UNIVERSITY OF CRETE
DEPARTMENT OF COMPUTER SCIENCE
FACULTY OF SCIENCES AND ENGINEERING

Analyzing the Impact of Digital Advertising on User Privacy

Panagiotis Papadopoulos

PhD Dissertation

Submitted in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

Heraklion, September 2018

UNIVERSITY OF CRETE

DEPARTMENT OF COMPUTER SCIENCE

Analyzing the Impact of Digital Advertising on User Privacy

PhD Dissertation Submitted

by **Panagiotis Papadopoulos**

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy

APPROVED BY :

Supervisor: Prof. Evangelos P. Markatos, University of Crete, Greece

---

Committee Member: Dr. Sotiris Ioannidis, FORTH, Greece

Committee Member: Asst. Prof. Xenofontas Dimitropoulos, University of Crete, Greece

---

Committee Member: Dr. Nicolas Kourtellis, Telefonica Research, Spain

---

Committee Member: Asst. Prof. Michalis Polychronakis, Stony Brook University, USA

Committee Member: Dr. Nikolaos Laoutaris, Data Transparency Lab, Spain

---

Committee Member: Asst. Prof. Alexandros Kapravelos, North Carolina State University, USA

Department Chairman:

Heraklion, September 2018

Acknowledgments

During this amazing trip called PhD, I was fortunate to have some amazing people supporting me. First of all, I am grateful to my supervisor Prof. Evangelos Markatos for shaping the researcher I am today, and teaching me a unique research mentality. This thesis would not be possible without his advices and our countless talks. I learned so many things next to you that I will be always grateful for the influence you had in my life. I am also grateful to my supervisor in Telefonica Research, Dr. Nicolas Kourtellis for his continuous support and passion for work. It was always a pleasure working with you.

I consider myself very fortunate for being surrounded by brilliant minds, and therefore I need to express my appreciation to Sotiris Ioannidis, Nikolaos Laoutaris, Elias Athanasopoulos, Michalis Polychronakis, Kostas Magoutis and Xenofontas Dimitropoulos for their advices, mentoring and all the creative brainstorming time we spent together. Also I want to thank every single co-author I had all these years: Elias Papadopoulos, Ilias Leontiadis, Giorgos Vasiliadis, Panagiotis Ilia, Antonis Papadogiannakis, Alexandros Kapravelos, Eirini Degkleri, Michalis Diamantaris, Michalis Pachilakis, Antonios Chariton, Giorgos Christou, Thanasis Petsas, Alexandros Kornilakis, Manolis Karabinakis. We worked together, we got rejected together, we resubmitted together. I really enjoyed working with you guys.

After spending a bit more than 7 years in Distributed Systems (DCS) lab, I feel like I have found a second family there. I met and worked alongside great people in this lab and I want to express my deepest gratitude to Antonis Krithinakis, Dimitris Deyannis, Panagiotis Garefalakis, Nikos Tsikoudis, Demetris Antoniades, Iason Polakis, Christos Papachristos, Lazaros Koromilas, Eva Papadogiannaki, Despoina Antonakaki, Meltini Christodoulaki, Antonis Papaioannou, Giorgos Tsirantonakis, Michalis Athanasakis, Evangelos Ladakis, Stamatis Volanis, Zacharias Tzermias, Evangelos Dimitriadis, Laertis Loutsis, Manolis Stamatogiannakis and all other past and present members of DCS for preserving the balance between computers and real life, making the lab a fun place all these years.

I want to express my sincere thanks to Nikos Skotis, Aris Koutsouras, Giorgos Oikonomou, Dimitris Patelis, Marina Aeraki, Stelios Ninidakis, Stephen Tripodianakis, Nikos Patsiouras, Dimitris Chasapis, Dimitris Ioannidis, Bruno Cardoso, Minos Katevas and all other friends for their support, love, tolerance, and all the great moments we have shared.

Finally, I would never have been able to reach this point without the unconditional love and support from my parents Manolis and Evangelia, and my sister Kelina. Thank you for bearing with me and being always by my side through my studies and life.

Abstract

Digital advertising is a multi-billion dollar business that has the power to fuel the entire free Internet. The recent years, it progressively moves towards a programmatic model in which ads are matched to actual interests of individuals collected as they browse the web. The advertiser pays a monetary cost to buy ad-space in a publisher's medium (e.g., website) thus delivering their digital advertisement along with the publisher's interesting content in the visitor's display.

Unlike traditional advertisements in mediums such as newspapers, TV or radio, in the digital world, the end-users are also paying a cost for the advertisement delivery. Whilst the cost on the advertiser's side is clearly monetary, on the end-user, it includes both quantifiable costs, such as network requests and transferred bytes, and qualitative costs such as privacy loss to the ad ecosystem. Indeed, as advertisements become more and more personalized to match the users interests and become as effective as possible, more personal information about the visiting users is needed. Motivated by that, tracking companies deploy sophisticated user-tracking mechanisms retrieving any piece of information can reveal the user's interests and preferences.

Such information may include current and historical geolocations, installed apps, browsing histories, and so forth. All this information is used to form rich user profiles and large audience segments that can be shared with or sold to anyone interested (e.g., advertisers, data brokers, data management platforms, etc.) beyond the control of the users. To conduct such data markets and before performing any background user database merges, different entities perform synchronizations of the different userIDs they have set for the same users. This way they reduce the number of the different "aliases" with which they know a user, increasing this way their capability of re-identifying users when they erase their browser state (i.e., cookies) or even when they browse through VPN to preserve their privacy.

Besides the continuous growth of digital advertising and its impact on our everyday lives, little we know about the flow of information within the participating companies and the interconnections between them. Motivated by that, in this dissertation, we aim to enhance the transparency in this large ecosystem and investigate the bidirectional effect between user privacy and programmatic ad-buying. In particular, we explore the impact of personalized advertising on the users privacy and anonymity given the elaborate deployed user tracking and personal data collecting techniques. We experimentally measure the user information leaks appeared while using websites and mobile apps. Based on the insight gained from these experiments, we design countermeasures to mitigate the privacy loss.

Towards the opposite direction, we study how these collected user data affect the pricing dynamics of programmatic ad-auctions and how much advertisers pay to reach a user. Then, we compare the costs imposed by digital advertising to both users and advertiser for the very same delivered ad traffic. These costs include network overhead, temperature, energy consumption, loss of privacy. Finally, in an attempt to investigate privacy-preserving alternatives for web monetization that can be completely detached from any personal data requirement, we perform a detailed analysis of the profitability and the user-side overheads of the emerging technology of web cryptomining .

Supervisor: Professor Evangelos P. Markatos

Περίληψη

Η ψηφιακη διαφημιση ειναι μια επιχειρηση πολλων δισεκατομμυριων δολαριων που εχει την δυναμη να τροφοδοτει ολοκληρο το δωρεαν διαδικτυο. Τα τελευταια χρονια, προχωρα προοδευτικα προς ενα προγραμματιστικο μοντελο στο οποιο οι διαφημισεις ταιριαζουν με τα πραγματικα ενδιαφεροντα των ατομων τα οποια συλλεγονται καθως αυτοι περιηγουνται στον διαδικυο. Ο διαφημιστης πληρωνει ενα χρηματικο κοστος για να αγορασει διαφημιστικο χωρο στο ψηφιακο μεσο ενος εκδοτη (π.χ. ιστοτοπο) παρεχοντας ετσι την ψηφιακη του διαφημιση μεσα στο περιεχομενο του οπου καταληγει στην οθονη του επισκεπτη.

Σε αντιθεση με τις παραδοσιακες διαφημισεις σε μεσα οπως οι εφημεριδες, η τηλεοραση η το ραδιοφωνο, στον ψηφιακο κοσμο, οι τελικοι χρηστες πληρωνουν επισης ενα κοστος για να λαβουν μια διαφημιση. Ενω το κοστος απο την πλευρα του διαφημιζομενου ειναι σαφως νομισματικο, στον τελικο χρηστη, περιλαμβανει τοσο ποσοτικα, αμεσα προσδιορισιμα κοστη (οπως HTTP requests και μεταφερομενα bytes), οσο και ποιοτικα οπως η απωλεια ιδιωτικοτητας μεσα στο οικοσυστημα των ψηφιακων διαφημισεων. πραγματι, καθως οι διαφημισεις γινονται ολο και πιο εξατομικευμενες ωστε να ταιριαζουν με τα ενδιαφεροντα των χρηστων και να γινονται οσο το δυνατον πιο αποτελεσματικες, χρειαζονται περισσοτερες προσωπικες πληροφοριες για τους επισκεπτες. Με τον τροπο αυτο, οι εταιρειες παρακολουθησης αναπτυσσουν εξελιγμενους μηχανισμους παρακολουθησης χρηστων που ανακτουν οποιαδηποτε πληροφορια που μπορει να αποκαλυψει τα συμφεροντα και τις προτιμησεις του χρηστη.

Αυτες οι πληροφοριες μπορει να περιλαμβανουν τρεχουσες και ιστορικες γεωγραφικες θεσεις, εγκατεστημενες εφαρμογες, ιστορικο περιηγησης, κ.ο.κ. Ολες αυτες οι πληροφοριες χρησιμοποιουνται για την δημιουργια πλουσιων προφιλ χρηστων που μπορουν να μοιραστουν η να πουληθουν σε οποιονδηποτε ενδιαφερομενο (π.χ. διαφημιστες, μεσιτες δεδομενων, πλατφορμες διαχειρισης δεδομενων κ.λ.π.) περα απο τον ελεγχο των ιδιων των χρηστων.

Για την διεξαγωγη τετοιων αγορων δεδομενων και πριν απο την πραγματοποιηση συγχωνευσης οποιασδηποτε βασης δεδομενων, διαφορετικες οντοτητες εκτελουν συγχρονισμους των διαφορετικων αναγνωριστικων χρηστη που εχουν ορισει για τους ιδιους χρηστες. Με αυτον τον τροπο μειωνουν τον αριθμο των διαφορετικων αναγνωριστικων με τα οποια γνωριζουν εναν χρηστη, αυξανοντας ετσι την ικανοτητα τους να ταυτοποιησουν ξανα τους χρηστες οταν αυτοι διαγραφουν ολα τα δεδομενα του browser τους (π.χ. τα cookies) η ακομα και οταν περιηγουνται στο δικτυο μεσω VPN για να διατηρησουν την ανωνυμια τους.

Εκτος απο την συνεχη αναπτυξη της ψηφιακης διαφημισης και την επιδραση της στην καθημερινοτητα μας, ελαχιστα γνωριζουμε για την ροη πληροφοριων μεταξυ των συμμετεχουσων εταιρειων και τις διασυνδεσεις μεταξυ τους. Με αυτη την διπλωματικη εργασια επιδιωκουμε να

ενισχυσουμε την διαφανεια σε αυτο το μεγαλο οικοσυστημα και να διερευνησουμε την αμφιδρομη επιδραση μεταξυ της ιδιωτικοτητας των χρηστων και της ψηφιακης διαφημισης. Συγκεκριμενα, διερευναμε τον αντικτυπο της εξατομικευμενης διαφημισης στην ιδιωτικοτητα και την ανωνυμια των χρηστων, δεδομενων των περιπλοκων εφαρμογων παρακολουθησης χρηστων και τεχνικων συλλογης προσωπικων δεδομενων. Μετραμε πειραματικα τις διαρροες πληροφοριων χρηστων που εμφανιζονται κατα την περιηγηση ιστοτοπων και χρηση εφαρμογων σε κινητα. Με βαση τις γνωσεις που αποκτηθηκαν απο αυτα τα πειραματα, σχεδιαζουμε αντιμετρα για να μετριασουμε την απωλεια της ιδιωτικης ζωης.

Επιπροσθετα, μελεταμε πως αυτα τα δεδομενα που συλλεγονται απο χρηστες επηρεαζουν την τιμολογηση των προγραμματιστικων δημοπρασιων διαφημισης και ποσα πληρωνουν οι διαφημιστες για να προσεγγισουν εναν χρηστη. Στη συνεχεια, συγκρινουμε το κοστος που επιβαλλεται απο την ψηφιακη διαφημιση τοσο στους χρηστες οσο και στους διαφημιζομενους για την ιδια διαφημιση. Αυτα τα κοστη μπορουν ειτε να προσδιοριστουν αμεσα (π.χ. θερμοκρασια συστηματος, bytes , ενεργεια) ειτε να ειναι ποιοτικα, οπως η ιδιωτηκοτητα. Τελος, σε μια προσπαθεια να διερευνηθουν εναλλακτικες λυσεις για την διατηρηση των διαδικτυακων εσοδων των παραγωγων περιεχομενου, οι οποιες δεν απαιτουν χρηση προσωπικων δεδομενων, διεξαγουμε μια λεπτομερη αναλυση της αναδυομενης τεχνολογιας του web cryptomining αναφορικα με την κερδοφορια της για τους παραγωγους περιεχομενου και του κοστους της στον χρηστη.

Επόπτης: Καθηγητής Ευάγγελος Π. Μαρκάτος

The research outlined in this dissertation has been conducted in the Institute of Computer Science of Foundation of Research and Technology, Hellas and Telefonica Research Labs in Barcelona, Spain.

Contents

Table of Contents . . . xv
List of Figures . . . xix
List of Tables . . . xxv
1 Introduction . . . 1
1.1 Research Questions . . . 3
1.2 Contributions . . . 4
1.3 Outline of Dissertation . . . 6
1.4 Publications . . . 7
2 Background . . . 9
2.1 User Tracking . . . 9
2.1.1 Third party tracking in web sites . . . 9
2.1.2 Third party tracking in mobile apps . . . 9
2.2 Real Time Bidding . . . 9
2.2.1 The key players . . . 11
2.2.2 RTB price notification channel . . . 12
3 Personal Data Collection and User Tracking . . . 15
3.1 Our dataset . . . 17
3.2 Monitoring outgoing traffic . . . 19
3.3 Privacy leak Analysis . . . 20
3.3.1 Encrypted sessions . . . 21
3.3.2 Identifiers leaked . . . 21
3.3.3 Diffusion of privacy leaks . . . 22
3.3.4 Mobile browsers leak too . . . 22
3.3.5 Performance cost of user tracking . . . 23
3.3.6 Lessons Learned . . . 24
3.4 Fortifying Apps from Trackers . . . 25
3.4.1 Our approach: antiTrackDroid . . . 25
3.4.2 Implementation . . . 26
3.5 Evaluation . . . 27
3.5.1 Privacy performance . . . 27
3.5.2 Latency overhead . . . 27
3.5.3 Benefits from the use of antiTrackDroid . . . 29
4 Common User Identification . . . 33

4.1 Cookie Synchronization . . . 35
4.1.1 How does Cookie Synchronization work? . . . 36
4.1.2 Privacy implications for users . . . 36
4.1.3 Cookie Synchronization and Personalized Advertising . . . 37
4.2 Cookie Synchronization Detection . . . 37
4.2.1 Heuristics-based detection . . . 38
4.2.2 Cookie-less detection . . . 40
4.3 Dataset . . . 43
4.3.1 Users . . . 44
4.3.2 Cookies . . . 44
4.4 Privacy Analysis . . . 45
4.4.1 Initiation of Cookie Synchronization . . . 46
4.4.2 How are users exposed to CSync? . . . 47
4.4.3 Buy 1 - Get 4 for free: ID bundling and Universal IDs . . . 48
4.4.4 Sharing sensitive information together with userIDs . . . 49
4.4.5 Who are the dominant CSync players? . . . 50
4.4.6 Spilling userIDs from Secure Sessions . . . 51
4.5 Measuring ID-Spilling in the wild . . . 54
4.5.1 Data analysis . . . 55
4.6 Evaluation of Cookie-less Detection . . . 58
4.6.1 Detecting CSync in ID-sharing HTTP . . . 60
4.6.2 Detecting CSync in HTTP with ID looking strings . . . 60
4.7 Discussion . . . 61
4.7.1 Lessons Learned . . . 62
4.8 Countermeasures . . . 63
5 The Impact of User Data on Ad-pricing dynamics . . . 65
5.1 Technical Challenges . . . 66
5.1.1 Encrypted vs. cleartext prices . . . 66
5.1.2 Encrypted prices on the rise . . . 68
5.2 Methodology . . . 68
5.2.1 Overall cost of the user's data . . . 69
5.2.2 Price Modeling Engine . . . 70
5.2.3 YourAdValue . . . 71
5.3 Bootstrapping PME . . . 72
5.3.1 Dataset analysis . . . 72
5.3.2 Geo-temporal features . . . 73
5.3.3 User-related features . . . 74
5.3.4 Ad-related features . . . 76
5.3.5 Summary . . . 78

5.4 Charge Price Estimation . . . 78
5.4.1 Dimensionality reduction of features . . . 79
5.4.2 Ad-campaigns setup . . . 80
5.4.3 Ad-campaigns analysis . . . 81
5.4.4 Encrypted price modeling . . . 83
5.5 User Cost for Advertisers . . . 84
5.5.1 Encrypted vs. cleartext price distributions . . . 84
5.5.2 How much do advertisers pay to reach a user? . . . 85
5.5.3 Summary . . . 87
5.6 Discussion . . . 87
5.6.1 Limitations . . . 87
5.6.2 Computing The financial worth of individuals . . . 88
6 Costs of Advertising on Users and Advertisers . . . 89
6.1 Cost Analysis with OpenDAMP . . . 90
6.1.1 Quantitative & Qualitative User Costs . . . 91
6.1.2 The OpenDAMP framework . . . 92
6.2 The view of the User . . . 93
6.2.1 Network resources consumption . . . 93
6.2.2 User privacy loss . . . 97
6.3 The view of the Advertiser . . . 98
6.4 Consolidating the two Views . . . 100
6.4.1 Cost on data plan vs. Cost of RTB . . . 101
6.4.2 Cost of Privacy vs. Cost of RTB . . . 102
6.5 Discussion . . . 103
6.5.1 Learnings . . . 103
6.5.2 Impact of Advertising Cost . . . 104
6.5.3 Reducing or rebalancing the costs . . . 105
7 Web-Mining as an Alternative Monetization Model . . . 107
7.1 Background . . . 109
7.1.1 Web-based cryptocurrency mining . . . 109
7.1.2 How does web mining work? . . . 109
7.1.3 Cryptojacking . . . 110
7.2 Data collection and analysis . . . 111
7.2.1 WebTestbench framework for utilization analysis . . . 111
7.3 Analysis . . . 113
7.3.1 Profitability of publishers . . . 113
7.3.2 Costs imposed on the user side . . . 115
7.4 Discussion . . . 122
7.4.1 User awareness . . . 122

7.4.2 Letting users choose . . . 122
7.4.3 Web-miner detection . . . 122
7.5 Summary . . . 123
7.5.1 Lessons Learned . . . 123
7.5.2 Can web-mining become the next web monetization model? . . . 124
8 Related Work . . . 125
8.1 User tracking and Device Fingerprinting . . . 125
8.1.1 User ID Sharing . . . 126
8.2 User data and the Ad-Ecosystem . . . 127
8.2.1 Costs of Advertising . . . 128
8.3 Web-Mining and Monetization . . . 130
9 Conclusion . . . 133
9.1 Synopsis of Contributions . . . 133
9.2 Lessons Learned . . . 135
9.3 Directions for Future Work and Research . . . 136
Bibliography . . . 139

