefore lower privacy probability), i.e., $(1 - w_O \times p(u_i|X_t))$.

3. In addition to these two probabilities, we have prior probabilities of the user based on previous searches that include PII (names, locations, and organizations). In order for the web data (related to sensitive/private topics other than PII topic) to be linkable to a user, the PII revealed by the user in his previous data needs to be taken into account. This prior probability of risk (of linkable using PII revealed) for a user u_i is calculated from the HMM of PII. The privacy risks of data related to PII topic are modelled in a separate HMM.

   For a given user u_i, the prior risk probability is calculated by getting the minimum privacy probability (maximum privacy risk) from all the paths in the PII HMM which include nodes X_t that contain an observation probability for the user, i.e., $p(u_i|X_t) > 0$. For users who do not have revealed any PII in

previous search history the prior privacy probability becomes 1.0.

The overall privacy probability of a user u_i along a sequence of web data $X_1 \rightarrow X_2 \rightarrow \cdots \rightarrow X_t$ is calculated as, following the Markov probability in Equation (5.1):

$$\begin{aligned} p(X_1, \cdots, X_t|u_i) = min(HMM_{PII}|u_i) \times w_T \times p(X_1) \\ \times (1 - w_O \times p(u_i|X_1)) \times \prod_{x=2}^{t} w_T \times p(X_x|X_{x-1}) \\ \times (1 - w_O \times p(u_i|X_x)), \end{aligned} \tag{5.2}$$

where $HMM_{PII}|u_i$ returns a list of privacy probabilities calculated from the PII HMM for all paths that include nodes where the user has an observation probability of > 0.0.

5.3 Evaluation

In this section, we present and discuss our findings on quantifying the privacy risk of web data. First, we present the datasets in use and then we discuss results of our experiments.

5.3.1 Datasets

To measure the privacy risks associated with online web data, we use two datasets: *(1)* AOL users' search queries; and *(2)* reviews of Android applications on Google Play[3]. We summarize our datasets in Table 5.1.

AOL Search Queries: In 2006, AOL released an anonymized version of 20 million user search queries of more than 650,000 users over 3 months period. Usernames were replaced by anonymous identifiers with the aim to protect user privacy. However, it failed to prevent de-anonymization for some users who performed ego-surfing, or searched for personal details such as social security number, phone number, or location directions. Therefore, we use this dataset to quantify sensitivity of web data. Each line in the in AOL search query data contains five fields: anonymous user ID, query string, query time, the rank of the item selected, and the domain of the selected item's URL path. We did not apply our method on the whole dataset, rather we set a criteria that selects only those users who have queries greater than 100. The statistics of our sampled dataset is given in Table 5.1.

[3] https://play.google.com

Table 5.1: Datasets in Use

	AOL Search Queries	**Android Apps Reviews**
# of Entries (E)	36,389,567	16,335,480
# of Users (U)	657,429	11,196,960
# of Apps (A)	–	1,0186,560
Condition	E ≥ 100	5M Reviews where E ≥ 15 & E ≤ 20
	Sampled dataset	
# of Entries (E)	23,927,203	13128
# of Users (U)	90,818	773
# of Apps (A)	–	6866

Moreover, to highlight the consequences of searching privacy sensitive topics that could potentially reveal user information, we focus on three topics: Cancer, Pregnancy, and Alcohol. In order to extract queries in these topics, we need to identify some must words for each topic. For this purpose, we used Free Keyword Tool offered by Wordstream[4] that utilizes the latest Google keyword API. We then performed topic modeling on these keywords to get most accurate and relevant must words. An example of must words for the cancer topic after applying topic modeling is *'Leukemia, Breast, Prostate, thyroid, Pancreatic, Bladder etc.'*, based on which cancer queries were extracted. We used `NLTK` [2] and `gensim` [1] to perform topic modeling and to extract relevant queries.

Android Apps Reviews: In order to collect users' reviews on Android apps from Google Play Store, we leveraged the crawlers developed in [110] and used the top 100 apps as a seed. Our crawler collects apps identifiers[5] and apps metadata by following a breadth-first-search approach for the apps which are *"similar"* in description or published by the same developer at Google Play. In summary, we crawled 1,018,656 apps in a 4-week period of December 2016 and collected 16,335,480 reviews from 11,196,960 unique users. A given user review consists of anonymous ID of a user, review text, review time and date, app ID, and app category.

We selected four categories of apps i.e., Social, Lifestyle, Health, and Games and extracted 5 Million reviews from our crawled dataset and then applied a criterion to select only those users that provide reviews in a range of 15 to 20 on different apps. We found that most of the reviews have been given for games followed by Lifestyle and Health apps.

[4]https://www.wordstream.com
[5]Each Android app has a unique identifier, termed as app ID in short.

5.3.2 Experiments and Results

We analyze privacy risk prediction results from the three aspects of uniqueness, uniformity, and linkability, and also present overall risk prediction results combining all three.

5.3.2.1 Experimental Setting

Before applying our method, we first pre-processed the data by filtering the broken, invalid, or empty sentences, and then re-ordered them based on time sequence. We then split the data into 20-80 testing approach where 20% of the data were used for testing, while 80% were used for training the HMM. Furthermore, to reduce training time, we applied k-means clustering that partitions the training data into k clusters, and then used multi-processing technique to run each training cluster simultaneously [94]. k-means algorithm helps grouping similar web data i.e., queries and reviews, based on the nearest mean (centroid). For our datasets, we selected 20 clusters using the elbow method [207]. Results from each multi-processed cluster are then combined to create one training model. For AOL dataset, we used semantic similarity algorithm for short sentences proposed by [142] to find similar queries, while term frequency-inverse document frequency (`TF-IDF`) was used to evaluate similarities of app reviews [94].

5.3.2.2 Privacy Risk Prediction

Our results indicate that privacy risk increases with sharing more data on the web. For the AOL dataset, we found that an average privacy risk reaches to 100% (1.0 privacy risk) when a user enters 10 queries. An exemplary user is shown in Table 5.2 (user ID 3058504), where the risk becomes 100% after entering 10 queries. Moreover, the average risk of predicting a user with just 1 sensitive query ranges between 78% and 83% (0.78 – 0.83). This is because our framework calculates risks based on three aspects, i.e., uniformity, uniqueness, and linkability. Even if a user does not have uniform data, he might be identified through the unique pattern of entering data, and vice versa. For instance, we can predict after 10 queries of the user shown in Table 5.2 with user ID '3058504' that either he or his family member is suffering from thyroid cancer. Similarly, we observe that another user (with user ID '3612363' as shown in Table 5.2) wants to know about Dr. Paul Mansfield, who worked at MD Anderson Cancer Center. Further queries would reveal that he is interested in prostate cancer at MD Anderson and its treatment. We also observe similar cases for pregnancy and alcohol topics, and found that users could be identified through

Table 5.2: Few Privacy Risk Evaluation Cases

User Anon. ID	**Web Entries**	**Topic**
3058504	'do you need surgery for underactive thyroid', 'why is physical therapy important after back surgery', 'why do you need physical therapy after back surgery', 'had back surgery but when i went to physical therapy my body hurt after', 'is it normal for my body to hurt after first visit to physical therapy', 'not being use to exercising can make physical therapy hurt', 'my husband did physical therapy one time and didnt go back due to pain', 'i dont like physical therapy because my body hurts after', 'physical therapy can be painful', 'is it normal for my body to hurt after first visit to physical therapy'	Cancer (10 Queries)
3612363	'md anderson cancer center and dr. paul mansfield'	Cancer (1 Query)
7894176	'getting pregnanct after being on birth control', 'getting pregnant with antiphospholipid disorder', 'having a healthy pregnancy with antiphosophlipid disorder', 'healthy pregnancy with antiphosophlipid disorder', 'chances of having a baby with antiphospholipid syndrome', 'costs of heparin during pregnancy', 'pregnancy and positive ana 1 640', ''pregnancy and positive ana 1 640', 'does lupus effect fertility', 'if i quit smoking in the middle of pregnancy will i miscarry', 'how much does smoking have an effect on fertility'	Pregnancy (10 Queries)
6143033	'pregnant no insurance denied by medicaid in florida'	Pregnancy (1 Query)
4320454	'cocaine drug testing', 'harms from herion addiction', 'opiate drug called suboxcine', 'national institute on drug abuse', 'how to clean out your urine for acocaine drug test .', 'how can we beat a cocaine urine drug test for employment', 'the longest time cocaine stays in our system for a drug test', 'how many days or hours for cocaine to leave the system to be clean for drug urine test for employment'	Alcohol (8 Queries)
3305139	'new jersey drug treatment rehab flynn house'	Alcohol (1 Query)
5995260	'Awesome app I am loving this app. Good work by the developers', 'Car wash for kids I am loving this app. Good work by the developers.', 'Awesome app I am loving this app. Good work by the developers.', 'Awesome app I am loving this app. Good work by the developers.', 'World hello Awesome game. I'm loving it. Good work by the developers', 'Car Racing Awesome game. Loved it'	Games (6 Reviews)
1559229	'Very useful tool This app is great for anyone going through health issues. Very easy to use, has many options for location of pain, what you were doing, and you can add different options. It's a great app if you have Fibromyalgia.'	Health (1 Review)

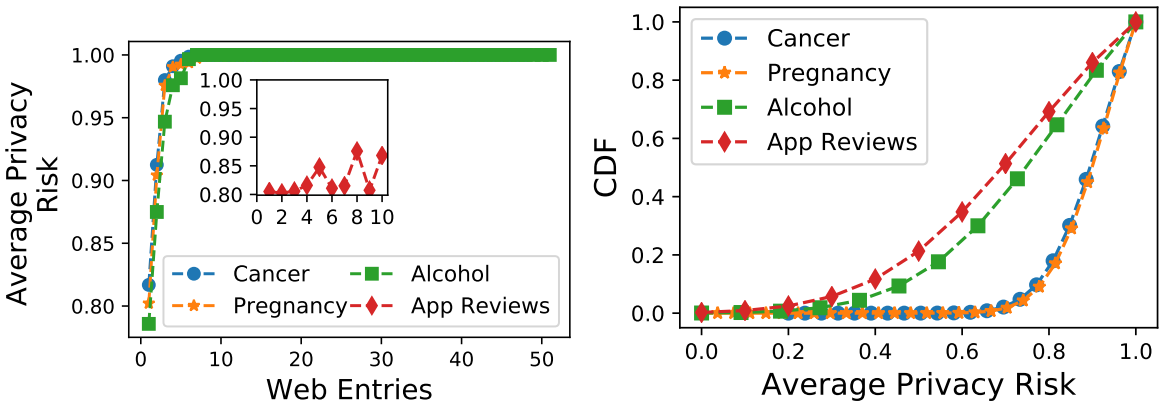
their unique web patterns. For instance, we discover that the user with ID '7894176' (shown in Table 5.2) is pregnant but has antiphospholipid and smoking problems. Likewise, the user with ID '4320454' wants to defy drug test by finding some ways.

For app reviews dataset, we found that average privacy risk associated with just 1 sensitive review is 80.5% (0.805), which increases to 87.5% (0.875) with 7 reviews. In Table 5.2, we observe that the user with ID '1559229' has some kind of association with Fibromyalgia disease and is using an app to improve his health issues. Similarly, we analyze that the user with ID '5995260' has the same writing pattern for all reviews and thus his privacy risk reaches to 99% (0.99) with only six reviews.

Considering our overall risk prediction results, we found that any data entry which contains words such as country name, person name, disease name, personal pronouns or uniformity has privacy risk of 75% (0.75) or above and is highly risky/sensitive. Therefore, we set our privacy risk threshold to 0.75, i.e., any entry which has a privacy risk above 75% is considered as highly risky which requires to be obfuscated with (semantically similar) entry.

Figure 5.4 shows the results of privacy risk prediction. It is clear in Figure 5.4a that our method is capable of re-identifying users even if the users' unique identities are not known. Our results indicate that an average risk reaches to 100% (1.0) if users have 10 or more data entries. The minimum average risk is 78% (0.78) for alcohol with 1 query. For app reviews, we achieve maximum of 87.5% (0.875) average risk with 7 reviews, and a minimum of 80.5% (0.805) with just 1 review. Figure 5.4b shows the CDF of users with their predicted privacy risks. For cancer and pregnancy, we found that more than 50% of users have risk higher than 0.85, while alcohol has a prediction rate of 0.7 for more than 50% of users. We found similar results for reviews dataset, where more than 50% of users have privacy risk of 0.7 involved in their reviews.

Uniformity: We now discuss our results on the uniformity of users' web entries. As mentioned earlier, uniformity refers to the number of observations of data entry by a user on the web. Our results compliment previous discussion, where users are exposed to higher risk with uniform data. We found that users who entered same entries two times have at least 85% (0.85) of privacy risk with all datasets. For instance, we observe that a user enters the query 'do i have liver disease if a small amount of billirubin is in urine' four times and thus gets the risk of 100% (1.0) being identifiable. Similarly, we found that a user enters the review 'NICE 1' 5 times, and has a privacy risk of 99.8% (0.998). Figure 5.5 shows the average risk for uniform queries. Overall, our results indicate that users involved in alcohol and pregnancy topics are 100% identifiable after entering 12 uniform queries, whereas users involved



(a) (b)

Figure 5.4: (5.4a) Average privacy risk with the increasing number of Web entries and (5.4b) average privacy risk per user.

in cancer topics are 100% identifiable with 10 uniform queries. Similarly, we analyzed that users who entered 10 similar reviews are 100% identifiable.

Uniqueness: Uniqueness refers to the distinctive sequence of a user's data entries on the web. Figure 5.6 shows the results. Our analysis shows that out of 700 unique data sequences of pregnancy, 680 sequences are 100% (1.0 risk) identifiable, and has the minimum average privacy risk of 98% (0.98). Likewise, cancer queries have 430 unique sequences out of which 410 are 100% identifiable and have the minimum average risk of 98.5% (0.985). For instance, in pregnancy topic, we found that a user is 98.5% identifiable after entering 7 unique queries in a sequence as shown below:

'how to increase fertility naturally', 'increasing fertility naturally', 'increasing the number of eggs released during ovulation naturally', 'increasing the number of eggs released during ovulation naturally', 'conceiving twins without fertility drugs', 'getting a baby girl', 'choosing babys sex with ovulation'

For alcohol queries, we realize that 40 out of 180 data sequences have 1.0 risk,

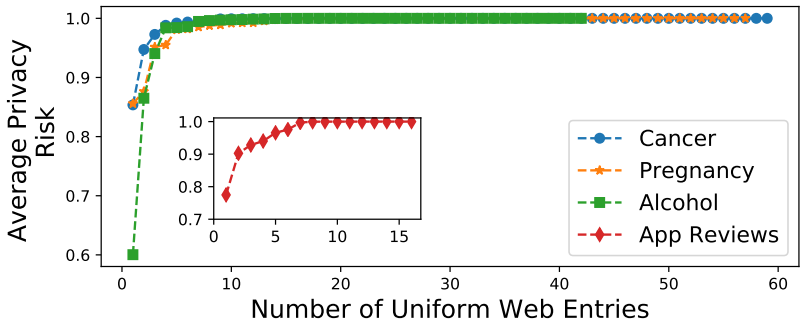


Figure 5.5: Risk Prediction Results of Uniform Web Entries.

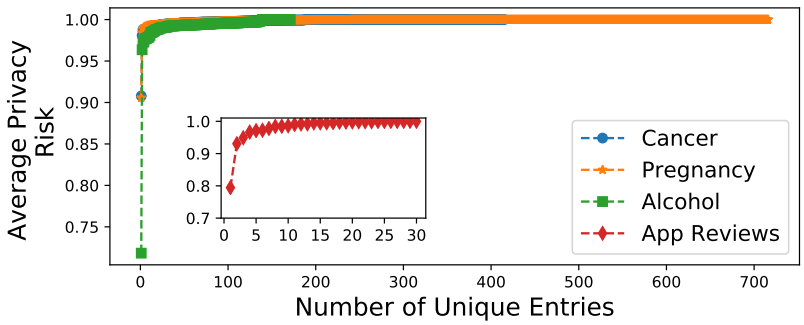


Figure 5.6: Risk Prediction Results of Unique Data Sequences

and these queries have the minimum average risk of 0.71. App reviews dataset has the lowest number of unique sequences, i.e., 20. The minimum average risk is 0.79 and it shows 1.0 privacy risk for 2 unique sequences only.

Linkability: We now investigate the linkability of users' web entries using their PII. We found few users who have PII information available in their web entries. For instance, a user in pregnancy topic entered a query 'place son long island to have a baby shower', and another user in alcohol topic entered PII query 'drug cases that been through the US appellate court'. Similarly, for app reviews dataset, we found that a number of users have entered either email IDs or names in their reviews.

Figure 5.7 shows the average privacy risk for the queries having PII available. We also present results without linkability information i.e., we remove PII and evaluate the privacy risk for the same set of entries. Our results indicate linking data with PII has more privacy risk as compared to data with no PII. For instance, cancer has the minimum average risk of 95% (0.95) for linkability, which reduces to 50% (0.5) if we remove PII. Similarly, pregnancy has 89.5% (0.895) minimum privacy risk with PII and 59% (0.59) without PII. We observe less difference in alcohol queries, i.e., a minimum of 98.5% (0.985) risk for linkability and 90.5% (0.905) for unlinkability. For app reviews, the linkable reviews have 90.6% (0.906) of minimum average risk, but reduces to 40.5% (0.405) without PII. However, we found that entries with or without PII can reach to 100% identifiability (uniqueness and uniformity) except for app reviews, for which the maximum risk involved with and without PII are 99% (0.99) and 98.5% (0.985), respectively.

5.3.3 Discussion

Our results reveal some important findings which we enlist below.

1. Privacy risk increases with sharing more data on the web even if the users' unique identities are not known. Users who share their personal interest in a

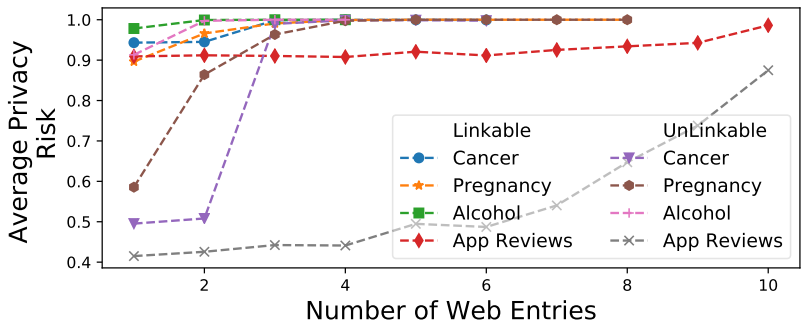


Figure 5.7: Linkable and Unlinkable Average Privacy Risks against Web Entries having PII.

specific field are likely to be more vulnerable to privacy attacks. For instance, users who searched for information related to specific medical center in a specific area are more easily identifiable in terms of their location and disease. For app review dataset, we found many users have same writing pattern in their reviews, thus making them identifiable against other users.

2. Privacy risk increases with sharing same data on the web. Users who entered same queries or reviews multiple times are easily recognizable. The identification reaches to 100% with 10 uniform entries. Similarly, privacy risk increases with the distinct sequence of web actions. This means that users who performed web actions or shared data in a different way than others are likely to be identifiable among others. Moreover, we found that users who share PII on the web are 100% identifiable in most cases.

5.4 Conclusion

In this chapter, we demonstrate the likelihood of tracking or identifying users through his web data actions e.g. search queries or comments on online forums. Our privacy risk prediction method depends on three key aspects: i) the uniqueness (or distinguishability) of a sequence of web data or activities by a user compared to other users, ii) uniformity (or non-diversity) of a sequence of web data or activities compared to the user's previous history, and iii) linkability of a sequence of web data to an individual using personal identifiable information (PII) available in the web data. We validate our method through two real web datasets and show that high privacy risk web data entries are likely to identify user, even if the user identities are anonymized. Our next chapter is yet another series on identifying privacy issues on the web. We explore potentially malicious third-party domains that are implicitly

trusted by first-party websites and also investigate the activities performed by these third-parties.

Chapter 6

Measuring and Analyzing the Chain of Implicit Trust

The web is a tangled mass of interconnected services, whereby websites import a range of external resources from various third-party domains. The latter can also load further resources hosted on other domains. For each website, this creates a dependency chain underpinned by a form of implicit trust between the first-party and transitively connected third-parties. The chain can only be loosely controlled as first-party websites often have little, if any, visibility on where these resources are loaded from. This chapter performs a large-scale study of dependency chains in the web, to find that around 50% of first-party websites render content that they did not directly load. We find that 73% of websites under-study load resources from suspicious third-parties, and 24.8% of first-party webpages contain at least three third-parties classified as suspicious in their dependency chain. By running sandboxed experiments, we observe a range of activities with the majority of suspicious JavaScript codes downloading malware.

In Section 6.2, we present our data collection methodology consisting of two parts: i) collecting information about websites and ii) classification of third-parties as suspicious vs. innocuous. Section 6.3 inspects the dependency chains across the Alexa's top-200K with the purpose to find the presence of implicitly trusted domains. We then proceed to inspect if *suspicious* or even potentially *malicious* third-parties are loaded via dependency chains in Section 6.4. In Section 6.5, we focus on *what* activities are undertaken within the dependency chains, by specifically targeting JavaScript programs. In Section 6.6, we summarize our key findings and finally conclude the chapter in Section 6.7.

6.1 Motivation

In the modern web ecosystem, websites often load resources from a range of third-party domains such as ad providers, tracking services and analytics services. This is a well known design decision that establishes an *explicit trust* between websites and the domains providing such services. However, often overlooked is the fact that these third-parties can further load resources from other domains, creating a *dependency chain*. This results in a form of *implicit trust* between first-party websites and any domains loaded further down the chain.

Consider the `bbc.com` webpage,[1] which loads JavaScript program from `widgets.com`, which, upon execution loads additional content from another third-party, say `ads.com`. Here, `bbc.com` as the first-party website, *explicitly* trusts `widgets.com`, but *implicitly* trusts `ads.com`. This can be represented as a simple dependency chain in which `widgets.com` is at level 1 and `ads.com` is at level 2. Past work tends to ignore this, instead collapsing these levels into a single set of third-parties [79, 168].

Here, we argue that this overlooks a vital aspect of website design. For example, it raises a notable security challenge, as first-party websites lack visibility on the resources loaded further down their domain's dependency chain. This potential threat should not be underestimated as errant active content (*e.g.,* JavaScript code) opens the way to a range of further exploits, *e.g.,* Layer-7 DDoS attacks [176] or ransomware campaigns [120].

This chapter studies dependency chains in the web ecosystem. Although there has been extensive work looking at the presence of third-parties in general [79, 168, 138], little work has focused on how content is indirectly loaded. We start by inspecting how extensive dependency chains are across the Alexa's top-200K (Section 6.3). We confirm their prominence, finding that around 50% of websites *do* include third-parties (e.g., content delivery networks (CDNs) such as `akamaihd.net` and ad and tracking services such as `google-analytics.com`) which subsequently load other third-parties to form a dependency chain (*i.e.,* they implicitly trust third-parties they do not directly load). The most common *implicitly* trusted third-parties are well known operators, *e.g.,* `google-analytics.com` and `doubleclick.net`: these are implicitly imported by 68.3% (134,510) and 46.4% (91,380) websites respectively. However, we also observe a wide range of more obtuse third-parties such as `pippio.com` and `51.la` imported by 0.52% (1,146) and 0.51% (1,009) of websites.

[1] This is an example (i.e., hypothetical case) to elaborate the (suspicious) resource dependency tree of `bbc.com`.

Although the majority (84.91%) of websites have short chains (with levels of dependencies below 3), we find first-party websites with dependency chains exceeding 30 in length. This not only complicates page rendering, but also creates notable attack surface.

With the above in mind, we then proceed to inspect if *suspicious* or even potentially *malicious* third-parties are loaded via these long dependency chains (Section 6.4). We do not limit this to just traditional malware, but also include third-parties that are known to mishandle user data and risk privacy leaks [96, 165, 205, 69, 35]. For instance, example threats include the re-identification of users in the anonymised AOL search histories, the Netflix training data that was attacked, and the Massachusetts hospital discharge data [96, 165, 205]. Furthermore, the collection of sensitive data by third parties also had devastating impacts on people's lives. For instance, it was shown that a person discovered his teenage daughter's pregnancy by observing her targeted adverts [69]. Similarly, Gmail was shown to use words from users' emails to target ads, exposing the nature of private correspondence in targeted ads [35].

Using the VirusTotal service [115] API, we classify third-party domains into innocuous *vs.* suspicious. When using a classification threshold (i.e., VTscore $\geq$ 10, further elaborated in Section 6.2.2 and 6.4.1), we find that 1.2% of third-parties are classified as suspicious. Although seemingly small, we find that this limited set of suspicious third-parties have remarkable reach. 73% of websites under-study load resources from suspicious third-parties, and 24.8% of first-party webpages contain at least 3 third-parties classified as suspicious in their dependency chain. This, of course, is impacted by many considerations which we explore — most notably, the power-law distribution of third-party popularity, which sees a few major players on a large fraction of websites.

We also focus on *what* activities are undertaken within the dependency chains. Hence, we *sandbox* all suspicious JavaScript programs to monitor their activities (Section 6.5). We build a sandbox and perform tests executing suspicious JavaScript codes. We find that JavaScript codes loaded at higher levels in the dependency chain ($Level \geq 2$) generated a larger number of HTTP requests. This is worrying as resources loaded at higher levels in the dependency chain are the most opaque to the website operator (*i.e.,* they rely on implicit trust). The activities of these scripts are diverse. For example, we find evidence of first-party websites performing malicious search poisoning activities when (implicitly) loading some JavaScript codes. The most typical purpose of the suspicious JavaScript

code is downloading dropfiles[2]. We also observe instances of *very* active JavaScript codes, *e.g.,* the most active (at level 4) downloads 129 files. We share all our datasets, experimental testbed code and scripts with the wider research community: `https://wot19submission.github.io`.

6.2 Dataset and Data Enrichment

We start by presenting our data collection methodology, and how we have validated its correctness. This consists of two key parts:(*i*) collecting information about individual websites, such that we can extract their dependency chains; and (*ii*) classifying all dependencies (*i.e.,* third-party domains) and suspicious *vs.* innocuous.

6.2.1 Alexa Dependency Dataset

We first present how we have obtained data on website dependencies, and how we construct their dependency chain. This critical first step underpins all subsequent analysis.

6.2.1.1 Data Collection

We obtain the resource dependencies of the Alexa top-200K websites' main pages[3] using the method described in [132]. This Chromium-based Headless [91] crawler renders a given website and tracks resource dependencies by recording network requests sent to third-party domains. The requests are then used to reconstruct the dependency chains between each first-party website and its third-party URLs. Note that each first-party can trigger the creation of multiple dependency chains (to form a tree structure).

Figure 6.1 presents an example of a dependency chain with 3 levels; level 1 is explicitly trusted by the first-party website, whilst level 2 and 3 are implicitly (or indirectly) trusted. For simplicity, we refer to any domain that differs from the first-party to be a third-party. More formally, to construct the dependency tree, we identify third-party requests by comparing the second level domain of the page (*e.g.,* `bbc.com`) to the domains of the requests (*e.g.,* `cdn.com` and `ads.com` via `widgets.com`). Those with different second level domains are considered third-party. We ignore the sub-domains so that a request to a domain such as `player.bbc.com`

[2]Dropfiles are executables (*e.g.,* malware, Exploitkits, Trojans, *etc.*) exploiting the browser to download and execute code without user consent (cf. Section 6.5.2.3).

[3]We select the top 200K as this gives us broad coverage of globally popular websites, whilst also remaining tractable for our subsequent data enrichment activities.

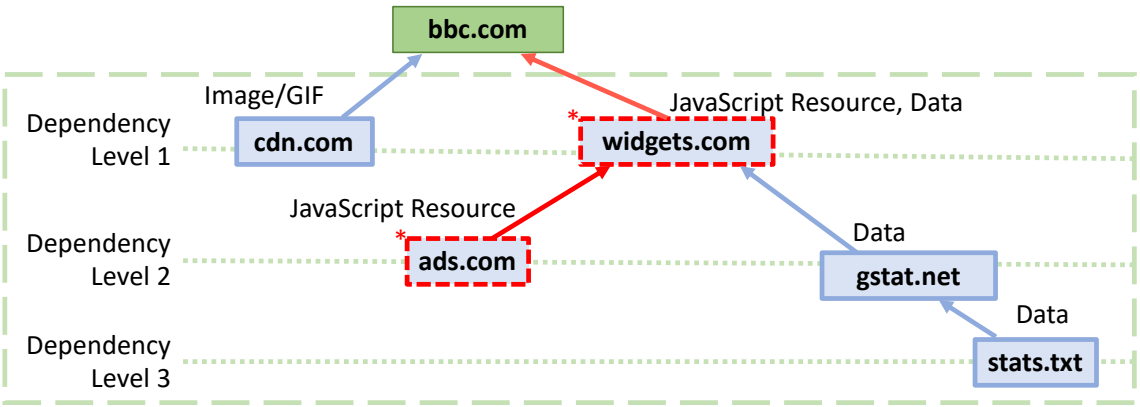


Figure 6.1: Example Dependency Chain of `bbc.com`. Arrows represent the inclusion of resources with red ones showing suspicious resource inclusion. For instance, `ads.com` is suspicious, and loaded by `widgets.com`, creating an implicit line of trust.

is not considered as third-party. Due to the lack of purely automated mechanism to disambiguate between site-specific sub-domains (*e.g.,* `player.bbc.com`) or country-specific domains (*e.g.,* `bbc.co.uk`), we leverage the Mozilla Public Suffix list [204] and `tldextract` [133] for this task. From the Alexa Top-200k websites, we collect 11,287,230 URLs which consist of 6,806,494 unique external resources that correspond to 68,828 and 196,940, respectively, unique second level domains of third- and first-parties.

Constructing the dependencies between objects in a webpage is a non-trivial task. In cases where third-party JavaScript program gets loaded into a first-party context, and then makes an AJAX request, the HTTP(S) request appears to be from the first-party (i.e. the *referrer* will be the first-party). To overcome such cases and to preserve the information on relations between the nested resource dependencies, we allow the crawler to include the URL of the third-party from which the JavaScript program was loaded by first-party.

6.2.1.2 Data Validation

As our main dataset relies on a single snapshot, we want to evaluate the stability of the resources loaded by websites to ensure that a single snapshot does not miss significant complexity within the ecosystem. Thus, we repeat the methodology from Section 6.2.1.1 on a daily basis to study how the dependency chains evolve. Unfortunately, performing daily crawls for the Alexa top-200k websites was not possible due to scalability reasons. We therefore selected 1,500 domains as a seed for the crawler. This list consists of the Alexa top-1K alongside 250 domains randomly

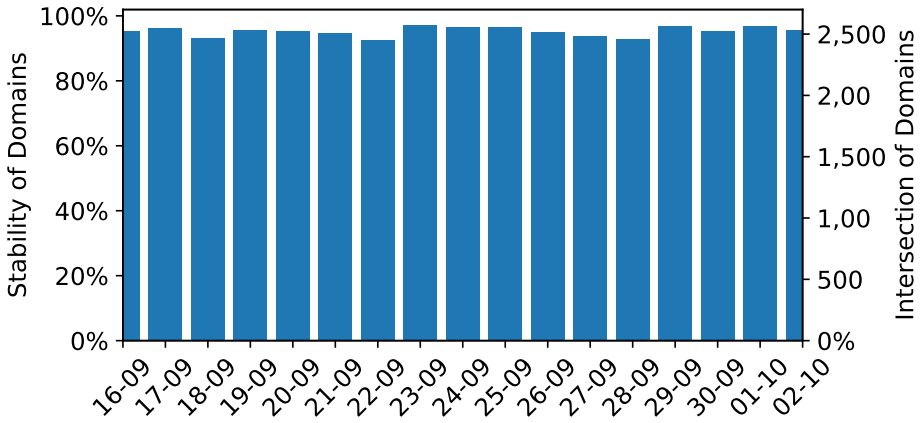


Figure 6.2: Stability of Day-by-Day Dependency Trees Analyzed per Domain.

selected from the Alexa rank ranging from 1K to 50K, and a final 250 domains randomly chosen from websites within the Alexa rank 50K–200K. This offers a broad sampling of the Alexa sites covered. In total, on a daily basis from September 15 to October 2 2018, we have collected on average 225,035 unique URLs per daily snapshot which covers 5,423 unique second level domains from the 1,500 first-parties.

Figure 6.2 presents the day-to-day stability of the domains we see within each website.[4] We observe that 95.07% of second level domains remain consistent across consecutive days, and only an average of 4.93% domains are absent in any two consecutive snapshots. On average, only 35 (0.66%) and 232 (4.27%) domains are absent at explicit and implicit dependency levels, respectively. Hence, we take this as a strong indicator that utilising a single snapshot is sufficient for gaining vantage into the use of third-parties.

6.2.2 Meta-data Collection From VirusTotal

The next challenge is to classify domains as suspicious *vs.* innocuous.For this we use VirusTotal — an online solution which aggregates the scanning capabilities provided by 68 Anti-Virus (AV) tools, scanning engines and datasets. It has been commonly used in the academic literature to detect malicious apps, executables, software and domains [122, 114, 127, 106, 111]. For each domain, we use the VirusTotal report API to obtain the VTscore for each third-party domain. This VTscore represents the number of AV tools that flagged the website as malicious (max. 68). The reports also contain meta-information such as the first scan date, scan history, domain name resolution (DNS) history, website or domain category, reverse DNS, and whois infor-

[4] We define the (normalized) stability as the count of domains present in the dependency trees crawled on day n and also present on day $n + 1$. More specifically, let C denoting the crawled data then, stability = $\frac{C_n \cap C_{n+1}}{C_n \cup C_{n+1}}$.

mation. We further supplement each domain with their WebSense [217] category[5] provided by the VirusTotal's record API. During the augmentation, we eliminate repeating, unresponsive or invalid URLs in each dependency chain. Thus, we collect the above metadata for each second level domain in our dataset. This results in a final sample of 196,940 first-party websites, and 68,828 third-party domains.

6.3 Exploring the Chains

We begin by exploring the presence and usage of implicit trust chains. We first confirm if websites do, indeed, rely on implicit trust and then explore how these chains are used. To this end, at each level of the dependency chain, we choose two metrics: number of requests and number of third-parties. The first metric, the number of requests, shows the significance or volume of resources imported from different levels, whereas the second metric characterizes coverage of third-parties in different levels.

6.3.1 Do websites rely on implicit trust?

Overall, the Top-200k dataset makes 11,287,230 calls to 6,806,494 unique external resources, with a median of 27 external resources per first-party website. To dissect this, Table 6.1 presents the percentage of webpages in each Alexa range that load explicitly and implicitly trusted third-parties. Confirming prior studies [79, 138], it shows that 95% of websites import external resources, with 91% importing externally hosted JavaScript codes. More important is that around 50% of the websites *do* rely on implicit trust chains, *i.e.,* they do include third-parties to load further third-parties on their behalf. The propensity to form dependency chains is marginally higher in more popular websites; for example, 55% in the Alexa top 10K have dependency chains compared to 48% in the bottom 10K (*i.e.,* rank 190-200K). In other words, more popular websites tend to rely more on implicitly trusted third-parties.

These implicitly trusted third-parties appear at various positions in the dependency chain. Intuitively, long chains are undesirable as they typically have a deleterious impact on page load times [216] and increase attacks surface. Figure 6.3a presents the CDF of chain length for all first-party websites. For context, websites are separated into their sub-categories.[6] It shows that 80% of the first-party web-

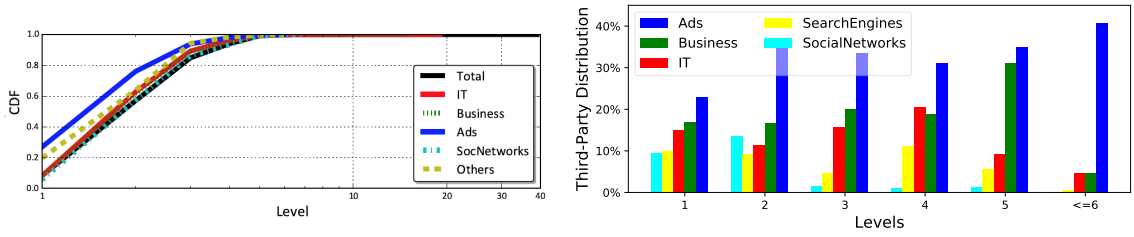
[5]For details on the websites or domains classification, we refer the reader to WebSense's, also known as ForcePoint, domains classification repository [81].
[6]We only include the most popular categories.