List of Figures

2.1 High level overview of the RTB ecosystem. Several entities interact with each other, exchanging user's personal data before it is finally converted to money. 10

3.1 High level overview of the data collection process. 17
3.2 Classification of apps based on the different content categories. 17
3.3 Number of analytics- or ad-related libraries per app. 43.33% of apps does not contain any such library with the remaining 56.67% containing at least one. . 17
3.4 Percentage of apps, in which each ad-libraries is detected. 18
3.5 Overview of the monitoring methodology of apps and web related traffic. . . . 19
3.6 Use of SSL in apps and web. 20
3.7 Number of 3rd party tracking domains, with which the device interacts when the user accesses each of the top 20 privacy-leaking services. 22
3.8 Percentage of apps, in which the top tracking domains were detected. 22
3.9 Number of requests to tracking domains for each browser, while fetching google.com. 23
3.10 Distribution of total requests sent by services when accessed from web and app. 24
3.11 Distribution of tracking requests sent. 24
3.12 Distribution of total KBytes transferred. 25
3.13 Distribution of the tracking related KBytes transferred. 25
3.14 Defense mechanism overview. 26
3.15 Number of leaked ID without and with antiTrackDroid for the 30 apps with the higher number of ID leaks. 28
3.16 Overall request forwarding time with and without antiTrackDroid. 28
3.17 Benefits from the use of antiTrackDroid. 28

4.1 Example of two entities (advertiser.com and tracker.com) synchronizing their cookieIDs. Interestingly, and without having any code in website3, advertiser.com learns that: (i) cookieIDs userABC==user123 and (ii) userABC has just visited the particular website. Finally, both entities can conduct server-to-server user data merges. 35
4.2 High-level overview of the internal components of CONRAD. 38
4.3 Heuristics-based Cookie Synchronization detection mechanism. 40

4.4 Distribution of number of unique domains visited per user per month. The median user in our dataset visits 20 - 30 different domains per month. 43
4.5 Distribution of number of times a user revisits the same domain per month. The median user revisits the same domain around 7-10 times per month through their mobile browser, while the 90th percentile may revisit the same domains up to 25-29 times. 43
4.6 Number of (first and third-party) cookies per domain per user. We see that the median user receives around 10 cookies per visited website. 44
4.7 Number of unique userIDs set per domain across the year. 80% of the users are known to a single domain with only 2 aliases, on average, throughout the entire year. 44
4.8 Distribution of the time it takes for the first CSync to appear per user. Around 20% of the users get their first userID synced in 1 day or less, when 38% of users get synced within their first week of browsing. 46
4.9 CSyncs per HTTP request for the average user through the year, normalized with their total number of requests. The average user receives 1 synchronization every 68 HTTP GET requests. 46
4.10 Distribution of the synced userIDs per user. The median user has 7 userIDs synced, and 3% of users has up to 100 userIDs synced. 47
4.11 Distribution of synchronizations per userID. The median userID gets synced with 3.5 different entities. 47
4.12 Distribution of the number of entities learned at least one userIDs of the user with and without the effect of Cookie Synchronization. As we can see, after syncing the entities that learned about the median user grew by a factor of 6.75. 48
4.13 Portion of synced userIDs per content category. As expected, the vast majority regards ad-related companies. 50
4.14 Portion of synced userIDs learned per entity over HTTP: 3 companies learn more than 30% of the total userIDs in our dataset; 14 companies learn more than 20% each. 50
4.15 High level overview of the TLS session leak. A privacy-aware user (1) visits a webpage (*example.com*) over TLS and VPN. (2) It sends tracking information to tracker1.com, and receives its cookie over TLS. (3,4) It takes only a HTTP-based Cookie Synchronization (among tracker1.com and tracker2.com) in order to spill user unique identifiers and visited website. Then, a snooping ISP can re-identify the user just by monitoring the synced cookies, even if their real IP address is hidden. 51
4.16 Distribution of the leaked TLS URLs per affected user. The median of those users has 70 TLS URLs leaked through Cookie Synchronization, when the 90th percentile has up to 226 TLS URLs leaked. 54

4.17 Distribution of the portion of TLS-based synchronizations per website for both TLS and non-TLS websites. As we see, the median non TLS website has around 27% of its Cookie Synchronizations over TLS when, most of the TLS-protected websites have 92% of their synchronizations over TLS. 56
4.18 Distribution of non-TLS synchronizations per TLS website. Few websites (1 in 13) include quite a lot (up to 100!) of plain-HTTP CSync redirections. . . 56
4.19 Distribution of the non-TLS synchronizations per leaked userID. There is a 10% of IDs that gets synced with more than 17 third parties. 57

5.1 Portion of encrypted and cleartext pairs of ADX-DSP over time (2015). . . . 67
5.2 Cumulative portion of cleartext prices vs. ad-entities' portion of RTB. 67
5.3 High level overview of our method. The user deploys YourAdValue on her device, which calculates in real-time the total cost paid for her by advertisers. In case of encrypted prices, it applies a decision tree model derived from the PME. 69
5.4 Distribution of charge prices per city (sorted by city size). 74
5.5 Distribution of charge prices for different times of day. 74
5.6 Distribution of charge prices for different days of week. 75
5.7 Portion of RTB traffic for top mobile OSes. 75
5.8 Portion of RTB traffic normalized by OS. 75
5.9 Distribution of charge prices per mobile OS. 75
5.10 CDF of the generated cost per IAB category. 76
5.11 Ad-slot size popularity through time (sorted by area size). 76
5.12 Distribution of the charge prices per ad-slot size (sorted by area size). 77
5.13 Accumulated revenue per ad-slot size (sorted by area size). 77
5.14 Comparison of CPM costs for the different IAB categories in our dataset and the 2 probing ad-campaigns. 82
5.15 Comparison of price distributions between cleartext and encrypted, for different time periods and datasets (D vs. $A1$ and $A2$). 84
5.16 Cumulative CPM paid per user in our year long dataset. 85
5.17 Total cleartext vs. total estimated encrypted cost of each user in D (color indicates nu- mber of users). 86
5.18 Average cleartext vs. average estimated encrypted price per impression of each user in D. 86
5.19 Preliminary implementation of YourAdValue Chrome extension in use. 88

6.1 An example use of CSync in programmatic advertising. Advertisers can track and re-identify users while they surf the web. 91

6.2 HTTP requests produced per user, across the year. Users create a relatively stable HTTP traffic, increased during holiday periods. 93
6.3 Volume of total consumed KBytes per user, across the year. Users consume an average of 5.9 GBytes per month. 93
6.4 Portion of HTTP requests produced across the day. As expected, users produce web traffic mostly from morning till early afternoon. 95
6.5 Portion of HTTP requests per content category the average user fetches through the year. On average, 77% of the HTTP requests is associated with the content the user is actually interested in. 95
6.6 HTTP requests received per user, per different resource type. 95
6.7 Bytes received per user, per different resource type. 95
6.8 KBytes per ad-related HTTP request per user, across the year. 96
6.9 Ad-related KBytes downloaded per user, through the year. 96
6.10 Portion of CSyncs per content category pair, through the year. 97
6.11 Synchronizations per HTTP request users receive through the year. The median user is exposed to a steady number of CSyncs. 97
6.12 Unique synced userIDs per user. The 50th (75th) percentile user gets up to 63 (195) unique IDs synced, at least once. 98
6.13 Portion of the total userIDs in our dataset each tracking entity learned. Some entities have learned more than 10% of all userIDs. 98
6.14 Number of entities having access to a portion of a user's IDs. The median user loses up to 20% of its anonymity to 22 tracking entities. 99
6.15 Although there is an OpenRTB standard [110], every company follows its very own protocol with different parameter naming, making RTB price filtering a challenging task. 99
6.16 The RTB market share of the different bidders in our dataset. As we see, the market share is mainly divided to a dozen of companies with the top 5 wining 67.7% of the RTB auctions. 100
6.17 Cost per user for advertisers to display ads across the year. The average cost per impression for the median user is 0.9 CPM. The total cost paid by advertisers for the median user is ∼22 CPM. 100
6.18 CDF of the average cost on the users' data plan, and cost paid by advertisers to deliver personalized ads to the same users. 101
6.19 CDF of the average CSyncs per impression retrieved per user, across the year. 101
6.20 Heatmaps of (a) average cost per impression for Bytes consumed by users in advertising requests, (b) average Cookie Synchronizations per impression, both compared against the average cost paid by advertisers to deliver RTB ads to the same users (1-1 mapping), across the year. 102

7.1 Cryptomining market share per third party library in our dataset. Coinhive owns the dominant share (69%) when JSEcoin follows with 13%. 110
7.2 High level overview of our measurement testbed. A Chrome-based platform fetches each website for a specific time and its different components measure the resources. 112
7.3 Estimation of monthly profit for the different monetization methods for a website with 100K visitors and average visit duration of 1 minute. Even for visitors with powerful devices (300Hashes/sec), a publisher gains 5.5× more revenue by including 3 ads in its website. 114
7.4 Revenue per visitor for a website running in a background tab. In order for a publisher to gain higher profit from mining than using ads (3 ad- slots), a visitor must keep his tab open for duration > 5.3 minutes (depending on the their device). 114
7.5 Revenue per visitor. In our Hybrid approach the revenue is bounded either before or after the break-even point, to be always higher or equal to both ads and web-mining 116
7.6 Distribution of average real and virtual memory utilization through time. Miner-supported websites although reserve (3.59x) larger chunks of virtual memory, require 1.7x more MBytes of real memory than ad-supported websites.117
7.7 Distribution of the total transmitted volume of bytes per website for a visit duration of 3 minutes. The median miner-generated traffic volume is 3.4x larger than the median ad-generated. In 20% of the websites the difference reduces significantly (less than 2x). 118
7.8 Distribution of the transmitted bit rate per miner-supported website in our dataset. The median in-browser miner communicates with its remote MSP by transmitting 1.17 Kbits per second. 118
7.9 Distribution of average temperatures per system's core. When the visited website includes miner, the average temperature of the cores may reach up to 52.8% higher ($73 - 77^{\circ}$ Celsius) than when with ads. 120
7.10 Impact of background running miner- and ad-supported websites to a user's process. When the majority of ad-supported websites have negligible effect in other processes, the median embedded miner in our dataset through its heavy CPU utilization may cause a performance degradation of higher than 46% to a parallel running process. 121

List of Tables

2.1 Examples of (A) cleartext, (B, C) encrypted RTB price notifications. "ID" is typically a hexadecimal number. 12

3.1 Description of each ID we investigate, their required permissions (Normal permissions are marked with blue, when Runtime/Dangerous permissions with red), their leakability by apps or browsers and the percentage of services found retrieving the corresponding value of each ID. 30
3.2 Identifiers leaked by the most popular browser apps when visiting google.com. 31

4.1 Examples of userIDs getting synchronized between different entities. 39
4.2 Examples of Cookie Synchronization between third parties with plaintext and encrypted cookie IDs. 41
4.3 Summary of contents in our dataset. 42
4.4 Breakdown of the Cookie Synchronization triggering factors. 45
4.5 Example of an *ID Summary* stored on the user's browser with userIDs and cookie expiration dates set by 4 different domains. 49
4.6 Example of leak in our dataset. Our crawler visited over TLS the *https://example.com*. 2 Cookie Synchronizations appear, where *https://tracker1.com* advertiser shares with *tracker2.com* and *tracker3.com* the ID it assigned to the user. By doing that over plain HTTP, the visited website is leaked through the referrer field to a monitoring ISP or other entity. 53
4.7 Summary of results 55
4.8 Performance of decision tree model trained on different subsets of features available at runtime for classification, given already identified id-sharing entries, and 10 cross-fold validation. 58
4.9 Performance of decision tree model trained on different subsets of features available at runtime for classification, given a pre-filter for ID-looking strings. All results besides the last row are with balanced dataset across the three classes, and 10-cross fold validation. The last row's results are computed given an unseen, and unbalanced test set, maintaining the original ratio of classes. . 59
4.10 Detailed performance of decision tree model trained on different subsets of features in a balanced dataset, and tested on an unseen, and unbalanced test set, which maintains the original ratio of classes (last row of Table 4.9. 60

5.1 Summary of notations. 68
5.2 Summary of dataset and ad-campaigns. 72
5.3 Features extracted by summarizing data from parameters embedded in each price notification detected in the dataset for users and advertisers. 73
5.4 Basic filters used in controlled ad-campaigns in Spain. In total, 144 experimental setups were attempted. 81

7.1 Summary of our dataset 111
7.2 Distribution of the average CPU Utilization for the different monetization methods. The median miner-supported website utilizes 59x more the user's CPU than the median ad-supported website. 116
7.3 Distribution of the average consumption of power for the different monetization methods. The median miner-supported website forces the user's device to consume more power than the median ad-supported website: 2.08x and 1.14x more power for the CPU and the memory component, respectively. 119

Chapter 1
Introduction

In today's data-driven economy, the amount of user data an IT company holds has a direct and non- trivial contribution to its overall market valuation [211]. Digital advertising is the most important means of monetizing such user data; it grew to $194.6 billion in 2016 [218] and $209 billion in 2017 [119] of which $108 billion were due to mobile advertising. In order to be more effective, digital advertising has become more personalized and targeted thus changing the model of modern ad-buying. Nowadays, there is a whole new ecosystem, where each ad-slot on a user's display is getting auctioned in programmatic real-time auctions where advertisers bid based on the profile of the user and how well their interests match the advertised product.

As a result, we see more and more IT companies rushing to participate in this rapidly growing advertising business either as advertisers, ad-exchanges (ADXs), demand-side platforms (DSPs), data management platforms (DMPs), or all of the above. All these different types of entities are parts of the same process: the process of converting user data to money.

Evidently thus, the model of digital advertising has been become heavily data-centric: the efficiency and effectiveness of the personalized ad delivery process depends on the quantity and quality of the data known for the current user. All of these necessary user data are nothing more than information that the user generates with their actions while surfing the Internet. This information, when collected and processed, can form a rich user profile that includes interests, preferences, financial status, etc. of a potential product buyer of an advertised product. Hence, it is apparent, that in personalized advertising such information is valuable and therefore the collection of such data the recent years has become more aggressive and sometimes even intrusive [96, 104]. This aggressiveness has raised a huge public debate around the tradeoffs between (i) innovation in advertising and marketing, and (ii) basic civil rights regarding privacy and personal data protection [139, 155].

These increasing privacy concerns, is the motive to explore the current advances of user tracking and measure the privacy loss of users. Although there is a significant body of research in desktop devices, the same does not applies in the in the booming era of IoT. We do not know what kind of personal information gets leaked while using websites or apps on

a mobile device. And given that (a) mobile devices have become strictly personal (i.e., each user uses it's own device which carries almost always with them), equipped with different kinds of sensors (e.g., accelerometer, gyroscope, GPS, etc.), (b) the heterogeneity of apps is huge with different developers being able to penetrate the app market getting their own fair share, and (c) considering the progressive shift of user traffic from desktops to these devices, it becomes apparent that transparency with respect to privacy is of paramount importance, and more investigation is required regarding what third party entities have access on what data (sensitive data, personal identifiable information (PII), user device fingerprinting information) and how do they collect them (with or without user's permission). Such privacy analysis, will increase the awareness of users regarding which of the two options (browser accessed website or mobile app) facilitates the most privacy leaks while accessing the same online service from their mobile device.

Apart from analyzing the user tracking techniques and their privacy implications, still little we know about how the leaked user personal data are getting shared among the different entities and what is the information flow within the ad ecosystem. In addition, there is a significant lack of transparency regarding the monetization of this information and how the quality and quantity of the personal data affect the pricing dynamics. Although, we receive hundreds of personalized advertisements in our everyday life, yet we do not know what is the monetary cost these advertisers pay to reach each one of us. There exist several reports about the average revenue per user (ARPU) from online advertising [43, 99, 188], but ARPU, as its name suggests, is just an average. It can be calculated by dividing the total revenue of a company by the number of its monthly active users. Computing the revenue per individual user is a completely different matter for which very limited work is available.

Given the impact of user's data to the ad-pricing, the above estimation of the monetary value advertisers pay to reach an individual, increase the user awareness regarding the monetary value spent for the privacy loss they sustain. But are the advertisers only the ones paying for the ad delivery? Do users indeed get content free of charge from an ad-supported service? Contrary to the traditional advertising (i.e. in newspapers, TV, radio), in the digital world, it is not only the advertiser that pays this cost, but the user as well! Indeed, besides the costs on privacy, users have also to sustain costs on their dataplan, energy consumption, network traffic while downloading the additional ad-, analytics- and tracking- related KBytes. Having these hidden costs in mind, it becomes questionable who pays the most for the same transmitted ad traffic: the receiver (the user) or the sender (the advertiser).

The existence of the hidden costs have also made users take measures to reduce the costs they pay. Indeed, a growing number of users (615 million devices – 30% growth since last year [46]) decided to abdicate from receiving ads by adopting all-out approaches (like deploying ad-blocking mechanisms [38, 179, 245] or ad-stripping browsers [25, 39, 75]). This increasing ad-blocking trend made some major web publishers, after seeing their income to significantly shrink (total losses of $22 billion [162]), to deploy ad-blocker detection tech-

niques [115, 161, 173] and deny serving content to ad-blocking users [42, 159, 216, 223]. Such aggressive actions from both sides escalated an inevitable arms race between the ad-ecosystem on the one side, and the ad-blockers and privacy advocates on the other side [20, 163, 173].

In such a dispute, evidently, publishers are trapped in the crossfire being unable to effectively monetize their services. To that end, it did not take long for some of them to look for reliable alternative schemes to support their websites. Some of these schemes include paid website versions and user compensation (e.g., Basic Attention Token [12]). Recently, the frenetic growth of cryptocoins made distributed in-browser mining an effective alternative for content monetization. Specifically, publishers borrow CPU cycles from the user's system while they browse the website's content in order to compute hashes required by the cryptominer. A great advantage of web mining over advertising is its zero user data requirements (i.e., no user tracking and personal data collection is needed), thus making it unequivocally GDPR compliant. Besides the popularity web mining enjoys, its profitability and the imposed costs on the user side are not explored adequately, yet. As a result, it is still unclear in what extent web mining can constitute the next primary monetization model for the post-advertising era of free Internet.

1.1 Research Questions

This thesis raises a number of research questions which motivate the work presented in this dissertation:

1. What kind of personal information gets leaked while using websites and apps on a mobile phone?

2. Which of the two: browser accessed website or mobile app facilitates the most privacy leaks considering the same online service?

3. How trackers and advertisers synchronize the user IDs they have set for the same user profiles before they participate in data markets?

4. In the era of personalized advertising how much do advertisers pay to reach an individual? And how the personal data affect the pricing dynamics in programmatic ad-buying?

5. Are the users indeed receiving the content they want free of charge in the ad-supported web? If not, what are the costs the median user has to sustain and how comparable these costs are with the monetary cost advertisers pay to deliver them ads?

6. Are there alternatives to the data-centric model of digital advertising nowadays? Can the resource-borrowing model of cryptomining constitute the next primary monetization model in the post-advertising era of free Internet?