	Alexa Rank					
	1-200K	1-10K	190-200K	10-50K	50-100K	100-200K
First-parties that trust at least one third-party which loads:						
Any Resources:						
Explicitly (Lvl. 1)	95%	95%	95%	94%	95%	95%
Implicitly (Lvl. ≥ 2)	49.7%	55.1%	47.9%	51.8%	50.23%	48%
JavaScript Resources:						
Explicitly	91%	92%	91%	91%	91%	90%
Implicitly	49.5%	55%	47.8%	51.69%	50%	47.8%

Table 6.1: Overview of the Dataset for Different Ranges of the Alexa Ranking. The rows indicate the proportion of Alexa's Top-X websites that explicitly and implicitly trust at least one third-party (*i*) resource (of any type); and (*ii*) JavaScript code. It shows that 95% of websites import external resources, with 91% importing externally hosted JavaScript codes. Moreover, around 50% of the websites *do* rely on implicit trust chains, *i.e.*, they allow third-parties to load further third-parties on their behalf.

sites create chains of trust of length 3 or below. However, there is also a small minority that dramatically exceed this chain length: we find that all website categories import ≈2% of their external resources from level 3 and above. In the most extreme case, we see `rg.ru` (news) with a chain containing 38 levels, consisting of mutual calls between `adriver.ru` (ad provider) and `admelon.ru` (IT website). Other notable examples include `thecrimson.com` (Harvard's student newspaper), `argumenti.ru` (news), `mundomax.com` (IT news), `lifestyle.bg` (entertainment), which have a maximum dependency level of 15. We argue that these complex configurations make it extremely difficult to reliably audit such websites, as a first-party cannot be assured of which objects are later loaded.

Briefly, we also note that Figure 6.3a reveals subtle differences *between* different categories of third-party domains. For example, those classified as adverts are



(a) (b)

Figure 6.3: (a) CDF of Dependency Chain Lengths (broken down into categories of first-party websites); and (b) Distribution of Third-party Websites across Various Categories and Levels.

most likely to be loaded at level 1; this is perhaps to be expected, as many ad brokers naturally serve and manage their own content. In contrast, Social Network plugins and widgets (*e.g.,* Facebook plugins) are least likely to be loaded at level 1. We found that social networks are typically (99% of the times) loaded via CDNs (*e.g.,* `akamaihd.net` which is hosting the JavaScript codes belonging to Facebook) and in some cases (1% of the times) via third-parties, *i.e.,* analytics services (*e.g.,* `addthis.com`). Business third-parties are also very common: As per the Websense [81, 217] categorisation, this includes websites devoted to business firms, business associations, industry groups, *e.g.,* banks, credit unions, credit cards, and insurance. This also includes websites that provide access to business-oriented web applications and allow storage of sensitive data. Whilst the "IT" category includes websites providing information about computers, software, the Internet and related business firms, including sites supporting the sale of hardware, software, peripherals and services.

6.3.2 What objects exist in the chain?

The previous section has confirmed that a notable fraction of websites create dependency chains with (up to) tens of levels. We next inspect the types of resources imported within these dependency chains. We classify resources into four main types: Image, JavaScript codes, Data (consisting of HTML, JSON, XML, plain text files), and CSS/Fonts.[7] Overall, first-party websites import a median of 9 JavaScript codes and/or 6 images from external websites. Table 6.2 presents the volume of each resource type imported at each level in the trust chain. We observe that the make-up of resources varies dramatically based on the level in the dependency chain. For example, the fraction of images imported tends to increase with each level— this is largely because third-parties are in-turn loading images (*e.g.,* for adverts). In contrast, the fraction of JavaScript programs decreases as the level in the dependency chain increases: 30.6% of resources at level 1 are JavaScript codes compared to just 12.3% at level 3. This trend is caused by the fact that new levels are typically created by JavaScript code execution (thus the fraction of JavaScript codes is likely to deplete along the chain).However, it remains at a level that should be of concern to web engineers as this confirms a significant fraction of JavaScript code is loaded from potentially unknown implicitly trusted domains (cf. Section 6.6 for further discussion).

To build on this, we also inspect the *categories* of third-party domains hosting

[7] We classify using the HTTP headers and URL extensions (*i.e.,*`*.js, *.html, *.css`); this allowed us to classify 85% of resources.

Level	Total	# Unique Res. Calls	# Unique Third-Parties	Image	JavaScript Codes	Data	Font/ CSS	Uncategorised
1	9,212,245	8,866,074	57,375 (83.36%)	34.4%	30.6%	16.0%	7.8%	11.3%
2	1,566,841	1,295,322	8,617 (12.52%)	48.8%	16.7%	11.7%	3.3%	19.4%
3	405,390	223,080	1,618 (2.35%)	45.0%	12.3%	11.1%	1.3%	30.2%
4	78,107	90,984	647 (0.94 %)	41.8%	18.4%	8.0%	8.1%	23.6%
5	14,413	8,928	310 (0.45%)	40.6%	18.0%	12.8%	2.0%	26.4%
≥6	10,208	4,764	548 (0.8%)	36.6%	12.3%	13.0%	1.2%	36.8%

Table 6.2: Breakdown of Resource Types Requested by the Top-200K Websites across Each Level in the Dependency Chain. Total column refers to the number of resource calls made at each level.

these resources. Figure 6.3b presents the make-up of third-party categories at each level in the chain. It is clear that, across all levels, advertisement domains make up the bulk of third-parties. We also notice other highly demanded third-party categories such as search engines, Business and IT. These are led by well known providers, *e.g.,* `google-analytics.com` (web-analytics[8]) is on 68.3% of pages. The figure also reveals that the distributions of categories vary across each dependency level. For example, 23.1% of all loaded resources at level 1 come from advertisement domains, 37.3% at level 2, and 46.2% at level 3. In other words, the proportion increases across dependency levels. In contrast, social network third-parties (*e.g.,* Facebook) are mostly presented at level 1 (9.58%) and 2 (13.57%) with a significant drop at level 3. The dominance of advertisements is not, however, caused by a plethora of ad domains: there are far fewer ad domains than business or IT (see Table 6.3). Instead, it is driven by the large number of requests to advertisements: Even though ad domains only make-up 1.5% of third-parties, they generate 25% of resource requests. Importantly, these popular providers can trigger further dependencies; for example, `doubleclick.com` imports 16% of its resources from further implicitly trusted third-party websites. This makes such domains an ideal propagator of malicious resources for any other domains having implicit trust in it [144, 212, 120, 138].

6.4 Finding Suspicious Chains

The previous section has shown that the creation of dependency chains is widespread, and there is extensive implicit trust within the web ecosystem. This, however, does not shed light on the activity of resources within the dependency chains, nor does it mean that the implicit trust is abused by third-parties. Thus, we next study the existence of *suspicious* third-parties, which could lead to abuse of the implicit trust.

[8]Grouped as in business category as per VirusTotal reports.

				VTscore ≥ 3		**VTscore ≥ 10**		**VTscore ≥ 20**		**VTscore ≥ 40**		**VTscore ≥ 55**	
Category	**Third-Parties**	**Total Calls**	**Suspicious JS**	Num.	Vol.	Num.	Vol.	Num.	Vol.	Num.	Vol.	Num.	Vol.
All	68,828	11,287,204	270,758 (2.4%)	1.6%	6.4%	1.2%	6.2%	1.0%	6.1%	0.6%	5.7%	≤ 0.1%	≤ 0.1%
Business	6,786	1,924,591	184,360 (9.6%)	1.5%	21.5%	1.1%	21.5%	1.0%	21.4%	0.5%	20.6%	0%	0%
Ads	1,017	2,870,482	7,924 (0.3%)	3.5%	0.1%	3.3%	0.1%	2.9%	0.1%	1.6%	≤ 0.1%	0%	0%
IT	8,619	1,646,287	10,547 (0.6%)	2.2%	3.8%	1.5%	3.6%	1.2%	3.5%	0.6%	3.0%	≤ 0.1%	≤ 0.1%
Other	52,406	4,845,844	67,927 (1.4%)	1.4%	4.6%	1.1	4.3%	0.9%	4.2%	0.6%	3.8%	≤ 0.1%	≤ 0.1%

Table 6.3: Overview of Suspicious Third-Parties in Each Category. **Col.2-4:** number of third-party websites in different categories, the number of resource calls to resources, and the proportion of calls to suspicious JavaScript. **Col.5-9:** Fraction of third-party domains classified as suspicious (*Num.*), and fraction of resource calls classified as suspicious (*Vol.*), across various VTscores (i.e., ≥ 3 and ≥ 55).

Within this section we use the term *suspicious* (to be more generic than malicious) because VirusTotal covers activities ranging from low-risk (*e.g.,* sharing private data over unencrypted channels) to high-risk (malware).

6.4.1 Do chains contain suspicious parties?

First, we inspect the fraction of third-party domains that trigger a warning by VirusTotal. From our third-party domains, 2.5% have a VTscore of 1 or above, *i.e.,* at least one virus checker classifies the domain as suspicious. If one treats the VTscore as a ground truth, this confirms that popular websites *do* load content from suspicious third-parties via their chains of trust. However, we are reticent to rely on VTscore ≥ 1, as this indicates the remaining 67 virus checkers did not flag the domain.[9] Thus, we start by inspecting the presence of suspicious third-parties using a range of thresholds.

Table 6.3 shows the fraction of third-parties that are classified as suspicious using several VTscore thresholds. For context, we separate third-parties into their respective categories (using WebSense). The table confirms that a noticeable subset of suspicious third-party domains exist; for example, if we classify any resource with a VTscore ≥ 10 as suspicious, we find that 1.2% of third-party domains are classified as suspicious with 6.2% of all resource calls in our dataset going to these third-parties. Notably this only drops marginally (to 5.7%) with a *very* conservative VTscore of ≥ 40. We observe similar results when considering thresholds in the [3..50] range. This confirms, with a high certainty, that approximately 6% of resource calls in the dependency chains are towards domains that engage in suspicious activity (see Section 6.5) for further details). We will conservatively refer to domains with a VTscore ≥ 10 as suspicious in the rest of this analysis.

The proportion of suspicious third parties and resource calls can be related to a *prominence* metric defined in [73] that measures the frequency with which a user browsing encounters the third-party. The paper showed that the top 5 third-parties (`doubleclick.net`, `google-analytics.com`, `gstatic.com`, `google.com`, and `facebook.com`) have a prominence of 5.7 on average. The exact relationship between the prominence and the number of suspicious third-parties (and their volume of resource calls) is not important to us. However, a high prominence of a suspicious third party means that users have a high probability of becoming a target, *i.e.,* the effect of the 1.2% suspicious third parties becomes more devastating when a user is accessing those websites multiple times. For instance, in table 6.5

[9] Diversity is likely caused by the virus databases used by the different virus checkers [38]

we show that `google-analytics.com` is the top most suspicious third-party which has a prominence of 6.20 implying that a user is hit 6.2 times by this website.

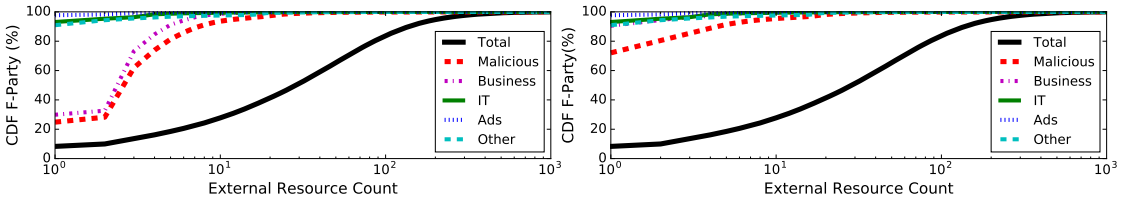
Additionally, we inspect first-party domains that inherit suspicious JavaScript resources from the explicit and various implicit levels. We focus (cf. Section 6.5) on JavaScript programs as active web content that poses great threats with significant attack surfaces consisting of vulnerabilities related to client-side JavaScript codes, such as cross-site scripting (XSS) and advanced phishing [138]. Table 6.4 shows the top first-party domains, ranked according to the number of unique suspicious third-parties in their chain of dependency. We note that the top ranked (most vulnerable) first-party domains belong to various categories such as Content Sharing, News, or IT. This indicates that there is not any single category of domains that inherits suspicious JavaScript codes. However, we note that first party websites categorised as "Business" represent the majority of most exposed domains at Level ≥2: 16% of first-party domains implicitly trusting suspicious JavaScript codes belonging to the Business Category. The distant second is the "News & Media" Category, and the third is "Adult". The number of suspicious JavaScript codes loaded by these first-party domains ranges from 4 to 31. For instance, we note the extreme case of `amateur-fc2.com`, which *implicitly* imports 31 unique suspicious JavaScript programs from 4 unique suspicious domains.

6.4.2 How widespread are suspicious parties?

We next inspect how widespread these suspicious third-parties are at each position in the dependency chain, by inspecting how many websites utilize them. Figure 6.4a displays the cumulative distribution (CDF) of resource calls to third-parties made by

#	First-party Domains	Alexa Rank	# Malicious JSes	Unique Suspicious Domains	Category	Chain Length
Unique Suspicious Domains at Level = 1						
1	theinscribermag.com	46,242	6	5	Blogs	5
2	skynet-system.com.ua	192,549	6	5	Busin.	4
3	nodwick.com	194,823	13	4	Enter.	4
4	iphones.ru	12,045	4	4	IT	4
5	privet-rostov.ru	193,024	6	4	LifeStyle	4
Unique Suspicious Domains at Level ≥ 2						
1	traffic2bitcoin.com	33,513	6	5	Games	7
2	radionetplus.ru	166,003	8	4	SW Download	6
3	studiofow.tumblr.com	85,483	11	4	Adult	4
4	amateur-fc2.com	52,556	31	4	Adult	5
5	fasttorrent.ru	24,250	9	4	File Sharing	7

Table 6.4: Top 5 most exposed first-party domains (with VTscore ≥ 10) ranked by the number of unique suspicious domains.



(a) All websites (b) Without google-analytics

Figure 6.4: CDF of Resources Loaded per-website from Various Categories of Third-parties.

each first-party webpage in our dataset. Within the figure, we decompose the third-party resources into various groups (including total *vs.* suspicious). As mentioned earlier, we take a conservative approach and consider a resource suspicious if it receives a VTscore ≥ 10. We purposefully select a relatively low VTscore threshold to balance the need for broad coverage against high confidence[10]. Figure 6.4a reveals that suspicious parties within the dependency chains are commonplace: 24.8% of all first-party webpages contain at least 3 third-parties classified as suspicious in their dependency chain. Remarkably, 73% of first-party websites load resources from third-parties at least once. Hence, even though only 1.6% of third-party domains are classified as suspicious, their reach covers nearly three quarters of websites (indirectly via implicit trust).

The above is demonstrated in Table 6.5, which presents the top 10 most frequently encountered suspicious third-party domains that are providing suspicious JavaScript codes. It can be seen that popular third-party domains exist across *many* first-party sites. The top 20% of third-party domains cover 86% (9,650,582) of all resource calls. Closer inspection shows that it is driven by one prominent third-party: `google-analytics.com`. At first, we thought that this was an error, however, during the measurement period `google-analytics.com` obtained a VTscore of 51, suggesting a high degree of certainty. This was actually caused by `google-analytics.com` loading another third-party, `sf-helper.net`, which is known to distribute adwares and spywares. It is unclear why Google was performing this. We therefore repeated these checks in October 2018, to confirm that this activity has ceased, and `sf-helper.net` is no longer loaded. To understand the impact its new de-classification has, Figure 6.4b shows the distribution of resource calls to third-party categories when `google-analytics.com` is benign. This reduces the number of first-party websites exposed to suspicious resources by 63%. This

[10]Note that the difference between 3 and 10 only results in an increase of 0.2% resource calls classified as malicious.

#	Third-party Domain	Alexa Rank	# First Parties	Category
Prevalence of Third-parties at Level = 1				
1	google-analytics.com	13,200	43,156	Business (Web Analytics)
2	gravater.com	2,292	3,520	IT
3	charter.com	12,714	3,425	Business
4	vk.com	13	2,815	Social Network
5	statcounter.com	2,265	2,327	Business (Web Analytics)
Prevalence of Third-parties at Level ≥ 2				
1	charter.com	12,714	3,452	Business
2	vk.com	13	2,290	Social Network
3	livechatinc.com	888	851	Web Chat
4	onesignal.com	950	467	Business
5	rambler.ru	291	370	SearchEngine

Table 6.5: Top 5 most prevalent suspicious third-party domains (with VTscore ≥ 10) on level 1 (explicit trust) and beyond (implicit trust) providing resources to first-parties. Here, First-party domains having the corresponding suspicious third-party domain in their chain of dependency.

highlights effectively the impact of high centrality third-parties being permitted to load further resources: the infection of just one can immediately effect a significant fraction of websites.

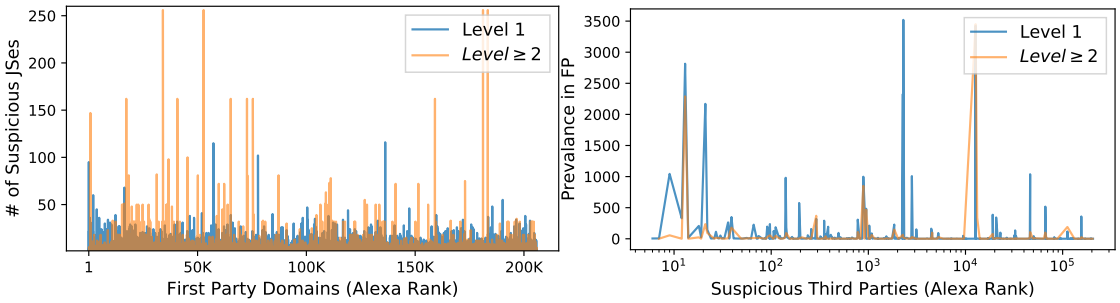
6.4.3 How popular are suspicious third-parties?

We next test if widespread suspicious third-parties are also highly ranked within Alexa. We treat this as a proxy for global popularity. Beyond `google-analytics.com` we find several other suspicious third-party domains from the Top 100 Alexa ranking. For-instance, `vk.com`, a social network website mostly geared toward East European countries has been used by 3,094 first-parties and is ranked 13 by Alexa. This website is found to be one of the most prevalent suspicious third-party domains at both level 1 and levels ≥ 2. An obvious reason for this domain's presence is because of other infected (malware-based) apps that try to authenticate users from such domains [181]. Other websites such as `statcounter.com` or `gravater.com` are also among the most prevalent third party domains in level 1. These websites were reported to contain malware in their JavaScript codes [78]. For instance, users in statcounter forums reported it as malicious because a JavaScript code running on its website redirects users to a malware website `gocloudly.com`, and forces users to click the button [82].

More generally, we observe the presence of a wide range of Alexa ranks in the list

of most prevalent domains at levels ≥ 2. In Figure 6.5a, we show the number of suspicious JavaScript codes imported by the first-party domains (Y-axis) according to their Alexa rank (X-axis). Overall, first-party domains import a larger number of suspicious third-party JavaScript codes at levels ≥ 2. However, the first-party domains seem to be equally vulnerable to the implicit importing of suspicious content regardless of their rank. There are exceptions though, signified by the peaks in the number of suspicious JavaScript codes — these are near exclusively driven by a large number of ≥level-2 scripts (implicit trust). We also encounter an interesting case, which we exclude from the graphs for readability purposes: The first-party domain `kikar.co.il` imports 2,592 JavaScript codes originating from the third-party `hwcdn.net`, a well-known browser hijacker that has been reported to force users to visit spam pages [213]. The VirusTotal API indicates a VTscore of 22 for this suspicious domain. We also note that 35 other first-party domains have this domain in their chain of dependency. Again, this highlights the risk of implicit trust.

In Figure 6.5b we show the number of impacted first-party domains as a function of the Alexa Rank of suspicious third-party domains (limited to a maximum Alexa Rank of 1 million) — note the log scale of X-axis. Some very prevalent third-parties have a high Alexa ranking (even excluding `google-analytics.com`). For instance, note a spike around the 2000 rank, which reaches a prevalence of 3500 first-party domains at level 1. This spike is caused by `gravatar.com`, propagating suspicious Javascript resources. This supports our statements earlier (from Table 6.5) where `gravatar.com` is ranked the second top most suspicious domain. Similarly, a spike around 10K rank indicates the presence of `charter.com` both at level 1 and 2 respectively. These findings demonstrate the wide variety of third-party suspicious



(a) (b)

Figure 6.5: Figure (a) depicts the number of suspicious JavaScript content imported (explicitly and implicitly) by first-party domains shown according to their Alexa ranking; and (b) shows the number of impacted first-party domains as function of the ranking of domains of Suspicious JavaScript.

JavaScript content loaded from various, not necessarily "obscure", third-party domains.

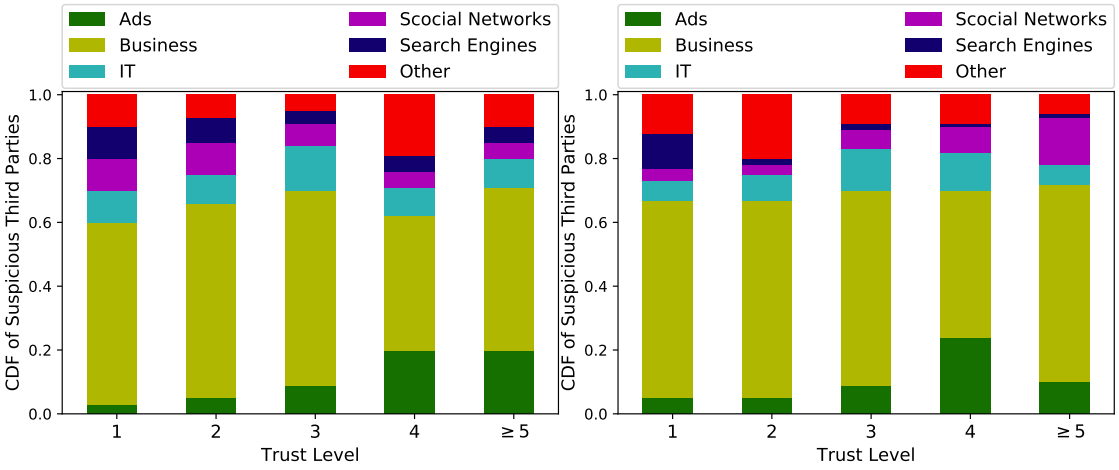
6.4.4 At which level do suspicious third-parties occur?

Next, we inspect the location(s) in the dependency chain where these suspicious third-parties are situated. This is vital, as implicitly trusted (≥level 2) resources are far more difficult for a first-party administrator to remove. Table 6.6 presents the proportion of websites that import at least one resource with a VTscore ≥ 10. We separate resources into their level in the dependency chain. The majority of resources classified as suspicious are requested at level 1 in the dependency chain (*i.e.,* they are explicitly trusted by the first-party). 73% of websites containing suspicious third-parties are "infected" via level 1. This might include websites that purposefully utilise such third-parties [107]. Perhaps more important, the above leaves a significant minority of suspicious resources imported via *implicit* trust (*i.e.,* level ≥ 2). In these cases, the first-party is potentially unaware of their presence. The most vulnerable category is news: over 15% of news sites import *implicitly* trusted resources from level 2 with a VTscore ≥ 10. Notably, among the 56 news websites importing suspicious JavaScript resources from trust level 3 and deeper, we find 52 loading advertisements from `adadvisor.net`. One possible reason is that ad-networks could be infected or victimized with malware to perform malvertising [145, 212].

	All		**News**		**Sports**		**Entertainment**		**Forums**	
Lv.	All	JS	All	JS	All	JS	All	JS	All	JS
1	61.30%	57.70%	75.40%	73.50%	75.70%	73.20%	69.30%	65.60%	67.40%	65.50%
2	5.20%	2.20%	13.40%	5.60%	11.10%	3.70%	8.60%	4.10%	9.10%	4.05%
3	1.30%	0.18%	2.90%	0.45%	3.60%	0.28%	2.70%	0.30%	3.20%	0.15%
4	0.22%	≤ 0.1%	0.64%	0.08%	0.80%	≤ 0.1%	0.70%	0.08%	0.60%	0.00%
≥ 5	≤ 0.1%	0	0.002	≤ 0.1%	0.001%	≤ 0.1%	0.002%	≤ 0.1%	≤0.001%	0.00%

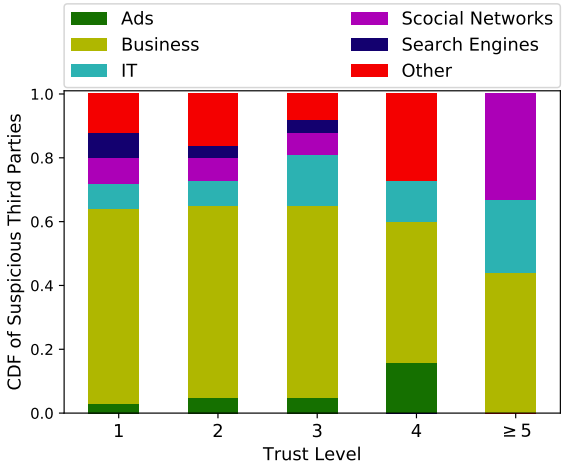
Table 6.6: Proportion of Top-200K Websites Importing Resources Classified as Suspicious (with VTscore ≥ 10) at Each Level.

Similar, albeit less extreme, observations can be made across Sports, Entertainment, and Forum websites. Briefly, Figure 6.6 displays the categories of (suspicious) third-parties loaded at each level in the dependency chain — it can be seen that the majority are classified as business. This is, again, because of several major providers classified as suspicious such as `convexity.net` and `charter.com`. Furthermore, it can be seen that the fraction of advertisement resources also increases with the number of levels due to the loading of further resources (*e.g.,* images).



(a) All Websites

(b) News Websites



(c) Sports Websites

Figure 6.6: Distribution of Suspicious Third-Party Websites per Category at Each Level, for all Top-200K Websites (Figure 6.6a) and most vulnerable first-party categories (Figures 6.6b, 6.6c).

Next, we again focus on JavaScript content as, when loaded, it can represent significant security risks: Our analysis is motivated by well known attack vectors underpinned by JavaScript codes, *e.g.,* malvertising [145], malware injection and exploit kits redirection. These are exemplifed by the recent reporting that Equifax and TransUnion were hit by a third-party web analytics script [136, 194]. Figure 6.7 presents the breakdown of the domain categories specifically for suspicious JavaScript resources. Clear trends can be seen, with IT (*e.g.,* `dynaquestpc.com`), Business (*e.g.,*`vindale.com`), News and Media (e.g., `therealnews.com`), and Entertainment (*e.g.,* `youwatchfilm.net`) dominating. Clearly, suspicious JavaScript resources cover a broad spectrum of activities. We observe that 70% and 67%, re-

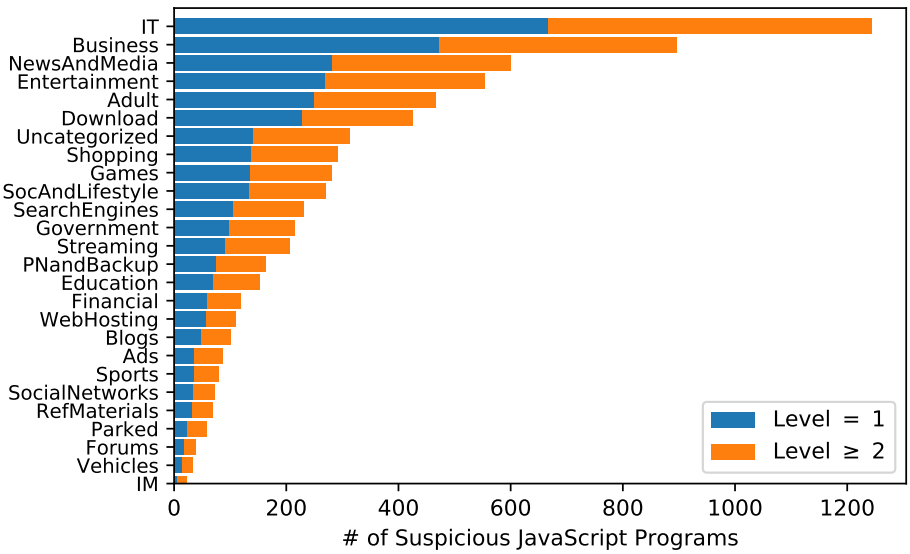


Figure 6.7: Breakdown of Unique, Suspicious JavaScript Resources at Explicit and Implicit Trust Levels. Previous work [54] used the domain category to group suspicious JavaScript resources. In the same spirit, we use the domain category to group JavaScript resources into different groups such as IT, Business, etc. Here, the Uncategorized category includes suspicious JavaScript resources whose domain's categories are *unknown* to WebSense, *e.g.*,`newmyvideolink.xyz` and `cooster.ru`. We observe that the suspicious JavaScript resources hosted by domains of IT, Business, News and Media, and Entertainment dominate at explicit and implicit trust levels.

spectively, of Business (Web analytics) and Ads JavaScript codes are loaded from level ≥ 2 in contrast to 17% and 31% of JavaScript programs of Government and Shopping loaded at level 1.

We next strive to quantify the level of suspicion raised by each of these JavaScript programs. Intuitively, those with higher VTscores represent a higher threat as defined by the 68 AV tools used by VirusTotal. Hence, Figure 6.8 presents the cumulative distribution of the VTscores for all JavaScript resources loaded with VTscore > 0. We separate the JavaScript programs into their location in the dependency chain. Clear difference can be observed, with level 2 obtaining the highest VTscore (median 52). In fact, 78% of the suspicious JavaScript resources loaded on trust level 2 have a VTscore > 52 (indicating *very* high confidence).

This is a critical observation; whereas suspicious third-parties at level 1 can be ultimately removed by first-party website operators if flagged as suspicious, this is much more difficult for implicitly trusted resources further along the dependency chain. If the intermediate (non-suspicious) level 1 resource is vital for the webpage, it is likely that some operators would be unable or unwilling to perform this action.

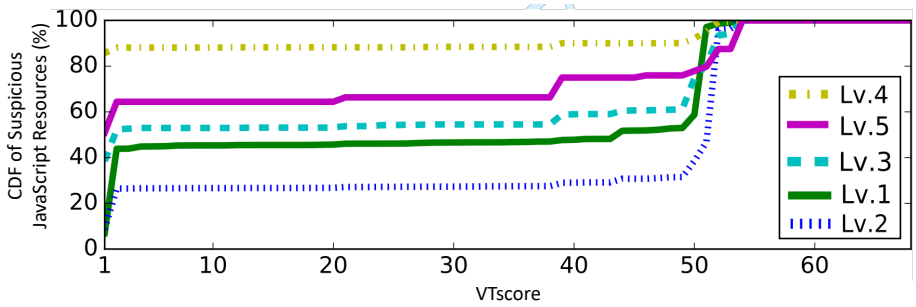


Figure 6.8: CDF of VTscores for JavaScript Programs (with VTscores > 0) at Different Levels in the Chain.

The lack of transparency and the inability to perform a vetting process on implicitly trusted loaded resources further complicates the issue. It is also worth noting that the VTscore for resources loaded further down the dependency chain is lower (*e.g.,* level 4). For example, 80% of level 4 resources receive a VTscore below 5. This suggests that the activity of these resources is more contentious, with a smaller number of AV tools reaching consensus. It is impossible to state the reason for this, hence in Section 6.5 we analyze the dynamic activities of these JavaScript content.

6.5 Analysis of Suspicious JavaScript resources

JavaScript codes are arguably the most dangerous resource to import, as JavaScript codes have the potential to execute diverse functions (including the downloading of further resources). Thus, we proceed to inspect the activities of the 7,166 JavaScript resources that were classified as suspicious in our dataset. We achieve this by executing the JavaScript resources in an isolated sandbox, and studying their activities.

6.5.1 Methodology

We use a dedicated testbed composed of three Virtual machines (VMs) that connect to the Internet via a computer running the Cuckoo sandbox and `tcpdump`. These VMs are configured with Windows 7, and are utilised to log all system-level events and to intercept all traffic being transmitted between the virtual machines and the Internet.Moreover, we use `Volatility` [214] to collect and analyse memory dumps of JavaScript code running in the browser. `Volatility` allows us to reveal information (*i.e.,* kernel-level processes and network connections) about the analysed JavaScript codes.This allows us to observe the traffic generated by each JavaScript code when it is rendered by the browser. For instance, our logs keeps a record of the network traffic generated, all file operations, memory changes, registry changes *etc.*.

For each test, we first create an HTML document and inject suspicious JavaScript code via the `<script>` tag. We then load the HTML in the browser of our VM testbed. We configured our testbed to wait 120 seconds for each target JavaScript code, embedded in an HTML code, to be rendered by the VM browser. The yielded logs are stored in a JSON object and pushed to our storage server for further analysis. It took, on average, an additional 3 seconds transferring and saving data at our server. Prior to each test, we turn-off and restore the VM to a clean snapshot. This ensures that any malicious software downloaded by prior JavaScript code's test does not remain on the VM. We share the code and data for the testbed at `https://wot19submission.github.io`.

6.5.2 Results

Using our sandbox testbed, we next measure which resources are accessed by suspicious JavaScript programs, as well as any dropfiles that are generated on the VM.

6.5.2.1 HTTP Request Frequency

We start by inspecting the underlying HTTP requests triggered by the JavaScript programs. Table 6.7 provides a list of the JavaScript resources that generate the most HTTP requests (separated into implicit and explicitly trusted). There is significant network activity generated by the suspicious JavaScript resources within our testbed, with downloads initiated at various locations in the dependency chain: 44.7% of requests are triggered at level 1 (explicit trust), whereas 55.3% at level ≥ 2 (implicit trust).

To explore this further, Figure 6.9a presents the distribution of the number of HTTP requests generated per suspicious JavaScript. The figure splits the JavaScript programs into their respective positions in the dependency chains. Although 47% of JavaScript resources generate fewer than 5 requests, there are notable differences among the different levels. JavaScript resources at level 1 generate the fewest HTTP requests (median 2), yet level ≥ 4 are extremely active (median 30 requests). 36% of the JavaScript programs imported from level 5 generate at least 30 HTTP requests in contrast to 15% of the JavaScript programs sourced from level 2. This is in contrast to a typical behavior of legitimate JavaScript programs that have been previously measured to generate on average just 4 HTTP requests [192].

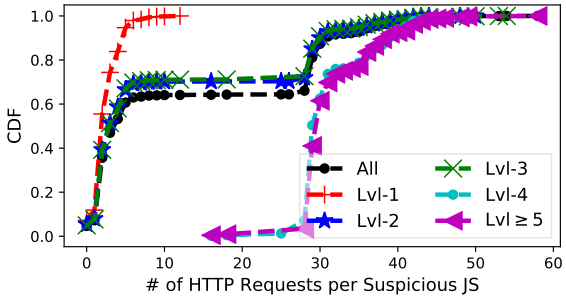
Furthermore, VirusTotal shows that those at level 1 tend to have lower VTscores (average 13), compared to those at ≥ 2, which tend to have a higher score (average 21). This is worrying as resources loaded further down the dependency chain are

#	Level	JavaScript	Category	VTscore	A-Rank	HTTP	Domains	Observed Behavior
1	Lvl-1	http://pinshan.com/js/union/new-play-1.js	Business	12	25,574	12	5	PUP[a] activity e.g., Installing Fake AV and mediaplayers
2	Lvl-1	http://newyx.net/js/dui_lian.js	Games	11	22,057	12	6	Displaying annoying ads and perform click fraud
3	Lvl-1	http://loxblog.com/fs/clocks/02.js	IT	10	86,505	10	3	Installing additional SW with elevated privileges
4	Lvl-1	http://mecum.com/js/jquery.fancybox.pack.js	Business	13	51,897	9	3	PUP activity, Installing Fake AV and mediaplayers
5	Lvl-1	http://bubulai.com/js/xp.js	Enter.	10	117,261	9	5	Displaying annoying ads and perform click fraud
1	Lvl≥2	http://yourjavascript.com/3439241227/blog.js	PNandBackup	13	2,007,688	58	51	Displaying annoying ads and perform click fraud
2	Lvl≥2	http://negimemo.net/alichina/login.js	SW Download[b]	10	13,093,855	49	21	Displaying annoying ads and perform click fraud
3	Lvl≥2	http://funday24.ru/js/c/funday-index.js	News	11	2,017,900	42	9	Displaying annoying ads and perform click fraud
4	Lvl≥2	http://netcheckcdn.xyz/optout/set/strtm.js	Business	14	18,064,762	42	7	Displaying annoying ads and perform click fraud
5	Lvl≥2	http://pushmoneyapp.com/js/main.js	Business	17	8,757,970	41	6	Installing additional SW with elevated privileges

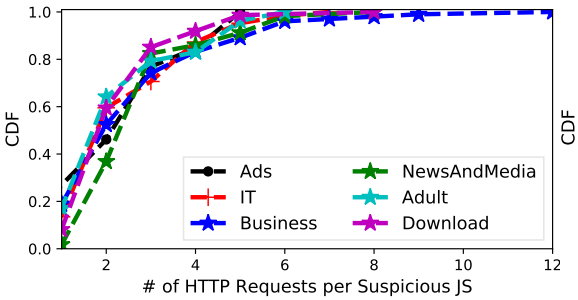
Table 6.7: Top 5 Suspicious JavaScript Resources Measured by Number of HTTP Requests Generated. We separate into JavaScript loaded at Level 1 (upper part of table) and Level ≥2 (lower). Here 'A-Rank' and 'Category' represent *Alexa* rank and category of the domain of the JavaScript code, respectively. Here, PUP stands for Potentially Unwanted Programs including "bogus" software such as free screen-savers or fake AV scanners that surreptitiously generate advertisements or perform redirections to collect personal identifiable information. SW Download means websites that share or facilitate downloading software executables.