1.2 Contributions

Thesis Statement: In this dissertation, we show that *while users seemingly retrieve online content for free, in the data-centric model of the ad-supported Internet, they have to sustain hidden costs like privacy loss and in some cases monetary costs greater than what advertisers pay to deliver the ad traffic. Experimental results show that data-less models based on resource borrowing may become an effective and reliable alternative way of web monetization under specific circumstances.*

The key contributions of this dissertation are the following:

1. We conduct a comparative study regarding the device-related privacy leaks of mobile apps and mobile browsers in Android. Our dataset includes a set of online services accessible by both mobile apps and mobile-friendly web pages. For each of them, we investigate if the associated app leaks more or less information than its website, accessed through mobile browser. We design antiTrackDroid: a novel anti-tracking mechanism for mobile devices. Similar to state-of-the-art browser ad-blockers, our approach blocks any possible request may deliver to third parties data that can be used either for user profiling or device fingerprinting. We implement antiTrackDroid as an integrated filtering module for Android. antiTrackDroid uses a mobile- based blacklist, which we publicly release, and it does not require changes in the respective OS or any kind of external infrastructure (i.e. proxy). We experimentally evaluate our prototype and show that it is able to reduce the leaking identifiers of apps by 27.41% on average, when it imposes an insignificant latency of less than 1 millisecond per request.

2. In order to explore how trackers construct a universal identification for each user, we design a heuristics-based mechanism to detect Cookie Synchronization (CSync) events, along with the participating entities. Our detection mechanism, contrary to the early proposed ones, is able to detect synchronizing userIDs wherever they are piggybacked in the: 1) URL parameters, 2) URL's path, or 3) referrer field. Using this detection mechanism, we conduct the first to our knowledge in depth analysis of Cookie Synchronization in the real (mobile) world, using a year-long dataset of 850 volunteering users. Our results show that 97% of regular web users are exposed to CSync. In addition, the average user receives around 1 synchronization per 68 GET requests, and the median userID gets leaked, on average, to 3.5 different ad-entities. Furthermore, CSync increases the number of entities that track the user by a factor of 6.7. Additionally, we present in full detail how a last mile observer can snoop parts of a user's browsing history through Cookie Synchronization. As a proof of concept, we perform active measurements through our secure VPN, by probing the top 12K Alexa sites, and explore the feasibility and spread of these attacks in the wild. By fetching the landing pages, we collect a dataset of 440K HTTP(S) requests, and by using it as input to our

analyzer, we show that 1 out of 13 of the most popular websites worldwide expose their users to these two privacy-breaching attacks.

3. To measure the impact of the collected user data on the pricing dynamics of the ad ecosystem, we propose the first to our knowledge holistic methodology to calculate the overall cost of a user for the RTB ad ecosystem, using both encrypted and cleartext price notifications from RTB-based auctions. We study the feasibility and efficiency of our method by analyzing a year-long weblog of 1600 real mobile users. Additionally, we design and perform an affordable (a few hundred dollars cost) 2-phase real world ad-campaign targeting ad-exchanges delivering cleartext and encrypted prices in order to enhance the real-users' extracted prices. We show that even with a handful of features extracted from the ad-campaign, our methodology achieves an accuracy $> 82\%$. The resulting ARPU is 55% higher than that computed based on cleartext RTB prices alone. Our findings challenge the related work's basic assumption that encrypted and plain text prices are similar (we found encrypted prices to be around 1.7x higher). Finally, we validate our methodology by comparing our average estimated user cost with the reported per user revenue of major advertising companies. Using lessons from this study, we design a system where the users, by using a Chrome browser extension, can estimate in real-time, in a privacy-preserving fashion on the client side, the overall cost advertisers pay for them based on their exposed personal information. In addition, they can also contribute anonymously their impression charge prices to a centralized platform for further research.

4. We design a methodology to measure the costs a user pays when receiving ad-related traffic. These costs may be either directly quantifiable (e.g., requests, bytes, energy), or qualitative such as loss of privacy. In addition, our methodology estimates the costs advertisers pay to display each of the advertisements a user receives through the contemporary programmatic RTB auctions. We implement our methodology in OpenDAMP (open Digital Advertising Measurement Platform): a framework for passive weblog analysis oriented to digital advertising. OpenDAMP can analyze user HTTP traffic and detect ID sharing incidents among third parties (known as Cookie Synchronizations). In addition, by incorporating information from external resources and blacklists, OpenDAMP can classify the traffic based on the content the domains deliver, and extract metadata and charge prices from RTB ad-auctions. To assess the effectiveness of our methodology, we apply it on the above year-long dataset of real volunteering users. Our analysis shows that the costs advertisers and users pay are largely unbalanced, In fact, users pay 3x more through their data plan to download ads, than what the advertisers pay to deliver them to these users. Furthermore, the majority of users sustains a significant loss of privacy to receive these personalized advertisements.

5. Finally, we explore the potential of web-mining to be used as an alternative, privacy-preserving monetization method for web publishers. Specifically, we conduct the first study on the profitability of web-based cryptocurrency mining. Our results show that for the average duration of a website visit, ads are 5.5x more profitable than crypto-mining. However, a miner-supported website can produce higher revenues if the visitor remains in the website for longer than 5.3 minutes. We design a methodology to assess the resource utilization patterns of ad- and miner- supported websites on the visitor's device. We implement our approach in WebTestbench framework and we investigate what costs these utilization patterns impose on the visitor's side with regards to the user experience, the system's temperature, and energy consumption and battery autonomy. We collect a large dataset of around 200K ad- and miner- supported websites that include different web-mining libraries and cryprocurrencies. We use this dataset as input for the WebTestbench framework and we compare the resource utilization and costs of the two web monetization models. Our results show that while browsing a miner-supported website, the visitor's CPU gets utilized 59 times more than while visiting an ad-supported website, thus increasing the temperature (52.8%) and power consumption (2x) of her device.

1.3 Outline of Dissertation

The rest of this dissertation is organized as follows. Chapter 2 provides some background information about User Tracking and Device Fingerprinting techniques with which entities collect information about the browsing user. Additionally, necessary knowledge about programmatic ad-buying is provided before presenting how contemporary real-time ad-auctions work. Chapter 3 studies the data trackers collect from mobile users through both web browsing and mobile apps in attempt to compare the generated privacy implications of these two content receiving models. After that, a track-blocking countermeasure is proposed, able to reduce the privacy risks in both apps and websites.

Chapter 4 explores how all the collected data are attributed to individual user profiles and how web entities form universal user identifiers by syncing the different userIDs they assign for the same user. By doing so, different companies can merge their databases on the background and participate in user data markets. After presenting our novel technique to detect such userID synchronization events, in this chapter, we analyze a large dataset to explore the characteristics of this tracking mechanism. In Chapter 5, we analyze how the above collected user data affect the pricing dynamics within the ad-ecosystem in an attempt to enhance the transparency regarding the growing real-time programmatic ad-auctions. Specifically, we develop a first of its kind methodology for computing exactly the price paid for a browsing user by the ad ecosystem in real time. Our approach (namely YourAdValue) is based on tapping on the Real Time Bidding (RTB) protocol to collect cleartext and encrypted prices

for winning bids paid by advertisers in order to place targeted ads.

Building on the above methodology, in Chapter 6, we compare the costs advertisers and users pay for the same ad delivery. These costs may be either directly quantifiable (e.g., requests, bytes, energy), or qualitative such as loss of privacy. To achieve that we implement our methodology in OpenDAMP (open Dig- ital Advertising Measurement Platform): a framework for passive weblog analysis oriented to digital advertising. After analyzing the above user-side costs, in Chapter 7, we explore privacy preserving alternatives for publishers that could be used as content monetization methods like in-browser cryptomining. To determine if such approach could indeed constitute a solid option for publishers, we study the profitability of cryptomining but also the costs it imposes to the user's system.

Chapter 8 surveys prior work and in Chapter 9 we summarize the contributions and results of this dissertation, and outline research directions that can be explored in future work.

1.4 Publications

The research activity related to this thesis has so far produced the following publications (ordered by publication date):

1. Elias Papadopoulos, Michalis Diamantaris, Panagiotis Papadopoulos, Thanasis Petsas, Sotiris Ioannidis and Evangelos P. Markatos. *The Long-Standing Privacy Debate: Mobile Websites Vs Mobile Apps*, In proceedings of the 26th International World Wide Web Conference (WWW'17), 2017, Perth, Australia. [179]

2. Panagiotis Papadopoulos, Nicolas Kourtellis, Pablo Rodriguez Rodriguez, Nikolaos Laoutaris. *If you are not paying for it, you are the product: How much do advertisers pay to reach you?*, In proceedings of the 17th Internet Measurement Conference (IMC'17), 2017, London, UK. [185]

3. Panagiotis Papadopoulos, Nicolas Kourtellis, Evangelos P. Markatos. *The Cost of Digital Advertisements: Comparing User and Advertiser Views*, In proceedings of the 27th International World Wide Web Conference (WWW'18), April 2018, Lyon, France. [183]

4. Panagiotis Papadopoulos, Nicolas Kourtellis, Evangelos P. Markatos. *Exclusive: How the (synced) Cookie Monster breached my encrypted VPN session*, In proceedings of the 11th European Workshop on Systems Security (EuroSec'18), April 2018, Porto, Portugal. [184]

5. Panagiotis Papadopoulos, Panagiotis Ilia, Evangelos P. Markatos *Truth in Web Mining: Measuring the Profitability and Cost of Cryptominers as a Web Monetization Model*, arXiv preprint arXiv:1806.01994, 2018. [181]

6. Panagiotis Papadopoulos, Nicolas Kourtellis, Evangelos P. Markatos. *Cookie Synchronization: Everything You Always Wanted to Know But Were Afraid to Ask*, arXiv preprint arXiv:1805.10505, 2018. [182]

Chapter 2
Background

2.1 User Tracking

2.1.1 Third party tracking in web sites

Traditionally, web sites keep track of users and sessions by using *cookies*. So by storing some state (in the form of cookies) on the client side, an advertising or analytics company can identify a user along with her interests, preferences, or even past purchases. The need for a more centralized infrastructure, which will work as a data warehouse containing rich data for several individual users and their interactions with different web sites, brought in the surface mechanisms like web beacons [106] and cookie synchronization [2]. These mechanisms allow third parties to collect user data bypassing the same origin policy [247], thus enriching their profiles of the users.

2.1.2 Third party tracking in mobile apps

To earn ad revenue, app developers display ads from third parties in their apps. These ads are usually delivered through RTB auctions [82]. To deliver effective advertisements to the end-user, third party ad-networks request from app developers to embed in their apps external third party ad-libraries, also known as in-app libraries [50]. The scope of these ad-libraries is to allow the developer request at runtime an ad-impression to fill the app's available ad-slots. To facilitate the delivery of personalized advertisement, ad-libraries inherit all the permissions enjoyed by the original app such as: access to the phone, access to contacts list, access to device characteristics, etc. In this way, ad-libraries can track users as they use services in cyberspace.

2.2 Real Time Bidding

An RTB auction is a programmatic, instantaneous type of auction, where a publisher's advertising inventory is bought and sold on a per ad-slot basis. RTB accounts for 74% of programmatically purchased advertising, reaching a total revenue of $8.7 billion in US [19],

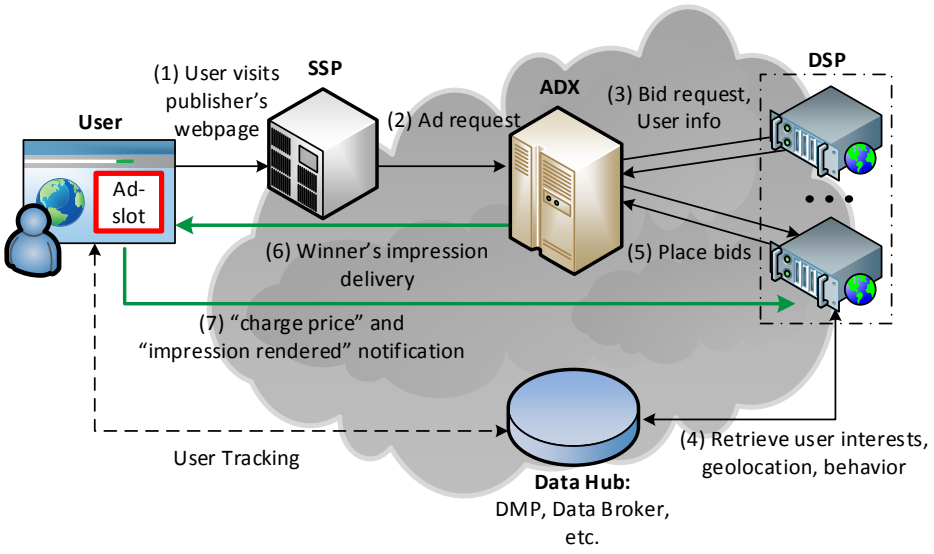


Figure 2.1: High level overview of the RTB ecosystem. Several entities interact with each other, exchanging user's personal data before it is finally converted to money.

allowing advertisers to evaluate the collected data of a given user at real-time and bid for an ad-slot in the user's display.

A typical transaction for an ad-slot begins with a user visiting a website. This triggers a bid request from the publisher (or SSP) to an ADX (step 2 and 3 in Figure 2.1), usually including various pieces of user's data (e.g. browsing history, demographics, location, cookie-related info, minimum acceptable price, etc.). Then multiple DSPs automatically submit their impressions and their bids in CPM (i.e. cost per thousand impressions [113], typically in US dollars or euros, based on the market) to the ADX. All bids are sealed so every participant places only one bid for a particular ad-slot, this allows the RTB auction to finish within milliseconds (the entire RTB protocol usually runs in around 100 ms). The ad slot goes to the highest bidder and its impression is served in the user's display. The charge price for the ad slot is the second higher price following the Vickrey auctions [237]. The main advantage of this type of auctions is that forces all bidders to have their bids truly reflect on what they think the value of the ad slot should be. Note that minimum acceptable price defined by the publisher can act as the second price in an auction, if the second highest bid price falls below it.

2.2.1 The key players

As it can be seen in Figure 2.1, the key roles of the RTB ecosystem include the *Advertiser*, *Publisher*, *DSP*, *Ad-exchange*, and *SSP*, which interact with each other in several ways [11]. Note that it is very common for some (large) companies to play simultaneously different roles even inside the same auction (e.g. Google's DoubleClick Bid Manager [94] and DoubleClick for Publishers [87].

Publisher: (e.g., CNN.com) is the owner of a website, where users browse for content and receive ads (step 1). Each time a user visits the website, an auction takes place for each available ad slot. The ad impression of the winning advertiser is finally displayed in each auctioned slot of the website.

Advertiser: is the buyer of a website's ad slots. The advertiser creates ad campaigns and defines the audience that has to be targeted according to his marketing objectives, budgets, strategies, etc. In each auction, the one with the highest bid wins the ad slot and places its impression on the screen of the website's visitor.

Supply-Side Platform (SSP): is an agency platform, which enables publishers to manage their inventory of available ad slots and their pricing, allocate ad impressions from different channels (e.g. RTB or backfill in case of unsold inventory [128]) and receive revenue[1]. SSP is also responsible for interfacing the publisher's side to multiple ad-exchanges (step 2) and aggregate/manage publisher's connections with multiple ad networks and buyers. In addition, by using web beacons and cookie synchronization, SSPs perform user tracking in order to better configure their ad slots' pricing and achieve as many re-targeting ads as possible and thus higher bids [125]. Popular vendors selling SSP technology are OpenX, PubMatic, Rubicon Project, Right Media.

Ad-exchange (ADX): is a digital, real-time marketplace that, similarly to a stock exchange, enables advertisers and publishers to buy and sell advertising space through RTB-based auctions. ADX is responsible for hosting an RTB-based auction and distribute the ad requests along with user information it owns (i.e. browsing history, demographics, location, cookie-related info) among all the interested auction participants (step 3).

After the auction, the winning impression is served to the user's display within 100 ms of the initiating call (step 6) and the winning bidder is notified about the final charge price. Popular ad-exchanges include: DoubleClick, MoPub, and OpenX.

Demand-Side Platform (DSP): is an agency platform, which employs decision engines with sophisticated audience targeting and optimization algorithms aiming to help advertisers buy the best-matched ad slots from ADXs in a simple, convenient and unified way. DSPs retrieve and process user data from several sources (step 4) such as ADXs, Data Hubs, etc. The result of this processing is translated to a decision in practice: *How much is it worth to bid for an ad slot for this user, if any?* If the visitor's profile matches the audience the

[1]Publishers can also interface directly with ADXs and handle their inventory on their own.

Winning Price Notification URLs
(A) cpp.imp.mpx.mopub.com/imp?ad_domain=amazon.es& ads_creative_id=ID&**bid_price=0.99**&bidder_id=ID&... &bidder_name=..&**charge_price=0.95**&country=..&... &currency=USD&latency=0.116&mopub_id=ID&pub_name=..
(B) tags.mathtag.com/notify/js?exch=ruc&... &**price=B6A3F3C19F50C7FD**&... &3pck=http%3A%2F%2Fbeacon-eu2.rubiconproject.com%2F beacon%2Ft%2Fce48666c-6eb4-46db-b0e9-6f4155eb557d%2F
(C) adserver-ir-p.mythings.com/ads/admainrtb.aspx?googid=ID&.. &width=300&height=250&...&cmpid=ID&gid=ID&mcpm=60&... **rtbwinprice=VLwbi4K21KFAAAm2ziqnOS_O5oNkFuuJw**&..

Table 2.1: Examples of (A) cleartext, (B, C) encrypted RTB price notifications. "ID" is typically a hexadecimal number.

advertiser has focused his ad-campaign on, the DSP will submit to the ADX the impression and a bid in CPM on behalf of the advertiser (step 5). Popular DSPs are MediaMath, Criteo, DoubleClick, AppNexus and Invite Media.

User Data Hub, Data Exchange Platform (DXP): is a centralized data warehouse such as a Data Management Platform (DMP) [22, 127] or a Data Broker [133] which aggregates, cleans, analyzes and maintains user private data such as demographics, device fingerprints, interests, online and offline contextual and behavioral information [118,120]. These user data are typically aggregated in two formats: 1) a full, audience user profile for offline analytics and data mining, 2) a run-time user profile, optimized for real-time requests such as RTB queries from DSPs, before submitting their bids to ADXs [63, 123] (step 4).

Such user profiles are sold to ad entities [14] because they increase the value of an RTB inventory by enabling a more behavioral-targeted advertising (2.7× more effective than non-targeted advertising [15, 251]). In fact, Data Hubs are considered the core component of the digital ad-ecosystem as they perform the attribution and labeling of users' data and create groups, namely *audience segments*, which are useful (i) to the publishers for their customer understanding, (ii) to the SSPs for retrieving more re-targeted ads and (iii) to the DSPs for feeding their bid decision engines. Further, quality scores are impartially assigned to users' private data based on the success of ad-campaigns they were used, thus driving the bid prices of future ad-campaigns. Notable DXPs are Turn, Adobe, Krux, Bluekai, Lotame.

2.2.2 RTB price notification channel

When an ADX selects the winning bid of an auction, the corresponding bidder must be notified about its win to log the successful entry and the price to be paid to the ADX. One

could implement this notification in two ways: (i) with a server-to-server message between ADX and DSP, (ii) with a notification message conjoined with the price, passed through the user's browser as a call-back to the DSP.

The first option is straightforward and tamper-proof; no one can modify or block these messages, allowing companies to ensure that their logs are fully synced at any time. In addition, DSPs can hide information about the transactions, the purchased ad-slots and the prices paid from the prying eyes of competitors. However, DSPs do not have any indication of the delivery of each ad, in order to inform their campaigns and budget.

Instead, the second option not only can ensure the DSP that the winning impression was indeed delivered (the callback is fired soon after the impression is rendered on the user's device), but also gives the opportunity to drop a cookie on the user's device. Therefore, the second option is the dominant one in the current market: the ADX piggybacks a notification URL (nURL) in the ad-response, which delivers to the user the winning impression and the ad (steps 6 and 7 in Figure 2.1). This nURL includes basically the winning DSP's domain, the charge price, the impression ID, the auction ID and other relevant logistics (see Table 2.1 for some examples). In this present work, we study such nURLs and the prices embedded in them, as well as how they associate with the users' browsing behavior and other personal information.

Chapter 3
Personal Data Collection and User Tracking

In the online era, where behavioral advertising fuels the Internet world as we know it, user privacy has become a commodity that is being bought and sold in a complex, and often ad-hoc, data ecosystem [83, 100, 199]. Indeed, users' personal data collected by IT companies constitute a valuable asset, whose quality and quantity significantly affect each company's overall market value [211]. As a consequence, in order to gain advantage over their competitors, IT companies participate in a user data collecting spree, aiming to retrieve and process as much information as possible. This collection of user personal data has become so aggressive (and sometimes even intrusive [96, 104]) that has raised a huge public debate around basic civil rights regarding privacy and personal data protection [139, 155]. These increasing privacy concerns, drew the attention of a significant body of research [2, 60, 67, 172, 180, 202, 232], which studied users' privacy loss in conjunction to existing user tracking techniques that companies deploy in the web.

However, although some years ago the only way to connect and interact with online services (or online web sites) was through *web browsers*, the proliferation of mobile devices and developer tools gave service providers the chance to create their very own mobile *applications* (or apps) that the users download and install in their devices. Each of the above two access choices (i.e. apps vs. web browsers), offers different kinds of advantages [221]. For example, web browsers can be found in most/all mobile devices and provide easy access to any mobile-friendly web site. On the other hand, native apps may offer better support for specific functionalities such as interactive gaming and offline access.

Choosing between the app and the browser is not easy. There is a lot of debating on the web, with several studies trying to compare these two options across different dimensions [24, 27,76,220,221,248]. Fortunately, the majority of service providers support both options: they provide *both* a mobile app *and* a mobile website. Although each option may have different benefits, in this study, we are interested in exploring their privacy-related characteristics.

That is, *which of the two options protects the users' privacy in the best way? apps or browsers?* Or similarly, *which of the two options facilitates the most privacy leaks?*

Certainly, we are not the first ones to deal with this problem. For example, C. Leung *et al.* attempt a comparison of mobile apps and vanilla browsers based on the amount of personally identifiable information (PII) they leak [142]. The authors manually examined a small group of 50 online services and they monitored the PII each of them shares, over plaintext or encrypted connections. Their results show that there is no clear single answer. Therefore, they implemented an online service [141] aiming to recommend the users the best option for accessing a small sample of online services, based on the PII they care about the most.

In this study, we conduct a similar comparison, but we significantly broaden the definition of privacy. Apart from personal data, such as gender, email address, name, username, birthdate, etc., that a service may leak, there is also device-specific information that can be used as identifiers. Such identifiers may include: (i) installed applications, (ii) known SSIDs, (iii) connected wifi, (iv) operating system's build information, (v) carrier, etc. These identifiers although seemingly unable, at a first glance, to reveal any possible sensitive data for the user, they are able, when combined, to allow a monitoring entity to persistently track mobile users without using any deletable cookies or resettable Advertising IDs [60, 154, 244]. This way, the monitoring entity can uniquely identify the mobile user and monitor her behavior, her actions or her interests in the online world: information, which is usually more useful for the web entities (advertisers, analytics, etc.) to obtain than individual and possibly sensitive parts of personal data.

To make matters worse, by deploying device fingerprinting a third party can also: (i) decloak user's anonymous sessions: by linking for example Tor sessions of the same device with non anonymous ones [3,233] and (ii) link web with app sessions, one of the biggest challenges of mobile ad networks [228]. Surprisingly, our results show that in case of device-specific privacy leaks, there is a clear winner: mobile browsers leak significantly less information compared to mobile apps. In most of the cases we studied, mobile apps leaked tons of information that mobile browsers did not (or could not) leak. Thus, we urge users to consider more seriously the use browsers whenever they have the choice.

Unfortunately, this choice may not always be available. For example, web sites may provide poor functionality to mobile devices and thus, the use of apps may seem the only reasonable choice from a user experience point of view. To improve the privacy of users, who *must* use mobile apps, we propose *antiTrackDroid*, an anti-tracking mechanism for mobile apps, tantamount to the current state-of-the-art ad-blockers of mobile browsers. Our approach constitutes an integrated monitoring and filtering module, which contrary to alternative approaches works solely in the users' device, without requiring any additional infrastructure (i.e. proxy or VPN). Our evaluation shows that *antiTrackDroid* is able to reduce the leaking

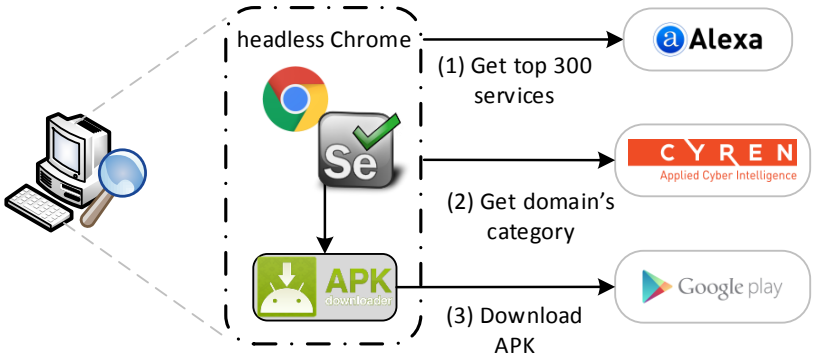


Figure 3.1: High level overview of the data collection process.

identifiers of apps by 27.41% on average, when it imposes an insignificant latency of less than 1 millisecond per request.

3.1 Our dataset

Our dataset contains several popular online services along with their mobile application (app) counterpart. We started with the 300 top online web services from Alexa and, for each one of them, we tried to find its corresponding mobile app. To automate the mobile apps collection process, we used the Selenium suite [212] to instrument Chrome browser and the APK downloader plugin [196] to download the full APKs of the Android apps from

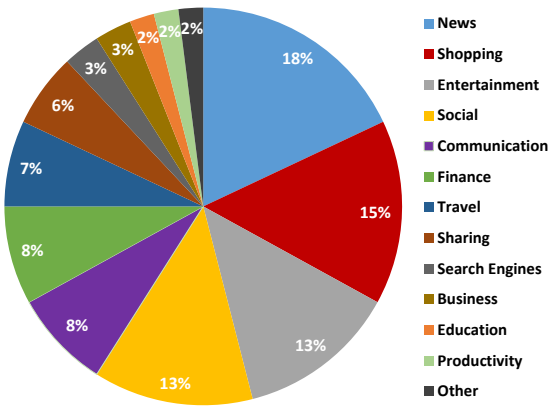


Figure 3.2:
Classification of apps based on the different content categories.

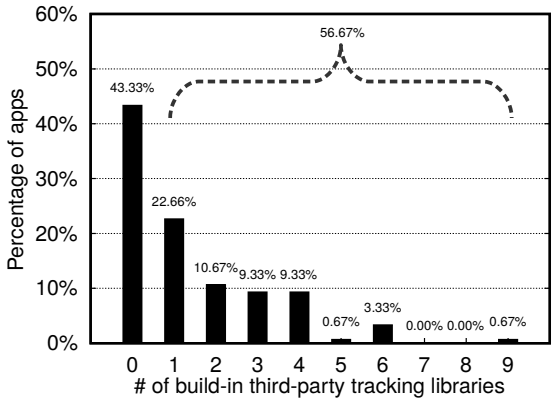


Figure 3.3:
Number of analytics- or ad-related libraries per app. 43.33% of apps does not contain any such library with the remaining 56.67% containing at least one.

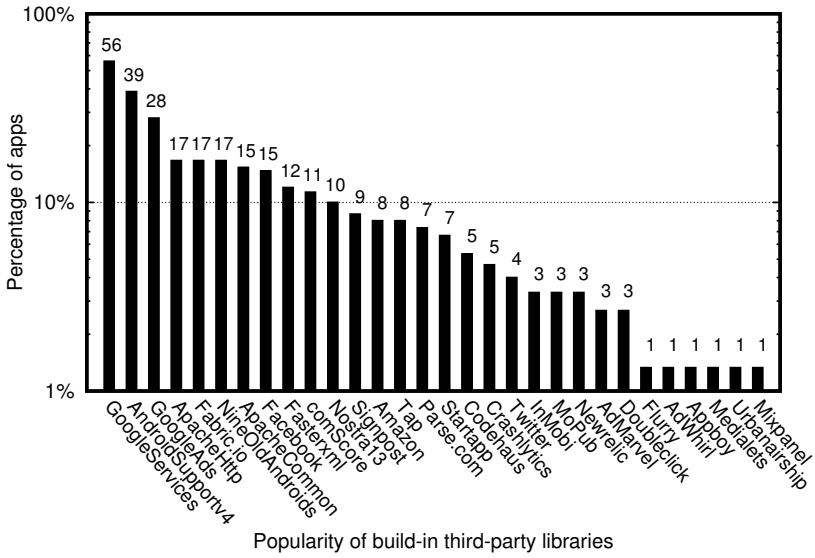


Figure 3.4: Percentage of apps, in which each ad-libraries is detected.

Google Play. Figure 3.1 summarizes our dataset collection process. Note that at the time of experimentation (February 2016) only 116 (of the top 300 Alexa sites) provided a mobile app. Thus, our final dataset consists of 116 apps along with their associated mobile-friendly web counterpart, making it larger than datasets explored in similar manually investigative studies([142]: 50 apps, [198]: 100 apps , [255]: 110 apps).

Before we analyze the privacy leaks of apps and web in our dataset, we first explore the characteristics of our dataset:

Application Categories. As presented in Figure 3.1, we used CYREN [49] to extract the category of each service in our dataset. Figure 3.2 shows the breakdown of our services into different categories. As expected, News-, Shopping- and Social-related services dominate the dataset, but it seems that we have services from all over the spectrum.

Third party in-app libraries. As we discussed in Section 7.1, the majority of free apps embed a third party, in-app library. These third party libraries are used for analytics- and ad- related purposes, thus sending to third parties information about the user, which is important for the targeted advertising delivery. To identify these in-app libraries in our dataset, we use LibRadar [147], a tool for Android which detects embedded libraries, even if obfuscation methods were deployed. After the in-app library extraction, we manually filtered out libraries other than analytics- and ad-related. Figure 3.3 presents the analytics- and ad-related libraries found per app and as we see, 56.67% of apps contain at least one such library, when there is also 1 app with 9 of such in-app libraries. Note that these results, are verified by other studies as well [166]. In Figure 3.4, we see the popularity of the top third party

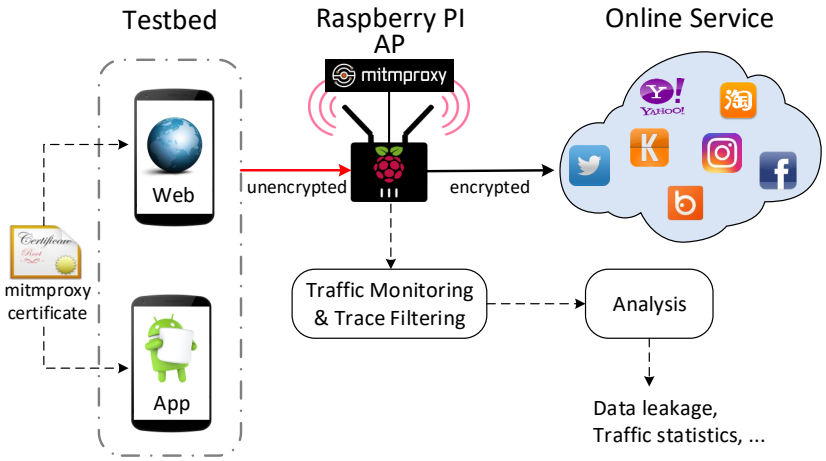


Figure 3.5: Overview of the monitoring methodology of apps and web related traffic.

libraries in our dataset, with GoogleAds residing in 28% of the apps and Fabric.io following with 16.67%.

3.2 Monitoring outgoing traffic

In Figure 3.5, we see an overview of our testbed. Using a NEXUS 6 smartphone running Android 6.0.1, we run each online service (i) from the corresponding app and (ii) from its website by using Firefox mobile browser[1]. Each run lasts for approximately 20 minutes where we perform the same user actions in both counterparts of the service including login, registration, search, share, etc. In order to capture the devices' network traffic (both HTTP and HTTPS), we used a dedicated monitoring component, which captures all the outgoing requests of both apps and browsers. For our monitoring component, we use a raspberry PI 2 [194] device configured as an access point and by running mitmproxy [45], an SSL-capable monitoring proxy, we are able to monitor SSL sessions as well. After capturing both HTTP and HTTPS traffic, the traces are forwarded to our *Traffic Monitoring & Trace Filtering* module, in which the tracking related requests are identified by using a filtering list based on a popular mobile-based blacklist [122], enhanced with entries we collected after manual inspection. Finally, the categorized traffic is passed to the *Analysis* module to produce statistics and the privacy leak analysis results. In this module, we filter possible leaked identifiers by performing pattern matching using a list of ID keywords we discover after studying device's settings. Moreover, we implemented an app able to collect all these IDs, with a view to find and verify the correctness of them. Then, we manually inspect the results to eliminate possible false positives.

Muffling Background Apps. To prevent apps running simultaneously during our experi-

[1]We choose Firefox mobile browser for its ability to support browser extensions

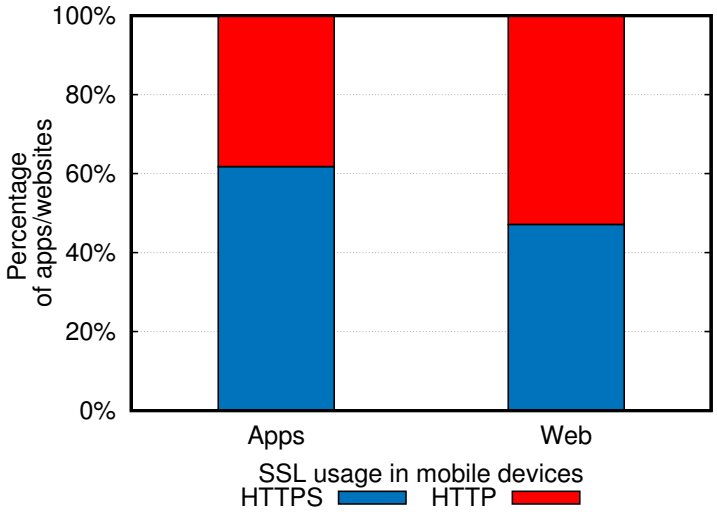


Figure 3.6: Use of SSL in apps and web.

ments, we limit the background app activities while capturing the trace of each app in our dataset. More technically, we use the available developer options of the mobile device and by using a custom bash script that employs the adb toolkit [7], we kill the background processes of the device.

Bypassing SSL Certificate Pinning. Certificate (or SSL) Pinning [95, 239] is a technique used by several mobile apps to avoid MITM attacks. Through SSL pinning, a mobile app checks the certificate received by the server during the SSL handshake, and compares it to a known copy of the particular certificate, that was bundled with the app. To bypass SSL Certificate pinning and allow our monitoring component to capture the SSL traffic unimpeded, we use a modified version of JustTrustMe [241] module of Xposed Framework [204]. By using this module, we hook into SSL-related functions and nullify the code responsible for performing SSL pinning checks.

3.3 Privacy leak Analysis

In this section, we present the core analysis of our privacy-related study. Specifically, by using the network traces we collected with our monitoring proxy, we measure and compare the quantity and the type of information leaked, as well as the diffusion of these leaks in both web and app versions of our collected online services.

Apart from the personal data a service may leak (such as birth-date, email addresses, gender, etc.), there is also device-specific information, which, if leaked, can also be used as identifiers. Although unable, at a first glance, to reveal any personal data for the user, these identifiers are able to allow a tracking entity to follow mobile users inside a network

without using any deletable cookies or resettable Advertising IDs. Table 3.1 presents a short description of the leaks and identifiers we detected.

3.3.1 Encrypted sessions

Both apps and web browsers need to communicate with their associated online service in order to send and retrieve updated information. In our first experiment, we measure the adoption of SSL in the transactions of both apps and websites to explore the possibility of a passive observer to learn user info by monitoring the traffic.

First we measure the use of SSL in apps, and our results indicate that **only 18.97% of the apps use exclusively HTTPS**, 2.58% use solely HTTP, and 78.45% a susceptible [91] mixture of both. Consequently, it should not come as a surprise that we found 2 apps sending user's credentials in plaintext over HTTP. We informed the corresponding providers and we can confirm that at least one of them has fixed it. In Figure 3.6, we compare the use of SSL in web and apps. We see that **apps are more likely to use HTTPS (62% of total traffic) compared to web browsers (47%)**.

3.3.2 Identifiers leaked

In our next experiment, we set out to explore what kinds of identification information is leaked. Table 3.1 presents a list of such identifiers including "Advertising ID", "Android ID", "AP SSID", etc. along with its required permissions. We immediately see that apps are very aggressive at leaking such information (see column "Services(%))"). For example, we see that 57.76% of the apps leak the "Android ID" identifier. Surprisingly, we observe that none of the web browsers (see 4th column) leak this information. This happens, contrary to apps, because *web browsers typically do not have access to this information.* In summary, we see that **apps are much more prone than web browsers to leak device identification information**.

Table 3.1 also shows that mobile apps leak a huge variety of information including the list with the rest of the installed apps, too. This list allows an observing entity to easily infer important PII about the user including gender, age, preferences, interests etc. There are 3.45% of the applications, sending the whole list of the installed apps to a remote domain. Interestingly, in one of them, the remote server after conducting some analysis on the data, responded back with an approximation of the user's gender, age range, a list of possible interests and a number of recommended brand names.

In addition, there are also five applications and one website leaking the nearby WiFi Access Points. Such data can be correlated with online AP maps [23] and reveal the exact location of the user and possible interpersonal relations of people being in the same location at the same time. In addition, there is one app leaking the entire list of known APs, which

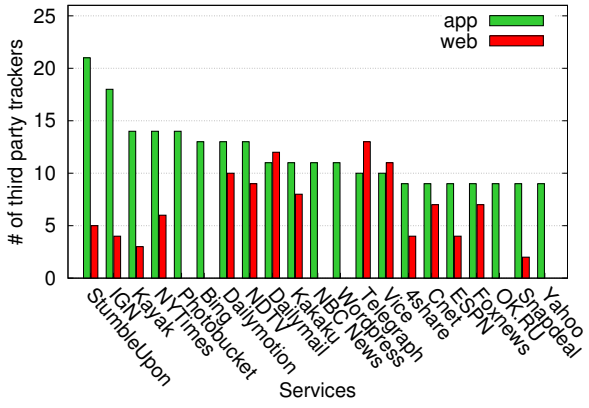


Figure 3.7:
Number of 3rd party tracking domains, with which the device interacts when the user accesses each of the top 20 privacy-leaking services.