[a] PUP stands for potentially unwanted programs, it includes bogus software such as free screen-savers or fake antivirus (AV) scanners that surreptitiously generate advertisements or perform redirection to collect user credentials or personal identifiable information.

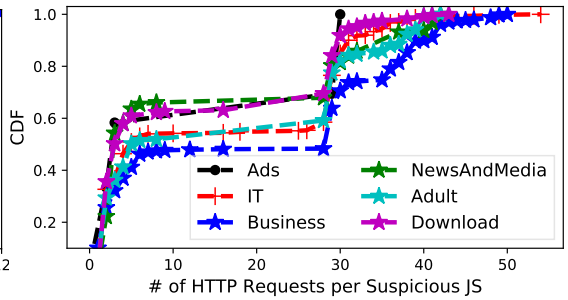
[b] SW Download means websites that share or facilitate downloading software executable.



(a) CDF of number of generated HTTP requests per suspicious JS at different dependency levels.



(b) CDF of the number of generated HTTP requests per suspicious JS at Level = 1



(c) CDF of number of generated HTTP requests per suspicious JS at Level ≥ 2

Figure 6.9: CDFs of number of HTTP requests generated per suspicious JavaScript resources viewed across, *(a)* different levels of dependency chain and categories of domains and dependency *(b)* Level = 1 and *(c)* Level ≥ 2.

the most opaque to the website operator. The most regularly observed JavaScript at level 1 is `new-play-1.js`, a relatively highly ranked (22,574 Alexa) script which downloads dropfiles. In contrast, at level ≥ 2, the most regularly observed JavaScript is `blog.js`, which show intrusive adverts that perform click fraud.

We are also interested in how behaviours might differ across categories of website. Hence, Figure 6.9b and 6.9c separate the JavaScript resources into their respective content categories. They then plot the distribution of number of requests per JavaScript within these categories. Whereas those at level 1 (explicit trust) exhibit relatively similar traits across all categories (Figure 6.9b), we find that those at level ≥ 2 (implicit trust) have far more divergence across categories (Figure 6.9c). Those classified as Business, IT or Adult are the most active, whereas News, Ads and Download generate the fewest HTTP requests. This is largely driven by the fact that most Business (*i.e.,* a subcategory of web-analytics) domains download more JavaScript codes, which then subsequently trigger further downloads (creating a cumulative effect).In contrast, other categories (*e.g.,* IT, Adult and Blogs) tend to download more static content, *e.g.,* images (which do not trigger further requests).

When inspecting what exactly the requests contain, we find that the overwhelming majority are downloading dropfiles (see Section 6.5.2.3). A remarkable 99.5% of all suspicious JavaScript codes download at least one dropfile, with the vast majority (98.62%) involving malvertising and click fraud (as identified via VirusTotal). This creates a heavy traffic footprint: 22% of HTTP requests are downloading dropfiles (further discussed in Section 6.5.2.3).

6.5.2.2 HTTP Request Targets

We next inspect the domains that these requests are accessing, *i.e.,* the domains hosting the content and files downloaded. For ease of presentation, we consider the top 25 domains targeted by the suspicious JavaScript codes (in terms of total number of HTTP requests targeting them). Figure 6.10 presents a heatmap illustrating the number of requests to them, with the X-axis showing the suspicious JavaScript code and the Y-axis listing the targeted domains. The heat is defined as the fraction of requests that each JavaScript issued to each domain. We find that most JavaScript codes have a distinct preference towards a small set of domains.For instance, 18% of JavaScript programs submit over 50% of their HTTP requests to a single domain. One particularly popular domain is `bing.com`; 65% of all suspicious JavaScript programs access this domain at least once. Closer inspection suggests that most JavaScript resources targeting `bing.com` undertake some form of search engine optimisation (SEO), *e.g.,* launching exploits to elevate the ranking of certain URLs in the results [102, 117].

Figure 6.11 also presents the count of HTTP requests across the top 25 targeted domains. We separate JavaScript codes into level 1 *vs.* level ≥ 2. We observe that the majority of fetches are triggered by level 1 (*i.e.,* they are explicitly trusted by the first party). However, we also note a large number of fetches to these domains are from JavaScript resources loaded at level 2. Revisiting our earlier example, 79.5% of HTTP requests to `bing.com` are triggered by level ≥ 2, indicating that the first party domain might be unaware that they are responsible for this attack (these requests are primarily for search engine manipulation [117]).As well as search engines, we note the existence of various certification authorities. We leave further inspection of these activities to future work, but conjecture that one of the reasons behind sending requests to the certification authorities is that digitally signed suspicious JavaScript resources can bypass system protection mechanisms that only install or launch programs with valid signatures. Such malware could also evade anti-virus programs which often forego scanning signed binaries.

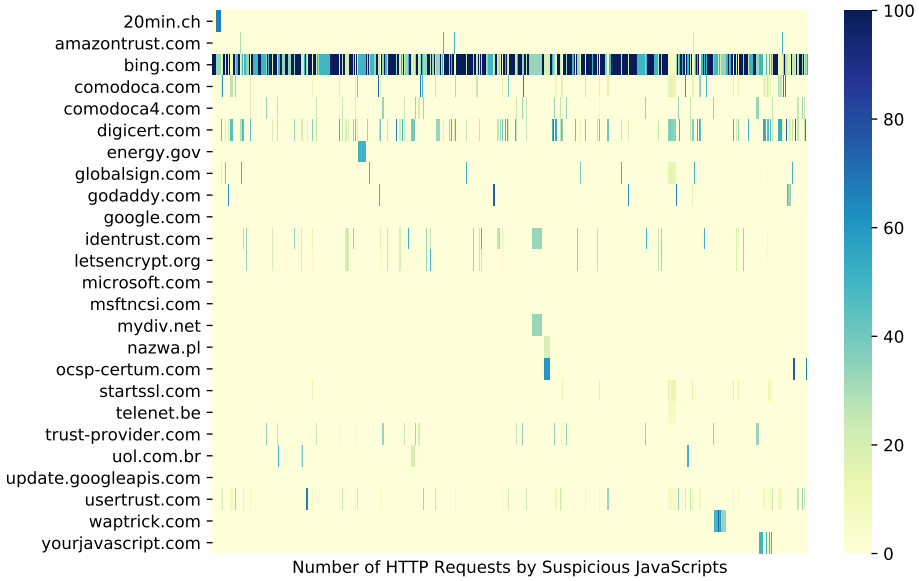


Figure 6.10: Heatmap of Number of Requests to Domains by Suspicious JavaScript Codes and Histogram of Top 25 Contacted Domains by Suspicious JavaScript Codes.

6.5.2.3 Analysis of Dropfiles

The above has revealed that a large number of suspicious JavaScript resources download files. The use of these dropfiles is commonplace, and they are often used during the infection process [16]. For instance, these files can potentially contain the

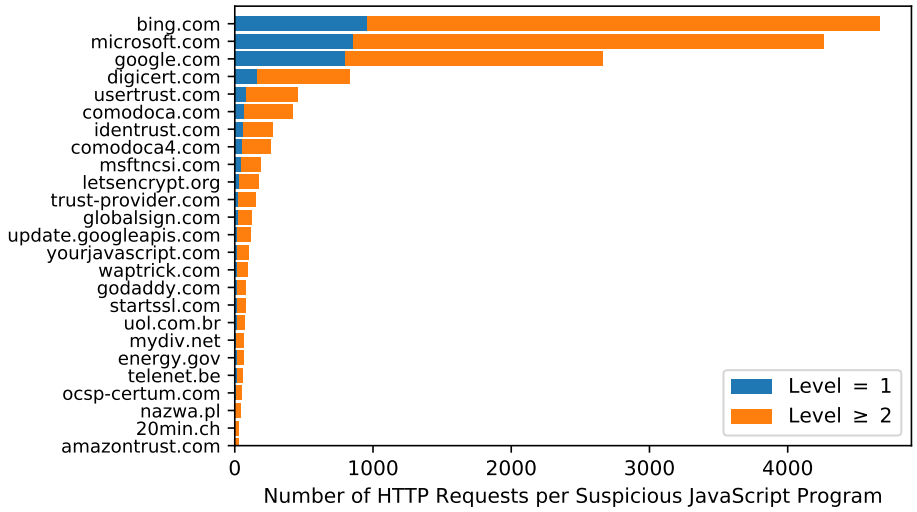


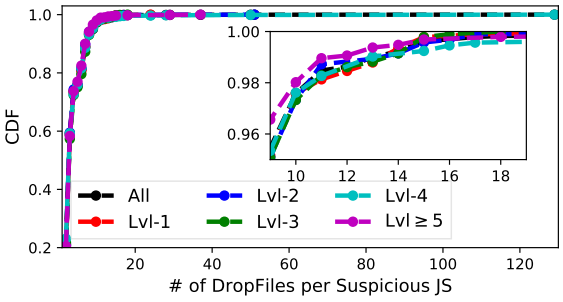
Figure 6.11: Number of HTTP Fetch Requests Issued by Suspicious JavaScript Resources, at Level = 1 vs. Level ≥ 2, to Top 25 Domains. Here X-axis shows the sum over all the loads of all the JavaScript programs across all analysed domains. The figure shows that most commonly occurring HTTP fetch is to `bing.com`.

unpacked malware binary that could potentially present worrisome vulnerabilities. Hence, we next inspect the creation of local files by JavaScript.

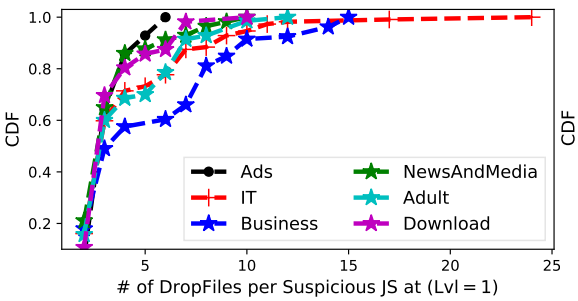
We observe a significant number of active memory operations[11]. as depicted by the number of dropped files. Note that dropfiles are executables (*e.g.,* malware, Exploitkits, Trojans, *etc.*) exploiting browsers to download and execute code without user consent. As previously identified, 99.5% of the suspicious JavaScript codes generate at least one dropfile, indicating that this is one of the most common activities undertaken. We observe significant differences in the number of dropfiles downloaded by each JavaScript. Figure 6.12 depicts the number of files dropped by the JavaScript content as a CDF. Whereas the majority download below 5, a small minority exceed 30. 22% of JavaScript codes generate at least 8 files from memory by compiling the dynamically loaded code in active memory and saving them to OS specific executables (*i.e.,* a "Trojan") confirming that memory exploits are being used by these JavaScript codes.

Table 6.8 presents the top-10 JavaScript codes based on how many dropfiles

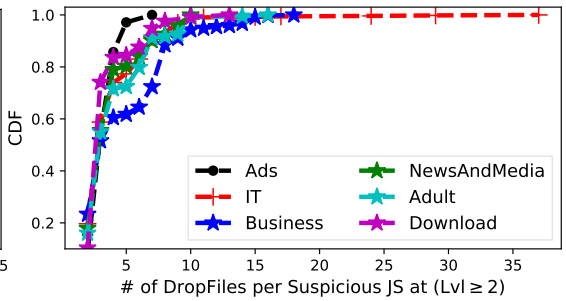
[11] Active memory operation mean processes that operates in memory. New types of malware differ from the traditional ones in the sense that they dynamically load suspicious codes from servers controlled by cybercriminals and run suspicious instructions from memory (*i.e.,* random access memory (RAM))



(a) Dropfiles by JavaScript Resources



(b) Dropfiles by JavaScript Resources at Level = 1



(c) Dropfiles by JavaScript Resources at Level ≥ 2

Figure 6.12: CDF of Dropfiles Downloaded and Operated (i.e., Read/Write Operation) by Suspicious JavaScript Codes.

#	Level	JavaScript Code	# of Drop files	% of Mal. DropFiles	VTscore	Mal. Type	Observed Behavior
1	Lvl-1	http://xbigg.com/adv.js	16	64%	9	AdCB	Displaying annoying ads and perform click fraud
2	Lvl-1	http://passback.free.fr/webmails/js/edito.js	10	82%	4	AdCB	Displaying annoying ads and perform click fraud
3	Lvl-1	http://via-midgard.info/engine/ajax/loginza.js	10	89%	5	AdCB	Displaying annoying ads and perform click fraud
4	Lvl-1	http://pokesnipe.de/js/app.min.js	10	90%	4	Torjan	Installing additional SW with elevated privileges
5	Lvl-1	http://hdvideo18.com/includes/ace.min.js	8	100%	3	AdCB	Displaying annoying ads and perform click fraud
1	Lvl≥2	http://yourjavascript.com/3439241227/blog.js	129	20%	15	AdCB	Displaying annoying ads and perform click fraud
2	Lvl≥2	http://s2d6.com/js/globalpixel.js	25	69%	11	AdCB	Displaying annoying ads and perform click fraud
3	Lvl≥2	http://cmsdude.org/wp-includes/js/jquery/jquery.js	12	71%	11	AdCB	Displaying annoying ads and perform click fraud
4	Lvl≥2	http://widih.com/js/like.js	10	90%	10	PUP	PUP activity, Installing Fake AV and mediaplayers
5	Lvl≥2	http://pichak.net/blogcod/clock/04/clock.js	8	100%	10	Exploitkit	Installing Exploitkit and performing Web redirects

Table 6.8: Top 5 Suspicious JavaScript Codes with Dropfiles (i.e., executables (e.g., Malware, Exploitkits, Trojans, etc) Exploiting the Browser to Download and Execute Code without User Consent) at Explicit and Implicit Dependency Level. AdCB means Adware and Click Bots.

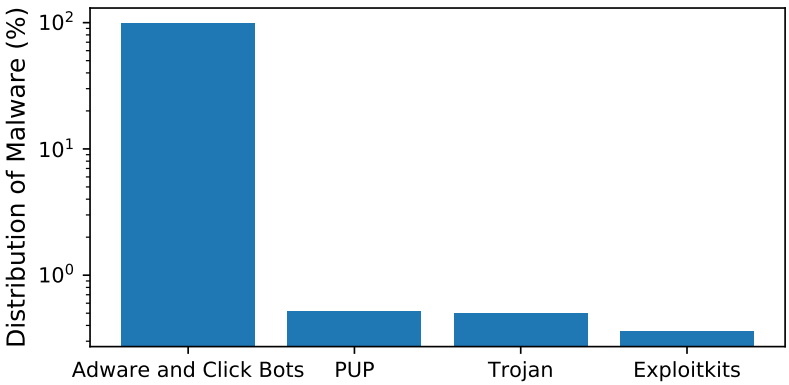


Figure 6.13: Histogram of Type of Malware (*i.e.,* Dropfiles) as per VirusTotal Reports (*cf.* Section 6.2.2).

they generate. Some of these JavaScript codes are extremely active. For example, `xbigg.com/adv.js` (loaded at level 1) downloads 16 files.Although there is not any significant difference between the number of dropfiles per level, most active JavaScript code (`http://yourjavascript.com/3439241227/blog.js`) is loaded at level 4 and downloads 129 files." It is also interesting to observe that the resources at level ≥ 2 tend to have higher VTscores, indicating that their activities are blocked by a large number of virus checkers. The actual content of the files are quite diverse. Figure 6.13 plots the distribution of file types, as classified by VirusTotal (*cf.* Section 6.2.2).We exclude the 8% which are encrypted, and therefore cannot be examined. The vast majority of remaining files (98.62%) are Adware and Click bots, suggesting that these types of financial gain are a major driving force in this domain. The remainder are Potentially Unwanted Programs (0.52%), Exploitkits (0.36%), Adware and Click Bots (98.62%), and Trojan (0.50%).For instance, `videowood.tv/assets/js/poph.js` uses and exploits `eval()` — JavaScript's dynamic loading method — to download and execute `1832-fc204a9bcefeab3d.exe` (with VTscore=5). This then enables the attacker to take over web browser for displaying a wide range of adverts and garner fraudulent clicks.

6.6 Discussion

In this Section, we summarise our key findings and explore simple solutions that may mitigate the impact of the vulnerabilities discussed.

6.6.1 Discussion and Mitigation

Our measurement results have identified a number of websites that load resources via implicit trust (*i.e.,* at level > 1 in the dependency chain). We have also confirmed that these chains often contain suspicious JavaScript resources, which expose users to risks. Unfortunately, these are not necessarily trivial to identify without appropriate expertise. Hence, the presence of these dependency chains can create challenges for identifying and filtering such resources. From the perspective of the first-party website, filtering an unwanted dependency can only be done by removing an intermediate third-party in the chain. This can be problematic if the dependency is performing a critical task (which may have taken a lot of development time to integrate). Similarly, many developers may simply not be aware of this practice and might therefore not know to check the dynamic loading of such resources.

To minimise such risks, users may leverage security and privacy preserving tools such as NoScript [116] to reduce the risk of (suspicious) JavaScript execution. However, an ordinary user is not expected to be well aware of such risks, or to install security tools. Hence, there are several methods that could be used by third-party services, resource providers, websites or browser developers to minimise the adverse impact of including resources from potentially suspicious third-parties. Most obviously, web developers and site operators should be made more aware of the risks identified in this chapter. This is particularly the case as these stakeholders have the capacity to curtail the problem by blocking any resources that depend on other malicious domains. The challenge here is providing greater transparency. This could, for example, be achieved by website operators running standard development tools and using VirusTotal to classify each domain contacted. There would also be value in sharing such information across websites (*e.g.,* to crowd source a blacklist of suspicious third-party resources which depend on other third parties). Of course, this list should be communicated to the third-party operators, as they may be unaware themselves of their dependency chain. We argue that such operators should also monitor their chains to ensure that they do not load any malicious resources.

Much of these checks could be automated within the browser. Existing AV tools could be used to block malware blacklists that are loaded via implicit trust. Implementation of best practices such as sub-resource integrity checks can also mitigate issues. Cross-origin resource sharing checks [162] could similarly be performed to prevent third-party resources gaining access to the first party context (*e.g.,* to access cookies). Websites should also ensure they utilise appropriate security headers to communicate their access policies (*e.g.,* using Access-Control-Allow-Origin).

6.7 Conclusion

This chapter has explored dependency chains in the web ecosystem. Inspired by the lack of prior work focusing on how resources are loaded, we found that over 40% of websites *do* rely on implicit trust. We classified the third-parties using VirusTotal to find that 1.2% of third-parties are classified as potentially malicious. Worryingly, our "confidence" in the classification actually increases for implicitly trusted resources (*i.e.,* trust level ≥ 2), where 78% of suspicious JavaScript resources have a VTscore > 52. In other words, more implicitly trusted JavaScript resources have high VTscores than explicitly trusted ones. We also performed sandbox experiments on the suspicious JavaScript to understand their actions. We witnessed extensive download activities, much of which consist of downloading dropfiles and malware.

Here, we end the first part of our thesis i.e. 'Privacy Risk Identification and Quantification' for web and mobile platforms. The next two chapters are our contributions towards the protection against the identified privacy leaks. The next chapter presents a privacy preserving framework that reduces the user tracking and identification issues on mobile devices.

Chapter 7

Privacy Preserving Framework for Mobile Sensor's Data

Sensors embedded in smart devices monitor user's environment with high accuracy and provide variety of services to a device user, from finding routes, to health monitoring and handwritten words recognition. However, as explained in chapter 4, these sensors have a potential of disclosing private information about a user, that eventually leads to user tracking or identification. To address this problem, in this chapter, we propose a privacy preserving framework that minimizes privacy leakages by bringing down the risk of trackability and distinguishability of individual users while preserving the functionality of the existing apps/services. We formulate our problem as time-series modeling and forecasting that overcomes the problem of handling unpredictable data and balancing privacy-utility when sensor data is fluctuating. The proposed framework is generic and keeps running in isolation without the interaction of a user or service providers. In addition, the proposed framework is resilient against noise filtering attacks by an adversary as it adds correlated noise-series to the forecasted time-series such that the noise is indistinguishable by an adversary. Rigorous experiments on benchmark datasets show that out framework limits trackability and distinguishability threats while maintaining a reasonable level of utility.

We organize this chapter by briefly highlighting the privacy problem and relating this to our contribution. In Section 7.2, we describe mobile ecosystem (Background) and then define the threat model of users information leakage from their mobile devices. In Section 7.3, we first present a sketch of our system model and then introduce our privacy-preserved framework that includes time-series data training and time-series privacy preservation as main phases. In Section 7.4, we discuss our experimental setup and then show results against privacy and utility metrics in Section 7.5. Finally, we discuss the results and conclude our work in Section 7.6.

7.1 Motivation

With the increasing use of mobile devices, a vast amount of temporal data generated by individuals through the use of applications on the mobile devices are being collected, such as sensor data ranging from microphones and cameras to accelerators, touch gestures and GPS trajectories. In many cases, service providers (app owners) capture and analyze raw sensor data in order to perform the required app functionalities or for other analytics purposes. For instance, a handwritten letter recognition app needs raw sensors data, such as x and y coordinates, to recognize letters or words written through a stylus or finger on a touch screen; a health monitoring app needs motion sensor data to detect activity (sitting, walking, running etc.) of a person. Some apps also require such data to personalize user experience, for example, appsee is an in-app mobile analytics service that performs in-depth analysis of user's behavior using scroll and touch gestures [140]. On the contrary, the collection of such mobile sensory data is highly associated with the privacy risk of individuals being identified and tracked through their data. It is quite evident from the literature that multiple sensors equipped within these devices contain subtle information and measurable variation (highly sensitive data), which allows users to be fingerprinted [57, 64, 159, 157]. For instance, a user could be identified from motion sensors signals produced from mobile phones [57]. In some cases, the captured sensor data is exchanged with third-parties such as advertisers or publishers for marketing and advertisement or various analytics purposes, for example, inferring health information about a user from motion sensor data [180, 225] , or inferring users' shopping interest or place of interest from GPS location [69].

We consider a situation where a user performs some gesture (swipe, tap, write) on a mobile device and data related to that gesture (raw x, y coordinates, finger pressure) is sent to a server, in order to acquire a service in return. While the data from a user gesture is essential to provide the desired functionality, there may be the cases where this data is unnecessarily captured for some intended/hidden purposes. For instance, Berend et.al [17] show that the sensors data, such as accelerometer and gyroscope, can reveal mobile password and PIN numbers to the apps. Similarly, sensor data can also be used to track physical location of the users even when the GPS is turned off, or to detect bots [119, 54]. Although the user and the service might not distrust each other, both parties sometimes appeal to symmetrically different outcomes. It is always in the service's best interest to gather knowledge about the user. The user, on the other hand, has a better interest in leaking a sparse amount of information. We illustrate this by looking at three concrete examples:

- Bob would like a service which optimizes his journey by reducing the travel time. The service does not need to know fined grained information e.g. favorite restaurant, health condition, as they divulge a lot of private information about Bob [131].

- Alice would like to use an app that transforms her handwritten letters into a notebook document. The app service does not need to extract her writing style as it may profile Alice and link her across different sessions [157].

- Georges would like to use a document reader service to read articles on his mobile. The service does not need to know any biometric information about Georges e.g. speed, duration, pressure of his finger, as it may lead to tracking [118].

In this work, *we investigate whether we can overcome user trackability and distinguishability issues associated with mobile sensory data release by providing a generic on-device privacy preserving framework.* Designing an on-device privacy-preserving approach is a challenge as there exists no clear distinction between the data which is required for the service to be functional and the one used for profiling purposes. Another challenge in this scenario is that service providers must not be involved in the privacy preserving mechanism. This assumption is justified since past incidents have revealed the distribution of users' data to third parties for tracking or marketing purposes [35]. Moreover, a privacy preserving mechanism must have an ability to preserve data without users' active participation. We make the following contributions in this chapter:

1. We design an on-device privacy preserving framework that minimizes the private information of a user before releasing to a server, whilst maintaining the intended utility of an application/service. The framework addresses two privacy risks, *trackability* and *distinguishability* by obfuscating raw data coming from various apps.

2. To the best of our knowledge, our framework overcomes the drawback of previous solutions by solving the problems of application and data specifity, user interaction with a privacy preserving mechanism, and trust issues with service providers. While models such as Deep Neural Network (DNN) are convenient to implement, they do not offer genercity as they are trained on known data types and for specific scenarios.

3. We formulate our problem as time-series data that utilizes time-series methods such as TBATS modeling, Dickey and Fuller (ad-fuller) test, and time-series forecasting. The representation of data as time-series enables our framework to be generic and keeps running in isolation without the interaction of a user or service providers. In addition, the framework uses box-cox transformation method to stabilize the time-series data that in return helps in accurate forecasting and adding considerable level of noise for utility preservation.

4. Our framework is resilient against noise filtering attacks by an adversary as it generates correlated noise-series which is added to the forecasted time-series data. Correlated noise implies that the correlation of the noise series with that of the original series is high and that they are indistinguishable by an adversary. This means that the refinement methods such as filtering cannot sanitize the noise in the released time-series. Thus, our framework offers two levels of protection; first by replacing original time-series data with forecasted data and then by adding correlated noise to the forecasted data.

5. Our framework provides a balance between privacy and utility by fine-tuning parameter values. These parameters can be changed anytime, through OS updates, to satisfy the requirements of privacy-utility tradeoff.

6. We empirically show the effectiveness of our framework through three different scenarios: (i) Handwriting Letters, (ii) Handwriting Digits, and (iii) Touch Swipes. We perform a comprehensive experimental study by evaluating our framework in terms of trackability, indistinguishability, utility and efficiency.

7. Experiments show that our framework limits trackability and distinguishability threats while mainitaining a reasonable level of utility. We found that an average untrackability increases to 38 – 40%, while indistinguishability on average increases from 19% to 50% for all the datasets. The utility loss ranges from a mean absolute error (MAE) of 0.31 to 0.5 on average for all the datasets. In general, we find that the maximum utility loss is 0.54 with 38% of untrackability and 32% of indistinguishability.

7.2 Preliminaries

In this section, we begin with a description of mobile eco-system (Background). Then, we define the threat model for users information leakage from their mobile devices.

7.2.1 Background

We assume a scenario where a user performs a certain task on his mobile device to get a particular functionality, for example, scrolling the touchscreen to read an article, writing or typing a message/email to send to a friend, tapping on a torch icon to turn on mobile torch light, tweeting or commenting on a social network app, using GPS to know the the nearby famous places or to get a route, etc. All these services are either built-in in a phone or offered by various service providers via mobile apps. In most of the situations, the data related to the task or a gesture performed by a user on a mobile device has to be sent to a remote server (operated by a service provider) on the fly to get a desired functionality. Moreover, users perform the same tasks repeatedly either in regular or irregular intervals. For instance, turning on an alarm clock every night, playing game after every few hours, checking emails every other hour, posting something on social networks every week, etc. Figure 7.1 shows the overview of a mobile device usage echo system. It is clear from the figure that our mobile echo system revolves around four major entities. We define these entities as follow:

1. *Users:* $\mathbb{U} = \cup_{i \epsilon [1,k]} u_i$ corresponds to the set of users of mobile devices where k corresponds to the total number of users.

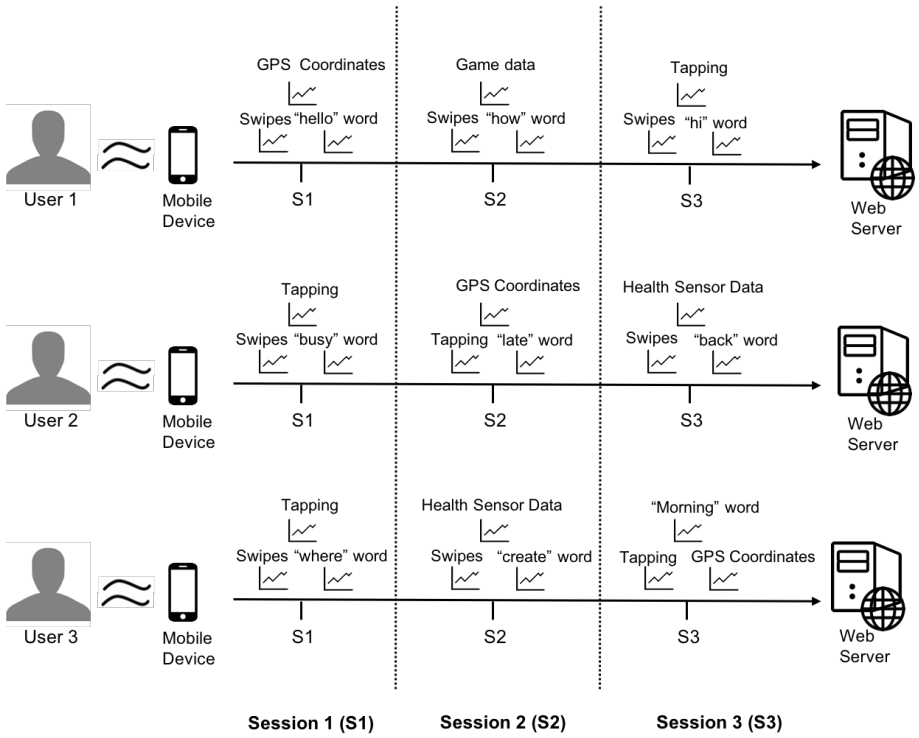


Figure 7.1: Overview of a Mobile Usage Echo System

2. *Gestures:* Let $\mathbb{G} = \cup_{i\epsilon[1,j]} g_i$ be a finite succession of gestures performed by a user on his mobile device. We assume that each time when data is sent to the external server, a set of gestures $\mathbb{G}$ are being performed by a user. Intuitively, the gesture corresponds to the data that a user is willing to release with the expectation that the user's privacy is not compromised.

3. *Sessions:* A session $S \in \mathbb{S}$ is a collection of gestures from a particular user u_i. Intuitively, this corresponds to one continuous use of the service. If a user is writing on a mobile phone, his session can be decomposed into n tasks, where each task consists of writing a word or a letter.

4. *Time-Series:* A time-series $\mathbb{T} = \cup_{i\epsilon[1,l]} t_i$ is a data from gestures $\mathbb{G}$ of a user at different point of time that is sent to a remote server at a fixed sampling rate. The data is then processed by a server to provide the requested functionality.

7.2.2 Threat Model

It has been demonstrated in the past that releasing raw mobile sensor information directly to service providers opens numerous ways for privacy leakage and user tracking [157, 155, 58, 54, 57]. We describe this scenario in Fig.7.2, where a service provider offers a service, e.g. an online food ordering, to a user on a mobile device. However, at the backend, the service provider can use the information, such as delivery destination, restaurants location, food menu, to analyze private information about a user e.g. home address, favorite foods, ethnicity etc. Moreover, a service provider can also exchange this information with third-parties (e.g. advertiser) to maximize the revenues. In this case, user information is leaked for various purposes, e.g. display related advertisements, link with another dataset to reveal health status etc. This can lead to two possible threats: *trackability* and *distinguishability*.

- In a *trackability* attack, an adversary infers private information about a user in one session, and then tries to link/track a user by maximizing the probability of inferring private information in future sessions.

- In a *distinguishability* attack, an adversary tries to uniquely identify a user from the data of other users in one session or a combination of all sessions. The unique identification is extracted by measuring the amount of identifying information in the data.

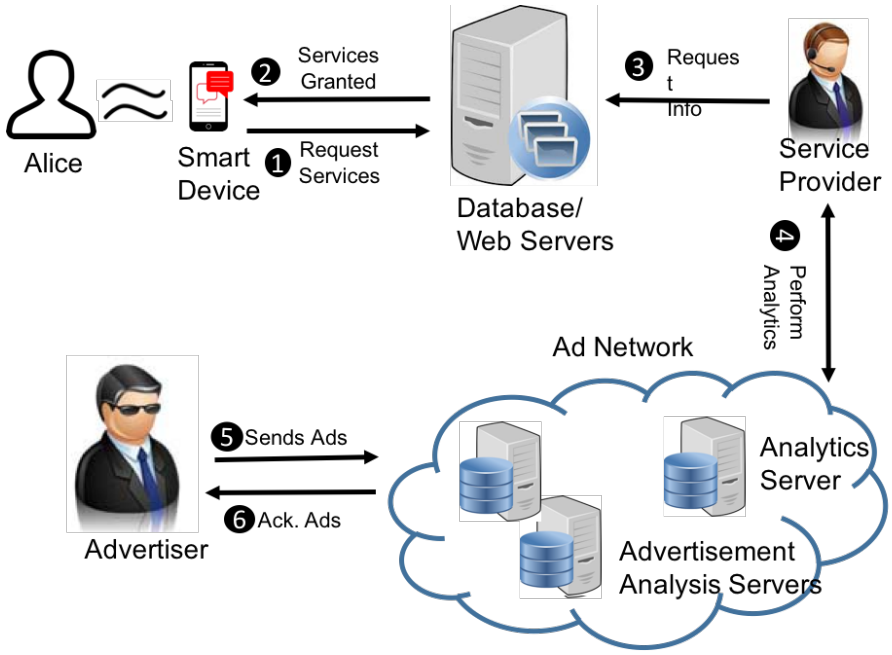


Figure 7.2: An Exemplary Threat Model of User Privacy Leakage

7.2.2.1 Trackability

Each user's u data is collected from various sessions $\mathbb{S}$ and can be characterized by a distribution of *public* and *private* information $Pub \times Pri$, where Pri corresponds to the private information that the user does not wish to disclose and Pub is the information that must be available to a service provider to provide a functionality. For-example, Pub corresponds to writing words like {'hello', 'hi', 'howdy'}, while Pri is the calligraphy specificity of the user.

By looking at the data $pub \in Pub$ in different sessions, a third party can gain insights on user's private information Pri, that could lead to user tracking or identifiablity. We call this as *trackability attack* where an adversary can link a set of sessions $\mathbb{S}$ to a particular user u by observing data $\forall_{S \in \mathbb{S}} pub_S$ and inferring private (uniquely identifiable) information Pri about a user. Here, we assume that the user's private information Pri is linked to his data Pub by the prior joint probability distribution $P_{Pub,Pri}$.

Definition 7.2.1. *Trackability Attack* A trackability attack takes a set of similar or matching observations from different sessions i.e. $sim(pub_{S_i}, pub_{S_j}) \geq \tau$[1], infers the private information Pri, and outputs the probability distribution $P_{Pub,Pri}$, such that the probability to link(track) Pub to user's $u's$ Pri is maximized $q^* : pri \rightarrow [0,1]$ from the set $P_{Pub,Pri}$ of distributions.

[1]where, $S_i, and S_j \in \mathbb{S}$

$$q^* = \arg\max_S P(Pri.Pub|S \in \mathbb{S})$$

here q^* is a belief distribution that Pub belongs to a user u with his Pri.

7.2.2.2 Distinguishability

The distinguishability attack uniquely identifies a user from a set of all other users by quantifying the amount of unique information from the dataset. Consider a scenario where an adversary gathers the data from all users with the purpose to uniquely identify each user based on specific patterns that occur in a data e.g. writing style or GPS locations. Let D denote the data of all users U in a dataset containing the set of features $\mathbb{F} = \cup_{i\in[1,n]} f_i$. Each feature f_i is associated with a time-series t such that $\forall_{t\in\mathbb{T}} f_i^t$, then the probability of a feature time-series belonging to a users u is given as:

$$\Pr(U = u \mid \mathbb{F} = \mathbf{f_i^t}) = \frac{\#(\mathbb{U} = u, \mathbb{F} \approx \mathbf{f_i^t})}{\#(\mathbb{F} \approx \mathbf{f_i^t})},$$

Now the mutual information or information gain between a user u and a set of feature $\mathbb{F}$ is defined as:

$$IG(U = u; \mathbb{F}) = H(U) - H(U = u \mid \mathbb{F}),$$

where $H(U)$ is the entropy to indicate the minimum number of bits of information required to distinguish each user in U and $H(U = u \mid \mathbb{F})$ is a conditional probability given as:

$$H(U = u \mid \mathbb{F}) = -\sum_{f\in\mathbb{F}} \Pr(\mathbb{F} = f) H(U \mid \mathbb{F} = f)$$

The given equations explain the methodology to quantify the unique information against each user thus leading to user identification among others.

7.3 Methodology

In order to overcome the trackability and distinguishability attack, we propose a privacy preserving framework that prevents user tracking across different sessions and also makes a user indistinguishable from other users. In this section, we first present a sketch of our system model and then introduce our privacy-preserved framework

that includes time-series data training and time-series privacy preservation as main phases.

7.3.1 System Overview

To thwart privacy leakage, a standard strategy is to send a distorted version of the data Pub called $\hat{Pub}$ to obfuscate the behavior [206, 189, 5, 190]. However, such solutions suffer from the issues of balancing privacy and utility, handling fluctuations/instability in mobile sensory data, and noise filtering attacks. Mobile sensor data is fluctuating and is not always stable because of changing environment and user activities. Therefore, a consistent obfuscation mechanism (that does not change over time) is not suitable since it cannot correctly predict the future data and is tuned to obfuscate based on previous learnings. Moreover, such obfuscation mechanisms cannot balance between privacy and utility; consider for example, an obfuscation mechanism that adds noise to the original data. If a mechanism is unable to predict future data then adding a constant noise value may either impact utility or privacy and thus cannot achieve a balance. Moreover, if a noise is independent and identically distributed (IID) then an adversary can separate the noise from the original data [36].