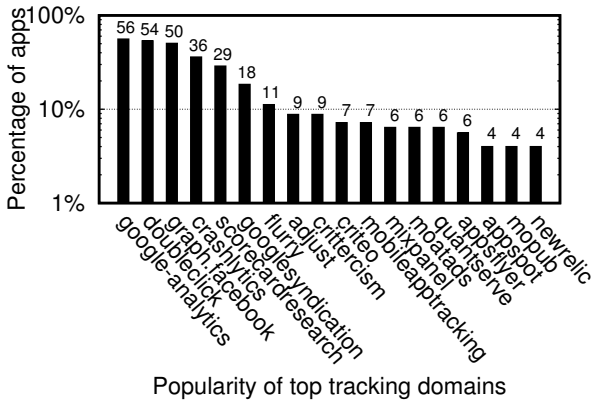


Figure 3.8:
Percentage of apps, in which the top tracking domains were detected.

can reveal previous locations the user has visited, even before the installation of the app. Finally, there is one app leaking the device's current running processes.

3.3.3 Diffusion of privacy leaks

After analyzing the type of information an app and a web browser can leak, we set out to explore the number of third party entities that receive this kind of information. First, we measure the number of third party trackers in apps and web browsers. Figure 3.7 shows that in most cases, it is the app that provides identifying information to more third party trackers. Indeed, we see that apps leak information to an average of 11.7 third party trackers while web browsers leak information to an average of 5 trackers. Similarly, as many as 93.9% of Android apps leak data to one or more third-party trackers, while the corresponding percentage for web browsers is 69.3%.

Finally, in Figure 3.8, we present the most popular tracking domains in the dataset of the collected apps. As we see, Google's analytics and advertising domains (google-analytics, doubleclick) dominate, having access in 56% and 53.6% of the apps respectively. Facebook, one of the most common social media apps, follows with 50.4%.

3.3.4 Mobile browsers leak too

Up to this point, we compare app and web counterparts of the online services. However, let us not overlook that mobile websites are being accessed by web browsers, which are mobile

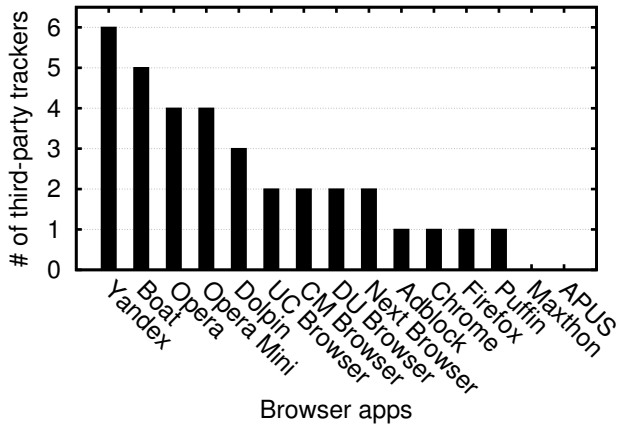


Figure 3.9: Number of requests to tracking domains for each browser, while fetching google.com.

apps themselves. As a consequence, browser apps may leak data to remote entities as well. To explore this, we experimented with the 15 most popular web browsers in Android. We fetch a simple, tracker-free website like google.com using each one of them and we monitor their traffic. As we see in Figure 3.9, the vast majority of browsers sends a significant number of third party tracking requests to the network. Surprisingly, we see that **even the Adblock browser sends a request to a tracking domain** (i.e. adjust.com).

In order to further-investigate the possible privacy leaks of web browsers, we analyze the content of the above tracking requests. In Table 3.2, we present the identifiers each browser leaks to both first and third parties. We see that there is a significant number of browsers leaking an abundance of identifiers (more than 10 in some cases). A careful reader may have observed that although some identifiers are not leaked from a website (see Table 3.1), interestingly, they are getting leaked through the browser app itself (see Table 3.2). Hence, we see that mobile web browsers constitute ordinary apps, thus including third-party trackers of their own. As a consequence, when a user visits a website (e.g cnn.com), the website running inside the browser's sandbox cannot access, for example AdvertisingID, but the browser app (along with its included third-party trackers) can, pairing this way the website visit with the specific AdvertisingID.

3.3.5 Performance cost of user tracking

Trackers do not cost only in the privacy of the user, they also consume resources of both web and apps by generating requests not relevant with the content the user chose to browse. In Figure 3.10 and Figure 3.11, we present the number of the total and tracking requests respectively for both web and app versions of our collected online services. We see that apps,

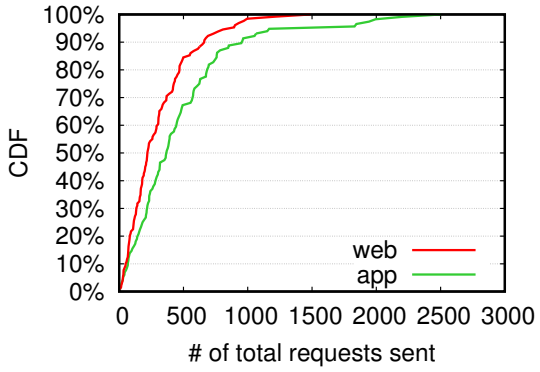


Figure 3.10:
Distribution of total requests sent by services when accessed from web and app.

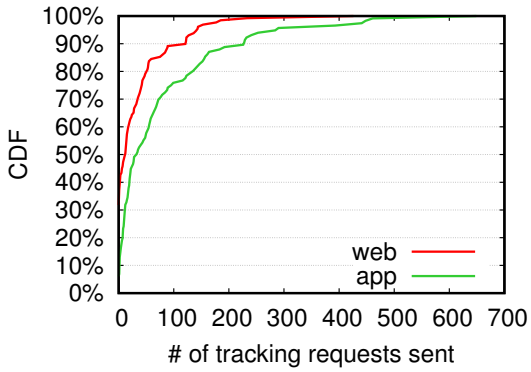


Figure 3.11:
Distribution of tracking requests sent.

in general, send more requests to the network (367 for the 50% of the apps) than web (221 for the 50% of the websites), when the portion of tracking requests is 8.5% and 5% respectively. Regarding bytes, we see in Figure 3.12 and Figure 3.13, that the transferred volume of bytes is similar in both apps and web, as expected given their similar functionality. The tracking related bytes are 192 KB in apps and 77 KB in web per service. Apparently, this amount of Bytes regard unnecessary tracking content, constituting a significant monetary cost for the user's data plan.

3.3.6 Lessons Learned

In the above analysis, we explore the information leaked in each of the two versions of an online service. We see that both web and apps leak important fingerprinting information about the user's device. This allows third parties to not only cross-channel track the users by linking web with app sessions but also correlate eponymous with anonymous sessions. In addition, we see apps leaking information (e.g. installed and running apps, nearby APs, etc.) that may allow a tracking domains to infer the user interests, gender, even behavioral patterns. Furthermore, we studied the diffusion of the above privacy leaks and we see that apps tend to send requests to more tracking domains than their web counterpart.

Our results prove that both versions of the online service leak information that can be used beyond the control of users (for targeted advertising purposes or an asset for sale to other entities [133]). However, recalling the motivating question of our study: *to identify which of the two, web or app, facilitates the most privacy leaks*, the answer is straightforward. Apps leak significantly more device-specific information.

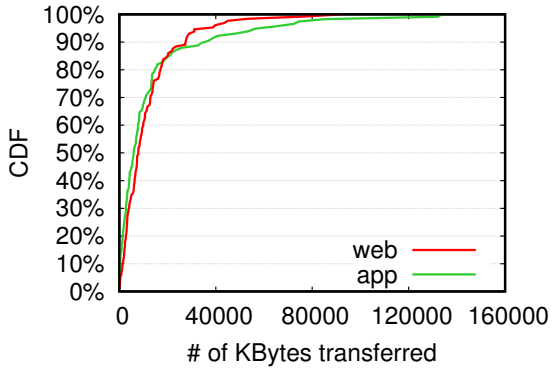


Figure 3.12:
Distribution of total KBytes transferred.

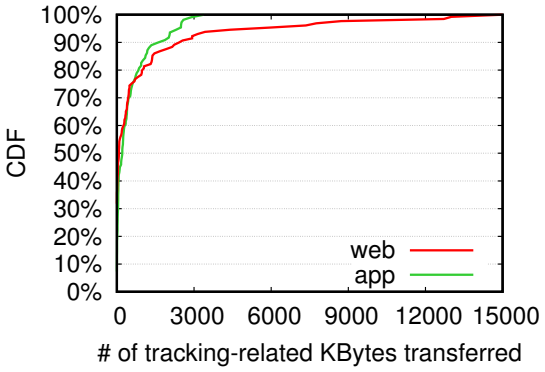


Figure 3.13:
Distribution of the tracking related KBytes transferred.

3.4 Fortifying Apps from Trackers

3.4.1 Our approach: antiTrackDroid

Our findings so far suggest that apps leak more information than web browsers. Thus, it is reasonable for privacy-aware users to prefer using web browsers instead of apps to access online services. However, this is not always possible, or desired [221]. As a result, the use of mobile apps is, in many cases, unavoidable. To provide these users with better privacy guarantees, we propose *antiTrackDroid*: an anti-tracking mechanism able to preserve the privacy of the users by blocking many personal and device information leaks to any third parties. Specifically, antiTrackDroid is a module which filters all outgoing requests and blocks the ones delivering tracking information.

The core design principles of antiTrackDroid include the ability to operate (i) for all apps, and (ii) without the need for any additional infrastructure (e.g. VPN, Proxy, etc.). To meet these principles, antiTrackDroid leverages Xposed [204]: a popular Android framework, which allows system-level changes at runtime without requiring installation of any custom ROM or modifications to the application. By using Xposed, antiTrackDroid is able to intercept every outgoing request and check if the destination's domain name exist in a blacklist of mobile trackers. In case of match (i.e. the destination is blacklisted), the outgoing request is blocked. Figure 3.14 summarizes the design of our approach.

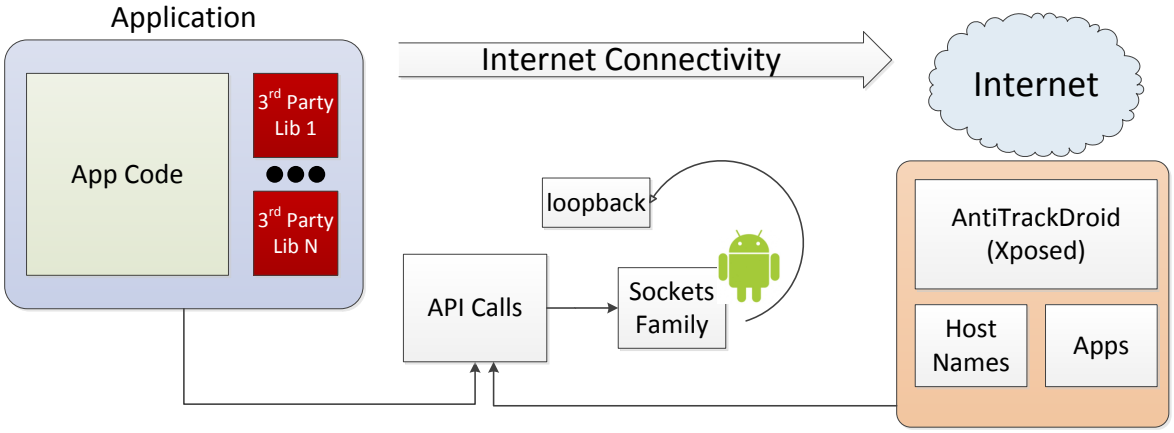


Figure 3.14: Defense mechanism overview.

3.4.2 Implementation

To assess the effectiveness and feasibility of our approach, we implemented a prototype of antiTrackDroid for Android. Our system consists of the following main components:

1. The *Filtering module*, which implements the *IXposedHookLoadPackage* and filters the tracking requests based on the Xposed module.

2. An *Android Activity* (hereafter named *Launcher*), with a graphical user interface to allow the users configure the Filtering module.

3. The *AppList Updater*, which listens for newly added or removed packages and updates the list of applications being monitored, using a Broadcast Receiver.

Launcher Activity. *Launcher* acts as an interface between the Filtering module and the user. It contains a menu allowing the user to (i) load a different blacklist or exclude an application from the filtering procedure. *Launcher* is also responsible of maintaining two different data structures: a HashSet with the tracking domain names loaded from the blacklist, and a HashSet with the applications being monitored. By using HashSets, antiTrackDroid is able to perform look-ups with O(1) complexity reducing significantly the per request latency overhead.

Filtering Module. Mobile applications send data over HTTP/HTTPS requests by using TCP sockets. Therefore, Filtering module dynamically hooks on the constructor of the TCP socket opened by the applications residing in the HashSet of monitored apps. In addition, it re-writes the destination IP address with *localhost* in case of a blocked request. This loop-back interface is a virtual network interface that does not correspond to any actual hardware, so any packets transmitted to it, will not generate any hardware interrupts. By redirecting to loop-back, antiTrackDroid avoids possible crashes of apps caused by aborted connections.

AppList Updater. Since users may install or remove applications at any time, our system must be able to update the list of monitored apps. In Android, every time an app is added or removed in the system, a broadcast message is send through the *PackageManager* component, which can reach any app in the device. By using a *Broadcast Receiver* [8], the AppList updater, running as a background service, can listen such messages and update the list of monitored apps.

Blacklist of Trackers. To determine if a request is a tracker or not in the *Filtering Module*, we use the popular mobile-based blacklist of AdAway [122], which we extended by including the tracking domains we collected manually during our privacy leak analysis. Our publicly released blacklist of antiTrackDroid which we update frequently, currently contains 66k entries in total. Recall that in Launcher Activity, the user is free to change the used blacklist by loading one of her choice.

3.5 Evaluation

In this section, we evaluate the effectiveness and performance of our antiTrackDroid, and we explore its benefits.

3.5.1 Privacy performance

To evaluate the privacy preservation of antiTrackDroid, we inspect the identifiers leaked to the network with and without the use of antiTrackDroid. In Figure 3.15, we see the number of leaked IDs with and without antiTrackDroid for the 30 more leaking apps. Our results show that antiTrackDroid is able to reduce the number of leaked identifiers by 27.41% on the average. Note that since our approach blocks the majority of third party trackers, the rest of the leaking IDs exist due to requests destined to the developer's first party domains and content providers (e.g. CDNs). Blocking such requests would cause degradation of the user experience or even fatal error to the application.

3.5.2 Latency overhead

Although antiTrackDroid significantly improves privacy, it may have an impact on the overall latency of the apps as well. Indeed, antiTrackDroid may increase latency because it includes an extra check with the blacklist. On the other hand, it may significantly reduce the latency imposed by blacklisted tracker requests as these requests will be blocked and the app will not have to suffer their latency. To measure the impact on latency, we created an Android app, with which we can send arbitrary number of requests to a server of ours. Thus, we create 1000 requests carrying 15KB of data each, and we send these requests to the server sequentially after a short time interval. We run this experiment 3 times: (i) once with antiTrackDroid switched off, (ii) once with antiTrackDroid enabled and the domain not included in the

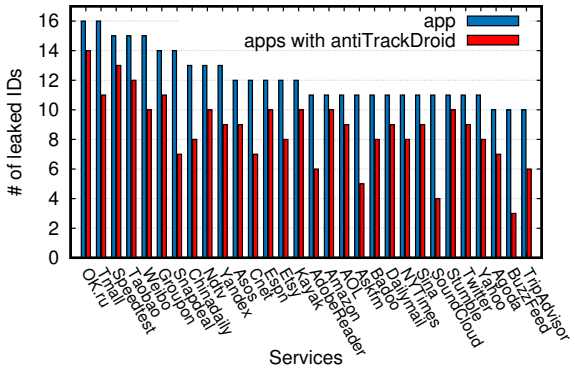


Figure 3.15:
Number of leaked ID without and with antiTrackDroid for the 30 apps with the higher number of ID leaks.

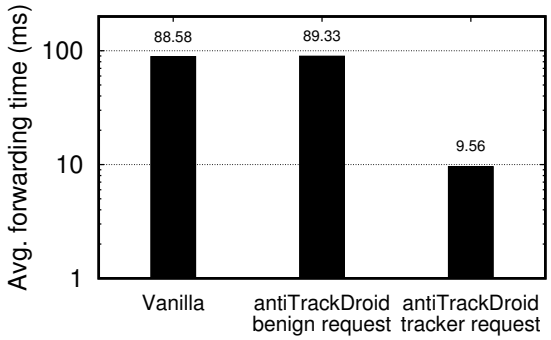
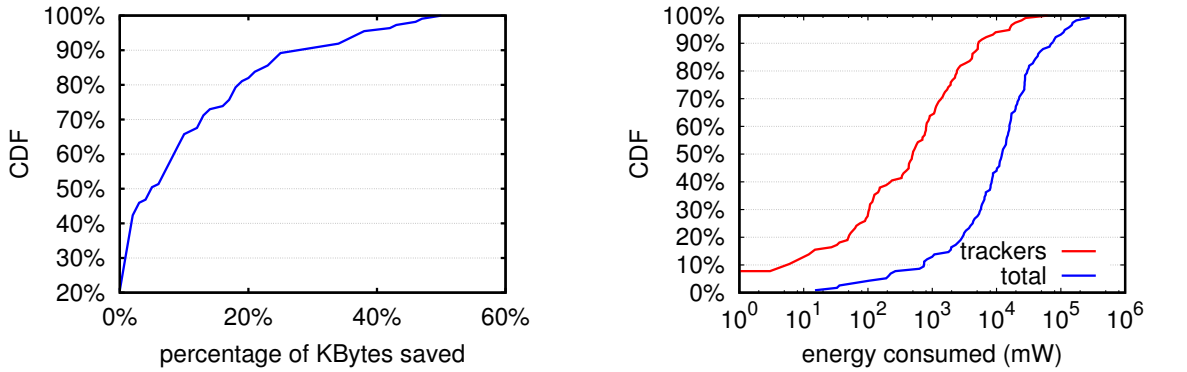


Figure 3.16:
Overall request forwarding time with and without antiTrackDroid.



(a) Percentage of KBytes saved.

(b) Watts saved due to the less tracking request sent.

Figure 3.17: Benefits from the use of antiTrackDroid.

blacklist (benign request), and (iii) one more with antiTrackDroid enabled and the domain of the server blacklisted (Tracker request).

Figure 3.16 shows that the vanilla (no antiTrackDroid) request (see 1st bar) takes about 100 ms. When we switch on antiTrackDroid (see 2nd bar), the latency is practically the same. Indeed, a few lookups in a blacklist do not add any overhead noticeable in the 100 ms range (less than 1 ms). Finally, when we switch on antiTrackDroid and make an access to a tracker (see 3rd bar), the latency drops to less than 10 ms as the request is blocked. We are happy to see that antiTrackDroid, not only improves privacy, but it also improves performance.

3.5.3 Benefits from the use of antiTrackDroid

Besides preserving the user's privacy, the blocking functionality of our approach improves also the performance of apps in the user's data-plan and battery.

Bytes transferred. By blocking the tracking related requests antiTrackDroid is able to save a significant amount of data, an aspect of great importance when it comes to mobile users with specific data plan. To determine the different volume of data transferred to/from the apps in a device running antiTrackDroid, we conduct the following experiment: we run all apps in our device as previously, but instead of blocking the requests we calculate the outgoing bytes of requests and the incoming bytes of the associate responses. In addition, we measure the overall traffic of the app and finally calculate the portion of traffic marked as tracker-related. Figure 3.17(a), presents the results, where we see that antiTrackDroid reduces the volume of transferred bytes by 8% for the 50% of the apps.

Energy cost There are several studies [103, 156] attempting to measure the energy cost imposed by the ad-related content to a user's device. It is apparent that every connection an app opens with a network entity, it imposes an overhead to the overall energy consumption of the device [168]. As a consequence, by reducing the requests an app sends or receives, along with their transferred data, antiTrackDroid is able to reduce the energy cost of the application as well.

Measuring the energy consumption in a mobile device is a challenging task. In order to estimate our gain with antiTrackDroid, we perform a simulation, based on the energy readings of Appscope [252]. Figure 3.17(b) presents the distributions of the per-app power consumption for (i) the total and (ii) the tracker-related transferred bytes. From our simulation we find that there is a significant reduction of about 7,5% for the 50% of the applications.

IDs	Description	Permission Group	App	Services(%)	Web	Services(%)
Build	The Android OS build version code	NONE	✓	100.00	✓	0.00
Model	The device model or its codename	NONE	✓	100.00	✓	9.48
OS Version	The OS or SDK version	NONE	✓	100.00	✓	100.00
Manufacturer	The device manufacturer	NONE	✓	78.45	✓	24.14
Screen Ressolution	The screen resolution of the device	NONE	✓	75.00	✓	42.24
Location	Device's GPS coordinates	LOCATION	✓	66.38	✓	85.34
Carrier	The Mobile Network Operator	PHONE	✓	64.66	-	0.00
Advertising ID	User-resettable, unique, anonymous ID for advertising, provided by Google Play services(ADID)	NONE	✓	62.93	-	0.00
Android ID	A random 64-bit number that is generated when the device boot's for the first time	NONE	✓	57.76	-	0.00
CPU	The device's CPU architecture	NONE	✓	35.34	✓	20.69
IMEI	International Mobile Equipment Identity	PHONE	✓	24.14	-	0.00
Timezone	User's timezone	NONE	✓	24.14	✓	9.48
City	The city name of the device's location	NONE	✓	22.41	✓	25.86
Device SN	A unique hardware serial number of the device	NONE	✓	14.66	-	0.00
MAC Address	The MAC address from the device WiFi NIC	WIFI STATE	✓	14.66	-	0.00
AP SSID	Access Point's MAC Address or SSID	WIFI STATE	✓	9.48	-	0.00
IMSI	International Mobile Subscriber Identity	PHONE	✓	9.48	-	0.00
Local IP	Device's local(LAN) IP address	WIFI STATE	✓	6.03	✓	0.00
Fingerprint	A string that uniquely identifies the device's build	NONE	✓	5.17	-	0.00
Memory Info	The device's (total/free) memory information	NONE	✓	5.17	-	0.00
Phone Number	The SIM number	PHONE	✓	5.17	-	0.00
WiFi Scan	Scan for nearby routers and devices and grab their MAC Address and SSID	WIFI STATE and LOCATION	✓	4.31	-	0.00
Contacts	The device's contacts list	CONTACTS	✓	3.45	-	0.00
Installed Apps	The device installed apps	NONE	✓	3.45	-	0.00
ICCID	The SIM card Serial Number	PHONE	✓	2.59	-	0.00
Kernel Version	The OS kernel version	NONE	✓	2.59	-	0.00
Baseband	The radio driver in which the info related to the telephone communications of the device is stored	NONE	✓	1.72	-	0.00
Bootloader	The system bootloader version number	NONE	✓	0.86	-	0.00
GSF	Google Services Framework Key ID, paired with the user's account	GSERVICES	✓	0.86	-	0.00
Stored SSIDs	The SSID/MAC of all connected Access Point's	WIFI STATE	✓	0.86	-	0.00
Logcat	The log of system messages, including stack traces	NONE	✓	0.86	-	0.00
SMS	The device's sent/received SMS	SMS	✓	0.00	-	0.00

Table 3.1: Description of each ID we investigate, their required permissions (Normal permissions are marked with blue, when Runtime/Dangerous permissions with red), their leakability by apps or browsers and the percentage of services found retrieving the corresponding value of each ID.

Leaked IDs	**Adblock**	**APUS**	**Boat**	**Chrome**	**CM Browser**	**Dolphin**	**DU Browser**	**Firefox**	**Maxthon**	**Next Browser**	**Opera**	**Opera Mini**	**UC Browser**	**Yandex**
Coordinates	-	✓	✓	-	✓	✓	-	-	✓	✓	✓	✓	-	✓
City	-	-	✓	-	✓	✓	-	-	✓	-	-	-	-	✓
Timezone	-	-	-	-	✓	-	-	-	-	-	-	-	-	✓
Android ID	-	-	-	-	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
Advertising ID	✓	-	✓	-	✓	-	-	✓	-	-	✓	✓	✓	✓
IMEI	-	-	-	-	-	-	✓	-	✓	-	-	-	✓	-
IMSI	-	-	-	-	-	-	✓	-	-	-	-	-	✓	-
ICCID	-	-	-	-	-	-	-	-	✓	-	-	-	-	-
MAC Address	-	-	-	-	✓	-	-	-	✓	-	-	-	✓	✓
Device SN.	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-
OS Version	✓	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
Build Version	✓	✓	✓	✓	✓	✓	✓	-	✓	✓	✓	✓	✓	✓
Carrier	-	-	✓	-	✓	✓	✓	-	-	-	✓	✓	✓	-
Manufacturer	✓	-	✓	-	✓	✓	✓	✓	-	-	✓	✓	-	✓
Model	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CPU Arch	-	-	-	-	✓	-	-	-	-	-	✓	✓	-	✓
Screen Resol.	✓	-	✓	-	✓	✓	✓	✓	-	-	✓	✓	-	✓
AP SSID	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
WiFi Scan	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
TOTAL	5	4	9	3	13	9	9	4	9	5	10	11	10	14

Table 3.2: Identifiers leaked by the most popular browser apps when visiting google.com.

Chapter 4
Common User Identification

In the online era, where behavioral advertising fuels the Internet world as we know it, user privacy has become a commodity that is being bought and sold in a complex, and often ad-hoc, data ecosystem [83, 100, 199]. Users' personal data collected by IT companies constitute a valuable asset, whose quality and quantity significantly affect each company's overall market value [211]. As a consequence, it is of no doubt that in order to gain advantage over their competitors, web companies such as advertisers and tracking companies participate in a user data collecting spree, aiming to retrieve and process as much information as possible (sometimes in both the online and the offline world [118, 120]) and reconstruct user profiles. These detailed profiles contain personal data[1] such as interests, preferences, personal identifying information (PII), geolocations, etc. [32, 235], and could be sold to third parties for advertising or other purposes beyond the control of the user [135, 189, 197]. Here, "sale" may be access to an API for ad-audience planning (e.g., Facebook), or even actual sharing of data to third parties [148].

Highlighting the importance of this data collection, web companies have invested a lot in elaborate user tracking mechanisms. The most traditional one includes the use of cookies: they have been commonly used in the World Wide Web to save and maintain some kind of state on the web client's side. This state has been used as an identifier to authenticate users across different sessions and domains. However, over the past few years, cookies have also been massively used to *track* users as they surf the web. Initially, *first-party* cookies were used to track users when they repeatedly visited the same site, and later, *third-party* cookies, were invented to track users when they move from one website to another. The same-origin policy (SOP) invented a few years later [247] placed constraints on what each party can access locally on the user's device. In effect, this policy forbids the cross-domain tracking of users, and consequently restricts the potential amount of information trackers can collect about a user and share with other third party platforms. Therefore, the SOP had the potential to slow down the increasingly aggressive tracking, and prevent web entities from tracking all users' activities on the web.

[1] `https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-personal-data_en`

To overcome this restriction, and also create unified identifiers for each user, the ad-industry invented the Cookie Synchronization (CSync) process: a mechanism that can practically "circumvent" the same-origin policy [89], and allow web entities to share (synchronize) cookies, and match the different IDs they assign for the same user while they browse the web. Sadly, recent results show that most of the third parties are involved in CSync: 157 of top 200 websites (i.e. 78%) have third parties which synchronize cookies with at least one other party, and they can reconstruct 62-73% of a user's browsing history [67]. Furthermore, 95% of pages visited contain 3rd party requests to potential trackers and 78% attempt to transfer unsafe data [253]. Finally, a mechanism for respawing cookies has been identified, with consequences on the reconstruction of users' browsing history, even if they delete their cookies [2].

Interestingly, past literature on CSync has focused so far on desktop devices and browsing, and there was no focus on mobile. Mobile devices, with their ubiquitous nature and the plurality of sensors embedded in them, have essentially allowed personalization of the web experience for each user-device owner, at the expense of user privacy. Considering the large population of mobile devices around the world, and the ever increasing usage of mobile devices in everyday activities, mobile traffic recently has taken over the majority of web traffic. In fact, in 2018, 52.2% of all website traffic worldwide was generated through mobile phones (up from 50.3% in the previous year) [219]. This increasing trend appears to be consistent and not expected to stop in the near future. Furthermore, mobile devices have become important sources of user personal information, leading to increased risk for privacy and anonymity loss for the mobile users - data owners, and to critical questions such as: *Which are the basic characteristics of CSync, and the mechanics used in the mobile ecosystem? Which entities are involved? How does CSync impact the user's privacy and anonymity on the mobile web?*

In this chapter, we aim to answer these and related questions on user mobile privacy and anonymity with respect to CSync, by studying this mechanism in depth, its use and growth through time, dominant entities, along with its side-effects on user privacy. More specifically, we design and implement CONRAD: (COokie syNchRonizAtion Detector) a holistic, hybrid mechanism to detect at real time CSync events and the privacy loss on the user side even when synced IDs are concealed or obfuscated by participating companies aiming to reduce identifiability from traditional, heuristic-based detection algorithms. In such cases when non-ID related features are used, our approach achieves high accuracy (84%-90%) and AUC (0.89 - 0.97). We perform the first of its kind, large-scale, longitudinal study of Cookie Synchronization on mobile users in the wild. This study is done on a large set of 850 volunteering mobile users, with a passive data collection lasting over an entire year. This means that the data collected are not crawled, like in past studies, and therefore do not capture a distorted or biased picture of CSync on the web. Instead, these data provide a rare glimpse of CSync on the mobile space, and an opportunity to study CSync and its impact on real mobile users' privacy and anonymity. Using the proposed detection mechanism, we conduct an in-depth privacy analysis of Cookie Synchronization. Our results show that

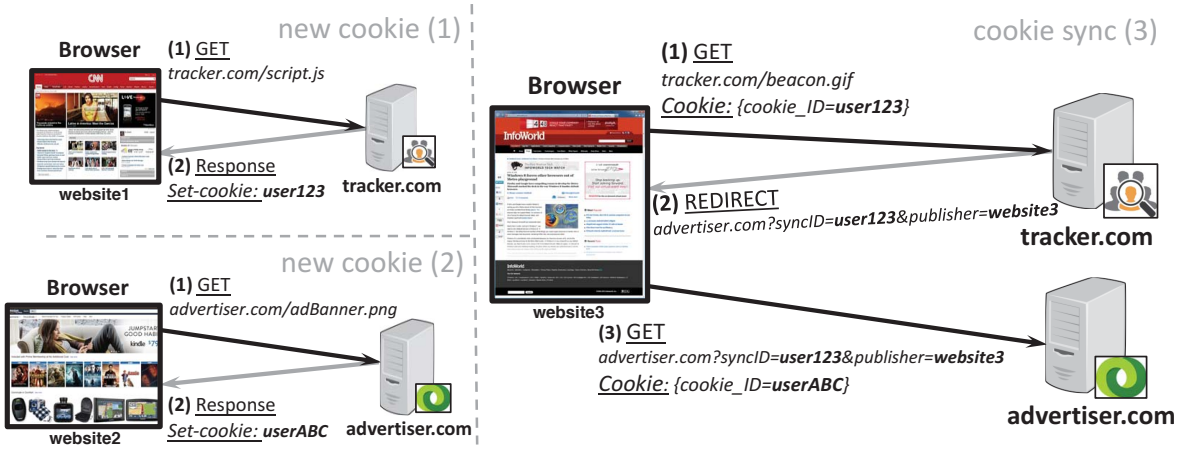


Figure 4.1: Example of two entities (advertiser.com and tracker.com) synchronizing their cookieIDs. Interestingly, and without having any code in website3, advertiser.com learns that: (i) cookieIDs userABC==user123 and (ii) userABC has just visited the particular website. Finally, both entities can conduct server-to-server user data merges.

97% of regular web users are exposed to CSync. In addition, the average user receives ∼1 synchronization per 68 GET requests, and the median userID gets leaked, on average, to 3.5 different ad-entities. Furthermore, CSync increases the number of entities that track the user by a factor of 6.7. We also detect CSync involved in different scenarios such as breaking-off an SSL session and exposing the CSync action in cleartext, and bundling IDs together in the same CSync event. Finally, we find at least eight different types of PII leakage triggered by CSync.

4.1 Cookie Synchronization

Cookies is an age-old technique, useful for maintaining a kind of state on the user's side and also works as an identifier to authenticate the user across different sessions and domains. Cookies, however, are domain-specific, which means those created by one third-party entity cannot be read by anyone else (see **same-origin policy** [247]: one of the foundational concepts in web security). This policy, by forbidding the cross-domain tracking of users, restricts the potential amount of information advertisers can collect about a user. In this section, we discuss the basic concepts of the Cookie Synchronization mechanism, which enables third parties to share information they acquired about a user bypassing the same-origin policy, as well as a heuristic-based methodology to detect CSync events.

4.1.1 How does Cookie Synchronization work?

Figure 4.1 presents a simple example to understand in practice what is Cookie Synchronization and how it works. Lets assume a user browsing several domains like `website1.com` and `website2.com`, in which there are third parties like `tracker.com` and `advertiser.com`, respectively. As a consequence, these two third parties have the chance to set their own cookies on the user's browser, in order to re-identify the user in the future. Hence, tracker.com knows the user with the ID `user123`, and advertiser.com knows the same user with the ID `userABC`. Now lets assume that the user lands on a website (say `website3.com`), which includes some JavaScript code from tracker.com but not from advertiser.com. Thus, advertiser.com **does not (and cannot) know which users visit website3.com**. However, as soon as the code of tracker.com is called, a GET request is issued by the browser to tracker.com (step 1), and it responds back with a REDIRECT request (step 2), instructing the user's browser to issue another GET request to advertiser.com this time, using a specifically crafted URL (step 3):

GET *advertiser.com?syncID=**user123**&publisher=**website3.com***

Cookie: {cookie_ID=**userABC**}

When advertiser.com receives the above request along with the cookie ID `userABC`, it finds out that `userABC` visited website3.com. To make matters worse, advertiser.com also learns that the user whom tracker.com knows as `user123`, and the user `userABC` is basically one and the same user. Effectively, CSync enabled advertiser.com to collaborate with tracker.com, in order to: (i) find out which users visit website3.com, and (ii) synchronize (i.e., join) two different identities (cookies) of the same user on the web.

4.1.2 Privacy implications for users

There are several privacy implications for the online users who access websites planted with such sophisticated tracking technologies. Using CSync, in practice, advertiser.com learns that: (i) what it knew as `userABC` is also `user123`, and (ii) this user has just visited website website3.com. This enables advertiser.com to track a user to a much larger number of websites than was initially thought. Indeed, by collaborating with several trackers, advertiser.com is able to track users across a wide spectrum of websites, even if those websites do not have any collaboration with advertiser.com.

To make matters worse, the ill effects of CSync may reach way back in the past - even up to the time before the invention of CSync. Assume, for example, that someone manages to get access to all data collected by tracker.com, and all the data collected by advertiser.com (e.g., by acquisition [135, 189], merging or hacking of companies [124]). In the absence of CSync, in those two datasets, our user has two different names: `user123` and `userABC`. However, after one single CSync, those two different names can be joined into a single user profile, effectively merging all data in the two datasets. Nowadays, such cases of *server-to-server*

user data merges are taking place at a massive scale [67], with the different web entities conducting mutual agreements for data exchanges or purchases, in order to enrich the quality and quantity of their user data warehouses [36, 135].

As if these threats to user privacy were not enough, CSync can rob users of the right to erase their cookies. Indeed, when coupled with other tracking technologies (i.e. evercookie [206], or user fingerprinting [60]), CSync may re-identify web users even after they delete their cookies. Specifically, when a user erases her browser state and restarts browsing, trackers usually place and sync a new set of userIDs, and eventually reconstruct a new browsing history. But if one of them manages to respawn [2] its cookie (e.g. through evercookie [206]), then through CSync, all of them can link the user's browsing histories from before and after her state erasure. Consequently: (i) users are not able to abolish their assigned userIDs even after carefully erasing their set cookies, and (ii) trackers are enabled to link user's history across state resets.

4.1.3 Cookie Synchronization and Personalized Advertising

Digital advertising has moved towards a more personalized model, where ad-slots are purchased programmatically (e.g., Real Time Bidding (RTB) based auctions) in auctions, based on how well the profile of the visitor matches the advertised product. Consequently, advertisers need to obtain user data (e.g., interests, behavioral patterns) to use as input in their sophisticated decision engines. The core component for this data purchasing/sharing includes Cookie Synchronization [81]. It allows different entities (e.g., trackers and advertisers) to perform common user identification and by participating in data markets enrich their knowledge base with user information from several data sources.

4.2 Cookie Synchronization Detection

In this chapter, we design CONRAD: a holistic methodology to detect Cookie Synchronization events in real time, on the user side. CONRAD, monitors the HTTP(S) traffic of the user on the browser level and detects userIDs when shared from one domain to the other. To achieve that, it uses a (i) Heuristics-based (stateful) detection mechanism (Section 4.2.1), where the IDs from cookies are tainted and alert is raised when they are exfiltrated to a domain other than the owner of the cookie. However, as also presented in past studies [11], more and more companies include encryption (or cryptographic hashing) in the Cookie Synchronization-related APIs, thus concealing the synced IDs. To deal with these cases, CONRAD uses a (ii) ML-based (stateless) detection mechanism (Section 4.2.2) capable of classifying with high accuracy such possible concealed Cookie Synchronizations, without relying on any previously stored cookie IDs, but only using characteristics from the connections themselves.