We design a privacy preserving framework that covers the abovementioned issues. We utilize the concept of time-series analysis where we consider mobile sensor data as time-series and apply tme-series modeling and forecasting methods. The purpose of modeling is to develop a model that captures the features of time-series based on past observations, whereas the aim of forecasting is to accurately predict the future and maintain stability across sensory data. The time-series analysis and modeling is discussed further in Appendix B.1. Moreover, to overcome the threat of noise filtering from obfuscated time-series data, we use the linear predictive filter technique from digital signal processing (DSP) that adds correlated noise to the data. The filter takes random white Gaussian noise as an input and produces correlated noise that helps in reducing the threat of noise filtering by an adversary from the obfuscated time-series. We provide more details about the noise filtering threat in Appendix B.1.1.

Our proposed privacy preserving framework is deployed inside a device operating system (OS) such that the data from any app is first processed at an OS level and then sent to a remote server. A device OS receives numerous out-going data records everyday which are sent to designated servers at fixed sampling rate. Depending on restrictions and rules deployed at OS level, the data can be sent with or with-

out processing. For our obfuscation framework, we assume that data coming from apps must be first processed such that the privacy threat of tracking users online is reduced to a sufficient level. We consider a scenario where our mechanism is deployed at OS level with pre-trained models so that when a user starts using a device, the privacy preservation level is maintained to some extent. Once sufficient data is collected from a user, our mechanism then updates the trained models after specified intervals for-example, a day or a week. Once the models are created and stored/updated at OS, our obfuscation framework then preserves user's sensitive data at run-time by using privacy preserving techniques. In particular, our framework consists of the following two main phases:

- *Time-Series Training:* In this phase, pre-trained models for our obfuscation mechanism are created and stored at OS. These models are called when apps start sending data at run-time. Also, these training models keep updating themselves at periodic intervals, for instance hourly, daily, or weekly. In particular, this phase includes building training models for time-series clustering and time-series stability (discussed later in Sec. 7.3.2.1).
- *Time-Series Privacy Perservation:* This phase interrupts the incoming data from apps and obscure at run-time before sending to a third-party server. It includes time-series forecasting and adding correlated noise to forecasted series (discussed later in Sec. 7.3.2.2).

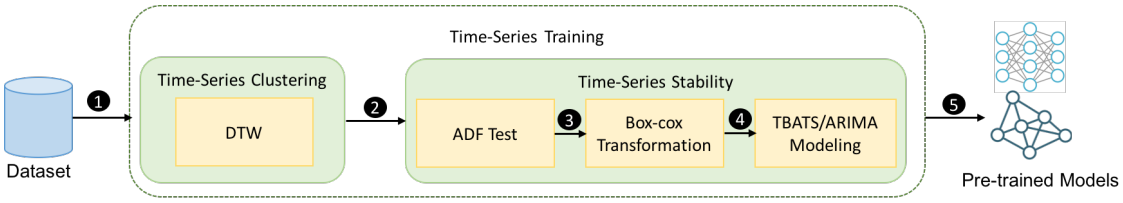
Figures 7.3a and 7.3b show the system architecture of training and privacy-preservation phases, respectively. We explain these in detail in the next Section.

7.3.2 Privacy-Preserving Framework

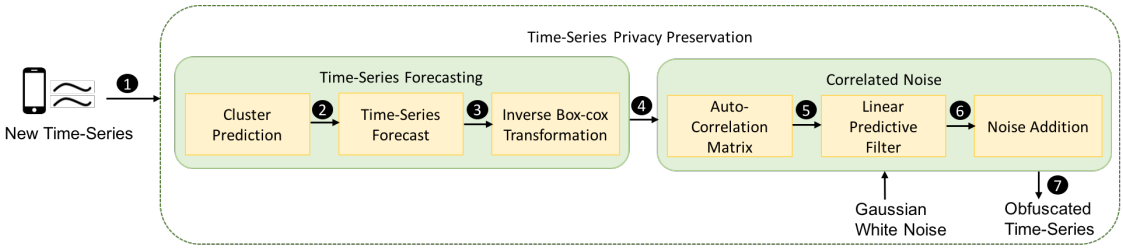
Our privacy preserving framework consists of the following two main phases namely: i) Time-Series Training and ii) Time-Series Privacy Preservation. In this subsection, we discuss in detail these two phases and their main components.

7.3.2.1 Time-Series Training

The Time-Series Training phase generates pre-trained models for our obfuscation mechanism, stores these models at OS of a mobile device, and also updates at regular intervals. In particular, the training phase has two main components namely: Time Series Clustering and Stability.



(a) In a time-series training phase, 1) time-series training data is fed into cluster model to generate pre-trained cluster model, 2) clusters stability is checked using ADF test, 3) unstable clusters are stabilized using box-cox transformation, 4) forecast modeling is performed on stable clusters using TABTS/ARIMA model, and 5) Pre-trained cluster and forecast models are stored in mobile OS



(b) In time-series privacy preservation phase, 1) pre-trained cluster, forecast models are readily stored in mobile OS and new time-series (at run-time) is fed for cluster prediction, 2) Based on the selected cluster, time-series forecasting is performed, 3) forecasted time-series is then transformed back using inverse box-cox transformation, 4) auto-correlation matrix is calculated for transformed time-series, 5) auto-correlation coefficients and white Gaussian noise is fed to Linear Predictive (LP) filter to generate correlated noise, 6) correlated noise is added to a time-series and 7) released to the server

Figure 7.3: System Overview

Time-Series Clustering refers to grouping similar time-series data into the same cluster and storing the trained cluster model in a device. This step is necessary to produce accurate forecasted time series at run-time. If the whole data is considered equivalent, then time-series forecasting will give inappropriate results as OS does not know what the data exactly is, for example if the data is a swipe, a word, a letter, or a GPS coordinate. It is, therefore, the requirement of the obfuscation mechanism to cluster similar time-series data during training phase and store the cluster model on a device. Once trained, a new time-series is ready to be associated with a cluster based on the trained model. We use Dynamic Time Wrapping (DTW) for grouping similar time-series data however, selecting the appropriate number of clusters is a challenge. Based on prior knowledge of the training dataset and the availability of computing resources, the number of clusters could vary.

Time-Series Stability refers to analyzing the seasonality and stability in a time-series and then making it stable for future predictions. The traditional classification and regression predictive modeling assumes that the summary statistics of observations are consistent [33]. These assumptions can be easily violated in time-series due

to the presence of trend, seasonality, and other time-dependent structures. In the time-series terminology, this is known as the time series being stationary or stable. Moreover, in the mobile sensory data, it is necessary to generate stable time-series in order to maintain high utility by forecasting accurately. If the time-series is not accurately forecasted then the utility of the functionality will be seriously affected. We use augmented Dickey–Fuller test (ADF), a statistical test, to check the stability of time-series data. Statistical tests make strong assumptions about the data. They can only be used to inform the degree to which a null hypothesis can be rejected or fail to be rejected. Moreover, they can provide a quick check and confirmatory evidence whether the time-series is stationary or non-stationary.

Once a time-series is determined to be non-stationary or unstable, a transformation such as box-cox needs to be applied to make a time-series stable. Box-cox is a data transformation technique used to stabilize variance and make the data look like a normal distribution [188]. The one-parameter Box–Cox transformation is defined as:

$$y_i^{\lambda} = \begin{cases} \frac{y_i^{\lambda}-1}{\lambda}, & if \lambda \neq 0 \\ \log(y_i), & if \lambda = 0, \end{cases} \tag{7.1}$$

where y is a time-series value and λ is a parameter with values ranging from -5 to 5. All values of λ are considered and the optimal value for the data is selected. The "optimal value" is the one which results in the best approximation of a normal distribution curve.

Once stabilized, the clusters of stabilized time-series data are then input into forecasting models such as TBATS, to train the model. These trained models are then used for time-series forecasting at run-time. The TBATS model basically belong to a general class of different models, such as random walk, random trend, seasonal and non-seasonal exponential smoothing and auto-regressive model, and it uses a systematic procedure for identifying the best TBATS model for any given time-series. These models are suitable for a huge amount of data that must be stationarized by differencing or using other mathematical transformations [31]. The most important step in such type of modeling is to select values for parameters of various components such as seasonality, trends, moving average, etc. There exist systematic procedures that select suitable parameter values based on time-series data [105], therefore, once the data is pre-processed (stationarized), an auto-forecasting model is applied that selects the parameter values and then fits the model on a time-series. Algorithm 2 illustrates the algorithmic description of the training phase.

Input : T = time-series data for training,
U=number of users,
F = dataset features
Output: T'= transformed/stabilized time-series,
V = pre-trained forecasted models,
C= pre-trained cluster models

```
1  for i ← 1 to U do
2      for f ← 1 to F do
3          Perform clustering on T using Dynamic Time Wrapping (DTW)
4          Store Cluster model C
5          for c ← 1 to C do
6              Check stability of time series in a cluster c w.r.t. variance σ using
                Augmented Dickey–Fuller test (ADF)
7              if not-stable then
8                  Apply box-cox transformation in a cluster c and get
                    transformed time-series T' using Eq. 7.1
9              Train ARIMA/TBATS model with transformed time-series T' to
                get a trained model V
10             Store trained model V
```

Algorithm 2: Time-series Training

7.3.2.2 Time-Series Privacy Preservation

After training, the next phase is to preserve user's sensitive data at run-time by using privacy preserving techniques. All the upcoming data from a user is first processed at OS and then sent to a remote server. The two main components in this phase are to forecast a time-series data and then add correlated noise to the forecasted time-series. Below we describe each of these in detail:

Time-Series Forecasting is an additional layer between original and perturbed data. The purpose of forecasting is to add noise to a stablized/stationarized time-series. User behavioural data such as swiping, typing, tapping, and writing is not consistent across sessions. In this case, noise parameters that are selected based on a certain training data, may not be suitable for data at run-time because inconsistency in data may impact the utility results. It is thus, necessary to stabilize the data before adding noise so that impact on utility has minimal effects.

In order to forecast, first the closest cluster to an upcoming data is predicted based on the pre-trained cluster model. Once the cluster is selected, the pre-trained forecasting model is called to forecast the future time-series. After forecasting, the last step is to reverse the box-cox transformation (back-transformation). The

purpose of back-transformation is to obtain forecasts on the original scale. The reverse Box-Cox transformation is given by:

$$y_i = \begin{cases} (\lambda y_i + 1)^{\frac{1}{\lambda}}, & if \lambda \neq 0 \\ \exp(y_i), & if \lambda = 0 \end{cases} \tag{7.2}$$

Correlated Noise is added to forecasted data to preserve the privacy. As described in Appendix B.1.1, the purpose of adding correlated noise is to reduce the privacy threat of getting estimated original series by filtering out noise from perturbed time-series data. Our correlated noise generation is based on the concept of linear prediction (LP) of DSP. The purpose is to find linear filter coefficients based on forecasted time-series such that when a random noise is passed through a filter, a noise correlated to forecasted data is generated. We explain the process of LP below:

Let $X[n]$ denote the elements of a random process, having outcomes $x[n]$. The goal of linear prediction is to find a set of coefficients $a_1, a_2, ...a_N$ such that

$$\bar{x}[n] = a_1 x[n-1] + a_2 x[n-2] + ...a_N x[n-N] \tag{7.3}$$

is as close as possible to $x[n]$, for all n, and for all realizations of the underlying random process. We say that $\bar{x}[n]$ is a "prediction" for the actual value of $x[n]$ and we refer to the a_1 through a_N as the LP coefficients of order N. There are many applications of LP, however we use LP to find its coefficients through the prediction error,

$$e[n] = x[n] - \bar{x}[n] \tag{7.4}$$

The prediction error reflects the idea that an ideal predictor would extract all possible information from the m number of samples which have already been seen, $x[n-1], x[n-2], ...x[n-m]$. To find a single set of coefficients which work for all n and all realizations of the underlying random process, the condition is to make $X[n]$ stationary, and look for the coefficients which minimize the expected (mean) squared prediction error. Once $X[n]$ is stationarized (as done in previous steps), the following auto-correlation equation is used to find the coefficients.

$$\underbrace{\begin{pmatrix} R_{XX}[0] & R_{XX}[1] & \cdots & R_{XX}[N-1] \\ R_{XX}[1] & R_{XX}[0] & \cdots & R_{XX}[N-2] \\ \vdots & \vdots & \ddots & \vdots \\ R_{XX}[N-1] & R_{XX}[N-2] & \cdots & R_{XX}[0] \end{pmatrix}}_{R_{X_{0:N}}} \underbrace{\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{pmatrix}}_{a} = \underbrace{\begin{pmatrix} R_{XX}[1] \\ R_{XX}[2] \\ \vdots \\ R_{XX}[N] \end{pmatrix}}_{r} \tag{7.5}$$

which may be solved for the unknown coefficients, by setting

$$a = R_{X_{0:N}}^{-1} r \tag{7.6}$$

The matrix, $R_{X_{0:N}}$ is the auto-correlation matrix of the random vector, $X_{0:N}$, whose elements are $X[0]$ through $X[N-1]$. We may call this the Nth order auto-correlation matrix for the stationary random process, $X[n]$.

To apply LP in our work, lets consider $X[n]$ as forested time-series and $R_{X_{0:N}}$ as its auto-correlation function. The coefficients a for N order could be extracted by finding the values of $R_{X_{0:N}}$ and r from Eq. 7.6. It is to be noted that to find a linear filter coefficients with the best correlation effects, the auto-correlation matrix of any order from 1 to N should be tested. To do so, we use auto-correlation matrix from 1 to N order and select the one which gives correlation greater than a threshold. Once coefficients are extracted, we then create a linear filter.

The next step is to generate a white Gaussian noise Z with mean (μ) of 0 and different noise scale levels. The random noise Z is passed through a linear filter to generate a correlated noise Z'. The correlated noise is then added to the forecasted data using Eq. 7.7.

$$X' = X + Z' \tag{7.7}$$

The noisy correlated data is then released to remote servers. Algorithm 3 illustrates the algorithmic description of the privacy preservation phase.

7.4 Experiments Settings

In this section, we first present the datasets that are used in our experiments followed by the definition of metrics that are used to evaluate our obfuscation mechanism. We then explain our experimental procedure that includes data preprocessing and experimental setup with different parametric values.

Input : X = time-series of length l,
V = pre-trained forecasted models,
C= pre-trained cluster models,
F = dataset features,
τ = correlation threshold
Output: X'= obfuscated time-series

for $f \leftarrow 1$ **to** F **do**
  Get a predicted cluster by giving time-series X as input to the pre-trained cluster C
  Forecast a time-series X' of length l using pre-trained forecasted model V
  Perform inverse box-cox transformation on time-series X' using Eq. 7.2
  for $n \leftarrow 1$ **to** N **do**
    Calculate auto-correlation matrices $R_{X_{0:n}}$ and r of X' for n order
    Calculate coefficients a using Eq. 7.5
    Create a linear filter F based on LP coefficients a
    Generate white Gaussian Noise Z
    Pass noise Z through linear filter F to get correlated noise Z'
    Calculate Correlation $Corr$ between X' and Z'
    if $Corr \geq \tau$ **then**
      Add noise to the time-series X' using Eq. 7.7

Algorithm 3: Time-Series Privacy Preservation

7.4.1 Datasets

We use two datasets to demonstrate the effectiveness and comprehensiveness of our obfuscation mechanism across three different scenarios. We use datasets containing (a) Pen-based Gestures [22], and (b) Mobile Swipes [149] as our experimental datasets. These two datasets demonstrate scenarios where a user is writing on a touchscreen, or swiping touchscreen, respectively. All these tasks are performed by users either using finger or stylus in different sessions. For readability, we use handwriting and swipes to refer these datasets in the rest of the chapter. Table 7.1 shows the summary statistics of the two datasets. The values are shown for total number of users, samples in each dataset, different functionality types (e.g. alphabets, swipe direction), and raw features associated with each dataset. We collected a total of 3,237,375 samples from a total of 95 users across all datasets. Among them, handwriting dataset has the most number of recorded samples. We now describe the context of each dataset in detail.

Handwriting Dataset: To demonstrate the applicability and effectiveness of our obfuscation mechanism in a latest and user-centered technology such as stylus, we use a dataset provided by [22]. The original purpose of collecting this dataset was to help developers and designers in determining the most suitable gesture recognition

Table 7.1: Statistics of Datasets

Dataset	# of Users	# of Samples	Possible Types	# of Raw Features
Handwriting (L)	30	779,800	26	4
Handwriting (D)	30	299,900	10	4
Swipes	35	2,157,675	4	3
Total:	95	3,237,375		

algorithm in various contexts/scenarios by fine-tuning different parameters such as speed, human distinguishability, etc. The features used in this dataset could be used in the context of end-user application where a user uses stylus pen to write on his mobile device to perform a certain functionality. In our experiments, we use gesture features for 26 letters of alphabets and 10 numeric digits of a total of 30 participants. Each alphabet and digit is recorded 10 times making a total of 300 gestures per letter/numeric digit.

Swipes Dataset: This dataset contains touch swipes performed by users on mobile devices for various purposes such as scrolling documents, images, playing games, etc. The dataset is provided by [149] with the purpose to investigate automated user authentication techniques on mobile devices. We filter this dataset based on the number of samples recorded[2] against each user, with a total of 35 users. We filter our all the swipes which have less than 5 data points, and finally extract 2,175,675 total number of samples. More details on the datasets is given in the paper [149].

7.4.2 Features Identification

Both datasets contain three common features i.e. *x, y coordinates and pressure* In addition to these two features, we use *pen angle* from handwriting dataset. We use these features because of two obvious reasons: i) these are the only raw features available in the given datasets and ii) these features have wide usage and high diversity across different applications. However, other features such as *phone orientation, Screen Orientation, Finger Orientation, X-Tilt , Y-Tilt,* and *Stroke Time* could also be used, if available. In general, the type and the number of features should not effect a privacy preserving mechanism. An ideal privacy preserving mechanism should be effective against different range of features.

[2] A gesture sample generates a sequence of data points of different length. This length depends on the sampling rate and duration of a sample, and varies across different devices.

7.4.3 Evaluation Metrics

In this section, we define privacy and utility metrics to evaluate the efficacy of our obfuscation framework. Privacy is indeed a subjective term and it varies greatly from application to application. It is more often based on the requirements and goal of applications. For-instance, one aspect of privacy is to protect a user from being tracked using the data in different sessions, another aspect is not being identified among group of users.

Hence, in our work, we use two different metrics for privacy in terms of untrackability and indistinguishability. Similarly, utility is highly subjective and it depends on the functionality that a user wants to achieve from a system. For-example, one individual is interested to get nearby locations from a mobile app while another individual only requires weather forecast based on his current location. We measure the utility of data after being obfuscated by our method in terms of error rate between original and obfuscated data. In the following, we describe these metrics in detail.

Untrackability: We define "untrackability" as an inability of an adversary/third-party to track a user across different sessions, for example, a user that reads and scrolls document at time t should not be identified by an adversary while reading another document at time *t+1* using his/her touch gestures. Similarly, letters or words written across different sessions should not be linked to the same individual. Theoretically, let's denote $X_t^{'}$ as an obfuscated time-series data of a user u_1 at time t consisting of n data points $\{x_{t,1}^{'}, x_{t,2}^{'}, ...x_{t,n}^{'}\}$ whereas $X_{t+1}^{'}$ is another obfuscated time-series of the same user at time $t+1$ having data points $\{x_{t+1,1}^{'}, x_{t+1,2}^{'}, ...x_{t+1,n}^{'}\}$. We then define "untrackability" as an inability to identify and link $X_t^{'}$ and $X_{t+1}^{'}$ as corresponding to user u_1 based on the distance/similarity between $X_t^{'}$ and $X_{t+1}^{'}$.

There are several distance or similarity metrics, such as euclidean distance, DTW, and cosine similarity, that could be used to measure the distance/similarity between two data streams. For our work, we use the Mean Absolute Error (MAE) to measure trackability of a user across sessions. MAE is a common measure of forecast error in time series analysis. Given two series $X_t^{'}$ and $X_{t+1}^{'}$ of a user u, MAE is calculated as:

$$MAE/untrackability = \frac{\sum_{t=1}^{n} \left| X_t^{'} - X_{t+1}^{'} \right|}{n} \tag{7.8}$$

i.e. our obfuscation mechanism should have high MAE between different time-serieses of the same user indicating that user cannot be tracked across his sessions.

Indistinguishability: This metric identifies whether an adversary is able to uniquely identify/distinguish user's data from a set of other users' data. In other

words, this metric checks the ability of an adversary to perform an inference attack on user $u_1's$ time-series data $X_t^{'}$, given a dataset of time-series D of multiple users $\{u_1, u_2, u_3, ...u_n\}$. Formally, the conditional probability of inferring an individual's time-series data $X_t^{'}$ from a dataset D of n time-series data where k time-series in D are highly likely to be $X_t^{'}$ is:

$$H(X_t^{'}|D) \geq \log_2 k$$

and the Information Gain (IG) is:

$$IG(X_t^{'}|D) \leq \log_2 n - \log_2 k$$
$$IG(X_t^{'}|D) \leq \log_2 n/k \tag{7.9}$$

Thus, distinguishability indicates how much information an adversary can infer about a user to distinguish him/her from other users and indistinguishability is the inverse of it.

Utility: The utility refers to correctly identifying the input and performing the intended functionality of an application, for instance, correctly recognizing letters (such as a, b, c) and swipe directions (left, right, up, and down) after the data obfuscation. In our experiments, we measure utility by computing MAE between original time-series X and obfuscated time-series $X^{'}$ as given in Eq. 7.10.

$$MAE/Utility = \frac{\sum_{i=1}^{n} \left|X_i - X_i^{'}\right|}{n}, \tag{7.10}$$

where X is an original time-series with data points $\{x_1, x_2, ...x_n\}$, $X^{'}$ is an obfuscated time-series having $\{x_1^{'}, x_2^{'}, ...x_n^{'}\}$ data points, and n is the number of data points. A high value for MAE indicates low utility and vice versa.

7.4.4 Experimental Setup

We categorize our experimental setup into two parts: i) Data Pre-processing and ii) Parameter Setting.

Data Pre-processing: This part involves preparing data to input into the obfuscation mechanism. First the data is filtered by removing broken, invalid, or empty values. Next, each time-series data, for example a gesture such as a single swipe or letter, is resampled to a fixed number of data points. This is necessary because mobile devices send data at a fixed sampling rate. In order to mimic this behaviour, we fixed sampling rate to 100 data points per gesture. After fixing sampling rate,

we normalized datasets using min-max scaling. This step is performed to make illustration of results consistent across different datasets. Finally, we partitioned the data from each dataset into two random but mutually exclusive sets. The first set has 80% of the samples, and the second has the remaining 20% of the samples. The larger set – 80% was used as training dataset, while 20% of the samples were used for "testing". We shall call this approach as the '80-20 approach' throughout the rest of this chapter. The training samples are used to train cluster and forecasting models, whereas testing samples are used to check the effectiveness of our obfuscation mechanism.

Parameter Setting: This part involves the procedure for parameter setting to evaluate the effectiveness of obfuscation mechanism expalined in Sec. 7.3. For each dataset, the obfuscation mechanism is evaluated against two parameters i.e. "Gaussian Noise Scale", and "Correlation Threshold". For each of these parameter values, the mechanism is run 10 and 5 times, respectively. The "Gaussian Noise Scale" parameter generates random noise within a given scale. We therefore tested our mechanism for different noise scales ranging from 0.1 to 1.0, with an interval of 0.1. The "Correlation Threshold" parameter decides the correlation level of noise as mentioned in Sec. 7.3.2.2 and Algo. 3. This parameter sets the noise correlation level with the data. We therefore, evaluated our mechanism for different threshold values ranging from 0.1 to 0.5, with an interval of 0.1. The different values of these two parameters explain the trade-off between privacy and utility of our obfuscation mechanism i.e. an increase in noise scale value indicates high privacy but less utility, and a decrease in scale value indicates good utility but less privacy. On contrary, an increase in correlation threshold reflects low privacy and high utility and a decrease in correlation threshold reflects high privacy and low utility.

7.5 Evaluation

In this section, we present and discuss the results of applying our privacy preservation mechanism in three different scenarios of mobile sensory data. Our aim of this experimental study is to show that a user tracking (privacy risk) from mobile application data is (significantly) dropped after applying our privacy preserving mechanism, with a reasonable level of utility of the mobile application functionality. We specifically evaluate in terms of (a) the ability of the obfuscated data to allow identifying/tracking returning user, (b) indistinguishability of an individual user after data obfuscation, and (c) that the utility of the intended functionality is preserved without much loss after the obfuscation.

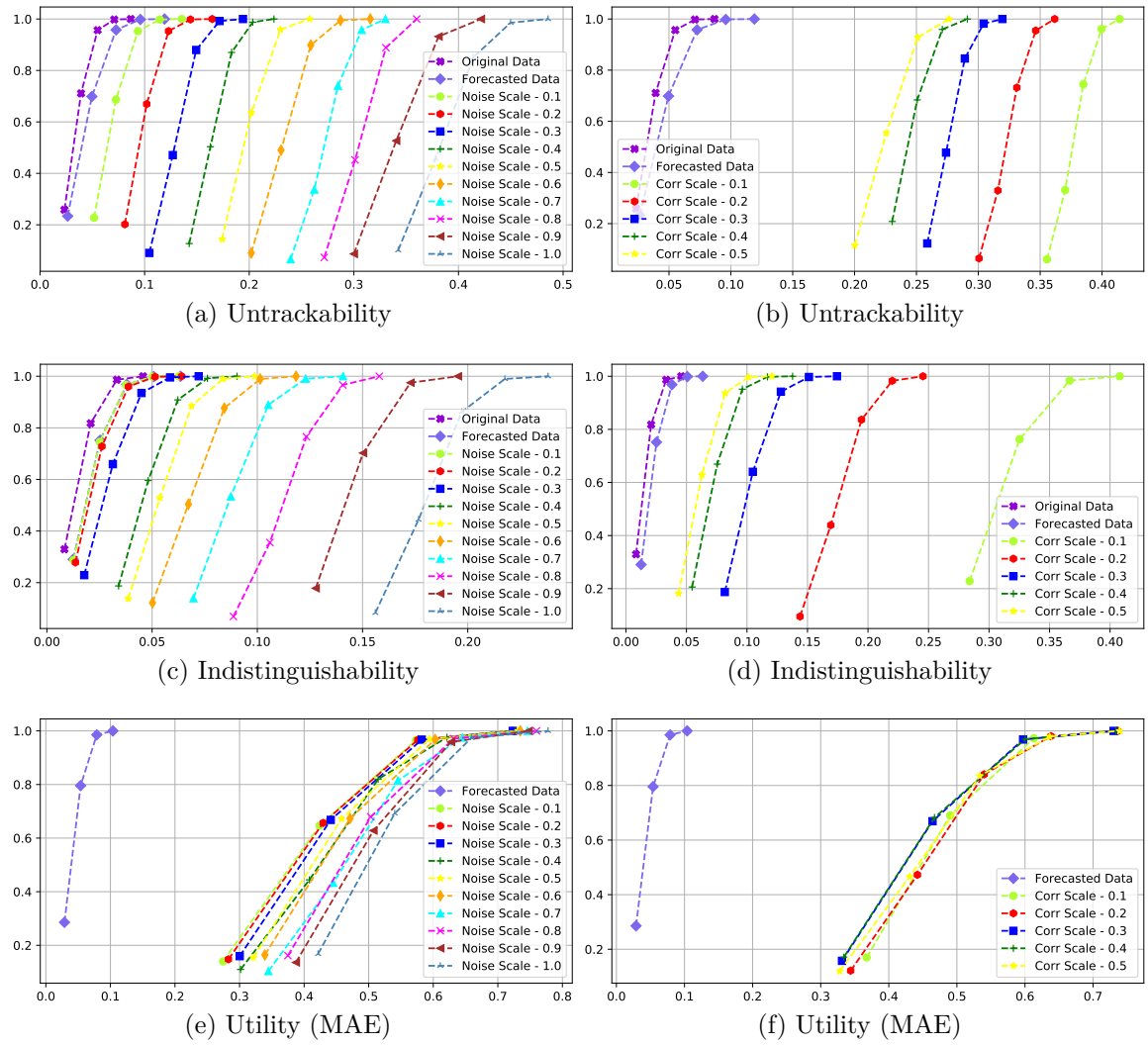
7.5.1 Privacy Evaluation

We start our discussion by first looking at the privacy aspects of original, forecasted, and obfuscated time-series data.

Handwriting Dataset (Letters): Our results indicate that in the original data users can be tracked across different sessions from the way they write. It indicates that user writing styles are quite unique from each other such that they could be recognized across sessions. We found similar results for forecasted data since it is a stabilized version of the original data. However, untrackability is significantly improved after applying obfuscation mechanism. Fig. 7.4a shows Cumulative Distribution Function (CDF) of untrackability against original, forecasted, and obfuscated data with 10 different noise scales and a fixed correlation threshold τ of 0.6. From the figure, it is clearly evident that increasing the noise scale increases the untrackability, thus increasing privacy. In the original data, 50% of the users have untrackability rate of just 4%, which slightly increases to 5% with forecasted data and then eventually increases to 38% with a noise scale of 1.0. Untrackability rate can be further increased using lower correlation threshold. It is worth noting here that we are trying to minimize two different threats here i.e. inference attack and noise filtering attack from an adversary. To reduce inference attack, a guassian noise scale must have high value whereas to reduce noise filtering attack, correlation threshold must have high value. Both of these parameters work in contrast to each other for instance, increasing correlation threshold increases the chances of inference attack while reducing the likelihood of noise filtering attack. Moreover, the above value are selected by keeping resulting utility loss parameter in mind as well. Increasing noise level or lowering correlated threshold, adversely effects utility.

We also examine results with different correlation threshold values and a fixed noise scale of 0.6. Results are shown in Fig. 7.4b indicating a clear trade-off between untrackability and correlation threshold, i.e. increasing correlation between noise and original data reduces privacy (decreases untrackability across sessions). The untrackability rate is 23.6% for 50% of users with a correlation threshold of 0.5 and increases to 38% with a correlation threshold of 0.1. The untrackability could be increased further by increasing the noise scale (currently fixed at 0.6).

Next, we evaluate our mechanism against indistinguishability. We found that an increase in noise scale increases indistinguishability and vice-versa. Similarly, a decrease in correlation threshold increases indistinguishability. Fig. 7.4c and 7.4d show CDF of indistinguishability against different noise scale and correlation values respectively. From Fig. 7.4c, we can observe that indistinguishability increases



(a) Untrackability (b) Untrackability

(c) Indistinguishability (d) Indistinguishability

(e) Utility (MAE) (f) Utility (MAE)

Figure 7.4: Cumulative Distribution Function (CDF) of Handwriting Dataset (Letters). Y-axes represents fraction of the participant population and X-axes shows Untrackability, Indistinguishability, and Utility (MAE) of varying noise scale levels (fig.(a), (c), (e)) and varying correlation threshold values (fig.(b), (d), (f)), respectively.

with an increase in noise scale. We found that indistinguishability increases to 19% with a noise scale of 1.0. Though less, these results are an indication that user uniqueness among other users can be decreased by our obfuscation mechanism. The indistinguisability can be further increased by lowering the correlation threshold from 0.6. In Fig. 7.4d, we observe similar trend across different correlation threshold values, where a decrease in correlation increases indistinguishability and vice versa. With a correlation threshold of 0.1 and a fixed noise scale of 0.6, 50% of users achieve 31% of indistinguishability, which starts decreasing with an increase in correlation

threshold.

Handwriting Dataset (Digits): We also analyzed the applicability of our obfuscation mechanism on digits (numeric) written with the stylus. Fig. 7.5a shows untrackability results for different noise scales and correlation threshold values. The untrackability rate of original and forecasted data is 3% approximately which increases with an increase in noise scale. For instance, with a noise scale of 1.0, untrackability reaches 38% for 60% of users. As mentioned earlier, untrackability rate may further increase if the correlation threshold is decreased from a fixed value of 0.6 to a lower value. However, to balance between the utility and indistinguishability, we used correlation threshold of 0.6 here. In Fig. 7.5b, we show untrackability rate for different correlation thresholds. The maximum untrackability rate with a correlation threshold of 0.1 is 38.5% for 60% of users.

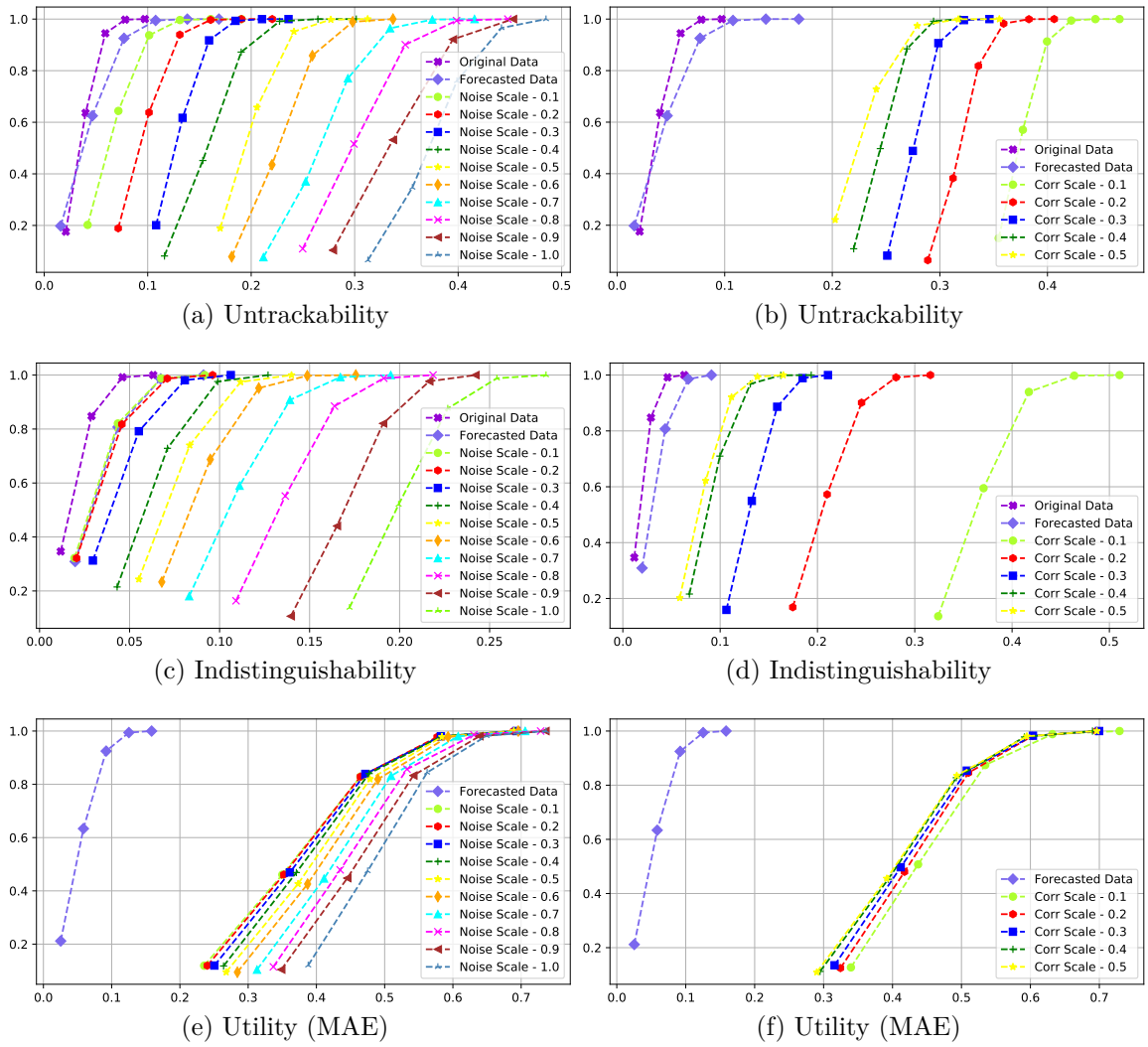
We also evaluate indistinguishability after applying obfuscation mechanism and found an expected trend i.e. an increase in noise scale increases indistinguishability. With the noise scale of 1.0 and correlation threshold of 0.6, the mechanism achieves 21% of indistinguishability, whereas the correlation of 0.1 and a noise scale of 0.6, offers indistinguishability of 38% for 60% of users. Fig. 7.5c and 7.5d show the respective CDFs of results.

Swipes Dataset: We apply our obfuscation mechanism on swipes dataset and found that untrackability and indistinguishability increases either with the increase in noise scale or a decrease in correlation threshold. For-instance, with a noise scale of 0.8, 80% of users have untrackability of 40% while indistinguishability reaches 50%. Similarly, if we decrease correlation to 0.1, untrackability reaches 35% for 60% of users and indistinguishability is 50%. Figure 7.6 shows results from Swipe dataset. These results indicate that a balance could be acheived between untrackability and indistinguishability if parameters are properly tuned.