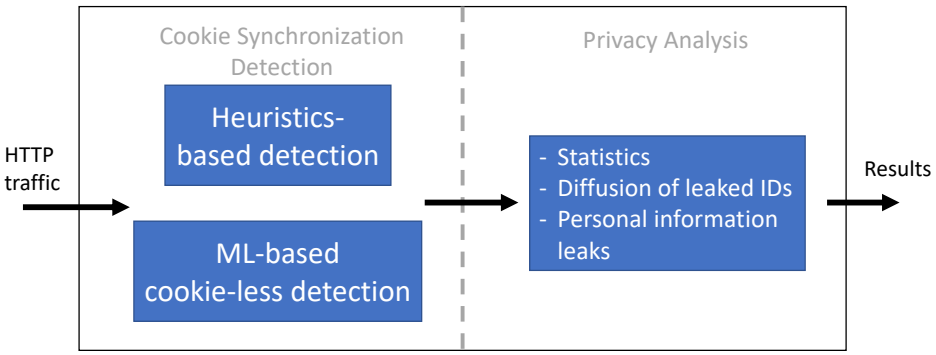


Figure 4.2: High-level overview of the internal components of CONRAD.

For each Cookie Synchronization detection method, CONRAD extracts its information flow: the chain of entities that share the synced ID, the domain that triggered the Cookie Synchronization event, the entities that (without having access to the website) used this sync request to set and sync their own userIDs (see Section 4.4.1). This way, our tool is able to measure the diffusion of anonymity loss for the given user by analyzing what number of their overall userIDs budget got synced, and to how many third party domains. In addition, by using a simple pattern matching technique, CONRAD extracts possible personal information leaks tailored with the synced ID (see Section4.4.4). Figure 4.2, provides a high level overview of CONRAD's internal components.

Although there are several existing techniques for detecting ID-sharing events even when cookies are encrypted, accurate CSync detection in real time is a hard task. The main advantages of our approach, contrary to existing detection mechanisms [2, 11, 67] are as follows: (i) It offers the ability to detect synchronizations when the userID is embedded not only in the URL's parameter, but also in its path (either in case of request/response URL or Location URL of the referrer). (ii) By filtering-out domains of the same provider, our approach can discriminate between intentional CSync and unequivocally legitimate cases of internal ID sharing, thus avoiding false positives. (iii) It is capable of detecting Cookie Synchronization *at real time*, even when shared IDs are encrypted.

4.2.1 Heuristics-based detection

Technically, as we see in Table 4.1, CSync is nothing more than a request from the user's browser to a third party domain carrying (at least one) parameter that constitutes a unique ID set by the calling domain. However, what CSync typically enables is a multiple, back-to-back operation with several third party domains getting updated with one particular ID. This multiple synchronization happens by utilizing URLs of HTTP requests (i.e., the Location HTTP header), in which the cookie ID (i.e., the userID) of the triggering entity is embedded. The userID may be embedded in the: (i) parameters of the URL, (ii) URL path, or (iii)

URLs of Cookie Synchronization HTTP Requests
1. a.atemda.com/id/csync?s=**L2zaWQvMS9lkLzMxOUwOTUw**
2. bidtheater.com/UserMatch.ashx?bidderid=23& bidderuid=**L2zaWQvMS9lkLzMxOUwOTUw**& expiration=1426598931
3. d.turn.com/r/id/**L2zaWQvMS9lkLzMxOUwOTUw**/mpid/

Table 4.1: Examples of userIDs getting synchronized between different entities.

referrer field. In some cases, detection may be straightforward: one can simple look for specific parameter names (e.g., syncid, user_id, uuid). However, different companies use different APIs and parameter naming; relying only on string matching for Cookie Synchronization detection will lead to a large number of false negatives in case of newcomer syncing domains. To remedy this, we design a stateful heuristics-based detection algorithm, which relies on the previously set cookies to taint userIDs that may get synced with entities different than the cookie setter. In particular, our Cookie Synchronization detection methodology includes the following steps, which are also illustrated in Figure 4.3:

1. First of all, we extract all cookies set on the user's browser. To accomplish that, we parse all HTTP requests in our dataset and extract all *Set-Cookie* requests.
   (a) We filter out all session cookies. These are cookies without expiration date, which get deleted right after the end of a session.
   (b) We parse the cookie value strings using common delimiters (i.e. ":", "&"). By extracting potentially identifying strings (cookie IDs), we create a list with the cookie IDs that could uniquely identify the user in the future.

2. Second, we exclude any possible ID-sharing URL from the HTTP trace:
   (a) We identify ID-looking strings carried either (i) as parameters in the URL, (ii) in URL path, or (iii) in the referrer URL. As ID-looking strings, we define strings with specific length (> 10 characters) – false positives do not matter at this point.
   (b) Upon detection, each of such ID-looking strings is stored in a hashtable along with the URL's domain (receiver of the ID).
   (c) In case we have already seen the same ID in the past, we consider it as a shared ID and the requests carrying it as *ID-sharing requests*.
   (d) To check if the above ID-sharing requests regard different entities, we use several external sources (DNS whois, blacklists etc.) to filter-out cases where the IDs are shared among domains owned by the same provider (e.g., amazon.com and

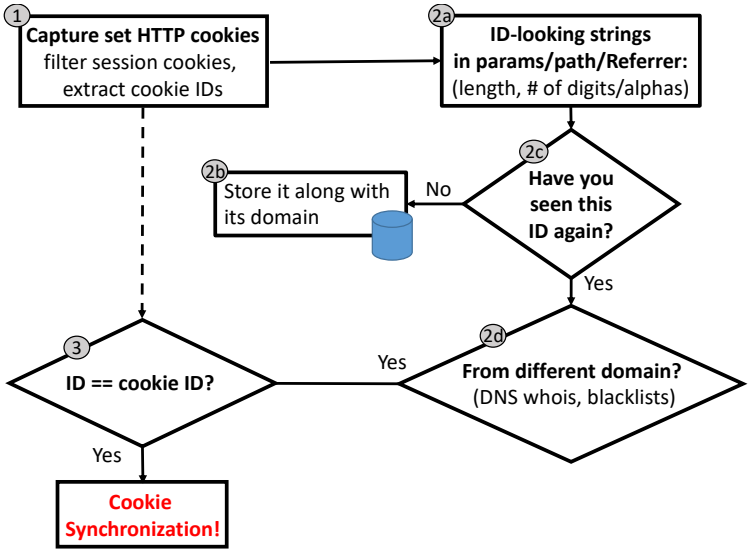


Figure 4.3: Heuristics-based Cookie Synchronization detection mechanism.

amazonaws.com) [152]. This way, our approach can discriminate between intentional ID leaking and legitimate cases of internal ID-sharing, thus avoiding false positives.

3. Finally, in order to verify if the detected shared ID is a userID, able to uniquely identify a user, we search this ID in the list of the cookie IDs that we extracted in the first step. If there is a match, then we consider this request as Cookie Synchronization.

4.2.2 Cookie-less detection

It is apparent that in order for the above methodology to be a viable CSync detection method, cookie IDs need to be shared in plaintext. However, major web companies [89] have started encrypting the cookie ID in an attempt to protect the actual cookie from being revealed to unwanted parties. Such entities can be third parties snooping on the user traffic (plugins or even ISPs), as well as syncing partners.

While the former case is obvious why companies would want to block such snooping, the latter case may not be clear why and, thus, we discuss this case further. In particular, under the traditional, plaintext case of cookie ID syncing, the same source company can sync independently with multiple third parties for the same user cookie ID. Thus, no-one forbids these other third parties from syncing their IDs with each other, and find out that they have information about the same user, something that goes beyond what the source company intended to do (top example of Table 4.2). With hashing or encryption of the cookie ID, these third parties are unable to do this syncing (bottom example of Table 4.2).

ID syncs beyond the 1-1 of source entity (plaintext):
Domain syncs ID with Tracker1 ID1; → ID1=ID;
Domain syncs ID with Tracker2 ID2; → ID2=ID;
Tracker1 syncs ID1 with Tracker2 ID2; → ID2=ID1=ID;
ID syncs beyond the 1-1 of source entity (encrypted):
Domain syncs h(ID,A) with Tracker1 ID1; → ID1=h(ID,A);
Domain syncs h(ID,B) with Tracker2 ID2; → ID2=h(ID,B);
Tracker1 syncs ID1 with Tracker2 ID2; → h(ID,A)!=h(ID,B), i.e., ID1!=ID2;

Table 4.2: Examples of Cookie Synchronization between third parties with plaintext and encrypted cookie IDs.

As a consequence of this encryption, CSync events can proceed undetected, if the previously used detection method is employed.

In order to address this scenario, in this chapter, we propose a novel method for identifying CSync which is oblivious to the IDs shared. This mechanism is able to identify with high accuracy CSync events in web traffic, even when the leaked IDs are protected and cannot be matched. To build this mechanism, we employ machine learning methods, which we train on the ground truth datasets created with the previous, heuristic-based technique. In particular, we analyze various features extracted from the web traffic due to CSync, and train a machine learning classifier to automatically classify a new HTTP connection as being a CSync event or not. Here, we make the assumption that the various features used to characterize, and eventually detect, CSync with plaintext IDs, are equally used, and have the same distributions and variability as in the CSync with encrypted IDs. We believe this is a reasonable assumption, since the companies employing encrypted IDs are not expected to change the rest of their mechanism which delivers these IDs and triggers CSync with their partners; these companies only want to obfuscate the IDs to avoid further, and unwanted, CSync.

For training the classifier, we extract various relevant features from the network traces. As ground truth, we use confirmed CSync events which were detected with the heuristics-based method described earlier. Beyond these confirmed events, there are *id-sharing* events which were first selected by the method as potential CSync events, but eventually were rejected as *non-CSync*, as they did not match cookie IDs already seen by the method (step 1 in Figure 3).

The features available for these network events can be several; we constrain the machine learning algorithm to use only features available at run time, and during the user's browsing to various websites:

- EntityName: {domain of recipient company}
- TypeOfEntity: {Content, Social, Advertising, Analytics, Other}

Description	#
Total mobile users	850
HTTP requests captured	179M
Unique Cookies (C)	8978275
ID sharing requests	412805
Unique shared IDs (S)	68215
Unique userIDs got synced ($C \cap S$)	22329
Cookie Synchronization requests	263635

Table 4.3: Summary of contents in our dataset.

- ParamName: {aid, u, guidm, subuid, tuid, etc.}
- WhereFound: {parameter in URL, parameter in Referrer, in the URL path}
- StatusCode: {200, 201, 202, 204, etc.}
- Browser: {Firefox, Chrome, Internet Explorer, etc.}
- NoOfParams: {0, 1, 2, ..., etc.}

Various machine learning algorithms can be applied in this case: Random Forest, Support Vector Machines, Naive Bayes, and even more advanced methods such as Neural Networks. However, a balance must be found between training the given algorithm to reach a good accuracy, the computation cost for this training, as well as the capability of the algorithm to be used at real time on the user device.

This method aims to address two possible scenarios that can arise while IDs are being shared: First, we consider the realistic scenario that an already identified set of id-sharings are candidate CSync events (as found by the heuristics-based method), but cannot be validated as CSyncs because of the cookie ID being encrypted or unavailable, and therefore not matching the repository of IDs. Second, we consider the scenario where various HTTP connections are ingested by the method, and it needs to decide at run-time which are CSync events and which are not. This case is a more generalized version of the previous scenario, and attempts to detect CSync events, as an alternative method to the heuristic-based approach. In either of the two cases, we follow a generally accepted methodology that separates training such a machine learning model, from applying it at real-time. The training can be performed offline, on an existing dataset (e.g., the one we collected, or from anonymous user network traffic donations), and the model trained can be distributed accordingly with the tool at hand. Then, the tool can apply the classifier on each network connection under question, for real-time classification.

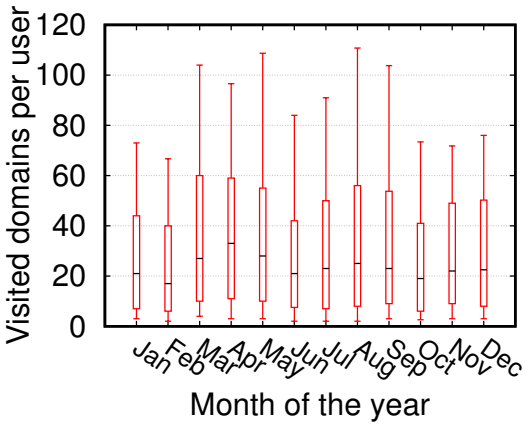


Figure 4.4:
Distribution of number of unique domains visited per user per month. The median user in our dataset visits 20 - 30 different domains per month.

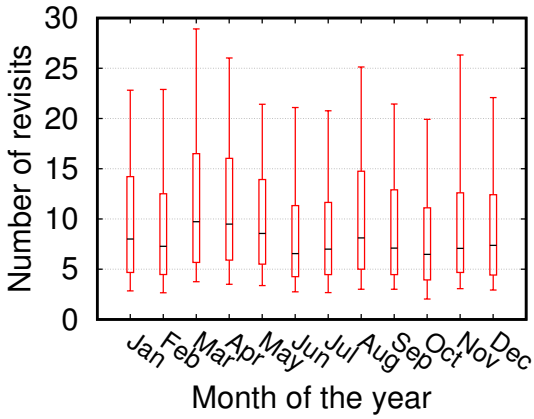


Figure 4.5:
Distribution of number of times a user revisits the same domain per month. The median user revisits the same domain around 7-10 times per month through their mobile browser, while the 90th percentile may revisit the same domains up to 25-29 times.

4.3 Dataset

In this section, we describe the data collection process and our year long dataset. In order to collect data from real users, we set a group of proxies fronted by a load balancer, and gathered 850 volunteering users residing in the same country. These users agreed to strip their browsers from any previous state (i.e., cookies, cache, webStorage) and have their devices to continuously redirect their network traffic through our proxies for 12 consecutive months (2015-2016). They signed a consent form allowing us to collect and analyze their data during this period, and publish any anonymized results. They were well-aware of the purposes of the data collection, and were compensated with free data plan, as long as they were using the proxies. Before the analysis was performed, all data were anonymized and never shared with any other entities.

Given the long duration of the experiment, and in order not to jeopardize the confidentiality of the users' secure sessions, we capture only their HTTP traffic. On the server-side and based on the user agent of each request, we filtered out any possible app-related traffic. Overall, we collected a dataset containing a total of 179M HTTP[2] requests, spanning an entire year. Table 4.3 summarizes the contents and CSync findings in our dataset.

[2]For confidentiality purposes, users agreed to provide only their HTTP traffic. However, our proposed methodology works in the same way for HTTPS traffic as well.

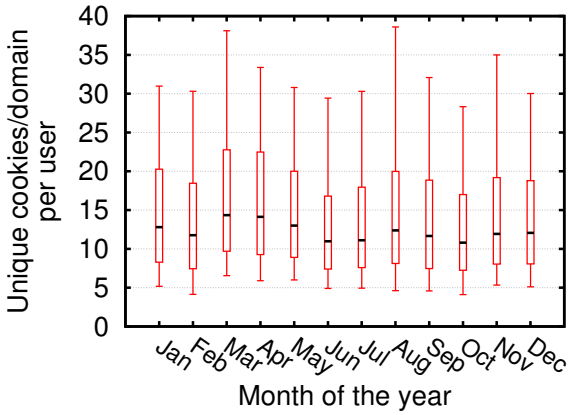


Figure 4.6:
Number of (first and third-party) cookies per domain per user. We see that the median user receives around 10 cookies per visited website.

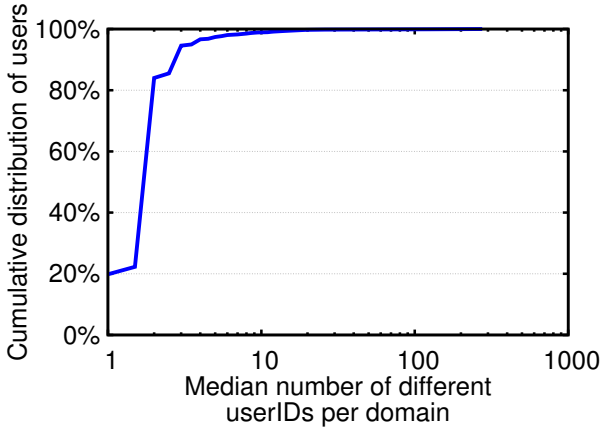


Figure 4.7:
Number of unique userIDs set per domain across the year. 80% of the users are known to a single domain with only 2 aliases, on average, throughout the entire year.

4.3.1 Users

To analyze our dataset, we create a simple HTTP weblog parser. As noted earlier, this dataset consists of web browser traffic from the mobile devices of 850 users. After separating the flows of each one of them, we produced their timelines, and in Figure 4.4, we present the number of different domains each user visits per month. As we see from the distribution (Percentiles: 10th, 25th, 50th, 75th, 90th), the median user in our dataset visits 20-30 different domains depending on the month (we observe a seasonal phenomenon with increases during spring break and summer holidays).

Similarly, in Figure 4.5, we present the number of times each of these domains gets revisited by the median user in our dataset. In every revisit, there is a new HTTP request that asks the user's browser if there is a previously set cookie. If this *Get-Cookie* request regards a previously set cookie, this means that the domain already knows the user and we consider it as a revisit. We observe that the median user revisits around 7 times the same domain from their mobile browser. Again, we observe the seasonal phenomenon as earlier, when users tend to have more time to spend browsing the web. Also, the 75th percentile of the users may revisit the same website more than 15 times (March, August).

4.3.2 Cookies

In order to have a good view of the cookie activity of users, we extract all (first and third party) cookies set in the users browsers across the year, and in Figure 4.6, we plot the

	Initiator	**Portion**
(i)	Publisher syncs its userID	2.692%
(ii)	Embedded third party triggers syncing of its own set userID	49.668%
(iii)	Third party uses sync request to share its own set userID	45.697%
(iv)	Third party uses sync request to share with other domains the publisher's set userID	0.2658%

Table 4.4: Breakdown of the Cookie Synchronization triggering factors.

distribution (percentiles: 10th, 25th, 50th, 75th, 90th) of the number of cookies per visited website. The median user receives a fairly constant number of cookies per month: 12.25 cookies per visited website on average.

Next, we extract the unique identifiers set in these cookies (cookie IDs). A cookie ID, in essence, constitutes a unique string of characters that websites and servers associate with the browser and, thus, the user in which the cookie is stored. Thereby, in the rest of this section, we consider a cookie ID as a unique user identifier called userID. As it can be seen in Table 4.3, in our dataset, there are almost 9 million such unique IDs.

In Figure 4.7, we plot the distribution of the number of unique userIDs assigned to the users per domain. The vast majority of the users (80%) receive, on average, only 2.2 userIDs per domain, across the year. This means that users tend not to erase their cookies frequently, thus, allowing web domains to accurately identify them through time and during the users' browsing. Only 1.13% of users erase their cookies (either manually or by browsing with Private/Incognito Browsing), receiving more than 9.5 different userIDs per domain, on average. This means that it is very rare for a domain to meet a previously known user with a different alias.

4.4 Privacy Analysis

In this section, we present the results of CONRAD when applied in the dataset we described in the previous section. As expected, there are several IDs passed from one entity to another. We detect 68215 such unique shared IDs. From these IDs, 22329 were actually cookie IDs from previously set cookies that were synced among different domains. In total, these cookie IDs were found in 263635 synchronization events (see Table 4.3 for a summary).