7.5.2 Utility Evaluation

This subsection discusses the utility of obfuscated time-series data. Our results on preserving the functionality (utility) of an application after the obfuscation shows that high/reasonable accuracy could be achieved by tuning the parameter values, 'noise scale', and 'correlation threshold'.

Handwriting Dataset (Letters): The utility requirement in handwriting dataset is to correctly recognize letters that are written from the stylus-pen. The author of this dataset showed accuracy as high as 95% and as low as 20% with different classification algorithms [22]. In our experiments, we want to analyze if



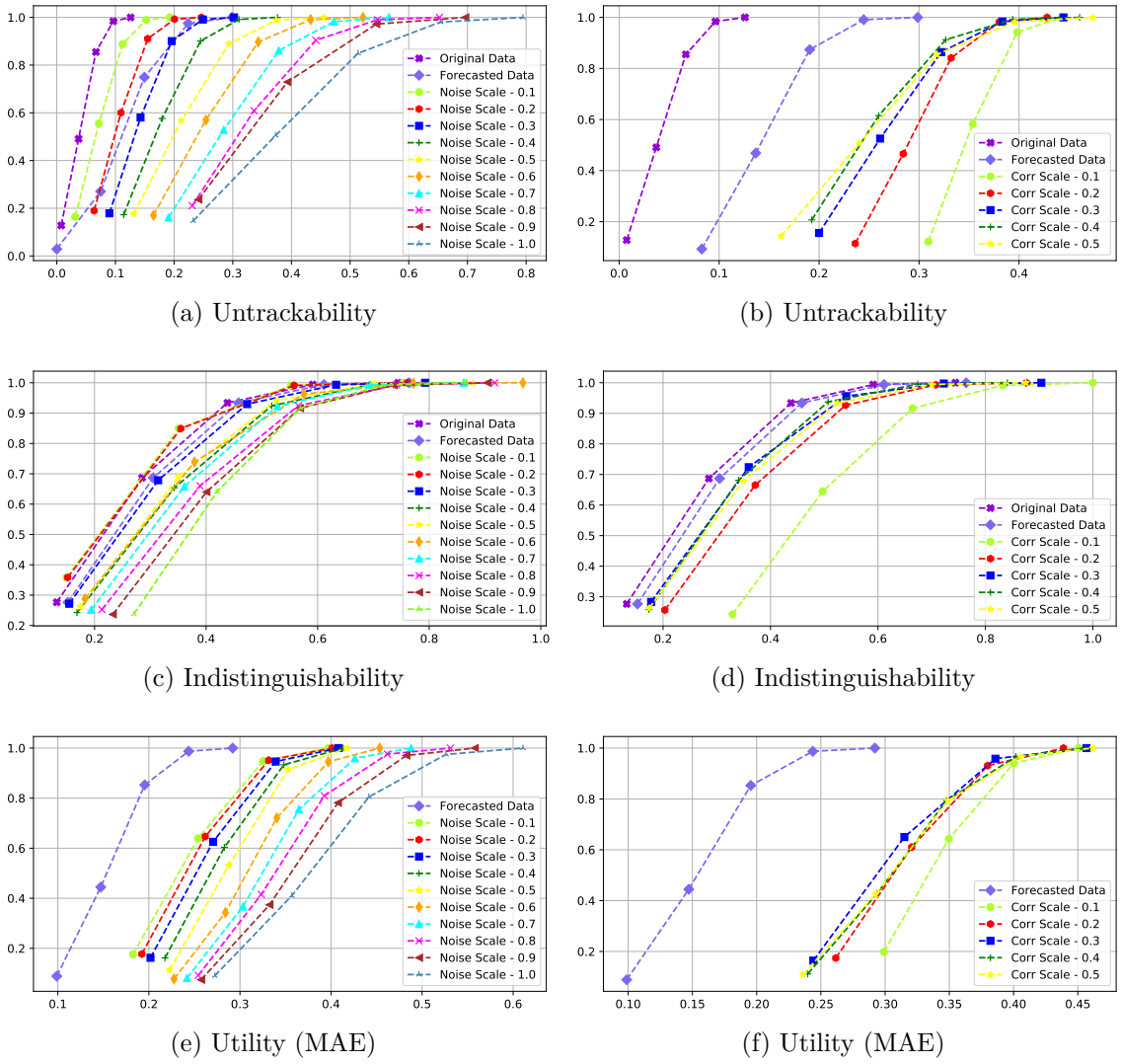
(a) Untrackability (b) Untrackability

(c) Indistinguishability (d) Indistinguishability

(e) Utility (MAE) (f) Utility (MAE)

Figure 7.5: Cumulative Distribution Function (CDF) of Handwriting Dataset (Digits). Y-axes represents fraction of the participant population and X-axes are Untrackability, Indistinguishability, and Utility (MAE) of varying noise scale levels (fig.((a), (c), (e)) and varying correlation threshold values (fig.(b), (d), (f)), respectively.

the MAE is low after the obfuscation. We observe the effect of our obfuscation mechanism on MAE with different noise scale and correlation threshold values. Results are indicated in Fig. 7.4e and Fig. 7.4f, respectively. In Fig. 7.4e, we can see that forecasted data has a MAE as low as 0.1. However, after the obfuscation, the MAE starts increasing with the increase in noise scale value. With the noise scale of 0.1, the MAE is 0.35 for 50% of users and 0.5 for 80% of users. We observe that increasing noise scale to 1.0 results in MAE of 0.48 for 50% of users. As mentioned earlier, we always have an option of increasing utility (decreasing MAE) by lowering fixed correlation threshold of 0.6 with an adverse impact on privacy. Similarly,



(a) Untrackability (b) Untrackability

(c) Indistinguishability (d) Indistinguishability

(e) Utility (MAE) (f) Utility (MAE)

Figure 7.6: Cumulative Distribution Function (CDF) of Swipes Dataset. Y-axes represents fraction of the participant population and X-axes shows Untrackability, Indistinguishability, and Utility (MAE) of varying noise scale levels (fig.(a), (c), (e)) and varying correlation threshold values (fig.(b), (d), (f)), respectively.

Fig.7.4f shows the effect of obfuscation mechanism on MAE with different correlation threshold values. The increase in correlation threshold decreases MAE to some extent but not as close to the original or forecasted data. The reason perhaps is that obfuscation mechanism is bounded with a fixed noise scale value.

Handwriting Dataset (Digits): The utility (MAE) of digits for different noise scales and correlation threshold values is shown in Fig. 7.5e and 7.5f, respectively. As expected, the utility loss increases with an increase in noise scales for instance, in Fig.7.5e the utility loss is 0.5 for 60% of users with the noise scale of 1.0 which

eventually decreases with the decrease in noise scale values. Similarly, in Fig. 7.5f, we see that an increase in correlation threshold, decreases utility loss. For instance, with a correlation threshold of 0.5, the utility loss is 0.42 for 60% of users which increases to 0.46 for a correlation threshold of 0.1.

Swipes Dataset: Our results on the utility of swipes dataset shows the MAE of 0.6 for a noise scale of 1.0 which decreases with the decrease in noise scale. For instance, 60% of users have an error of 0.3 with a noise scale of 0.5. Similarly, MAE decreases with an increase in correlation threshold. With a correlation scale of 0.3, MAE is 0.31 for 60% of users. Fig. 7.6e and 7.6f shows CDF of overall utility results.

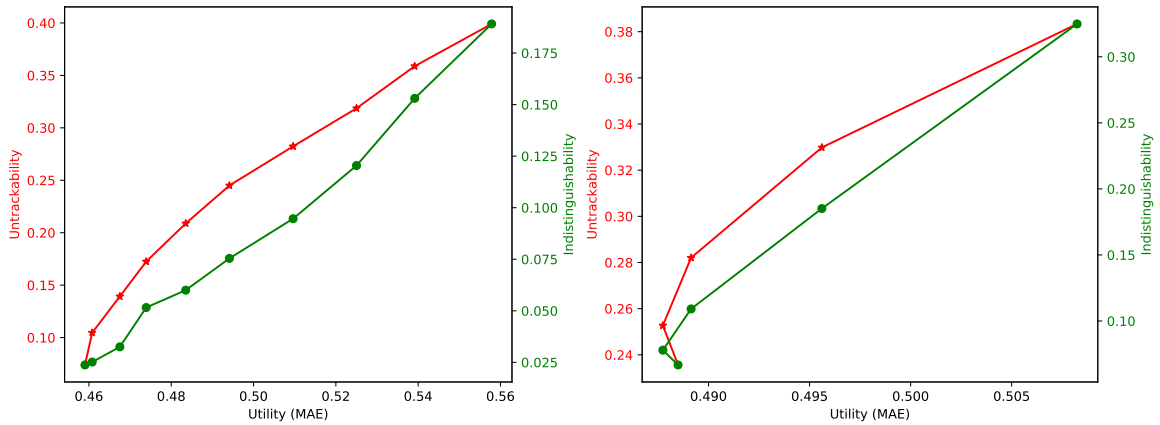
7.5.3 Privacy-Utility Trade-Off

This subsection summarizes the results in terms of tradeoff between privacy (untrackability and indistinguishability) and utility across different noise and correlation scale values. Fig.7.7 shows tradeoffs for all the datasets where the first row presents the results of different noise scale values and second row presents the results of different correlation thresholds. It is quite clear that an increase in MAE (x-axis) increases untrackability (y-axis right) and indistinguishability (y-axis left) for all values of noise and correlation threshold.

Fig.7.7a and 7.7b show the tradeoff result of handwriting dataset (letters). With different noise scale values (Fig.7.7a), the maximum achieved utility loss is 0.56 with 40% of untrackability and 19% indistinguishability, respectively. Similarly, for different correlation threshold values (Fig.7.7b), the maximum utility loss is 0.54 with 38% of untrackability and 32% of indistinguishability. As mentioned earlier, both privacy metrics are bounded by correlation and noise scales. This is necessary to limit threats of noise filtering and inference attack.

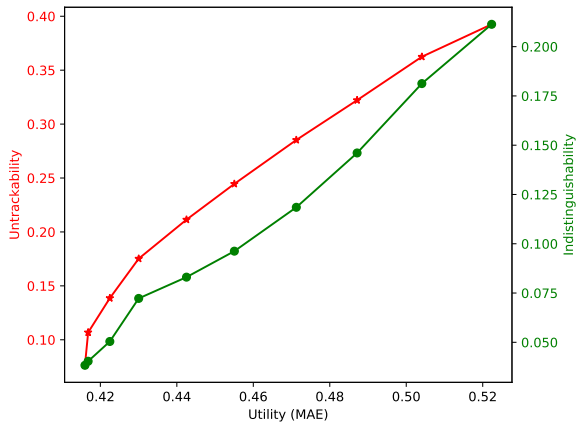
Fig.7.7c and 7.7d are tradeoff results of digits in handwriting dataset. In Fig.7.7c, results indicate a maximum utility loss of 0.52 with untrackability of 40% and indistinguishability of 22%, with different noise scale values. We observe similar trend with different correlation threshold values i.e. untrackability rate and indistinguishability is 38% with a utility loss of 0.48 (Fig. 7.7d).

Similarly, Fig. 7.7e and 7.7f shows tradeoff results of swipes dataset. We find that swipe achieves beteer utility-privacy tradeoff than handwriting dataset, particulalrly because a user handwriting style is almost similar throughout sessions whereas a swipe gesture is dependent on many factors such as mobile app, user environment, user current physical state (sitting, walking, etc.). We therefore, get better tradeoff results. For instance, with different noise scale, the average untrackability and

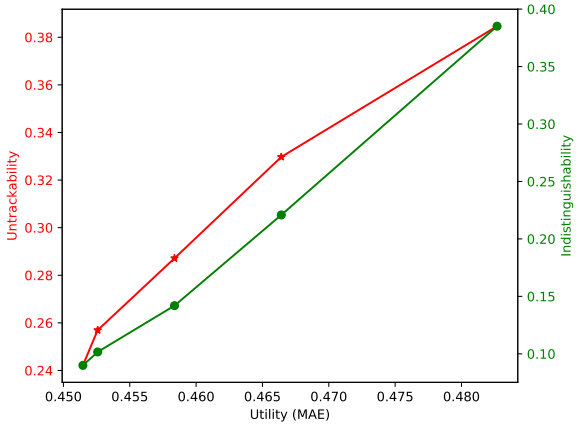


(a) Handwriting (Letters)

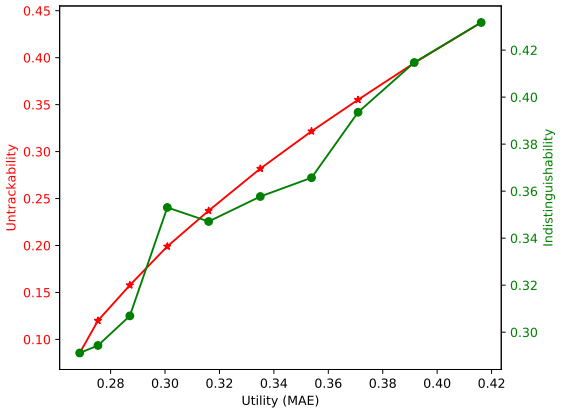
(b) Handwriting (Letters)



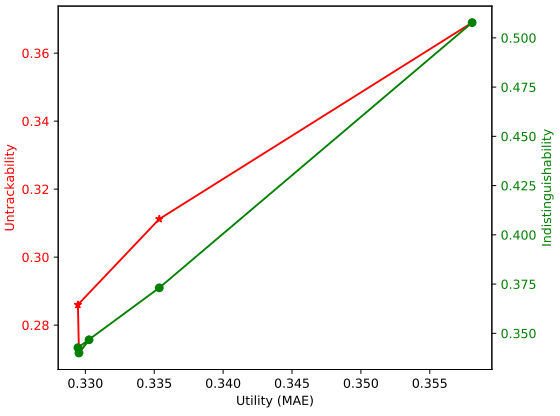
(c) Handwriting (Digits)



(d) Handwriting (Digits)



(e) Swipes



(f) Swipes

Figure 7.7: Privacy and Utility Trade-off. First Row shows trade-off with different Noise Scales (0.1 - 1.0), second row shows trade-off with different Correlation Thresholds (0.1 - 0.5)

indistinguishability is 45% and 43% respectively. On the other hand, we see that different correlation threshold performs better in terms of indistinguishability i.e.

50%.

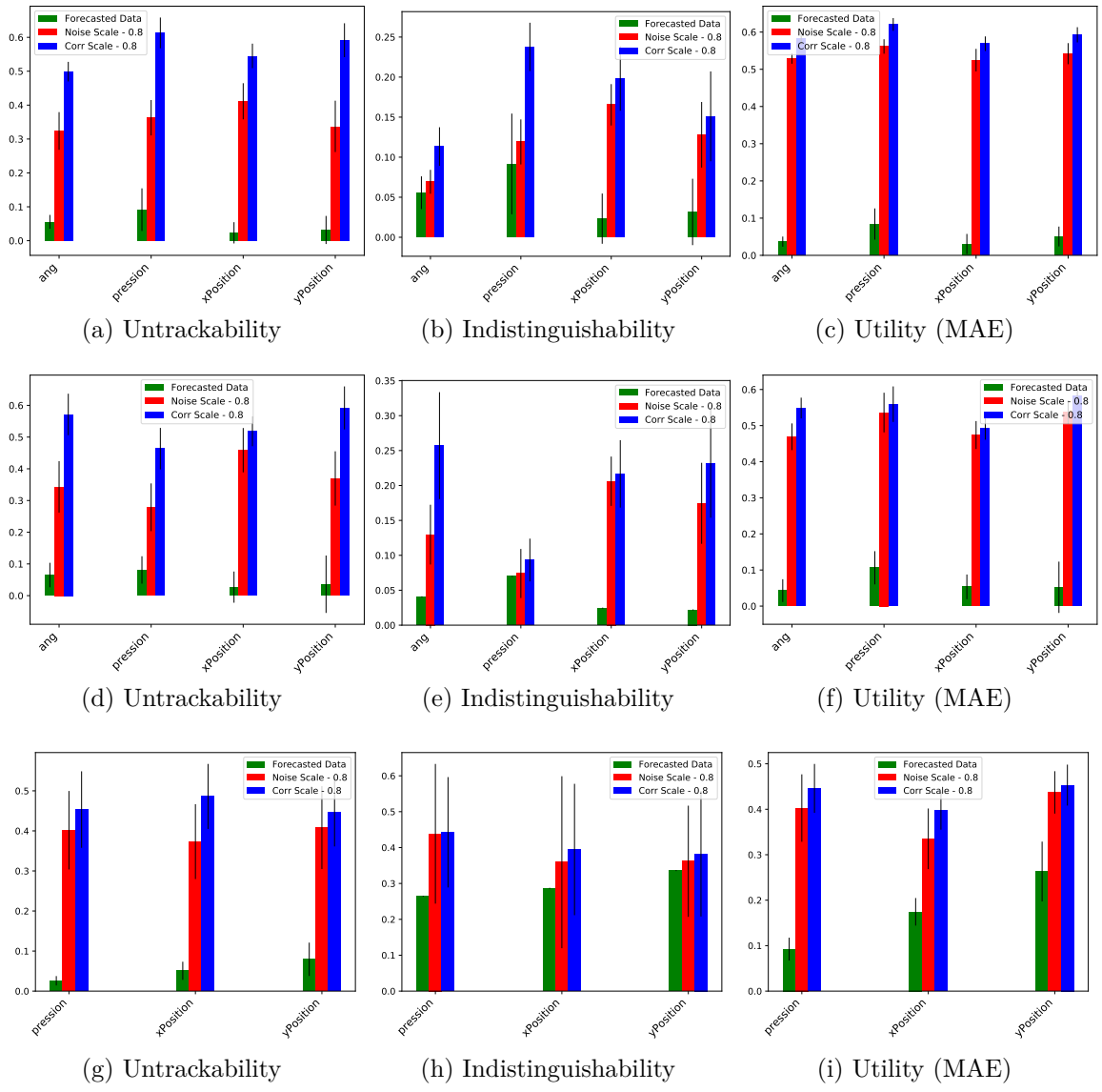
The trade-off results make it clear that our obfuscation mechanism is effective in achieving balanced privacy-utility for mobile based time-series data. Therefore, based on these results we select the default values of Gaussian noise and correlation threshold to discuss further results i.e. effect of obfuscation on features and specific functionality (letters, digits, swipe direction etc.) For handwriting datasets (letters and digits), we first fixed noise scale and correlation threshold to 0.8 and 0.6, and then also swap the values by fixing noise scale to 0.6 and correlation threshold to 0.8.

7.5.4 Effect on Features

This subsection discusses effect of our obfuscation mechanism on individual features. For handwriting dataset (letters), results are shown in Fig.7.8. With the correlation of 0.8, the highest untrackability rate is provided by *Pressure* followed by *yPosition*. However, with the noise scale of 0.8, *xPosition* offers high untrackability. These results indicate that *Pressure* and *yPosition* are effective against noise filtering attacks, while *xPosition* works best against inference attacks. We observe similar results for indistinguisbaility as illustrated in Fig.7.8b. We also found that MAE is normally higher for correlation scales as compared to noise scales (Fig.7.8c), for instance, *Pressure* is giving MAE of approximately 0.6 for a correlation scale of 0.8 whereas, it reduces to 0.56 for a noise scale. This trend is similar for other features as well.

Figures 7.8d, 7.8e, and 7.8f show feature-based untrackability, indistinguishability, and utility loss results of digits dataset. With the correlation threshold of 0.8, the highest untrackability rate is offered by *yPosition* of 58% followed by *pen angle* (55%). However, with the noise scale of 0.8, *xPosition* offers high untrackability of 48%. We observe that *pen angle* provides high indistinguishability of 25% with the correlation of 0.8 and *xPosition* offers indistinguishability of 22% with the noise scale of 0.8. Figure 7.8f indicates that utility loss is higher for correlation scales, for instance MAE of *yPosition* is 0.57 for a correlation of 0.8 which decreases to 0.51 for a nosie scale.

Results from the swipe dataset indicate that individual features offer better untrackability rate with a correlation threshold of 90.8. For instance, *xPosition*, offers 50% of average untrackability followed by *Pressure*. However, the untrackability rate with a noise scale of 0.8 is 42% for *yPosition*. For distinguishability, results from both parameters are quite comparable for example, *Pressure* achieves indis-



(a) Untrackability (b) Indistinguishability (c) Utility (MAE)

(d) Untrackability (e) Indistinguishability (f) Utility (MAE)

(g) Untrackability (h) Indistinguishability (i) Utility (MAE)

Figure 7.8: Effect of obfuscation mechanism on individual features. First row are the results from handwriting dataset (Letters), second row are the results from handwriting dataset (Digits), third row are the reuslts from swipes dataset.

tinguishability of 43% for both correlation threshold and noise scale value of 0.8. Similar to other datasets, we see that correlation threshold of 0.8 gives high MAE for all the features, but not more than 0.5.

7.5.5 Effect on Functionalities

We also analyze the effect of obfuscation on specific functionalities, for example, letters, digits, or swipes direction. Similar to features, we fixed the values of privacy

parameters and applied obfuscation. Table 7.2 shows results on functionalities for all the datasets. We observe that letters such as *j, t,* and *x* provides high untrackability rate of 40.2%, 40.8%, and 41.8% as well as high indistinguishablity of 61.3%, 59.5%, and 59.9% respectively. The untrackability of individual letters increases to 35% on average whereas indistinguishablity increases to 30% respectively. We find a utility loss of 0.5 on average for all the letters. This indicates that our mechanism is effectively balancing privacy and utility.

In digits datasets, we find that digit *8* provides high untrackability of 40.4% and digit *5* offers high indistinguishability of 53.9% respectively. Overall, untrackability rate is increased to 30%-32% whereas indistinguishability increases to 15%-17% from the original dataset. On average, the utility loss for all the digits is 0.5.

For swipes dataset, we find that *Down* swipe is least distinguishable, untrackable and also has low MAE. However, results from rest of the swipe directions, *Left, Right,* and *Up*, are also comparable and offer a balanced tradeoff against each metric.

7.5.6 Time Execution

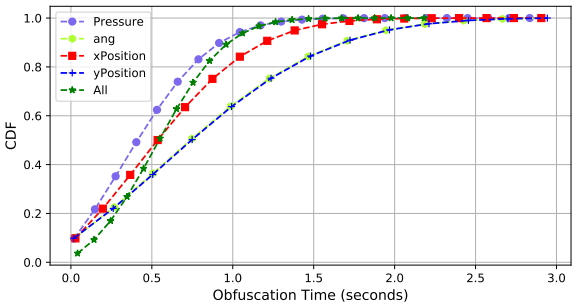
We showed that our obfuscation mechanism effectively provides a balance between privacy and utility. However, the gain in accuracy comes with the cost of the overhead of training the mechanism as well as applying the obfuscation at run-time. The overhead of training is reasonable as it could be performed during off-peak hours i.e. when a user is not using mobile device. On the other hand, the cost of applying obfuscation on-the-fly needs attention. Fig. 7.9a shows the time (seconds) it takes to obfuscate time-series data of handwriting dataset (letters). We find that 60% of *Pressure* time-series is obfuscated in 0.5 seconds, whereas *yPosition* and *pen angle* take 0.7 seconds for 50% of time-series. Overall, combining all the features, 50% of time-series data is obfuscated in 0.5 seconds.

Figure 7.9b shows obfuscation time of handwriting datasets with digits. Here we find that 60% of *yPosition* time-series is obfuscated in less than 0.5 seconds whereas 60% of *pen angle* takes 1 second for obfuscation. Overall, 50% of all time-series are obfuscated in 0.5 seconds.

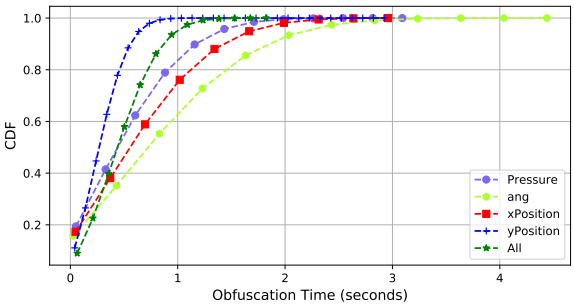
We observe that obfuscation time for swipes dataset is much less than the handwriting dataset. The average obfuscation time to obfuscate all the features is less than 0.07 seconds for 80% of time-serieses whereas individual time-series takes no more than 0.07 seconds for 60% of serieses.

Table 7.2: Effect of Obfuscation Mechanism on Utility Types with a Noise Scale 0.8

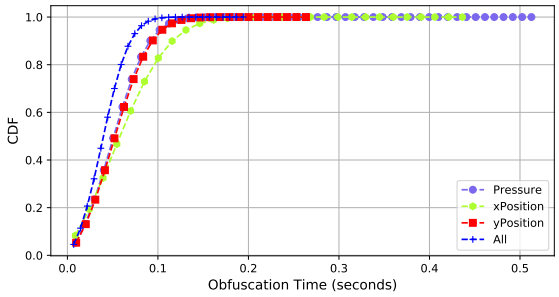
Types	Untrackability		Indisting-uishability		Utility		Types	Untrackability		Indisting-uishability		Utility	
	Org.	**Obf.**	**Org.**	**Obf.**	**Org.**	**Obf.**		**Org.**	**Obf.**	**Org.**	**Obf.**	**Org.**	**Obf.**
Handwriting Dataset (Letters)													
a	4.8%	34.0%	26.1%	56.0%	0.04	0.54	n	4.3%	37.1%	26.4%	57.2%	0.04	0.56
b	5.4%	35.0%	26.2%	58.9%	0.05	0.54	o	4.1%	33.8%	26.1%	58.6%	0.04	0.54
c	4.8%	36.1%	26.3%	58.7%	0.04	0.54	p	5.5%	36.5%	26.5%	58.5%	0.05	0.52
d	5.0%	35.9%	26.5%	57.9%	0.05	0.53	q	5.4%	34.2%	26.4%	56.8%	0.04	0.50
e	4.3%	35.5%	26.1%	56.9%	0.04	0.55	r	5.9%	34.0%	26.2%	57.0%	0.05	0.54
f	5.3%	37.2%	26.5%	58.7%	0.05	0.52	s	4.1%	32.6%	26.3%	56.1%	0.04	0.53
g	5.3%	38.7%	26.7%	57.7%	0.05	0.54	t	5.9%	41.8%	26.2%	59.9%	0.05	0.54
h	5.8%	34.2%	26.2%	56.7%	0.05	0.54	u	3.7%	33.8%	26.3%	57.3%	0.04	0.54
i	5.8%	38.5%	26.4%	57.6%	0.05	0.55	v	3.9%	32.6%	26.2%	55.9%	0.04	0.52
j	6.1%	40.2%	26.6%	59.4%	0.05	0.53	w	3.7%	34.1%	26.3%	56.8%	0.04	0.54
k	5.2%	36.4%	26.1%	56.6%	0.04	0.55	x	4.7%	40.8%	26.2%	61.3%	0.05	0.54
l	5.2%	31.9%	26.5%	56.3%	0.05	0.52	y	6.0%	37.7%	26.2%	57.9%	0.06	0.51
m	4.7%	36.9%	26.3%	57.2%	0.05	0.54	z	5.1%	32.3%	26.2%	57.0%	0.05	0.52
Handwriting Dataset (Digits)													
0	5.0%	36.9%	35.2%	51.7%	0.05	0.50	5	4.6%	36.5%	36.0%	53.9%	0.06	0.50
1	4.5%	34.3%	35.4%	50.9%	0.06	0.49	6	5.9%	35.6%	35.0%	50.6%	0.06	0.50
2	4.7%	34.5%	35.3%	52.0%	0.06	0.51	7	5.2%	36.1%	36.1%	51.3%	0.06	0.49
3	5.2%	34.6%	35.7%	51.7%	0.06	0.50	8	5.5%	40.4%	35.3%	52.3%	0.06	0.51
4	5.9%	38.4%	35.7%	52.0%	0.07	0.50	9	5.8%	35.4%	36.4%	52.8%	0.06	0.48
Swipes Dataset													
L	6.3%	37.0%	77.3%	33.6%	0.16	0.40	U	5.5%	40.6%	60.2%	37.1%	0.17	0.41
R	4.4%	37.1%	75.4%	35.8%	0.17	0.40	D	4.7%	41.6%	57.3%	32.6%	0.18	0.36



(a) Handwriting (Letters)



(b) Handwriting (Digits)



(c) Swipes

Figure 7.9: Obfuscation Time of Datasets

7.6 Conclusion

In this chapter, we propose a privacy preserving framework that protects user from the threat of trackability and distinguishability from mobile sensor data. The proposed framework overcomes the drawback of existing solutions by utilizing the concept of time-series modeling and forecasting. The framework is designed to handle unpredictable data and offers a balance between utility-privacy when sensors data is fluctuating. Moreover, the framework works in isolation from user or service providers/app developers and is data-type independent. We introduce correlated time-series noise in a framework that overcome the threat of noise filtering by an adversary. We empirically evaluate the framework on three different datasets and show that the risk from trackability and distinguishability is reduced whilst preserving the functionalities of the app even on the obfuscated data. We also demonstrate that our framework offers a balance between privacy-utility by fine-tuning correlation and noise parameters. In the next chapter, we present another privacy preserving method for protecting the privacy of web user data.

Chapter 8

Incognito: A Method for Obfuscating Web Data

Several works have been done on improving privacy of web data through obfuscation methods [103, 68, 196, 47]. However, these methods are neither comprehensive, generic to be applicable to any web data, nor effective against adversarial attacks. In this chapter, we propose a privacy-aware obfuscation method for web data addressing the identified drawbacks of existing methods. The web data with high predicted risk (as mentioned in Chapter 5) are obfuscated by our method to minimize the privacy risk using semantically similar data. Our method is resistant against adversary who has knowledge about the datasets and model learned risk probabilities using differential privacy-based noise addition. Experimental study conducted on two real web datasets validates the significance and efficacy of our method. Our results indicate that at most 0% privacy risk could be attained with our obfuscation method at the cost of average utility loss of 64.3%.

We organize this chapter as follows. In Section 8.1, we highlight the drawbacks of existing web obfuscation methods. Section 8.2 presents the methodology that we propose for obfuscating web users' data. In Section 8.3, we present our experimental results (Section 8.3.1), and conclude our work in Section 8.4.

8.1 Motivation

While there have been several works done on improving the privacy of users' web data through obfuscation methods, these existing methods primarily lack in considering all key aspects/features of web data privacy and they are not applicable to all the types of web data (e.g., search queries, posts, comments, reviews). Furthermore, these obfuscation methods are not resilient against adversarial attacks,

where given the adversary's knowledge about the obfuscation mechanism and the users' web behavior, they break the guarantees of protecting the privacy of users' web data.

This chapter tries to answer the question *How to develop a resilient obfuscation mechanism to improve the privacy of web data predicted with high risk, given the adversary has access to anonymized web data and knowledge of obfuscation algorithm?*. We propose a privacy-aware obfuscation method for web data that obfuscate high risk data entries with semantically similar lower risk data entries. The proposed obfuscation method can be applicable to any type of web applications, such as social networks, search engines, blogs, product review sites, and online forums. The main contributions of this chapter are as follows:

- We propose a novel obfuscation method to obfuscate high risk (predicted) data using semantically similar low risk data retrieved from the trained HMM at the cost of some loss in utility. Using differentially-private noise addition, our proposed method is resilient against adversary who has knowledge about the method, HMM probabilities and the training dataset and therefore is able to estimate the privacy risk values and could differentiate between the original and the obfuscated data by getting all possible paths in the HMM that have higher risks.

- We conduct an extensive empirical study using two real web datasets, the AOL dataset and our new app reviews dataset. Our results indicate that some obfuscated entries offer 0% privacy risk at the low cost of utility, however, there are some cases where obfuscated entries totally change the meaning of original entries. The addition of differentially private noise in the HMM model does not show significant difference in risk prediction, however, we see significant increase in utility loss for app reviews dataset i.e., 50% of the obfuscated entries has the utility loss of 64.3%, which increases to 90% for the perturbed entries by noise.

8.2 The Methodology for Obfuscating Web Data

In this Section, we describe how users' privacy risk in web data can be obfuscated if the predicted risk is high.

8.2.1 Obfuscation

Once a user data is identified as a privacy risk by our method based on the predicted privacy probability, the second step is to replace or modify the original high risk data with alternative data from different paths in the HMM to overcome the privacy risk with a loss in utility.

We quantify the utility loss (ul) in terms of semantic similarity between the original data X_x and the suggested data X_y.

$$ul(X_x, X_y) = 1.0 - sim(X_x, X_y), \tag{8.1}$$

where $sim(X_x, X_y)$ is a semantic similarity function [143] which returns the similarity value between the two data in the range 0 and 1. The larger the semantic similarity is the lower the utility loss is by using the alternative data.

The obfuscation module generates a list of alternative data suggestions (learned from the HMM model) along with their predicted privacy risk and calculated utility loss, from which one alternative data is chosen by the system to overcome privacy risk. It is important to note that the utility loss for the original data is 0.0 ($1.0 - sim(X_y, X_y) = 1.0 - 1.0 = 0.0$).

8.2.2 Adversarial Machine Learning

Given the training datasets and the HMM model learned probabilities can be accessed by an adversary, similar to all other existing obfuscation techniques our privacy-aware obfuscation technique can be susceptible to privacy attacks to learn the original data. The adversary is able to calculate or estimate the privacy risk values using the learned HMM probabilities and this could lead to privacy violation. For example, if a user's privacy risk increases with the data entered by the user and suddenly if the risk gets lower then the adversary might be able to guess that this could be a perturbed data by the system. In such a case, the adversary would be able to guess the actual data by getting all possible paths in the HMM that have higher risks.

In order to overcome this attack, we propose an adversarial machine learning technique by combining differential privacy-based noise addition with our HMM model. Noise is added in terms of counts/probabilities in the HMM model in order to perturb the original probability distribution. The magnitude of the noise depends on a privacy parameter ϵ and sensitivity S of query functions on the HMM model

by an adversary.

Definition 8.2.1 (**L1-sensitivity**). Given two count dictionaries T_1 and T_2, such that $|T_1| = |T_2|$ and T_1 and T_2 differ in only one element/entry's count, the L1-sensitivity of q query functions on both dictionaries, is measured as:

$$S = max_{\forall T_1,T_2} \sum_{i=1}^{q} |Q_i(T_1) - Q_i(T_2)|, \tag{8.2}$$

where $Q(\cdot)$ is a query function on a dictionary and $|\cdot|$ denotes the cardinality of a dictionary.

Theorem 8.1 (Noise addition with differential privacy). *Let Q be a set of query functions and S be the L1-sensitivity of Q. Then, ϵ-differential privacy can be achieved by adding random noise r, i.e., $Q_i^T \leftarrow Q_I^T + r$, where r is a random, i.i.d. variable drawn from a Laplace distribution with magnitude $b \geq S/\epsilon$.*

A differentially private dictionary release (publishing) corresponds to issuing count queries by an adversary:

$$select\ count(*)\ from\ dictionary\ where\ count/probability \geq x \tag{8.3}$$

Given a set of query functions Q, differential privacy adds noise drawn from Laplace distribution with magnitude b to the true response value. As shown in Theorem 8.1, b is determined by two parameters: (1) a privacy parameter ϵ and (2) the sensitivity S of Q. In this context, it is known that a single update in the count/probability value of an element in a dictionary can change the result of at most two count queries by a magnitude of at most one. Therefore, we add Laplace noise to each element in the dictionaries with $b = 2/\epsilon$. Positive noise is incorporated by incrementing the count/probability values, while negative noise requires subtracting the count probability values.

8.3 Evaluation

In this section, we present and discuss our findings on adversarial machine learning based differentially private web data obfuscation method. We have already explain the datasets and experimental settings in Section 5.3.1 and Section 5.3.2.1 of Chapter 5. Here, we discuss only results of our experiments.

8.3.1 Experiments and Results

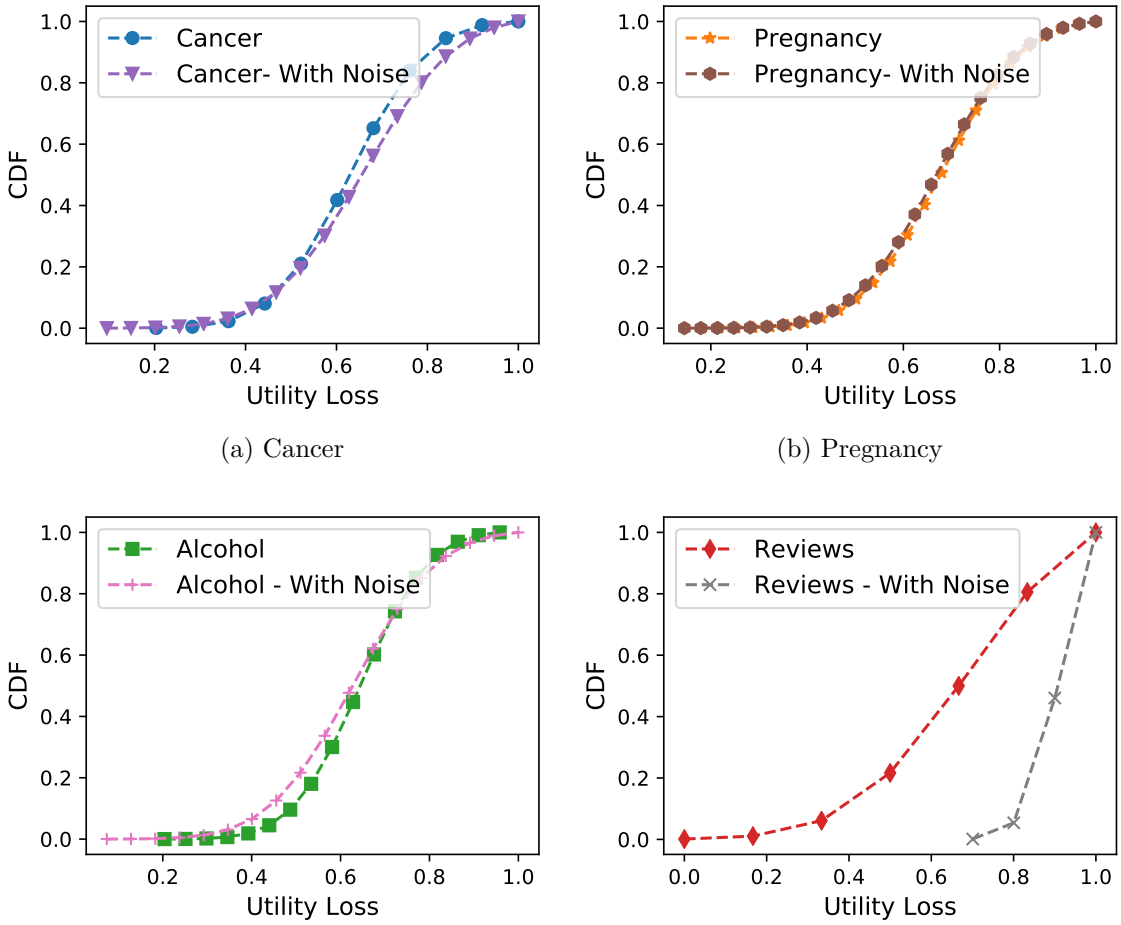
We first discuss our results on differentially private web data obfuscation method using some validation cases. We then present the efficiency results. For experimental purposes, we used ϵ-differential privacy based noise addition for adversarial machine learning where the privacy budget parameter is set to $\epsilon = 0.3$.

8.3.1.1 Obfuscation

In this Section, we discuss our results on the obfuscation of high risk web entries. We first present results for adversarial resistant obfuscation method and then move to few validation cases where original web entries are altered to low risk entries. As mentioned in Section 8.2, we obfuscated the data entries having higher privacy risk with lower risk entries that are semantically similar to original entries.

We compare original and obfuscated web entries using two metrics i.e., privacy risk and utility loss. We found that some obfuscated entries offer 0% privacy risk at the low cost of utility, however, there are some exceptions where obfuscated entries totally change the meaning of original entries. Moreover, our results indicate that the addition of differentially private noise in HMM model does not show significant difference in the risk calculations of web entries. However, we see a significant increase in utility loss for app reviews dataset, i.e., 50% of the altered entries has the utility loss of 64.3% (0.643), which increases to 90% (0.9) for the perturbed entries by noise. For AOL dataset, the utility loss remains between 58% to 63% (0.58 – 0.63) for 50% of both the perturbed entries with and without noise. The utility loss comparison between obfuscated data with and without noise for each topic is shown in Figure 8.1. Thus, these results indicate that the obfuscated entries come with the cost of loosing the original meaning of the data. We however are able to attain lower privacy risk, where the risk of all alternative entries suggested by our method are below 75% (0.75) and do not contain any name, location, specific writing pattern, uniformity in entries etc. On average, the privacy risk is reduced to almost 30% to 40% (0.3 – 0.4), however at the cost of utility.

Table 8.1 shows validation cases where some web entries are obfuscated to preserve privacy by our method at the cost of utility. We compare privacy risks of original and obfuscated web entries along with the utility loss before and after the addition of differentially-private noise. We take three cases (best, average, worst) from AOL topics and two cases (average, worst) from app reviews dataset. These cases indicate that the addition of noise not only improves privacy but also helps in securing the obfuscation method against adversary attacks. Similarly, in Figure 8.2 we show that



(a) Cancer

(b) Pregnancy

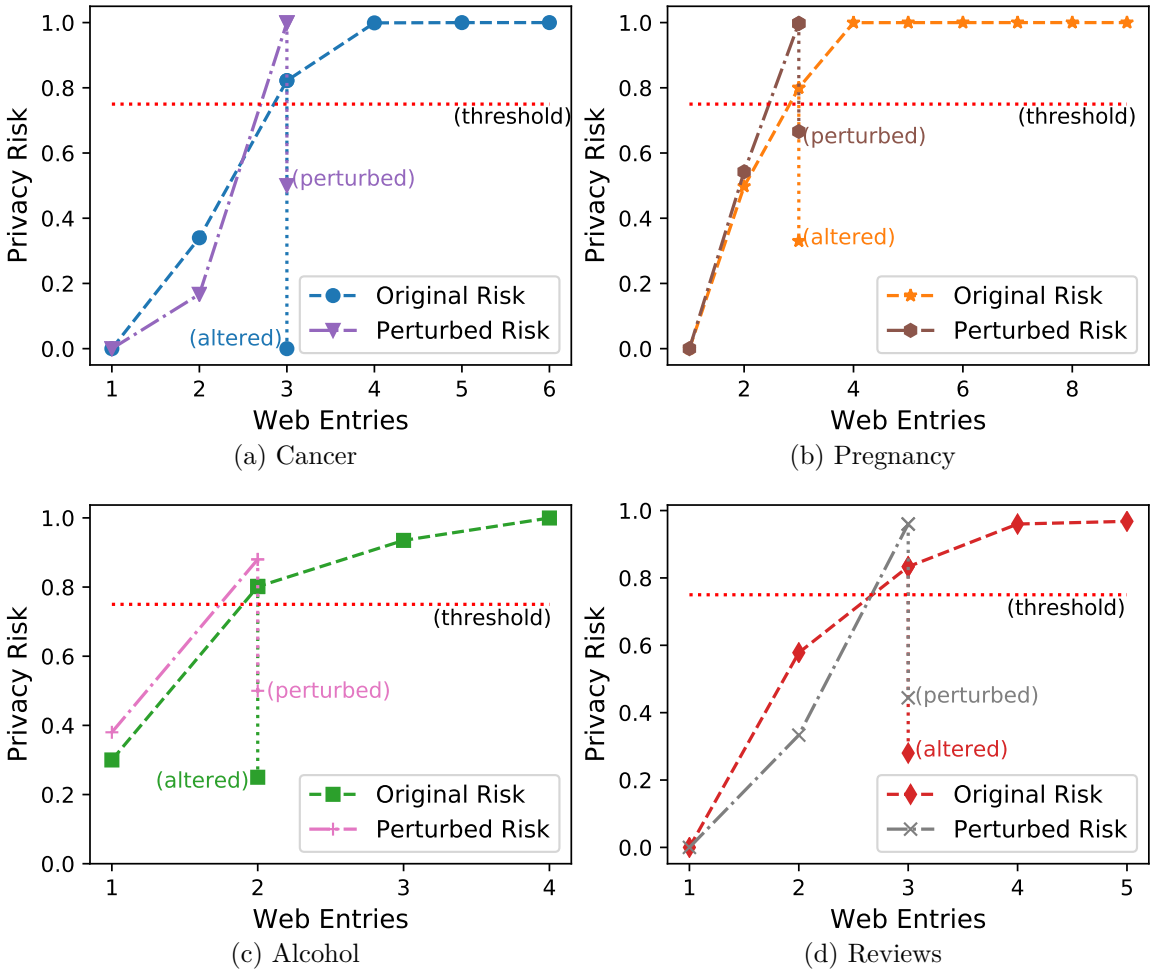
(c) Alcohol

(d) Reviews

Figure 8.1: Comparison of Utility Loss Between Obfuscated Data with and without Noise for Adversarial Machine Learning

the addition of noise dodges adversary by changing the original risk to perturbed risk. Even if the adversary has access to datasets, HMM probabilities, and knowledge of framework, our addition of differential-private noise does not allow the adversary to guess the original risk as well as the difference between original and obfuscated web data. Consider an example in Figure 8.2a where original risk reaches 83% (0.83) and suddenly falls down to 0% (0.0) risk by replacing original entry with obfuscated low risk entry. In this case, adversary is able to differentiate between original and obfuscated entries since sudden fall is an indication of obfuscated data. The inclusion of differential noise perturbs the risk such that it becomes difficult for an adversary to guess if it is an original or obfuscated entry. When a risk is above a certain threshold, the adversary model certainly replaces the original entry with low risk entries, however the addition of noise confuses the adversary to get to the

original entry.



(a) Cancer (b) Pregnancy

(c) Alcohol (d) Reviews

Figure 8.2: Improving Privacy and Resistance against Adversarial Attack

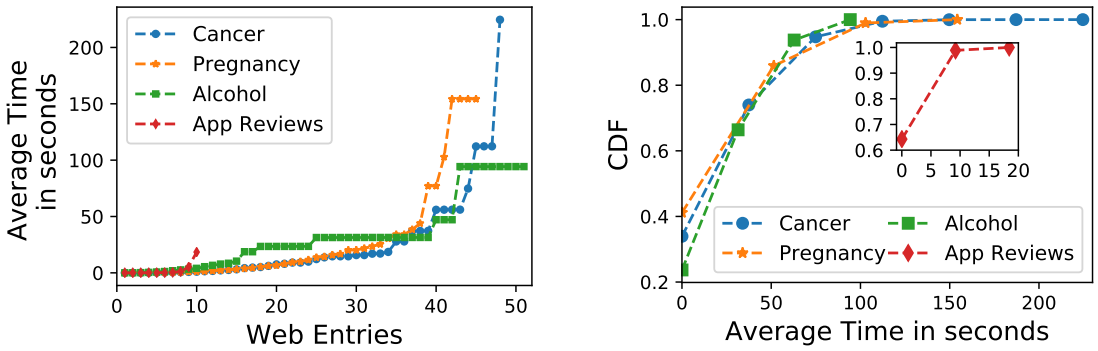
8.3.1.2 Efficiency

Finally, we investigate the time efficiency of our method. We found that time increases with the increasing number of data entries. The average time to predict, add noise, and then alternate high risk web entries is 0.0302, 0.0454, 0.0304, 0.0118 seconds per query of cancer, pregnancy, and alcohol, and app reviews, respectively. We found that the time to evaluate and obfuscate a query gets stable after entering certain number of queries. This is because either queries are repeating or we are training/updating our model continuously. However, when a new query comes in, which is not already seen by the training model, then it might take more time. Figure 8.3a shows the average time for each topic against the number of data entries. The maximum average time is 224 seconds for 47 cancer queries.

Table 8.1: Validation Cases - Comparison of Original and Perturbed Web Entries

Entry	Privacy Risk		Altered Entries		Privacy Risk		Utility Loss	
	Without Noise	With Noise	Without Noise	With Noise	Without Noise	With Noise	Without Noise	With Noise
stomach cancer signs	97.8%	98.30%	testiclular cancer	testiclular cancer	0%	50%	0.406	0.406
best clinic for prostate cancer	98.9%	99.11%	best clinic for prostate cancer research	best prostate cancer treatment centers	50%	66%	0.094	0.289
Inoperable bladder cancer	98.41%	98.81%	bladder sonogram	failure to diagnose bladder cancer	0%	50%	0.65	0.482
i need help on getting pregnant	95.11%	96.25%	chances of getting pregnant	tips on getting pregnant	0%	75%	0.52	0.296
First response early detection pregnancy tests	99.30%	99.34%	Early sign pregnancy	Early stages of pregnancy	50%	50%	0.364	0.411
can you take tyelnoe while pregnant	98.1%	98.33%	Can you take medicine while pregnancy	Can pregnancy women take tyelon 3	0%	50%	0.364	0.45
drug addiction help and new jersey	92.3%	94.66%	alcohol and tobacco law new jersey	drug abuse counseling	0%	75%	0.547	0.588
How to deal with an alcoholic	83.33%	97.57%	Christianity and the alcoholic	Programs for drug abuse	50%	66%	0.45	0.63
low cost drug addiction program in ny state	90.9%	97.43%	drug rehabilitation through programs	drug rehab programs	50%	75%	0.71	0.69
Nice excellent work...good work...excellent graphics	93.75%	93.89%	Nice good excellent	Pretty good	66%	66%	0.49	0.68
My 3 year old loves i!	88.64%	90.9%	My daughter loves it	Fun game for kids	50%	25%	0.26	0.76

Figure 8.3b shows the distribution of users against average time. We found that 85% of AOL users are processed within 50 seconds, while 62.5% of app reviews users are processed in 0.002 seconds. The significant time difference between the two datasets is because of two different techniques used for semantic matching. The `TF-IDF` approach [94] is pretty faster than the semantic similarity function [142] used for AOL because of various functions involved (word order, sentence order, `NLTK` semantic dictionary etc.).



(a) (b)

Figure 8.3: (8.3a) Average time in seconds in the increasing order of Web entries; and (8.3b) CDF average time per user.

8.4 Conclusion

Web data privacy has received much attention in the recent times due to the wide spread use of the internet and the growing concerns of privacy and confidentiality. Several works on obfuscation methods to counter privacy risks of web data have been conducted in the literature. However, these methods are not generic and applicable to any web data and they do not consider obfuscation for high risk predicted data using semantically similar data. In addition, adversarial machine learning for web data obfuscation has not been studied in the literature. In this chapter, we propose a privacy-aware obfuscation method that addresses the shortcomings of existing methods. We conducted experiments using two real web datasets and our experiment results show that our method is effective in predicting privacy risk in web data and obfuscating data that are predicted with high risk.

Chapter 9

Conclusion and Future Directions

This thesis analyzed the privacy issues in mobile and web platforms by specifically targeting the user tracking and identifiability problem. In addition, this thesis attempts to exploit suspicious third-party domains in a web dependency chain with a purpose to reveal their malicious intent such as malware execution, web tracking, and other privacy leaks. Finally, this thesis proposed privacy preserving frameworks for web and mobile platforms with an aim to reduce the identified privacy issues of user tracking and identifiability.

9.1 Summary and Conclusion

User touch gestures contain sufficient unique information for tracking

In chapter 4, we demonstrated that touch gestures can be used to uniquely identify (or fingerprint) users with high accuracy. This illustrates the threat of tracking based on touch gestures in general. Our results show that writing samples (on a touch pad) can reveal 73.7% of information (when measured in bits), and left swipes can reveal up to 68.6% of information. Combining different combinations of gestures results in higher uniqueness, with the combination of keystrokes, swipes and writing revealing up to 98.5% of information about users. We further show that, through our methodology, we can correctly re-identify returning users with a success rate of more than 90%. In summary, our results reveal some important findings, such as: i) Multiple samples of a gesture taken together reveal more accurate information about a user than a single sample of a gesture, ii) Swipes and handwriting carry more information as compared to taps and keystrokes, and iii) Features based on the area and pressure of the finger performing the gesture are the most informative.

Limitations: We have only used the cosine similarity metric to evaluate uniqueness. We have not investigated other similarity metrics such as Euclidean distance,

manhattan distance, and Jaccard index for comparison. Our main goal was to demonstrate the feasibility of touch-based tracking, and for that fixing a representative similarity metric was sufficient. Our quantitative methodology can also be extended by replacing the similarity metric with a machine learning classifier such as support vector machines (SVM) or k-nearest neighbours (kNN). Moreover, our focus in chapter 4 has been on four commonly used touch gestures. It is possible that other not-so widely used gestures such as pinch-in, pinch-out, drag, touch and hold, and multi-finger touches may lead to better unique identification rates of users and consequently user tracking.

We also need to verify the reliability and accuracy of our methodology as more users and more gesture data is collected through out TouchTrack app. Moreover, we did not at present take motion sensor features into account. There is likely a possibility that user uniqueness increases with the addition of these features. Our framework heavily relies on raw features extracted from android API; it assumes that raw features can be accessed from API's without requiring security permissions. While this assumption is valid for now, a number of other ways could be identified and used to extract raw features from mobile API's. Finally, we did not apply our framework on other mobile operating systems (OS) such as iOS and Windows, as we cannot access the raw features from these OS without having security permissions.

User web data entries may have high privacy risk even when the data is anonymized

In chapter 5, we presented a comprehensive privacy risk evaluation method for web data. We conducted experiments on two datasets, AOL search query and Android app reviews. The results show that our privacy prediction method is reliable enough to identify high risk web entries via three aspects of uniqueness, uniformity, and linkability. In general, our results reveal some important findings, such as: i) Privacy risk increases with sharing more data on the web even if the users' unique identities are not known, and ii) Privacy risk increases with sharing same data on the web i.e. users who entered same queries or reviews multiple times are easily recognizable. Therefore, user's privacy is at high risk when his web data is distinguishable from other users and is linkable with high confidence based on his previous history. Moreover, users who share or search for personal identifiable information (PII) on the web including names contact details, and address/location details of people are likely to be a victim of inference attack than users who do not share PII.

Limitations: Our proposed method has only used the basic HMM model to measure privacy probabilities and corresponding privacy risk. We have not investigated different probabilistic models such as Gaussian distribution, Dirichlet distribution, and maximum entropy Markov model (MEMM) for comparison. Our method can be extended by replacing the HMM model with other probabilistic methods. In addition, the AOL dataset (as used in most of the other related work) is outdated. Latest web datasets from search engines such as Google, and Yahoo as well as from social platforms such as Facebook may lead to high privacy risk rates. We also did not test our framework in an online environment.

We used fixed privacy budget parameter for our differentially private obfuscation method. Similarly, we fixed our privacy risk threshold to 0.75. We need to further investigate different parameter settings. Moreover, the semantic similarity function is not efficient to calculate risk in milliseconds, which requires other efficient and effective similarity measure approaches to be studied for real-time applications.

Trusted Websites unknowingly loading malicious content from 'suspicious' third-party domains

Chapter 6 explored the existence of dependency chains in the web ecosystem. By analyzing the Alexa's top-200K websites' dependency chains, we find that over 40% of websites have dependency chains, and therefore rely on an implicit trust model. Although the majority (84.91%) of websites have short chains (with levels of dependencies below 3), we found first-party websites with chains exceeding 30 levels. The most common *implicitly* trusted third-parties are well known operators (*e.g.,* `doubleclick.net`), but we also observed various less known implicit third-parties. Overall, 1.2% of all domains are classified as suspicious, yet their reach impacts 73% of Alexa websites. We hypothesised that this might create notable attack surfaces. These resources have remarkable reach — largely driven by the presence of highly central third-parties, *e.g.,* `google-analytics.com`. It was also particularly worrying to see that JavaScript resources loaded at level ≥ 2 in the dependency chain tended to have more aggressive properties, particularly as exhibited by their higher VTscore. This exposes the need to tighten the loose control over indirect resource loading and implicit trust: it creates exposure to risks such as malware distribution, search engine optimization (SEO) poisoning, malvertising and exploit kit redirection. We argue that ameliorating this can only be achieved through transparency mechanisms that allow web developers to better understand the resources on their webpages (and the related risks).

Limitations: Our study is dependent of the VirusTotal reports (and therefore the classification by various antivirus tools in VirusTotal)[1]. However, we acknowledge that VirusTotal is not perfect and therefore there may be noise within our classifications. To limit any issues, we do not rely on classifications by a single AV and, instead, set a VTscore threshold of above 10. This parameter was derived from a number of experiments and, where possible, we have presented results for multiple VTscore settings.

We also highlight the target of our study is a potentially dynamic landscape. Our reports from VirusTotal were captured in 2016, and our measurement are likely to not include domains which have been tagged as malicious in other years or periods that were not scanned by VirusTotal in 2016. Additionally, domains can be given an abnormally high VTscore for reasons identified earlier in this chapter (*e.g.,* malvertizing campaigns, *etc.*). While more accurate measurements could be achieved by scanning all URLs (rather than all domains) using VirusTotal's API, this solution might not be viable for a large-scale analysis due to limitations in the allowable request rate of the VirusTotal API.

Moreover, we would also like to highlight that similar issues exist when categorising websites (*e.g.,* as Business, IT, Adult *etc.*). WebSense did not return proper categories for 10% of third-party websites, and 15% of resources loaded in the dependency dataset. From these uncategorised websites, we found that 10% were deemed suspicious by VirusTotal, yet limitations in our methodoloy prevented us from a deeper understanding of their purpose.

Privacy preserving methods can protect web and mobile users from identification and tracking problems

The obfuscation method presented in the chapter 8 highlights two key aspects: *(1)* semantic similarity for obfuscated data and *(2)* resilient against adversarial machine learning. We obfuscate high risk entries by conducting experiments of our framework on two datasets, AOL search query and Android app reviews. The results show that our obfuscation method guarantees privacy against adversarial attacks with high effectiveness. Similarly, in chapter 7, we show that an effective privacy-utility balance can be achieved if time-series modeling and forecasting methods are used

[1]This was because malicious domains may circumvent a specific antivirus (AV) tool and, as suggested by previous studies [11, 21], some AV tools may not always report reliable results. Our experiments confirm this, as we found that AV tools often gave conflicting results (*i.e.,* they did not necessarily agree on classifications). We therefore used the VTscore records as an indicator of whether a domain (third party domains in this study) is benign or suspicious.

with privacy techniques. We have developed a framework that obfuscates mobile sensor data before sending to a remote server.

Limitations: Our proposed method used fixed privacy budget parameter for our differentially private obfuscation. Similarly, we fixed our privacy risk threshold to 0.75. We need to further investigate different parameter settings. Moreover, the semantic similarity function is not efficient to calculate risk in milliseconds, which requires other efficient and effective similarity measure approaches to be studied for real-time applications. We also did not test our framework in an online environment, and thus another important aspect for future investigation is to develop a real-time privacy risk prediction and obfuscation system where web entries are evaluated and obfuscated at run-time with or without user involvement. Perhaps a browser plug-in could be developed for our proposed method.

9.2 Future Directions

This thesis frames a number of avenues for future works.

Touch Gesture based Cross-Device Tracking on Mobile Devices

In chapter 4, we specifically focus on *single-user single-device* tracking scenario. An important aspect for future investigation is to verify our methodology for more complex scenario such as *multi-device single-user tracking*. This scenario tracks the same user across multiple devices and is quite realistic as the average number of connected devices per person is 3.5 [43]. Therefore, tracking user across multiple devices can create a potential risk to user privacy. Cross-device tracking is not straightforward to evaluate as it requires a more generalized approach. For example, it requires selecting "stable" features across all types of devices. This requires significantly more work in data collection and measurement to validate the stability of features across devices, which we intend to work on in the future. The stability of the gesture based fingerprint with time also needs to be investigated, as the user behaviour normally changes with time and it may have an impact on the accuracy of the uniqueness.

Mobile Bots Detection Using Touch Gestures

We can utilize our methodology given in chapter 4 to detect mobile bots. A Mobile bot is a fastest growing and most widespread types of mobile fraud. It runs

automatically once installed on a mobile device. These bots are embedded within seemingly legit mobile apps or websites. Recent advancements in mobile bots include i) Click Farms where a user can buy fake likes, dislikes, comments, or ratings of their published content, ii) Malicious Chatbots that can hack users credentials, iii) CrowdSource Bots that can perform tasks which are supposed to be performed by a user. A more advanced threat from Mobile Bots is Mobile Ad Fraud (for the advertisers) which are performed by bots hiding in legitimate apps. Examples of such threats include Click Spam, Click Injections, Fake Installs and SDK Spoofing. In this context, a future venue of work is to investigate users touch gestures to detect Bots action from a real user action. The aim is to utilize user touch gestures to identify a real user and a bot behavior. The crowd sourcing applications such as Mechanical Turk (Mturk) can be used for the data collection. Once data is collected having both Bot and real-user data, the uniqueness framework defined in chapter 4 can be used to identify bots from real users.

Measurement and Analysis of Mobile Apps and User Behaviour Study

Another possible future work is to perform static and dynamic analysis of mobile apps hosted on Google Play Store (perhaps Apple store as well) with the purpose to investigate if these apps are extracting raw sensor features for various purposes such as analytics, user experience, or tracking. Additionally, a study can be performed on users different touch behaviours while interacting with mobile. For this purpose, a framework should be designed that comprehensively investigate user touch behaviours in relation privacy and Human Computer Interaction (HCI).

Exploring the Temporal Behavior of Dependency Chains along with Dynamic Analysis of JavaScript Codes

In chapter 6, we do not investigate the impact of time on dependency chains. In future, we wish to perform longitudinal measurements to understand how these metrics of maliciousness evolve over time. We are particularly interested in understanding the (potentially) ephemeral nature of threats besides the inspection of temporal dynamics of resource dependency chains (*cf.* Section 6.2.1.2). Without this, we are reticent to draw long-term conclusions. Another line of work is understanding how level ≥ 1 JavaScript content creates inter-dependencies between websites. This is particularly noteworthy among hypergiants (*e.g.,* Google), who are present on a large number of first-party websites. Deep diving into other forms

of vulnerabilities (*e.g.,* cascading style-sheets (CSS)) would also be key for obtaining a wider understanding.

Similarly, it would be interesting to analyze the JavaScript running in the actual context, however, we are aware of the complexity that may be dealt to analyze the behavior of specific JavaScript code *i.e.,* it becomes difficult to extract all activities related to the individual JavaScript code. As a future work it would be interesting to further investigate their behaviour. For instance, performing graph analysis to understand how removing these hypergiants may impact the presence of interconnected suspicious third-parties. We are also keen to explore the efficacy of strategies like the same-origin policy, *e.g.,* to see how much third-party code is running in the first-party's context and thus can access its cookies. By opening our datasets and scripts to the wider research community, we hope that this will engender further research to help address the issues observed.

Privacy-Preserving Web Browser Plugin for Online Data

Based on our results from chapters 5 and 8, a possible future direction is to design and implement a browser plugin to predict privacy risks of user's online web data at run-time and then obscure high risk entries using probabilistic methods. The first part of this work learns privacy risk of users' Web entries at the run-time, using the probabilistic calculations of three different types of users' private/sensitive information: uniformity, uniqueness, and linkability of data. Based on the risk score predicted in the first part, the second part is to obscure high risk data entries by recommending alternative data entries that have low risk as calculated from our model. The utility loss of the recommended Web entries will also be presented to the user along with the calculated risk scores. The framework is made resilient to adversarial attacks, where the adversary with the knowledge of the model and calculated probabilities can make inferences about the actual data and the obfuscated data. This is achieved by adding noise to the probabilities using differential privacy method.

Bibliography

[1] gensim: Topic modelling for humans. https://radimrehurek.com/gensim/, 2018. Accessed on: 12-01-2018. (On page 78.)

[2] Natural language toolkit. http://www.nltk.org, 2018. Accessed on: 12-01-2018. (On page 78.)

[3] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. Fpdetective: dusting the web for fingerprinters. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1129–1140. ACM, 2013. (On page 31.)

[4] Jagdish Prasad Achara, Gergely Acs, and Claude Castelluccia. On the unicity of smartphone applications. In *Proceedings of the 14th ACM Workshop on Privacy in the Electronic Society*, pages 27–36. ACM, 2015. (On page 12.)

[5] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *ACM Sigmod Record*, volume 29, pages 439–450. ACM, 2000. (On page 124.)

[6] Rami Al-Rfou, William Jannen, and Nikhil Patwardhan. Trackmenot-so-good-after-all. *arXiv preprint arXiv:1211.0320*, 2012. (On page 34.)

[7] Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. Sensegen: A deep learning architecture for synthetic sensor data generation. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 188–193. IEEE, 2017. (On page 37.)

[8] Miguel E Andrés, Nicolás E Bordenabe, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. Geo-indistinguishability: Differential privacy for location-based systems. *arXiv preprint arXiv:1212.1984*, 2012. (On page 38.)

[9] AP. Google has been tracking your movements even if you told it not to. https://www.news.com.au/technology/gadgets/mobile-phones/google-has-been-tracking-your-movements-even-if-you-told-it-not-to/news-story/bb9eb906387ffd2295e8b17b24b7d883, Aug 2018. (On page 2.)

[10] Robert M Arlein, Ben Jai, Markus Jakobsson, Fabian Monrose, and Michael K Reiter. Privacy-preserving global customization. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 176–184. ACM, 2000. (On page 25.)

[11] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014. (On page 160.)

[12] Richard Atterer, Monika Wnuk, and Albrecht Schmidt. Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In *Proceedings of the 15th international conference on World Wide Web*, pages 203–212. ACM, 2006. (On page 13.)

[13] Ero Balsa, Carmela Troncoso, and Claudia Díaz. OB-PWS: obfuscation-based private web search. In *IEEE Symposium on Security and Privacy, SP 2012, 21-23 May 2012, San Francisco, California, USA*, pages 491–505, 2012. (On pages 6 and 36.)

[14] Muhammad Ahmad Bashir, Sajjad Arshad, William Roebertson, and Christo Wilson. Tracing information flows between ad exchanges using retargeted ads. In *USENIX Security Symposium*, 2016. (On page 30.)

[15] BBC. Facebook's data-sharing deals exposed. https://www.bbc.com/news/technology-46618582, Dec 2018. (On page 2.)

[16] Boldizsár Bencsáth, Gábor Pék, Levente Buttyán, and Márk Félegyházi. Duqu: Analysis, detection, and lessons learned. In *ACM European Workshop on System Security (EuroSec)*, volume 2012, 2012. (On page 110.)

[17] David Berend, Shivam Bhasin, and Bernhard Jungk. There goes your pin: Exploiting smartphone sensor fusion under single and cross user setting. In *Proceedings of the 13th International Conference on Availability, Reliability and Security*, page 54. ACM, 2018. (On page 117.)

[18] Shlomo Berkovsky, Nikita Borisov, Yaniv Eytani, Tsvi Kuflik, and Francesco Ricci. Examining users' attitude towards privacy preserving collaborative filtering. In *Workshop on Data Mining for User Modeling, Online Proceedings*, page 28, 2007. (On page 24.)

[19] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. Mediation of user models for enhanced personalization in recommender systems. *User Model. User-Adapt. Interact.*, 18(3):245–286, 2008. (On page 22.)

[20] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. The impact of data obfuscation on the accuracy of collaborative filtering. *Expert Syst. Appl.*, 39(5):5033–5042, 2012. (On page 26.)

[21] Antonia Bertolino, Gerardo Canfora, and Sebastian G. Elbaum, editors. *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 1*. IEEE Computer Society, 2015. (On page 160.)

[22] François Beuvens and Jean Vanderdonckt. Usigesture: An environment for integrating pen-based interaction in user interface development. In *Research Challenges in Information Science (RCIS), 2012 Sixth International Conference on*, pages 1–12. IEEE, 2012. (On pages 131 and 138.)

[23] Joanna Biega, Ida Mele, and Gerhard Weikum. Probabilistic prediction of privacy risks in user search histories. In *Proceedings of the First International Workshop on Privacy and Secuirty of Big Data, PSBD@CIKM 2014, Shanghai, China, November 7, 2014*, pages 29–36, 2014. (On pages 6, 36, and 71.)

[24] Joanna Asia Biega, Krishna P. Gummadi, Ida Mele, Dragan Milchevski, Christos Tryfonopoulos, and Gerhard Weikum. R-susceptibility: An ir-centric approach to assessing privacy risks for users in online communities. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 365–374, New York, NY, USA, 2016. ACM. (On page 36.)

[25] Igor Bilogrevic, Kévin Huguenin, Stefan Mihaila, Reza Shokri, and Jean-Pierre Hubaux. Predicting users' motivations behind location check-ins and utility implications of privacy protection mechanisms. In *22nd Network and Distributed System Security Symposium (NDSS)*, 2015. (On page 38.)

[26] Vincent Bindschaedler and Reza Shokri. Synthesizing plausible privacy-preserving location traces. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pages 546–563. IEEE, 2016. (On pages 21 and 38.)

[27] C Bo, L Zhang, and X SilentSense Li. Silent user identification via dynamics of touch and movement behavioral biometrics. In *The 19th Annual International Conference on Mobile Computing and Networking (MobiCom)*, pages 187–190, 2013. (On page 42.)

[28] Karoly Boda, Adam Mate Foeldes, Gabor Gyoergy Gulyas, and Sandor Imre. User Tracking on the Web via Cross-Browser Fingerprinting. *Information Security Technology for Applications*, 7161:31–46, 2012. (On page 28.)

[29] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. Mobile device identification via sensor fingerprinting. *arXiv preprint arXiv:1408.1416*, 2014. (On page 30.)

[30] Antoine Boutet and Mathieu Cunche. A privacy-preserving mechanism for requesting location data provider with wi-fi access points. 2018. (On pages 21 and 38.)

[31] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015. (On pages 127, 191, and 192.)

[32] Justin Brookman, Phoebe Rouge, Aaron Alva, and Christina Yeung. Cross-device tracking: Measurement and disclosures. *Proceedings on Privacy Enhancing Technologies*, 2:113–128, 2017. (On page 12.)

[33] Jason Brownlee. How to check if time series data is stationary with python. https://machinelearningmastery.com/time-series-data-stationary-python/, December 2016. (On page 126.)

[34] Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors. *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*. Springer, 2007. (On page 22.)

[35] Tomasz Bujlow, Valentín Carela-Español, Josep Sole-Pareta, and Pere Barlet-Ros. A survey on web tracking: Mechanisms, implications, and defenses. *Proceedings of the IEEE*, 105(8):1476–1510, 2017. (On pages 6, 14, 15, 17, 18, 88, and 118.)

[36] Fei Cai, Shangsong Liang, and Maarten De Rijke. Time-sensitive personalized query auto-completion. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pages 1599–1608. ACM, 2014. (On pages 124 and 193.)

[37] John Canny. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 238–245. ACM, 2002. (On page 26.)

[38] Julio Canto, Marc Dacier, Engin Kirda, and Corrado Leita. Large scale malware collection: lessons learned. In *IEEE SRDS Workshop on Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems*. Citeseer, 2008. (On page 97.)

[39] Jordi Castellà-Roca, Alexandre Viejo, and Jordi Herrera-Joancomartì. Preserving user's privacy in web search engines. *Computer Communications*, 32(13):1541–1551, 2009. (On page 33.)

[40] Jordi Castellà-Roca, Alexandre Viejo, and Jordi Herrera-Joancomartí. Preserving user's privacy in web search engines. *Computer Communications*, 32(13-14):1541–1551, 2009. (On page 34.)

[41] Sophie Cerf, Vincent Primault, Antoine Boutet, Sonia Ben Mokhtar, Robert Birke, Sara Bouchenak, Lydia Y Chen, Nicolas Marchand, and Bogdan Robu. Pulp: achieving privacy and utility trade-off in user mobility data. In *Reliable Distributed Systems (SRDS), 2017 IEEE 36th Symposium on*, pages 164–173. IEEE, 2017. (On pages 21 and 38.)

[42] Abdelberi Chaabane, Mohamed Ali Kaafar, and Roksana Boreli. Big friend is watching you: Analyzing online social networks tracking capabilities. In *Proceedings of the 2012 ACM workshop on Workshop on online social networks*, pages 7–12. ACM, 2012. (On page 15.)

[43] Dave Chaffey. How many connected devices do consumers use today?. http://www.smartinsights.com/traffic-building-strategy/

integrated-marketing-communications/many-connected-devices-use-today-chartoftheday/, 2016. (On page 161.)

[44] Prima Chairunnanda, Nam Pham, and Urs Hengartner. Privacy: Gone with the typing! identifying web users by their typing patterns. In *PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9-11 Oct., 2011*, pages 974–980, 2011. (On page 71.)

[45] Ramnath K Chellappa and Raymond G Sin. Personalization versus privacy: An empirical examination of the online consumer's dilemma. *Information technology and management*, 6(2-3):181–202, 2005. (On page 24.)

[46] Jingdong Chen, Jacob Benesty, Yiteng Huang, and Simon Doclo. New insights into the noise reduction wiener filter. *IEEE Transactions on audio, speech, and language processing*, 14(4):1218–1234, 2006. (On page 193.)

[47] Terence Chen, Roksana Boreli, Mohamed Ali Kâafar, and Arik Friedman. On the effectiveness of obfuscation techniques in online social networks. In *Privacy Enhancing Technologies - 14th International Symposium, PETS 2014, Amsterdam, The Netherlands, July 16-18, 2014. Proceedings*, pages 42–62, 2014. (On pages 20, 35, and 148.)

[48] Terence Chen, Abdelberi Chaabane, Pierre Ugo Tournoux, Mohamed Ali Kaafar, and Roksana Boreli. How much is too much? Leveraging ads audience estimation to evaluate public profile uniqueness. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7981 LNCS:225–244, 2013. (On pages 28 and 43.)

[49] Richard Chow and Philippe Golle. *Proceedings of the 2009 ACM Workshop on Privacy in the Electronic Society, WPES 2009, Chicago, Illinois, USA, November 9, 2009*, chapter Faking contextual data for fun, profit, and privacy, pages 105–108. 2009. (On page 36.)

[50] John H Cochrane. Time series for macroeconomics and finance. *Manuscript, University of Chicago*, pages 1–136, 2005. (On page 191.)

[51] J.R. Corripio, D.M.A. González, A.L.S. Orozco, L.J.G. Villalba, J. Hernandez-Castro, and S.J. Gibson. Source smartphone identification using sensor pattern noise and wavelet transform. *5th International Conference on Imaging for Crime Detection and Prevention, ICDP 2013*, 2013. (On page 30.)

[52] M. Cunche, Mohamed Ali Kaafar, and R. Boreli. I know who you will meet this evening! linking wireless devices using wi-fi probe requests. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9, June 2012. (On page 12.)

[53] Chris Cuomo, Jay Shaylor, Mary Mcguirt, and Chris Francesani. 'gma' gets answers: Some credit card companies financially profiling customers. https://abcnews.go.com/GMA/TheLaw/gma-answers-credit-card-companies-financially-profiling-customers/story?id=6747461, Jan 2009. (On page 18.)

[54] Anupam Das, Gunes Acar, Nikita Borisov, and Amogh Pradeep. The web's sixth sense: A study of scripts accessing smartphone sensors. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1515–1532. ACM, 2018. (On pages 21, 38, 104, 117, and 121.)

[55] Anupam Das and Nikita Borisov. Poster : Fingerprinting Smartphones Through Speaker. *35th IEEE Symposium on Security and Provacy*, pages 2–3, 2014. (On pages 30 and 43.)

[56] Anupam Das, Nikita Borisov, and Matthew Caesar. Do you hear what i hear?: Fingerprinting smart devices through embedded acoustic components. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 441–452. ACM, 2014. (On pages 15 and 30.)

[57] Anupam Das, Nikita Borisov, and Matthew Caesar. Tracking Mobile Web Users Through Motion Sensors : Attacks and Defenses. *Ndss*, (February):21–24, 2016. (On pages 2, 12, 15, 30, 38, 117, and 121.)

[58] Anupam Das, Nikita Borisov, and Edward Chou. Every move you make: Exploring practical issues in smartphone motion sensor fingerprinting and countermeasures. *Proceedings on Privacy Enhancing Technologies*, 2018(1):88–108, 2018. (On pages 15, 21, 38, and 121.)

[59] Carl De Boor. *A practical guide to splines*, volume 27 of *Applied mathematical sciences*. Springer-Verlag New York, 1978. (On page 56.)

[60] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. Semantics-aware content-based recommender systems. In *Recommender Systems Handbook*, pages 119–159. 2015. (On page 22.)

[61] Alysha M De Livera, Rob J Hyndman, and Ralph D Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011. (On page 192.)

[62] Loh Chin Choong Desmond, Cho Chia Yuan, Tan Chung Pheng, and Ri Seng Lee. Identifying unique devices through wireless fingerprinting. In *Proceedings of the First ACM Conference on Wireless Network Security*, WiSec '08, pages 46–55, New York, NY, USA, 2008. ACM. (On page 12.)

[63] Loh Chin Choong Desmond, Cho Chia Yuan, Tan Chung Pheng, and Ri Seng Lee. Identifying unique devices through wireless fingerprinting. *Proceedings of the first ACM conference on Wireless network security - WiSec '08*, page 46, 2008. (On page 31.)

[64] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable. *Network and Distributed System Security Symposium (NDSS)*, (February):23–26, 2014. (On pages 2, 12, 30, and 117.)

[65] David A Dickey and Wayne A Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431, 1979. (On page 192.)

[66] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004. (On page 33.)

[67] Josep Domingo-Ferrer, Agusti Solanas, and Jordi Castellà-Roca. h (k)-private information retrieval from privacy-uncooperative queryable databases. *Online Information Review*, 33(4):720–744, 2009. (On page 20.)

[68] Josep Domingo-Ferrer, Agusti Solanas, and Jordi Castellà-Roca. h (k)-private information retrieval from privacy-uncooperative queryable databases. *Online Information Review*, 33(4):720–744, 2009. (On pages 33 and 148.)

[69] Charles Duhigg. How companies learn your secrets. https://www.nytimes.com/2012/02/19/magazine/shopping-habits.html?pagewanted=all, Feb 2012. (On pages 6, 18, 88, and 117.)

[70] Peter Eckersley. How Unique Is Your Browser? *Proc. of the Privacy Enhancing Technologies Symposium (PETS)*, pages 1–18, 2010. (On pages 27, 30, 33, 42, 43, and 46.)

[71] Electronic Frontier Foundation (EFF). Do Not Track (DNT). https://www.eff.org/issues/do-not-track. (On page 20.)

[72] Steven Englehardt and Arvind Narayanan. Online Tracking: A 1-million-site Measurement and Analysis. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*, pages 1388–1401, 2016. (On page 28.)

[73] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 1388–1401. ACM, 2016. (On page 97.)

[74] Murat A Erdogdu, Nadia Fawaz, and Andrea Montanari. Privacy-utility trade-off for time-series with application to smart-meter data. In *AAAI Workshop: Computational Sustainability*, 2015. (On page 38.)

[75] European Commission (EU). COOKIE SWEEP COMBINED ANALYSIS, REPORT. https://ec.europa.eu/newsroom/article29/item-detail.cfm?item_id=640605, Nov 2016. (On page 12.)

[76] European Union Agency For Network and Information Security (ENISA). Online privacy tools for the general public, Dec 2015. (On page 19.)

[77] European Union Agency For Network and Information Security (ENISA). Online tracking and user protection mechanisms, Dec 2017. (On pages 2, 20, and 21.)

[78] IBM XForce Exchange. Statcounter session hijack. https://exchange.xforce.ibmcloud.com/vulnerabilities/20506, 2005. (On page 100.)

[79] Marjan Falahrastegar, Hamed Haddadi, Steve Uhlig, and Richard Mortier. Anatomy of the third-party web tracking ecosystem. *Traffic Measurements Analysis Workshop (TMA)*, 2014. (On pages 6, 29, 87, and 92.)

[80] David Fifield and Serge Egelman. Fingerprinting web users through font metrics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8975:107–124, 2015. (On page 28.)

[81] Forcepoint. Master database url categories | forcepoint. https://www.forcepoint.com/product/feature/master-database-url-categories, 2019. (On pages 92 and 94.)

[82] Stat Counter Forum. http://www.statcounter.com/ counter/counter.js has malware inside it ! https://forum.statcounter.com/threads/http-www-statcounter-com-counter-counter-js-has-malware-inside-it.43792/, 2016. (On page 100.)

[83] Mario Frank, Ralf Biedert, Eugene Ma, Ivan Martinovic, and Dawn Song. Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE Transactions on Information Forensics and Security*, 8(1):136–148, 2013. (On page 42.)

[84] Matthew Fredrikson and Benjamin Livshits. Repriv: Re-imagining content personalization and in-browser privacy. In *32nd IEEE Symposium on Security and Privacy, S&P 2011, 22-25 May 2011, Berkeley, California, USA*, pages 131–146, 2011. (On pages 19 and 30.)

[85] Arik Friedman, Shlomo Berkovsky, and Mohamed Ali Kâafar. A differential privacy framework for matrix factorization recommender systems. *User Model. User-Adapt. Interact.*, 26(5):425–458, 2016. (On page 26.)

[86] Arik Friedman, Bart P Knijnenburg, Kris Vanhecke, Luc Martens, and Shlomo Berkovsky. Privacy aspects of recommender systems. In *Recommender Systems Handbook*, pages 649–688. Springer, 2015. (On page 25.)

[87] Simon Gerber, Michael Fry, Judy Kay, Bob Kummerfeld, Glen Pink, and Rainer Wasinger. Personisj: mobile, client-side user modelling. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 111–122. Springer, 2010. (On page 25.)

[88] Arthur Gervais, Reza Shokri, Adish Singla, Srdjan Capkun, and Vincent Lenders. Quantifying web-search privacy. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 966–977, New York, NY, USA, 2014. ACM. (On pages 6 and 35.)

[89] Ian Goldberg. Improving the robustness of private information retrieval. In *IEEE Symposium on Security and Privacy*, pages 131–148. IEEE, 2007. (On page 33.)

[90] Richard Gomer, Eduarda Mendes Rodrigues, Natasa Milic-Fraying, and M.C. Schrafel. Network analysis of third party tracking: User exposure to tracking cookies through search. In *WI-IAT*, 2013. (On page 29.)

[91] Google. Headless chromium. https://chromium.googlesource.com/chromium/src/+/lkgr/headless/README.md, 2018. (On page 89.)

[92] Jeyanthi Hall, Michel Barbeau, and Evangelos Kranakis. Detection of transient in radio frequency fingerprinting using signal phase. *Wireless and Optical Communications*, pages 13–18, 2003. (On page 31.)

[93] Coşkun Hamzaçebi. Improving artificial neural networks' performance in seasonal time series forecasting. *Information Sciences*, 178(23):4550–4559, 2008. (On page 191.)

[94] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, 3rd edition*. Morgan Kaufmann, 2011. (On pages 79 and 156.)

[95] Africa Hands. Duckduckgo. *Technical Services Quarterly*, 29(4):345–347, 2012. (On page 33.)

[96] Saul Hansell. Aol removes search data on vast group of web users. http://query.nytimes.com/gst/fullpage.html?res=9504e5d81e3ff93ba3575bc0a9609c8b63, 2006. (On pages 3, 6, 18, 71, and 88.)

[97] Laura Hautala. These android apps have been tracking you, even when you say stop. https://www.cnet.com/news/these-android-apps-have-been-tracking-you-even-when-you-say-stop/, Feb 2019. (On page 2.)

[98] Michael Herrmann, Alfredo Rial, Claudia Diaz, and Bart Preneel. Practical privacy-preserving location-sharing based services with aggregate statistics. In *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*, pages 87–98. ACM, 2014. (On page 38.)

[99] Michael Hitchens, Judy Kay, Bob Kummerfeld, and Ajay Brar. Secure identity management for pseudo-anonymous service access. In *International Conference on Security in Pervasive Computing*, pages 48–55. Springer, 2005. (On page 25.)

[100] Philipp Holzinger, Stefan Triller, Alexandre Bartel, and Eric Bodden. An in-depth study of more than ten years of java exploitation. In *Proceedings of the 2016 ACM Conference on Computer and Communications Security*, CCS '16, 2016. (On page 29.)

[101] Bunke Horst and Caelli Terry Michael. *Hidden Markov models: Applications In Computer Vision*, volume 45. World Scientific, 2001. (On page 74.)

[102] Fraser Howard and Onur Komili. Poisoned search results: How hackers have automated search engine poisoning attacks to distribute malware. *Sophos Technical Papers*, pages 1–15, 2010. (On page 109.)

[103] Daniel C Howe and Helen Nissenbaum. Trackmenot: Resisting surveillance in web search. *Lessons from the Identity Trail: Anonymity, Privacy, and Identity in a Networked Society*, 23:417–436, 2009. (On pages 33 and 148.)

[104] Chong Huang, Peter Kairouz, Xiao Chen, Lalitha Sankar, and Ram Rajagopal. Context-aware generative adversarial privacy. *Entropy*, 19(12):656, 2017. (On page 37.)

[105] Rob J Hyndman, Yeasmin Khandakar, et al. *Automatic time series for forecasting: the forecast package for R*. Number 6/07. Monash University, Department of Econometrics and Business Statistics, 2007. (On page 127.)

[106] Damilola Ibosiola, Ignacio Castro, Gianluca Stringhini, Steve Uhlig, and Gareth Tyson. Who watches the watchmen: Exploring complaints on the web. In *Web Conference*, 2019. (On page 91.)

[107] Damilola Ibosiola, Benjamin Steer, Alvaro Garcia-Recuero, Gianluca Stringhini, Steve Uhlig, and Gareth Tyson. Movie pirates of the caribbean: Exploring illegal streaming cyberlockers. *International AAAI Conference on Web and Social Media (ICWSM)*, 2018. (On page 102.)

[108] Muhammad Ikram, Hassan Asghar, Mohamed Ali Kaafar, and Anirban Mahanti. On the intrusiveness of javascript on the web. In *CoNEXT Student Workshop*, 2014. (On page 29.)

[109] Muhammad Ikram, Hassan Jameel Asghar, Mohamed Ali Kaafar, Anirban Mahanti, and Balachandar Krishnamurthy. Towards seamless tracking-free web: Improved detection of trackers via one-class learning. *Proceedings on Privacy Enhancing Technologies*, 2017(1):79–99, 2017. (On pages 13 and 29.)

[110] Muhammad Ikram and Mohamed Ali Kâafar. A first look at mobile ad-blocking apps. In *16th IEEE International Symposium on Network Computing*

and Applications, NCA 2017, Cambridge, MA, USA, October 30 - November 1, 2017, pages 343–350, 2017. (On page 78.)

[111] Muhammad Ikram and Mohamed Ali Kaafar. A first look at mobile ad-blocking apps. In *Network Computing and Applications (NCA), 2017 IEEE 16th International Symposium on*, pages 1–8. IEEE, 2017. (On page 91.)

[112] Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kaafar, and Noha Loizon. Measuring and analysing the chain of implicit trust: A study of third-party resources loading. *ACM Transactions on Privacy and Security (TOPS)*, 2020. (Not cited.)

[113] Muhammad Ikram, Rahat Masood, Gareth Tyson, Mohamed Ali Kaafar, Noha Loizon, and Roya Ensafi. The chain of implicit trust: An analysis of the web third-party resources loading. In *The World Wide Web Conference*, pages 2851–2857. ACM, 2019. (Not cited.)

[114] Muhammad Ikram, Narseo Vallina-Rodriguez, Suranga Seneviratne, Mohamed Ali Kaafar, and Vern Paxson. An analysis of the privacy and security risks of android vpn permission-enabled apps. In *IMC*, 2016. (On page 91.)

[115] VirusTotal Inc. Virustotal public api. https://www.virustotal.com/en/documentation/public-api/, 2019. (On page 88.)

[116] InformAction. Noscript - javascript/java/flash blocker for a safer firefox experience! - what is it? https://noscript.net, 2019. Accessed: 2019-08-09. (On page 114.)

[117] Luca Invernizzi, Paolo Milani Comparetti, Stefano Benvenuti, Christopher Kruegel, Marco Cova, and Giovanni Vigna. Evilseed: A guided approach to finding malicious web pages. In *2012 IEEE symposium on Security and Privacy*, pages 428–442. IEEE, 2012. (On page 109.)

[118] Dongseok Jang, Ranjit Jhala, Sorin Lerner, and Hovav Shacham. An empirical study of privacy-violating information flows in javascript web applications. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 270–283, New York, NY, USA, 2010. ACM. (On page 118.)

[119] Andrea Day Jennifer Schlesinger. How gps can track you, even when you turn it off. https://www.cnbc.com/2018/07/13/gps-can-spy-on-you-even-when-you-turn-it-off.html, Jul 2018. (On page 117.)

[120] Sequa Jerome. Large angler malvertising campaign hits top publishers. https://blog.malwarebytes.com/threat-analysis/20/16/03/large-angler-malvertising-campaign-hits-top-publishers/, 2019. Accessed: 2019-01-18. (On pages 87 and 95.)

[121] Jinyuan Jia and Neil Zhenqiang Gong. Attriguard: A practical defense against attribute inference attacks via adversarial machine learning. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 513–529, 2018. (On page 38.)

[122] Alex Kantchelian, Michael Carl Tschantz, Sadia Afroz, Brad Miller, Vaishaal Shankar, Rekha Bachwani, Anthony D Joseph, and J Doug Tygar. Better malware ground truth: Techniques for weighting anti-virus vendor labels. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, pages 45–56. ACM, 2015. (On page 91.)

[123] Judy Kay. Scrutable adaptation: Because we can and must. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 11–19. Springer, 2006. (On page 26.)

[124] Judy Kay, Bob Kummerfeld, and Piers Lauder. Managing private user models and shared personas. In *UM03 Workshop on User Modeling for Ubiquitous Computing*, pages 1–11. Citeseer, 2003. (On page 26.)

[125] Meghan Keane. Instant personalization brings more privacy issues to facebook. https://econsultancy.com/facebook-s-instant-personalization-brings-yet-another-privacy-issue-to-the-site/, Apr 2010. (On page 24.)

[126] Simon Kemp. Digital 2019: Global internet user accelerates. https://wearesocial.com/blog/2019/01/digital-2019-global-internet-use-accelerates, Jan 2019. (On pages 1 and 2.)

[127] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer, 2015. (On page 91.)

[128] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pages 193–204. ACM, 2011. (On page 193.)

[129] Tadayoshi Kohno, Andre Broido, and Kimberly C Claffy. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005. (On pages 31 and 43.)

[130] Balachander Krishnamurthy and Craig Wills. Privacy diffusion on the web: a longitudinal perspective. In *Proceedings of the 18th international conference on World wide web*, pages 541–550. ACM, 2009. (On page 27.)

[131] John Krumm. Inference attacks on location tracks. In Anthony LaMarca, Marc Langheinrich, and Khai N. Truong, editors, *Pervasive Computing*, pages 127–143, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. (On page 118.)

[132] Deepak Kumar, Zane Ma, Ariana Mirian, Joshua Mason, J Alex Halderman, and Michael Bailey. Security Challenges in an Increasingly Tangled Web. In *Proceedings of the 2017 World Wide Web Conference on World Wide Web*, 2017. (On pages 30 and 89.)

[133] John Kurkowski. Accurately separate the TLD from the registered domain and subdomains of a url, using the public suffix list. https://github.com/john-kurkowski/tldextract, 2019. (On page 90.)

[134] Andreas Kurtz, Hugo Gascon, Tobias Becker, Konrad Rieck, and Felix Freiling. Fingerprinting Mobile Devices Using Personalized Configurations. *Proceedings on Privacy Enhancing Technologies*, 2016(1):4–19, 2016. (On pages 30, 31, and 43.)

[135] Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Foundations of Computer Science, 1997. Proceedings., 38th Annual Symposium on*, pages 364–373. IEEE, 1997. (On page 33.)

[136] Malwarebytes Labs. Malvertising on equifax, transunion tied to third party script (updated). https://blog.malwarebytes.com/threat-analysis/2017/10/equifax-transunion-websites-push-fake-flash-player/, 2017. (On page 103.)

[137] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints. *Proceedings - 2016 IEEE Symposium on Security and Privacy, SP 2016*, pages 878–894, 2016. (On pages 28 and 42.)

[138] Tobias Lauinger, Abdelberi Chaabane, Sajjad Arshad, William Robertson, Christo Wilson, and Engin Kirda. Thou shalt not depend on me: Analysing the use of outdated javascript libraries on the web. In *NDSS*, 2017. (On pages 6, 29, 87, 92, 95, and 98.)

[139] Byoungyoung Lee, Jinoh Oh, Hwanjo Yu, and Jong Kim. Protecting location privacy using location semantics. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1289–1297. ACM, 2011. (On page 38.)

[140] Hannah Levenson. Touch heatmap analytics: The future of mobile app usability testing. https://blog.appsee.com/touch-heatmap-analytics-future-mobile-app-usability-testing/, December 2017. (On page 117.)

[141] Chen Li, Houtan Shirani-Mehr, and Xiaochun Yang. Protecting individual information against inference attacks in data publishing. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications*, DASFAA'07, pages 422–433, Berlin, Heidelberg, 2007. Springer-Verlag. (On pages 20 and 35.)

[142] Yuhua Li, David McLean, Zuhair A. Bandar, James D. O'Shea, and Keeley Crockett. Sentence similarity based on semantic nets and corpus statistics. *IEEE Trans. on Knowl. and Data Eng.*, 18(8):1138–1150, August 2006. (On pages 79 and 156.)

[143] Yuhua Li, David McLean, Zuhair A Bandar, James D O'shea, and Keeley Crockett. Sentence similarity based on semantic nets and corpus statistics. *IEEE transactions on knowledge and data engineering*, 18(8):1138–1150, 2006. (On page 150.)

[144] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Knowing your enemy: understanding and detecting malicious web advertising. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 674–686. ACM, 2012. (On page 95.)

[145] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Knowing your enemy: understanding and detecting malicious web advertising. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 674–686. ACM, 2012. (On pages 102 and 103.)

[146] Timothy Libert. Exposing the Hidden Web: An Analysis of Third-Party HTTP Requests on 1 Million Websites. *International Journal of Communication*, 9(October):3544–3561, 2015. (On page 28.)

[147] Kun Liu and Evimaria Terzi. A framework for computing the privacy scores of users in online social networks. *ACM Trans. Knowl. Discov. Data*, 5(1):6:1–6:30, December 2010. (On page 36.)

[148] Jan Lukáš, Jessica Fridrich, and Miroslav Goljan. Digital camera identification from sensor pattern noise. *IEEE Transactions on Information Forensics and Security*, 1(2):205–214, 2006. (On page 30.)

[149] Upal Mahbub, Sayantan Sarkar, Vishal M Patel, and Rama Chellappa. Active user authentication for smartphones: A challenge data set and benchmark results. In *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–8. IEEE, 2016. (On pages 131 and 132.)

[150] Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. Mobile sensor data anonymization. *arXiv preprint arXiv:1810.11546*, 2018. (On page 37.)

[151] Mohammad Malekzadeh, Richard G Clegg, Andrea Cavallaro, and Hamed Haddadi. Protecting sensory data against sensitive inferences. In *Proceedings of the 1st Workshop on Privacy by Design in Distributed Systems*, page 2. ACM, 2018. (On page 37.)

[152] Mohammad Malekzadeh, Richard G Clegg, and Hamed Haddadi. Replacement autoencoder: A privacy-preserving algorithm for sensory data analysis.

In *Internet-of-Things Design and Implementation (IoTDI), 2018 IEEE/ACM Third International Conference on*, pages 165–176. IEEE, 2018. (On page 37.)

[153] Rahat Masood, Shlomo Berkovsky, and Mohamed Ali Kaafar. *Modern Socio-Technical Perspectives on Privacy*, chapter Tracking and Personalization. Springer, 2019. (Not cited.)

[154] Rahat Masood, Dinusha Vatsalan, Hassan Jameel Asghar, and Mohamed Ali Kaafar. Privacy preserving sensory data. *Proceedings on Privacy Enhancing Technologies*, 2020. (Not cited.)

[155] Rahat Masood, Dinusha Vatsalan, Muhammad Ikram, and Mohamed Ali Kaafar. Incognito: A method for obfuscating web data. In *Proceedings of the 2018 World Wide Web Conference*, pages 267–276. International World Wide Web Conferences Steering Committee, 2018. (On pages 18, 24, 25, and 121.)

[156] Rahat Masood, Benjamin Zi Hao Zhao, Hassan Jameel Asghar, and Moahmed Ali Kaafar. Poster: Touchtrack: How unique are your touch gestures? In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2555–2557. ACM, 2017. (Not cited.)

[157] Rahat Masood, Benjamin Zi Hao Zhao, Hassan Jameel Asghar, and Mohamed Ali Kaafar. Touch and you're trapp (ck) ed: Quantifying the uniqueness of touch gestures for tracking. *Proceedings on Privacy Enhancing Technologies*, 2018(2):122–142, 2018. (On pages 15, 30, 117, 118, and 121.)

[158] Jonathan R Mayer and John C Mitchell. Third-party web tracking: Policy and technology. In *2012 IEEE Symposium on Security and Privacy*, pages 413–427. IEEE, 2012. (On page 16.)

[159] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: your finger taps have fingerprints. *ACM Mobisys*, page 323, 2012. (On pages 2, 12, and 117.)

[160] Nitesh Mor, Oriana Riva, Suman Nath, and John Kubiatowicz. Bloom cookies: Web search personalization without user tracking. In *NDSS*, 2015. (On page 34.)

[161] Keaton Mowery and Hovav Shacham. Pixel Perfect : Fingerprinting Canvas in HTML5. *Web 2.0 Security & Privacy 20 (W2SP)*, pages 1–12, 2012. (On page 28.)

[162] Mozilla. Cross-origin resource sharing (cors) - http. https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS, 2019. (On page 114.)

[163] Mummoorthy Murugesan and Chris Clifton. Plausibly deniable search. In *Proceedings of the Workshop on Secure Knowledge Management (SKM 2008)*, 2008. (On page 34.)

[164] Mummoorthy Murugesan and Chris Clifton. Providing Privacy through Plausibly Deniable Search. *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 768–779, 2009. (On pages 6, 34, and 36.)

[165] Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society. (On pages 3, 6, 18, and 88.)

[166] Nam Tuan Nguyen, Guanbo Zheng, Zhu Han, and Rong Zheng. Device Fingerprinting to Enhance Wireless Security using Nonparametric Bayesian Method. *Infocom*, pages 1404–1412, 2011. (On page 31.)

[167] NHMRC. National statement on ethical conduct in human research (2007) - updated 2018. https://www.nhmrc.gov.au/about-us/publications/national-statement-ethical-conduct-human-research-2007-updated-2018, 2018. (On page 47.)

[168] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. You are what you include: Large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS'12, 2012. (On pages 6, 29, and 87.)

[169] Xia Ning, Christian Desrosiers, and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*, pages 37–76. 2015. (On page 22.)

[170] Łukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Diaz. The leaking battery: A privacy analysis of the HTML5 battery status API. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9481:254–263, 2016. (On pages 28 and 43.)

[171] Łukasz Olejnik, Claude Castelluccia, and Artur Janc. Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns. *5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012)*, pages 1–16, 2012. (On pages 42 and 43.)

[172] Adam Ostrow. Facebook fired: 8% of us companies have sacked social media miscreants. https://mashable.com/2009/08/10/social-media-misuse/, Aug 2009. (On page 24.)

[173] Georgios Paliouras. Discovery of web user communities and their role in personalization. *User Modeling and User-Adapted Interaction*, 22(1-2):151–175, 2012. (On page 24.)

[174] Jeffrey Pang, Ben Greenstein, Ramakrishna Gummadi, Seshan Srinivasan, and David Wetherall. 802. 11 User Fingerprinting. *Proceedings of the 13th*

Annual ACM International Conference on Mobile Computing and Networking, 9:99–110, 2007. (On page 31.)

[175] Sai Teja Peddinti and Nitesh Saxena. On the privacy of web search based on query obfuscation: A case study of trackmenot. In *Proceedings of the 10th International Conference on Privacy Enhancing Technologies*, PETS'10, pages 19–37, Berlin, Heidelberg, 2010. Springer-Verlag. (On pages 6 and 35.)

[176] Giancarlo Pellegrino, Christian Rossow, Fabrice J. Ryba, Thomas C. Schmidt, and Matthias Wählisch. Cashing out the great cannon? on browser-based ddos attacks and economics. In *USENIX*, 2015. (On page 87.)

[177] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (8):1226–1238, 2005. (On pages 44 and 56.)

[178] Daniele Perito, Claude Castelluccia, Mohamed Ali Kaafar, and Pere Manils. How unique and traceable are usernames? In *International Symposium on Privacy Enhancing Technologies Symposium*, pages 1–17. Springer, 2011. (On pages 28 and 43.)

[179] Albin Petit, Thomas Cerqueus, Sonia Ben Mokhtar, Lionel Brunie, and Harald Kosch. PEAS: Private, efficient and accurate web search. *Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, 1:571–580, 2015. (On page 34.)

[180] Kurt Plarre, Andrew Raij, Syed Monowar Hossain, Amin Ahsan Ali, Motohiro Nakajima, Mustafa Al'Absi, Emre Ertin, Thomas Kamarck, Santosh Kumar, Marcia Scott, et al. Continuous inference of psychological stress from sensory measurements collected in the natural environment. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 97–108. IEEE, 2011. (On page 117.)

[181] Bogdan Popa. 85 infected android apps stealing social network passwords found on play store. https://news.softpedia.com/news/85-infected-android-apps-stealing-social-network-passwords-found-on-play-store-518984.shtml, 2017. (On page 100.)

[182] Vincent Primault, Sonia Ben Mokhtar, Cédric Lauradoux, and Lionel Brunie. Time distortion anonymization for the publication of mobility data with high utility. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 1, pages 539–546. IEEE, 2015. (On page 38.)

[183] Nisarg Raval, Ashwin Machanavajjhala, and Jerry Pan. Olympus: Sensor privacy through utility aware obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2019(1):5–25, 2019. (On pages 20 and 37.)

[184] David Rebollo-Monedero and Jordi Forné. Optimized query forgery for private information retrieval. *IEEE Transactions on Information Theory*, 56(9):4631–4642, 2010. (On page 34.)

[185] Michael K Reiter and Aviel D Rubin. Anonymous web transactions with crowds. *Communications of the ACM*, 42(2):32–48, 1999. (On page 33.)

[186] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. *Proc. of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, (Nsdi):12, 2012. (On pages 2, 12, and 28.)

[187] Ian Rose and Matt Welsh. Mapping the urban wireless landscape with argos. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 323–336, New York, NY, USA, 2010. ACM. (On page 12.)

[188] RM Sakia. The box-cox transformation technique: a review. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 41(2):169–178, 1992. (On pages 127 and 192.)

[189] Salman Salamatian, Amy Zhang, Flávio du Pin Calmon, Sandilya Bhamidipati, Nadia Fawaz, Branislav Kveton, Pedro Oliveira, and Nina Taft. How to hide the elephant- or the donkey- in the room: Practical privacy against statistical inference for large data. In *IEEE Global Conference on Signal and Information Processing, GlobalSIP 2013, Austin, TX, USA, December 3-5, 2013*, pages 269–272, 2013. (On pages 20, 35, and 124.)

[190] Salman Salamatian, Amy Zhang, Flávio du Pin Calmon, Sandilya Bhamidipati, Nadia Fawaz, Branislav Kveton, Pedro Oliveira, and Nina Taft. Managing your private and public data: Bringing down inference attacks against your privacy. *IEEE Journal of Selected Topics in Signal Processing*, 9:1240–1255, 2015. (On page 124.)

[191] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007. (On page 26.)

[192] Fabian Schneider, Sachin Agarwal, Tansu Alpcan, and Anja Feldmann. The new web: characterizing ajax traffic. In *International Conference on Passive and Active Network Measurement*, pages 31–40. Springer, 2008. (On page 106.)

[193] David W. Scott. On optimal and data-based histograms. *Biometrika*, 66:605–610, 1979. (On page 52.)

[194] SecurityWeek. Malicious redirects on equifax, transunion sites caused by third-party scripts. https://www.securityweek.com/malicious-redirects-equifax-transunion-sites-caused-third-party-script, 2017. (On page 103.)

[195] Suranga Seneviratne, Aruna Seneviratne, Prasant Mohapatra, and Anirban Mahanti. Predicting user traits from a snapshot of apps installed on a smartphone. *ACM SIGMOBILE Mobile Computing and Communications Review*, 18(2):1–8, 2014. (On page 31.)

[196] Bracha Shapira, Yuval Elovici, Adlay Meshiach, and Tsvi Kuflik. PRAW - A privacy model for the web. *Journal of the American Society for Information Science and Technology (JASIST)*, 56(2):159–172, 2005. (On pages 20, 33, and 148.)

[197] Reza Shokri. Privacy games: Optimal user-centric data obfuscation. *Proceedings on Privacy Enhancing Technologies*, 2015(2):299–315, 2015. (On page 37.)

[198] Laurent Simon, Wenduan Xu, and Ross Anderson. Don't interrupt me while i type: Inferring text entered through gesture typing on android keyboards. *Proceedings on Privacy Enhancing Technologies*, 2016(3):136–154, 2016. (On page 31.)

[199] Ryan Singel. Facebook beacon tracking program draws privacy lawsuit. https://www.wired.com/2008/08/facebook-beacon/, Aug 2008. (On page 24.)

[200] Jan Spooren, Davy Preuveneers, and Wouter Joosen. Mobile Device Fingerprinting Considered Harmful for Risk-based Authentication. *2015 European Workshop on System Security (EuroSec 2015)*, (EuroSec):6:1–6:6, 2015. (On page 30.)

[201] Jessica Su, Ansh Shukla, Sharad Goel, and Arvind Narayanan. Deanonymizing web browsing data with social networks. In *Proceedings of the 26th International Conference on World Wide Web, (WWW) 2017, Perth, Australia, April 3-7, 2017*, pages 1261–1269, 2017. (On page 71.)

[202] Jingxiu Su, Zhenyu Li, Stephane Grumbach, Muhammad Ikram, Kave Salamatian, and Gaogang Xie. Web tracking cartography with dns records. In *IEEE 37th International Performance Computing and Communications Conference (IPCC)*, 2018. (On page 29.)

[203] Jingxiu Su, Zhenyu Li, Stephane Grumbach, Muhammad Ikram, Kave Salamatian, and Gaogang Xie. A cartography of web tracking using dns records. *Computer Communications*, 134:83 – 95, 2019. (On page 29.)

[204] Mozilla Public Suffix. View the public suffix list. https://publicsuffix.org/list/, 2019. (On page 90.)

[205] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics*, 25(2-3):98–110, 1997. (On pages 3, 6, 18, and 88.)

[206] Latanya Sweeney. Simple demographics often identify people uniquely. *Carnegie Mellon University, Data Privacy Working Paper 3. Pittsburgh 2000*, pages 1–34, 2000. (On pages 27 and 124.)

[207] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001. (On page 79.)

[208] Eran Toch, Yang Wang, and Lorrie Faith Cranor. Personalization and privacy: a survey of privacy risks and remedies in personalization-based systems. *User Modeling and User-Adapted Interaction*, 22(1-2):203–220, 2012. (On page 24.)

[209] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy preserving targeted advertising. In *Proceedings of the Network and Distributed System Security Symposium (NDSS), San Diego, California, USA, 28th February - 3rd March 2010*. The Internet Society, 2010. (On pages 19 and 30.)

[210] Imdad Ullah, Roksana Boreli, Mohamed Ali Kaafar, and Salil S Kanhere. Characterising user targeting for in-app mobile ads. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 547–552. IEEE, 2014. (On page 25.)

[211] David Vallet, Arik Friedman, and Shlomo Berkovsky. Matrix factorization without user data retention. In *Advances in Knowledge Discovery and Data Mining - 18th Pacific-Asia Conference, PAKDD 2014, Tainan, Taiwan, May 13-16, 2014. Proceedings, Part I*, pages 569–580, 2014. (On page 25.)

[212] Ashlee Vance. Times web ads show security breach. https://www.nytimes.com/2009/09/15/technology/internet/15adco.html, 2009. (On pages 95 and 102.)

[213] Quick Remove Virus. How do i remove hwcdn.net from my pc. https://quickremovevirus.com/how-do-i-remove-hwcdn-net-from-my-pc/, 2017. (On page 101.)

[214] volatilityfoundation. volatilityfoundation/volatility: An advanced memory forensics framework. https://github.com/volatilityfoundation/volatility, 2019. Accessed: 2019-08-09. (On page 105.)

[215] Hao Wang and Zhengquan Xu. Cts-dp: Publishing correlated time-series data via differential privacy. *Knowledge-Based Systems*, 122:167–179, 2017. (On pages 7 and 193.)

[216] Xiao Sophia Wang, Aruna Balasubramanian, Arvind Krishnamurthy, and David Wetherall. Demystify page load performance with wprof. In *Proc. of the USENIX conference on Networked Systems Design and Implementation (NSDI)*, 2013. (On page 92.)

[217] Websense. Real-time threat analysis with csi: Ace insight. https://csi.websense.com/, 2018. (On pages 92 and 94.)

[218] Zachary Weinberg, Eric Y Chen, Pavithra Ramesh Jayaraman, and Collin Jackson. I still know what you visited last summer: Leaking browsing history via user interaction and side channel attacks. In *2011 IEEE Symposium on Security and Privacy*, pages 147–161. IEEE, 2011. (On page 28.)

[219] Udi Weinsberg, Smriti Bhagat, Stratis Ioannidis, and Nina Taft. Blurme: Inferring and obfuscating user gender based on ratings. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 195–202, New York, NY, USA, 2012. ACM. (On page 35.)

[220] Hui Xu, Yangfan Zhou, and Michael R Lyu. Towards Continuous and Passive Authentication via Touch Biometrics: An Experimental Study on Smartphones. *SOUPS '14: Proceedings of the Tenth Symposium On Usable Privacy and Security*, pages 187–198, 2014. (On page 67.)

[221] Yabo Xu, Ke Wang, Benyu Zhang, Zheng Chen, and Ke Wang. Privacy-enhancing personalized web search. *Proceedings of the 16th international conference on World Wide Web - WWW '07*, page 591, 2007. (On page 35.)

[222] Shaozhi Ye, Felix Wu, Raju Pandey, and Hao Chen. Noise injection for search privacy protection. *Proceedings - 12th IEEE International Conference on Computational Science and Engineering, CSE 2009*, 3:1–8, 2009. (On page 34.)

[223] Ting-Fang Yen, Yinglian Xie, Fang Yu, Roger Peng Yu, and Martin Abadi. Host fingerprinting and tracking on the web: Privacy and security implications. In *NDSS*, volume 62, page 66. Citeseer, 2012. (On pages 28 and 43.)

[224] Michal Zalewski. Browser security handbook, part 2. *Google*, 2008. (On page 28.)

[225] Andong Zhan, Marcus Chang, Yin Chen, and Andreas Terzis. Accurate caloric expenditure of bicyclists using cellphones. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, pages 71–84. ACM, 2012. (On page 117.)

[226] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003. (On pages 191 and 192.)

[227] Sha Zhao, Julian Ramos, Jianrong Tao, Ziwen Jiang, Shijian Li, Zhaohui Wu, Gang Pan, and Anind K. Dey. Discovering different kinds of smartphone users through their application usage behaviors. *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '16*, pages 498–509, 2016. (On page 31.)

[228] Zhe Zhou, Wenrui Diao, Xiangyu Liu, and Kehuan Zhang. Acoustic fingerprinting revisited: Generate stable device id stealthily with inaudible sound.

In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 429–440. ACM, 2014. (On page 30.)

Appendix A

This supplementary section gives more insights on the information presented in chapter 4.

A.1 TouchTrack App Overview

The TouchTrack app consists of three well-known games and one purpose-built game. A brief description of games are given below:

1. *2048:* We used this game to collect swipes. It is a free and open-source game which is played on a 4 by 4 grid having numbered tiles that need to be swiped in any of the four directions. We selected this game since it is widely known and it captures swipes mimicking their usage in a natural way, i.e., while reading emails or swiping through an image gallery.

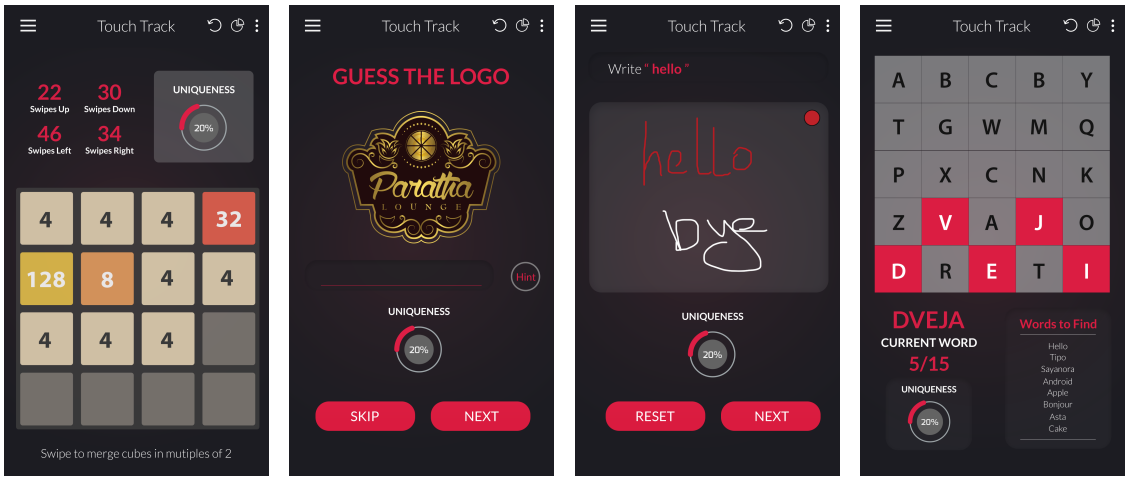
2. *Lexica:* We used this game to collect taps. It is another open-source free word game that gives the user three minutes to find as many words as possible on a 5 by 5 grid of random letters. The original behaviour of the game requires user to drag letters to make a single word. For our work, we changed the drag operation to a tap, and ask user to tap on the letter to select it, or tap on again the same letter for de-selection. The grid of 5*5 allows user to tap on almost every point of screen, thus simulating natural taps.

3. *Logo Maniac:* We used this game to collect keystrokes. The game tests the user's ability to recall popular brands by showing logos and asking them to type the brand name. We modify this game by only having the most popular brands in our database, and providing hints to user if they cannot recall it. We modified the keyboard layout of the game to make it similar to the keyboard layout used for entering texts in Android phones, to capture the user's natural typing behaviour on phones.

4. *Write Something:* We used this game to collect handwriting samples. This game was purpose-built by us. It asks users to write a word shown at top left

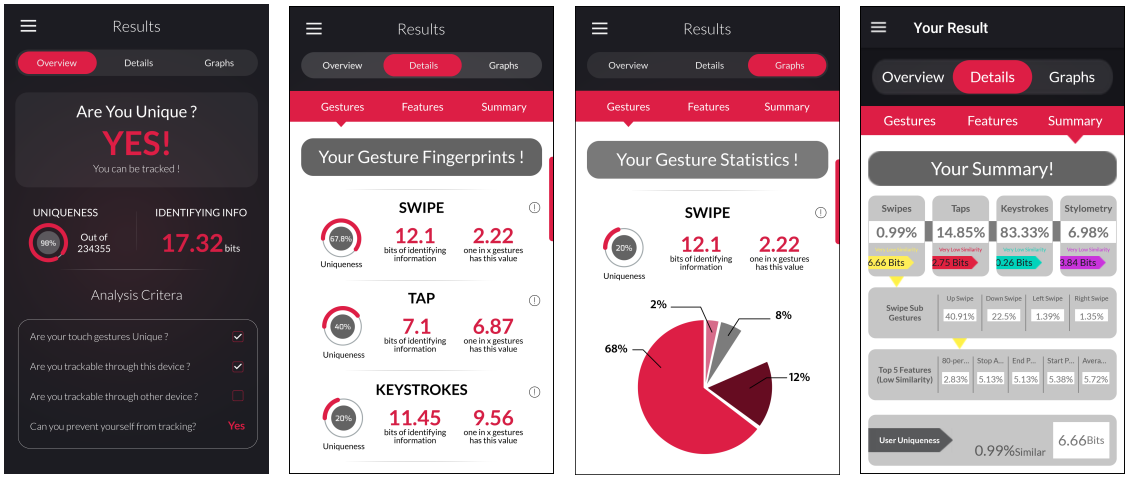
corner of the screen with a finger. User is provided with a large area on the screen to write in any direction or from any point.

The screen shots of the TouchTrack App are displayed in figure A.1, while figure A.2 shows the shots of result screen. We show uniqueness results for a feature, set of gesture samples, and multiple gestures to our app users.



(a) 2048 (b) Logo Maniac (c) Write Something (d) Lexica

Figure A.1: TouchTrack Game Screens

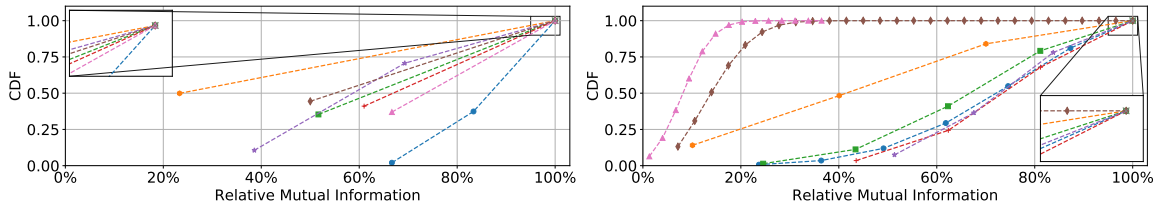


(a) Results Overview (b) Gesture Results (c) Graphs (d) Summary

Figure A.2: TouchTrack Result Screens

A.2 Users using Same Devices

We select “Nexus 5” as our primary device to analyze uniqueness results for users accessing our app through the same device. We chose “Nexus 5” because it is was



(a) CDF of Set of Gesture Samples on a Single Device. Relative Information of Respective Categories are: -●- Swipes: 95.8%, -⬡- Up Swipes: 54.0%, -■- Down Swipes: 80.9%, -+- Left Swipes: 83.7%, -★- Right Swipes: 66.0%, -◆- Taps: 77.7%, -▲- Keystrokes: 87.5%, -×- Handwriting: 100%

(b) CDF of a Gesture Sample on a Single Device. Relative Information of Respective Categories are: -●- Swipes: 80.8%, -⬡- Up Swipes: 52.4%, -■- Down Swipes: 77.9%, -+- Left Swipes: 87.1%, -★- Right Swipes: 82.0%, -◆- Taps: 14.4%, -▲- Keystrokes: 09.1%, -×- Handwriting: 100%

Figure A.3: Cumulative Distribution Function (CDF) of Gesture Sample(s) on a Single Device.

most used model of phone in our study, primarily because our test smartphone is also Nexus 5, which were given to users who did not possess an Android phone, for data collection. Table A.1 shows the statistics of analyzed data.

Our results indicate that users are highly recognizable on the same device. For a set of gesture samples, the performance of keystrokes and taps are fairly better on a single device as compared to multiple devices. We found that features with a single data point, such as Start X, Start Y, Start Pressure, Start Area, etc. highly contributes towards user uniqueness for keystroke and taps. Similarly, handwriting and overall swipes also show improved performance with the Finger Area being most prominent feature.

Figure A.3a shows the CDF of a set of gesture samples. We observe that handwriting reflects 100% of user identification followed by swipes with 95.83% of mutual relative information. Keystrokes and taps reveal 85.5% and 77.7% of user information respectively. The performance of swipe sub-types are also improved except for the up swipes (54%).

Figure A.3b is the CDF of a gesture sample. We found that the performance of a

Table A.1: Touch Data Statistics

Gesture	Users	Sp.	Gesture	Users	Sp.
Swipes	08	920	Up Swipes	08	244
Down Swipes	07	217	Left Swipes	07	214
Right Swipes	08	245	Handwriting	08	259
Taps	09	2653	Keystrokes	08	1614
Total Samples: 6366			Sp. = Samples		

handwriting and a swipe is overall improved, with 100% and 80.8% of identification respectively. It is noted that these results reflect a subset of the user data for a single Nexus 5X device. It could be concluded that these results are influenced from the device type and size of the subset. In order to confidently verify our suspicions, we need to collect and analyze data from other types of devices with more user data and consider it as part of future work.

A.3 Results Summary

Table A.2 and A.3 represent the summary of results corresponding to each gesture and combinations.

Table A.2: Summary of Results - Gesture Sample

Gesture	Gesture Sample		Set of Gesture Samples		Features List
	Rel. Inf.	TPR	Rel. Inf.	TPR	
Swipe	57.79%	76.11%	63.33%	89.85%	Stop Area, 80-percentile pairwise X-Tilt, Start Area, Mid-Stroke Pressure, 80-percentile pairwise Area, Std. Dev. of Pairwise Velocity, Std. Dev. of Pairwise Change of Area-Position, 20-percentile pairwise Area, 50-percentile pairwise Area, End to End Acc.*, Distance, Start Pressure, End Point of Pairwise Area, Start Point of Pairwise Area, Mid-Stroke Area, Std. Dev. of Pairwise Area, Average of Pairwise X-Tilt, Average of Pairwise Area, 50-percentile pairwise Pressure, 20-percentile pairwise Change of Area-Position, 80-percentile pairwise Pressure, Median of Last 3 Velocities Points, 20-percentile pairwise Pressure, Average of Pairwise Pressure, 80-percentile pairwise Change of Area-Position, Start Point of Pairwise Pressure, Stop Pressure, Start Point of Pairwise X-Tilt, Start Point of Pairwise Change of Area-Position, Std. Dev. of Pairwise Change of Pressure-Position, Start Point of Pairwise Velocity, End Point of Pairwise Pressure, 80-percentile pairwise Y-Tilt, Std. Dev. of Pairwise Pressure, Start Point of Pairwise Direction, Average of Pairwise Change of Area-Position, Start Y, 50-percentile pairwise X-Tilt, End to End Pressure Distance, 20-percentile pairwise X-Tilt, Start Point of Pairwise Change of Pressure-Position, Median of First 5 Acc. Points, Average of Pairwise Change of Pressure-Position, Start Point of Pairwise Y Velocity, Average Velocity, 20-percentile pairwise Change of Pressure-Position, Average of Pairwise Y-Tilt, 80-percentile pairwise Change of Pressure-Position, End Point of Pairwise X-Tilt, Direct End To End Direction, Average of Pairwise Y, Start Point of Pairwise Y-Tilt
Up Swipe	48.56%	74.11%	50.23%	84.44%	Stop Pressure, Std. Dev. of Pairwise X-Tilt, Average Velocity, Start Pressure, Start Y, Average of Pairwise X-Tilt, Std. Dev. of Pairwise Area, 20-percentile pairwise Pressure, 80-percentile pairwise Area, 20-percentile pairwise Area, Phone Orientation, End Point of Pairwise Pressure, Average of Pairwise Area, End Point of Pairwise X-Tilt, Stop Y, End Point of Pairwise Area, Start Point of Pairwise Pressure, Start X, Median of First 5 Acceleration Points, Average of Pairwise Pressure, 50-percentile pairwise Area Acc., 50-percentile pairwise Y Acc., 50-percentile pairwise Pressure Acc., 50-percentile pairwise Change of X-Tilt Position, Std. Dev. of Pairwise Pressure, Start Point of Pairwise X-Tilt, Start Point of Pairwise Area, Stop X, Average of Pairwise Direction, 80-percentile pairwise Pressure, Std. Dev. of Pairwise Y-Tilt, End to End Y Distance, 80-percentile pairwise Y Acc., 20-percentile pairwise X-Tilt, Length of Trajectory, 50-percentile pairwise Acc., 80-percentile pairwise Pressure Acc., Average of Pairwise Y-Tilt, 50-percentile pairwise X Acc., Std. Dev. of Pairwise X-Tilt Acc., 20-percentile pairwise Change of Area-Position, Start Point of Pairwise Y-Tilt, 20-percentile pairwise Direction, End to End X Distance, 50-percentile pairwise Area, End Point of Pairwise Y-Tilt, 50-percentile pairwise X, 20-percentile pairwise Change of Pressure-Position, Direct End To End Direction, 20-percentile pairwise Raw Y-Tilt
Down Swipe	52.22%	72.64%	54.58%	86.12%	Start Point of Pairwise Area, 50-percentile pairwise Acc., Median of First 5 Acc. Points, Start Y, Stop Pressure, Start Point of Pairwise Change of Area-Position, Start Pressure, 50-percentile pairwise Acc., 50-percentile pairwise Pressure Acc., Average Velocity, Std. Dev. of Pairwise Area, 80-percentile pairwise Pressure Acc., End Point of Pairwise Area, Average of Pairwise X-Tilt, Average of Pairwise Area, Start Point of Pairwise Pressure, Average of Pairwise Direction, 20-percentile pairwise Area, 20-percentile pairwise Pressure, 20-percentile pairwise Change of Area-Position, End Point of Pairwise Pressure, Start Point of Pairwise X-Tilt, 80-percentile pairwise Area, 50-percentile pairwise Y Acc., End to End Y Distance, Average of Pairwise Pressure, Start X, Stop Y, Average of Pairwise Change of Area-Position, Std. Dev. of Pairwise X-Tilt, Std. Dev. of Pairwise Pressure, Average of Pairwise Velocity, 80-percentile pairwise Y Acc., Ratio of End2End Dist. and Len of Trajectory, 80-percentile pairwise Pressure, 50-percentile pairwise Area, Average of Pairwise Y-Tilt, Average of Pairwise Y Velocity, End to End X Distance, Average of Pairwise X-Tilt Velocity, 20-percentile pairwise X-Tilt, 80-percentile pairwise Change of Y-Position, Start Area, 50-percentile pairwise Pressure, Start Point of Pairwise Y-Tilt, Start Point of Pairwise Change of Pressure-Position, 50-percentile pairwise Y, Std. Dev. of Pairwise X-Tilt Acc., Average of Pairwise Y, End Point of Pairwise X-Tilt
Left Swipe	53.96%	74.60%	68.66%	88.52%	20-percentile pairwise Area, Average of Pairwise X-Tilt, Stop Pressure, 80-percentile pairwise Area, Start Pressure, Average Velocity, 50-percentile pairwise Pressure Acc., Std. Dev. of Pairwise X-Tilt, Start Point of Pairwise Area, Median of First 5 Acc. Points, Start X, Std. Dev. of Pairwise Area, End Point of Pairwise Area, 80-percentile pairwise Pressure Acc., 50-percentile pairwise Acc., Average of Pairwise Area, 20-percentile pairwise Pressure, 20-percentile pairwise Change of Area-Position, 50-percentile pairwise Area Acc., End Point of Pairwise Pressure, Start Point of Pairwise Change of Area-Position, 50-percentile pairwise Y Acc., 80-percentile pairwise Area Acc., Start Y, 20-percentile pairwise X-Tilt, End to End Y Distance, Start Point of Pairwise Pressure, 50-percentile pairwise Change of X-Tilt Position, Average of Pairwise Pressure, Stop Y, Average of Pairwise Y-Tilt, 50-percentile pairwise Area, Std. Dev. of Pairwise Pressure, Start Point of Pairwise X-Tilt, 80-percentile pairwise Change of X-Tilt Position, 80-percentile pairwise Pressure, Average of Pairwise Change of Area-Position, Start Point of Pairwise Y, 50-percentile pairwise X-Tilt Acc., 80-percentile pairwise Y Acc., Average of Pairwise X-Tilt Velocity, Average of Pairwise Y Velocity, 20-percentile pairwise Y-Tilt, Average of Pairwise X-Tilt Acc., End Point of Pairwise X-Tilt, 20-percentile pairwise Change of Pressure-Position, 50-percentile pairwise X Acc., Start Point of Pairwise X-Tilt Velocity, Std. Dev. of Pairwise Y-Tilt, Average of Pairwise Velocity
Right Swipe	52.27%	76.37%	57.48%	86.24%	50-percentile pairwise Pressure Acc., Average of Pairwise X-Tilt, Stop Pressure, Start Pressure, 80-percentile pairwise Pressure Acc., Median of First 5 Acc. Points, Std. Dev. of Pairwise X-Tilt, Start Point of Pairwise Area, 20-percentile pairwise Area, Start Y, 80-percentile pairwise Area, End Point of Pairwise Area, Start X, Average of Pairwise Area, Std. Dev. of Pairwise Area, 50-percentile pairwise Area Acc., 20-percentile pairwise Pressure, Start Point of Pairwise Change of Area-Position, End Point of Pairwise Pressure, 80-percentile pairwise Change of X-Tilt Position, Average of Pairwise Direction, Average of Pairwise Pressure, Start Point of Pairwise X-Tilt, Average Velocity, End to End Y Distance, 20-percentile pairwise X-Tilt, 50-percentile pairwise Y Acc., Start Point of Pairwise Pressure, Average of Pairwise X-Tilt Velocity, 50-percentile pairwise Area, Average of Pairwise Y-Tilt, Stop Y, Std. Dev. of Pairwise Y-Tilt, 80-percentile pairwise Y Acc., 50-percentile pairwise Acc., 50-percentile pairwise X Acc., Std. Dev. of Pairwise Pressure, Average of Pairwise Change of Area-Position, 80-percentile pairwise Pressure, 80-percentile pairwise Area Acc., 50-percentile pairwise X-Tilt Acc., 50-percentile pairwise Change of X-Tilt Position, 20-percentile pairwise Y-Tilt, Average of Pairwise Velocity, 20-percentile pairwise X-Tilt Velocity, Start Point of Pairwise Y-Tilt, Start Point of Pairwise Y, End to End X Distance, End Point of Pairwise X-Tilt, Start Point of Pairwise Change of Pressure-Position
Keystroke	26.25%	60.00%	41.02%	75.00%	Inter-Stroke Time, Stroke Duration, Start X, Start Pressure, Start Y,Start Area, Mid-Stroke Finger Orientation, Key Error Rate
Tap	29.58%	63.33%	34.73%	79.54%	Inter-Stroke Time, Stroke Duration, Start X, Start Pressure, Start Y,Start Area, Mid-Stroke Finger Orientation
Handwriting	68.71%	81.16%	73.73%	91.11%	Average Pressure, Average Y, End Point of Pairwise X-Tilt, 80-percentile pairwise Pressure, 20-percentile pairwise Pressure, 80-percentile pairwise Area, Mid-Stroke Pressure, drawing width, 50-percentile pairwise Pressure, 80-percentile pairwise Y, Average of Pairwise Pressure, Std. Dev. of Pairwise X-Tilt, Start Pressure, Stop Pressure, End to End Y Distance, Start Point of Pairwise Pressure, End Point of Pairwise Y-Tilt, 20-percentile pairwise Y, End Point of Pairwise Pressure, 80-percentile pairwise X-Tilt, Std. Dev. of Pairwise Pressure, Start Point of Pairwise Direction, 50-percentile pairwise Y, 80-percentile pairwise X-Tilt Velocity, std. Dev. of Pairwise Change of Area-Position, TMP, Std. Dev. of Pairwise Change of Pressure-Position, Average of Pairwise Y, Std. Dev. of Pairwise X-Tilt Velocity, 20-percentile pairwise Area, Start Y, Std. Dev. of Pairwise Y-Tilt, 80-percentile pairwise Y-Tilt, drawing area, End to End Pressure Distance, 50-percentile pairwise Direction, 80-percentile pairwise Direction, 80-percentile pairwise Y-Tilt Velocity, Direct End To End Distance, Average of Pairwise X-Tilt, Start Point of Pairwise Y, Std. Dev. of Pairwise Y-Tilt Velocity, Stop Y, LMP, Stroke Duration, Start Point of Pairwise Change of Pressure-Position, 20-percentile pairwise Direction, 50-percentile pairwise X-Tilt, 80-percentile pairwise Change of X-Tilt Position, 20-percentile pairwise Change of Area-Position

* Acc. refers to Acceleration.

Table A.3: Summary of Results - Gestures Combinations

Gesture	Rel. Inf.	TPR	FPR	Gesture Combinations	Rel. Inf.	TPR	FPR
Swipes, Taps	48.11%	72.72%	16.75%	Swipes, Keystrokes	46.80%	91.10%	22.40%
Swipes, Handwriting	72.75%	93.75%	10.88%	Taps, Keystrokes	33.55%	91.11%	33.83%
Taps, Handwriting	66.31%	88.57%	12.68%	Keystrokes, Handwriting	68.26%	72.41%	13.17%
Swipes, Taps, Keystrokes	93.87%	39.4%	2.3%	Swipes, Taps, Handwriting	96.10%	68.75%	1.8%
Swipes, Keystrokes, Handwriting	98.54%	51.85%	0.85%	Taps, Keystrokes, Handwriting	95.06%	51.72%	2.46%
All Gestures	98.93%	40.74%	0.99%				

Appendix B

This supplementary section gives more insights on the information presented in the chapter 7.

B.1 Time-Series Analysis

The procedure to understand a time-series through a number of observations and then to make useful forecasting is known as *time-series analysis*. Time-series analysis has gained indispensable attention over last few decades in the fields such as business, economics, finance, science and engineering. Time-series analysis has two major goals: modeling and forecasting. The time-series modeling helps us developing a model that describes the inherent structure of the series while studying past observations. This model is then used to generate future values for the series i.e. to make forecasts. A variety of time-series forecasting models have been proposed in the literature with the aim to improve forecasting accuracy [31, 226, 50, 93].

In our obfuscation framework, we utilize the concept of time-series processing where we consider data as time-series that is being sent to a remote server to get some functionality. A time series is a sequence of data points, measured typically over successive times. It is mathematically defined as a set of vectors $x(t)$, where $t = 0, 1, 2, ...$ representing the time elapsed and variable $x(t)$ is treated as a random variable. A time-series $x(t), t = 0, 1, 2, ...$, generally follows a probability model which describes the joint distribution of the random variable x_i. Mathematically, this structure is termed as stochastic process, where the sequence of observations of the series is a sample realization of the stochastic process that produced it [50].

An important concept here is the stationary of a stochastic process which states that statistical properties of the process (mean and variance) do not change over time. This condition helps in making useful future forecasting and also reduces the mathematical complexity of a model. A process $x(t), t = 0, 1, 2, ...$ is said to be strongly stationary if the joint probability distribution function $\{x_{t-s}, x_{t-s+1}, ..., x_t, ...x_{t+s-1}, x_{t+s}\}$ is independent of t for all s. There are some mathe-

matical tests such as Dickey and Fuller [65] that are generally used to detect stationary in a time series data. In order to design a proper model and perform adequate forecasting, the underlying time series is expected to be stationary. Usually if a time series shows trend or seasonal patterns, then it is considered as non-stationary and methods such as differencing or power transformations are used to make the series stationary. A number of models have been proposed to address non-stationary in time-series data [31, 226, 61].

In this regard, one of the most well-known stochastic time series models is the TBATS model, which accounts for multi-seasonality (Trigonometric, Box-Cox transform, ARMA errors, Trend, and Seasonal components) in time-series data [61]. TBATS can cater a wide variety of seasonal patterns and can also avoid falling into non-linearity problems through the use of Box-Cox transformation [188]. The TBATS model can be expressed as follows:

$$y_t^{(w)} = \begin{cases} \frac{x_t(w)-1}{w}, & (x_t) - 1 \neq 0 \\ \log(x_t), & w = 0 \end{cases} \tag{B.1}$$

Here, $y_t^{(w)}$ represents the Box and Cox transformed observation with parameter w and x_t is the observation at time t. If the arguments are listed, the TBATS model is written as : $TBATS(w, \phi, p, q, \{m_1, k_1\}, \{m_2, k_2\}, ...\{m_t, k_t\})$, where w is the Box-Cox parameter, ϕ is damping parameter, p and q are ARMA model parameters, and $m_i, i = 1, 2, \cdots, t$ are seasonal periods with the number of harmonics $k_i, i = 1, 2, \cdots, t$ for the seasonal component, respectively. For more understanding on the TBATS model and its parameters, we refer the readers to De Livera et al.'s work [61].

B.1.1 Correlated Noise in a Time Series

In many situations, the behaviour of sequential data points in time-series affects each other in a dependent manner. For example, when a user writes a letter "a" on his mobile touchscreen using a stylus or a finger, the data points in this time-series are correlated, as there is a specific pattern associated with the writing. The deviation from a pattern can make it hard for a user to understand and also affects the functionality of a system. Similarly, in a GPS system, new coordinates (x_t, y_t) are dependent on previous ones (x_{t-1}, y_{t-1}) in order to make a clear route. In general, we can say that a time-series data coming from mobile or smart devices is highly correlated because of two reasons: i) human-beings naturally perform tasks in a certain manner ii) data must be input in a certain pattern or a format in order to

get a desired functionality or quality output. From a privacy perspective, Kifer et al. [128] demonstrated that ignoring the correlation in the dataset lowers the privacy of the published data This condition holds true for not only offline time-series but also makes sense for a time-series that is released/published on the fly i.e. at run time.

Wang et al. [215] presented an attack model where an adversary can easily separate independent and identically distributed (IID) noise (which is being added for privacy preservation) from a correlated time-series data. The correlation of a time-series can be represented by its auto-correlation function [36]. Let's assume that an original time-series X and a released time-series X' (after the noise addition) has an auto-correlation function of $R_{XX}(\tau)$ and $R_{X'X'}(\tau)$, respectively. The IID noise series Z has an auto-correlation function $R_z(\tau)$ which can be expressed as;

$$R_Z(\tau) = N_0\delta(\tau), \tag{B.2}$$

where N_0 is the Power Spectral Density (PSD) of Z and $\delta(\tau)$ is the impulse function. Now if, the original series X is also IID, then the IID noise in the time-series guarantees that the auto-correlation functions of X, X', and Z are the same, as mentioned in Equation B.3.

$$R_{XX}(\tau) = R_{X'X'}(\tau) = R_Z(\tau) = N_0\delta(\tau) \tag{B.3}$$

In other words, the noise retains the consistency of the noise and original series in terms of the statistical properties [215]. Thus, an attacker cannot make use of refinement methods to filter the noise from the released time-series to increase the probability of a successful attack.

However, on the other hand, if X is a correlated time-series but a noise series Z is still IID, then an attacker who has background knowledge and knows about the auto-correlation function $R_{XX}(\tau)$[1], can easily filter the noise from the released series using an optimal waveform estimation filter (e.g., a Wiener linear filter) [46]. After filtering noise, the attacker can get a sanitized estimation $\widetilde{X}$ of X'. Therefore, privacy-preserving techniques developed for correlated time series data need to address the correlation of data in the time-series.

[1]Here, the $R_{XX}(\tau)$ becomes a bilateral attenuation function of X.

Appendix C

C.1 TouchTrack Privacy Policy:

The goal of this project is to measure and study the uniqueness of touch-based behavioral biometric of a mobile device user. We are committed to protect the privacy of users of TouchTrack. All of the data for the project will be collected in an anonymized form which ensures that it is not Personally Identifiable Information, nor otherwise likely to lead to the exploitation of user identities.

We have established this TouchTrack Privacy Policy to explain what information we collect through the mobile app and how it is used. In this policy, "we" refers to the TouchTrack Team, i.e. Principal Investigator, Co-Investigator, Researchers, Interns, Developers, all of whom are bound by law or contract to keep information they receive as confidential.

C.1.1 Information Gathered by TouchTrack Mobile App

In general, TouchTrack collects anonymous raw data against four types of touch gestures i.e. swipes (left, right, up, down), tap, keystrokes, and handwriting. We use the term "raw data" for the touch data that is collected directly through Android API. This raw data, corresponding to each gestures, is collected when you play games provided in an app. The games (2048, Lexica, Logo Maniac) are very widely known and universally popular.

When you interact with a mobile device while playing games, our app collects raw data and sends it to a server, located in the networks group of data61-CSIRO (over HTTPS), for estimating a user uniqueness. When a user taps on "Results" button, the server processes the raw data to calculate uniqueness value and sends results back to the app for display. The specific list of raw features we collect includes:

- Screen Coordinates (i.e. X & Y positions)
- User Finger Pressure on a Screen

- User Finger Size on a Screen
- Screen Orientation (portrait or landscape)
- Finger Movement Type (up, down, move)
- Values from Sensors (Accelerometer, Gyroscope)
- Device Orientation (phone position in terms of angle)

Although these raw data may form a 'fingerprint' that could in principle be combined with information about mobile device or browser fingerprinting in order to track individuals, We will never do so. In addition, we collect 'housekeeping' information to assist us in analyzing the fingerprint data. The housekeeping information is:

- Event Timestamp
- User ID
- Android ID
- Mobile Model Name

Our practices and purposes for collecting these housekeeping records are discussed below:

Event Timestamp
TouchTrack collects a timestamp each time a user performs any gesture. This will be used to measure time-series features, such as stroke time, key hold time, duration of performing a swipe etc.

User ID
TouchTrack requests its users to register with a unique username so that their touch information is saved and retrieved afterwards. For security purposes, we are storing one-way cryptographic hash (SHA1) of usernames in our database. The main purpose of keeping username is to keep track of game progress such that a user can resume again. Moreover, we also want to determine how often user touch behavior change, when a user returns over time. Another temporary purpose is to establish a ground truth for our research and to know how reliable our uniqueness framework is. TouchTrack links the username with his/her touch gestures such

that a user can see previous results, whenever a user logs in.

Android ID

TouchTrack does not log Android ID, but we do compute cryptographic hash of each Android ID, using SHA1 and storing that hash in a database. This hashed Android ID will allow us to collect an anonymous dataset about a user interacting on multiple devices e.g. tablet and a mobile phone. We actually want to study how user behavior changes when they interact on multiple devices. Additionally, we may need to retain this information for situations such as app testing, diagnosis of technical problems, and handling a spike in traffic or other abnormal, short-term circumstances.

Mobile Model Name

We are collecting mobile model name to validate the study described in the above section. Additionally, we want to check what touch features are offered by different mobiles. This information is very necessary since we need common features that could be collected for every type of mobile device. For-example, few mobile phones have only accelerometer sensor while others have both accelerometer and gyroscope. This information is necessary to collect in order to show consistency among features.

Sharing of TouchTrack data

We will not share the data collected through TouchTrack with any external entities. It will remain within the boundaries of CSIRO research environment.

Security

Although we make good faith efforts to store information collected by TouchTrack in a secure operating environment, we cannot guarantee complete security. Information collected will be maintained for a length of time appropriate to our needs.

Should you have any questions about this privacy policy or any use of the data collected, please contact us.

Research Participant Information Sheet

Implicit Tracking Using Behavioural Biometrics

Project overview

This is the data collection phase of our project entitled *"TouchTrack: Implicit Tracking using Behavioural Biometrics"*. By participating, you will find out how unique your touch gestures are and how traceable you could be just by the use of specifics of your behavioural gestures. We call this notion **Implicit Tracking!** The purpose of our project is to quantify the uniqueness of touch-based behavioural biometrics of mobile device users such that their physical identification is possible. Findings from the study will help us protect the privacy of mobile device users by developing privacy-preserving techniques so that user mobile device interactions are not enabling user tracking. The study is being funded by CSIRO, the Australian Commonwealth Scientific and Industrial Research Organization. We would also like to ask you to spread the word about this project and encourage friends, relatives, and colleagues to download and participate in this study. More information about the project is available on the project website.

What does participation involve?

Your responses to this project will help us evaluate the uniqueness of user behavioural gestures on mobile devices such as tablets and phones. Participation in this project will involve downloading and installing a mobile app, named "*TouchTrack*", on your mobile phones. We have developed a *TouchTrack*, which incorporates three widely known games namely, "2048", "Lexica (find a word)", "Logo Maniac (Guess a Logo)," and one digital handwriting module. The purpose of selecting and using these games is to capture user touch gestures such as swipes, taps, keystrokes, and handwriting, in a most natural way. When you play these games, your touch interaction information will be send to our database server, in a complete secure way using HTTPS.

How to Use TouchTrack?

TouchTrack is available on the Google Play Store. You can download it either by searching with the name or by visiting the given link. The estimated time you need to play all four games is 5-7 minutes, but you do not need to play all games in one session. Rather, you can keep using the app as per your convenience. You can constantly check updated uniqueness value anytime after a minimum number of interaction has been scored.

Risk and benefits

Aside from giving up your time, there are no foreseeable risks associated with participating in this study (see Confidentiality section).

Withdrawal from the research project

Participation in this study is completely voluntary. Your decision whether to participate will not affect your current or future relationship with the researchers or anyone else at CSIRO. Similarly, you are free to stop using and uninstalling the app anytime. If you wish to withdraw your data from the database, simply notify the researchers listed below and your data will be destroyed. You may withdraw from this study at any time up until publication of the final outputs.

Confidentiality

Please note that we are committed to protect the privacy of participants of our project. All of the data will be collected in an anonymized form which ensures that it is not Personally Identifiable Information, nor otherwise likely to lead to the exploitation of user identities. All information about your gestures will be treated confidentially. We are not storing any personal information; even usernames and android id is hashed before storing in a database. The username has no link to your real identity as you can create a username with any combination of strings. The username and android id will not be included in any publications resulting from the study. All data collected in this study will be analysed and reported in such a way that responses will not be able to be linked to any individuals. De-identified, non-sensitive data collected by the project may be shared with other researchers for the purposes of verifying published results or advancing other research on this topic. To learn more about the project's privacy policy, visit this link.

How will my information be used?

It is anticipated that the information obtained through the mobile app will be published and/or presented in several research venues. This includes scientific journals, conferences presentations, seminars, and invited talks about security and privacy of emerging technologies, such as mobile devices, Internet of Things, and wearable devices, along with internet measurement and security venues. Data collected through the mobile app may also be used in future research being undertaken by CSIRO on privacy and security of mobile and wearable devices.

Ethical clearance and contacts

This study has been approved by CSIRO's Social Science Human Research Ethics Committee in accordance with Australia's *National Statement on Ethical Conduct in Human Research (2007)*. If you have any questions concerning your participation in the study please contact the researchers via their contact details below. Alternatively any concerns or complaints about the conduct of this study can be raised with the Manager of Social Responsibility and Ethics on (+61 7) 3833 5693 or by email at csshrec@csiro.au.

Prof. Dali Kaafar
Project Principal Investigator
Networks Group
Cyber Physical Systems Program
CSIRO Data61
Level 5, 13 Garden Street
Eveleigh, NSW 2015
Ph: +61 2 9490 5635
Email: Dali.Kaafar@data61.csiro.au

Dr. Hassan Jameel Asghar
Project Co-Investigator
Networks Group
Cyber Physical Systems Program
CSIRO Data61
Level 5, 13 Garden Street
Eveleigh, NSW 2015
Ph: +61 2 9490 5889
Email: Hassan.Asghar@data61.csiro.au

Rahat Masood
Researcher
Networks Group
Cyber Physical Systems Program
CSIRO Data61
Level 5, 13 Garden Street
Eveleigh, NSW 2015
Ph: +61 2 9490 5705
Email: Rahat.Masood@data61.csiro.au

Thank you for your time and we look forward for your responses. Should you have any questions, please do not hesitate to contact us.

CONTACT US
t 1300 363 400
+61 3 9545 2176
e csiroenquiries@csiro.au
w www.csiro.au

AT CSIRO, WE DO THE EXTRAORDINARY EVERY DAY
We innovate for tomorrow and help improve today – for our customers, all Australians and the world.
We imagine. We collaborate. We innovate.

FOR FURTHER INFORMATION
Insert Business Unit name
Insert contact name
t +61 0 0000 0000
e first.last@csiro.au
w www.csiro.au/ businessunit