From the total CSyncs detected in our dataset, userIDs were found in 91.996% of the cases inside the URL parameters, 3.705% in the Referrer URL, and in 3.771% of the cases in

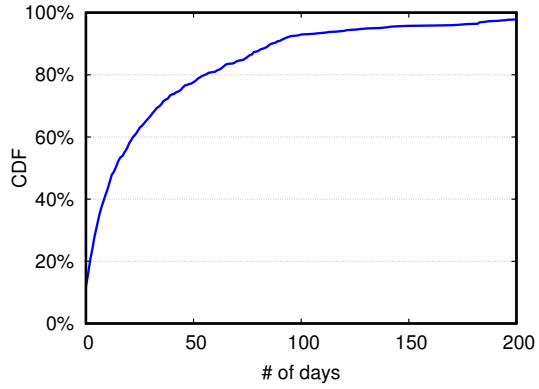


Figure 4.8:
Distribution of the time it takes for the first CSync to appear per user. Around 20% of the users get their first userID synced in 1 day or less, when 38% of users get synced within their first week of browsing.

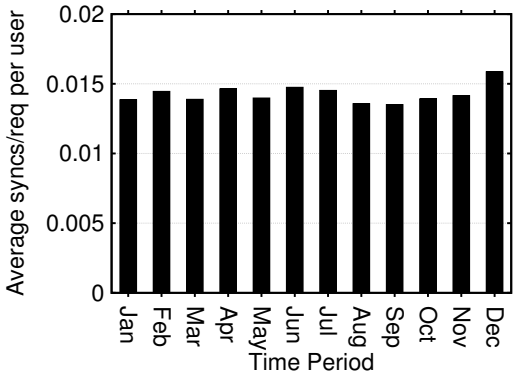


Figure 4.9:
CSyncs per HTTP request for the average user through the year, normalized with their total number of requests. The average user receives 1 synchronization every 68 HTTP GET requests.

the URI path. Therefore, our enhanced detection algorithm was able to detect 3.771% more cases of CSyncs than existing detection methods [2, 176, 183].

4.4.1 Initiation of Cookie Synchronization

First, we correlate the cookie domain (the setter), the synchronizing HTTP request's Referrer field, and the publisher that the user visited, in order to extract the entity that triggered the CSync on the user's browser. As seen in Table 4.4, there are 4 distinct cases: (i) the CSync was initiated by the publisher who syncs the userID he assigned for the user: we find 2.692% of these cases in our dataset, (ii) the synchronization was initiated from a publisher's iframe, by the guest third party which syncs its own userID (49.668%), (iii) a third party which participated in a previous synchronization (case (i) or (ii) above) and uses the sync request to share its own userID (45.697%). Lastly, there is a rare case (iv), where a third party which participated in a previous synchronization of the publisher's userID (case (i)) initiates a new round of syncs while it continues to share the publisher's userID (0.2658%). It is apparent, that in case (iv), the initiating third party shares with its third party affiliates, a userID assigned by an entity (the publisher) beyond its control, and possibly awareness.[3]

[3]We reported all such cases and we notified the respected publishers.

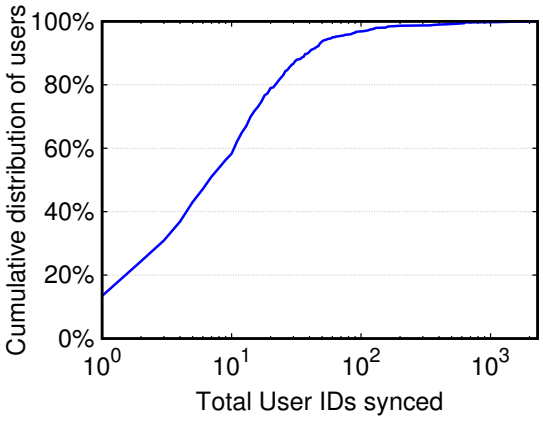


Figure 4.10:
Distribution of the synced userIDs per user. The median user has 7 userIDs synced, and 3% of users has up to 100 userIDs synced.

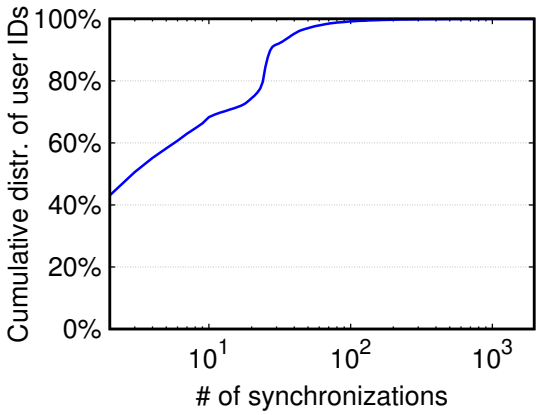


Figure 4.11:
Distribution of synchronizations per userID. The median userID gets synced with 3.5 different entities.

4.4.2 How are users exposed to CSync?

CSync impacts users' privacy by leaking assigned userIDs, and sharing them with third parties. In our dataset, we see that for users with regular activity on the web (> 10 HTTP requests per day), **97% were exposed to CSync at least once**. This means that CSync constitutes a phenomenon affecting the totality of online users.

Next, we study how long it takes for the first synchronization to happen, or in effect, how quickly a user gets exposed to CSync after she starts browsing. Recall that as mentioned in Section 4.3, all participating users, during bootstrapping phase, had all state from their browsers erased. This means that our proxy was able to capture the very first cookie that was set during the user's monitoring period. Of course, the time depends on the browsing patterns of each user, however, as we see in Figure 4.8, a median user experiences at least one CSync within the first week of browsing. In fact, a significant **20% of users gets their first userIDs synced in 1 day or less**. It is worth noting at this point, that users tend to browse the same top websites again and again (e.g., facebook.com, twitter.com, cnn.com), so the set cookies are already shared and no sync is fired.

Next, we investigate if the synchronizations the users are exposed to, change over time. Hence, we extract CSyncs per user, and normalize with the user's total number of requests. In Figure 4.9, we plot the average synchronizations per HTTP request across the year. As shown, CSync is persistent through the duration of an entire year, with the user being exposed to a steady number of synchronizations across time. Specifically, we see that the average user receives around 1 synchronization per 68 GET requests.

Considering the different userIDs that tracking entities may assign to a user, in Fig-

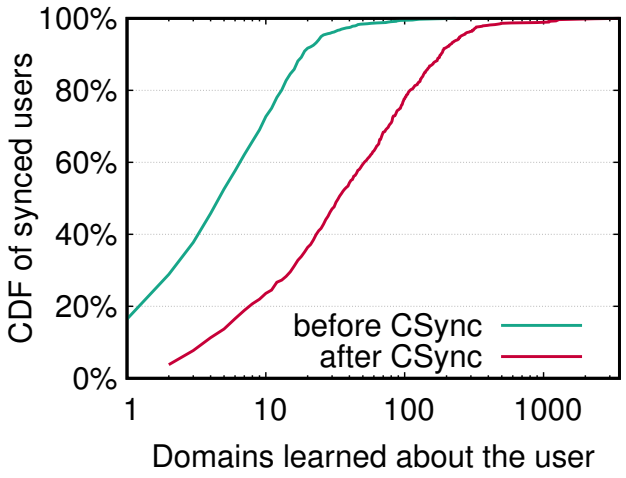


Figure 4.12: Distribution of the number of entities learned at least one userIDs of the user with and without the effect of Cookie Synchronization. As we can see, after syncing the entities that learned about the median user grew by a factor of 6.75.

ure 4.10, we measure the number of unique userIDs that got synced per user. Evidently, **a median user gets up to 6.5 userIDs synced**, and 3% of users has up to 100 userIDs synced. It becomes apparent that the IDs of a user may leak to multiple third party domains through CSync. To measure the userID leak diffusion, in Figure 4.11, we plot the distribution of synchronizing requests per userID. As we see, **the median userID gets leaked, on average, to 3.5 different entities**. There is also a significant 14%, that gets leaked to up to 28 different third parties.

To better understand the effect of CSync on the diffusion of the overall user privacy, we measure for each user the number of entities that learned about them (i.e., that learned at least one of their userID) before and after CSyncs. As we can see from the plotted distribution in Figure 4.12, **the entities that learned about the median user after CSyncs grew by a factor of 6.75**, and for 22% of users, this factor becomes > 10. This means that before the rise of CSync, when the user visited a website, the entities that could track them were only the publisher and the included third parties, but in an independent fashion. However, with the introduction of CSync, the number of entities that can track the user drastically increased (6.75x for the median user), severely decreasing their anonymity on the web.

4.4.3 Buy 1 - Get 4 for free: ID bundling and Universal IDs

In our dataset, we find **63 cases of domains which set on the users' browsers cookies with userIDs previously set by other domains**. For example, we see the popular *baidu.com*, the world's eighth-largest Internet company by revenue, storing a cookie with an

ID Summary stored in cookie by adap.tv
"key=**valueclickinc**:value=708b532c-5128-4b00-a4f2-2b1fac03de81:expiresat=wed apr 01 15:03:42 pdt 2015,key=**mediamathinc**:value=60e05435-9357-4b00-8135-273a46820ef2: expiresat=thu mar 19 01:09:47 pst 2015,key=**turn**:value=268483050575917034 5:expiresat= fri mar 06 16:43:34 pst 2015,key=**rocketfuelinc**:value=639511149771413484:expiresat=sun mar 29 15:43:36 pst 2015"

Table 4.5: Example of an *ID Summary* stored on the user's browser with userIDs and cookie expiration dates set by 4 different domains.

ID $baiduid = \{idA\}$, and more than 5 different domains after this incident setting their own cookie using the same ID $baiduid = \{idA\}$. This byproduct of CSync, enables trackers to use *universal IDs*, thus, bluntly violating the same-origin policy and merging directly (without background matching) the data they own about particular users.

In addition, we find **131 cases of domains storing in cookies the results of their CSyncs, thus composing *ID Summaries***. In these summaries, we see the userIDs that other domains use for the particular user previously obtained by CSyncs. An example of such summaries in JSON is shown in Table 4.5. As one can see, the cookie set by *adap.tv* includes the userIDs and cookie expiration dates of *valueclick.com*, *mediamath.com*, *turn.com* and *rocketfuel.com*. In our dataset, we find at least 3 such companies providing *ID Summaries* to other collaborating entities. This user-side info allows (i) the synchronizing entities to learn more userIDs through a single synchronization request, and (ii) *adap.tv* to re-spawn any deleted or expired cookies of the participating domains at any time, just by launching another CSync.

4.4.4 Sharing sensitive information together with userIDs

As mentioned earlier, the websites a user browses can easily leak through the referrer field during a CSync. Moving beyond this type of basic leak, in our dataset we find several cases of privacy-sensitive information passed to the syncing entity regarding the particular user with the particular synced ID. By deploying a simple string matching script, we find straightforward leaks of personal data of users to third parties:

- 13 synchronizations leaking the user's exact location at city level
- 2 synchronizations leaking the user's registered phone number
- 10 synchronizations leaking the user's gender
- 9 synchronizations leaking the exact user's age
- 3 synchronizations leaking the user's full birth date
- 2 synchronizations leaking the user's first and last name

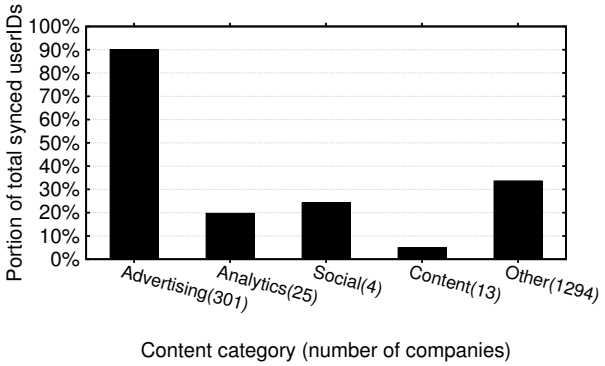


Figure 4.13:
Portion of synced userIDs per content category. As expected, the vast majority regards ad-related companies.

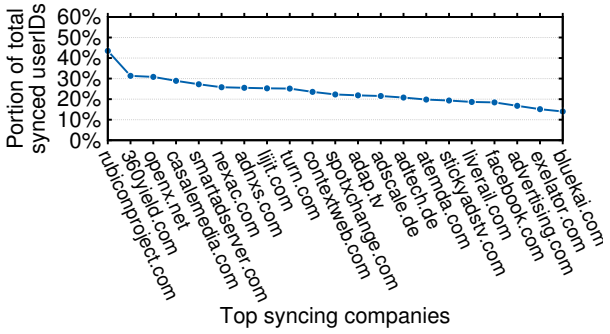


Figure 4.14:
Portion of synced userIDs learned per entity over HTTP: 3 companies learn more than 30% of the total userIDs in our dataset; 14 companies learn more than 20% each.

- 16 synchronizations leaking the user's email address
- 4 synchronizations leaking the user's full login credentials: username and password

Apparently, the above information constitutes not only a severe privacy threat for the user, but can also enable potential impersonation attacks.

4.4.5 Who are the dominant CSync players?

In order to assess the content that CSync entities provide, we extract all domains involved in CSync, and using EasyList, EasyPrivacy [72] and the blacklist of the popular Disconnect browser extension [55] (enriched with our additions after manual inspection), we categorize them according to the content they deliver. This way, we create five categories of entities related with: (i) Advertising, (ii) Analytics, (iii) Social, (iv) Third party content (e.g., CDNs, widgets, etc.), and (v) Other.

As we saw in Figure 4.11, the median userID gets shared with more than one entities. We find that **ad-related entities participate in more than 75% of the overall synchronizations through the year**. Consequently, as we can also observe in Figure 4.13, **ad-related entities have learned as much as 90% of all userIDs that got synced**, with Social and Analytics -related entities following with 24% and 20% respectively. In Figure 4.14, we plot the top 20 companies[4] that learned the biggest portion of the total userIDs through CSyncs in our dataset. Evidently, no more than 3 companies (i.e., rubiconproject.com, 360yield.com and openx.net) learned more than 30% of all userIDs in our

[4]In a dataset with both HTTP and HTTPS traffic, market shares may differ since there are companies operating over SSL only (e.g., DoubleClick)

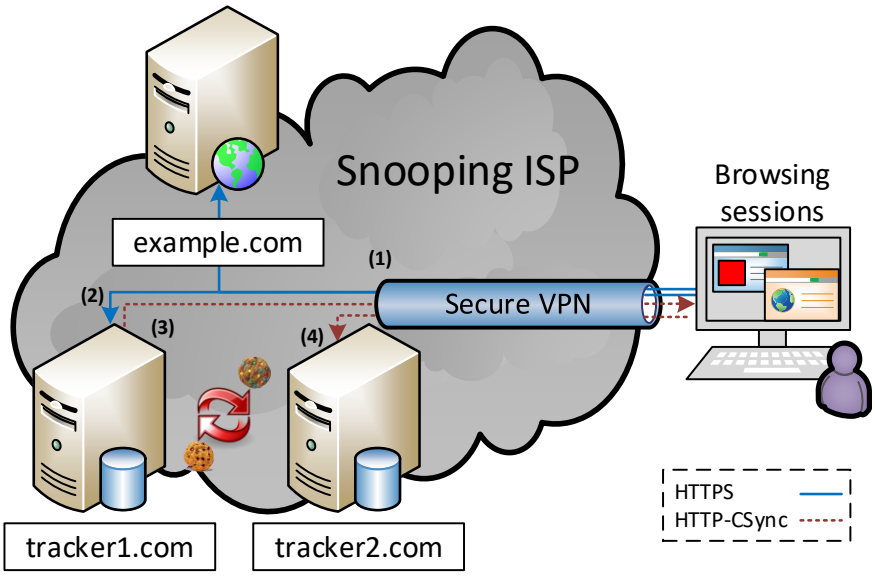


Figure 4.15: High level overview of the TLS session leak. A privacy-aware user (1) visits a webpage (*example.com*) over TLS and VPN. (2) It sends tracking information to tracker1.com, and receives its cookie over TLS. (3,4) It takes only a HTTP-based Cookie Synchronization (among tracker1.com and tracker2.com) in order to spill user unique identifiers and visited website. Then, a snooping ISP can re-identify the user just by monitoring the synced cookies, even if their real IP address is hidden.

dataset each. There is also a significant number (14) of companies that learn more than 20% of userIDs.

4.4.6 Spilling userIDs from Secure Sessions

It is well known, that the basic principle of TLS channels is the confidentiality of the transmitted information, which guarantees the privacy and security of the communication. However, TLS does not guarantee these principles in a plug-and-play manner. There are several guidelines [62] warning about the harmful practice of mixing encrypted and non-encrypted content in TLS sessions. Yet, there is still a significant portion of publishers that fail to maintain adequately the security of their TLS channels [179].

In this study, we demonstrate the massive privacy threat that HTTP-based Cookie Synchronization can impose to the user's TLS sessions. A possible scenario for this leak, as depicted in Figure 4.15, is the following: Let's assume a privacy-aware user, who conducts multiple browsing sessions from their PC, and uses a secure VPN to hide their true IP address. At some point, they visit *https://example.com*: a trustworthy reputable website,

which requires a secure TLS channel establishment (Step 1). The website is ad-supported and thereby it includes ad- and analytic- related third parties that deliver effective personalized advertisements to the visitors. However, as we noted, *https://example.com* is a well reputable website and therefore it includes only TLS supporting third parties.

One of these third parties is *https://tracker1.com*. This third party securely sets a cookie with a user unique identifier (userID) $ID = user123$ on the user's side, in order to re-identify this user in case of a next visit to a site that this third party is similarly included (Step 2). Immediately after, it performs a Cookie Synchronization in order to share the set userID with a remote collaborating domain *http://tracker2.com* (Step 3). Technically, *tracker1.com* redirects an HTTPS request coming from the user's browser to *http://tracker2.com* (Step 4), while it loads the location URL with its userID:

3xx Redirect request Headers
Location: tracker2.com?syncID=**user123**&partner=tracker1
Referrer:{example.com}

Response Headers
Set-Cookie: {cookie_ID=**userABC**}

This Redirect request allows *tracker2.com* to (i) learn the syncedID (i.e. the userID of tracker1.com), and (ii) respond back with a Set-Cookie, thus setting its own cookie in the user's browser (Note: Prior to Cookie Synchronization, tracker2.com was not included in any way in *example.com* and thus it had no reach to the user).

A careful reader may have already detect the severe privacy leaks in this scenario:

1. **Common userID leak:** The HTTP redirection exposes the TLS cookie to both the synced remote party and the snooping ISP. By allowing *tracker2.com* to set its own cookie on the user's browser in plaintext, the ISP is allowed to learn $user123 == userABC$. In this way, the ISP can re-identify the user in the web from now on, by leveraging the requests of *tracker2.com*. Specifically, whenever it captures HTTP requests to *tracker2.com* with cookie $ID = userABC$, it can identify who the user is, even if they use VPN or mixnets to hide their real IP address. Obviously, the more third party entities participate in the Cookie Synchronization the easier it is for the ISP to capture HTTP requests loaded with these synced cookies and, thus, re-identify the user.

2. **Browsing history leak:** There are specific directives (Section 14.36 in RFC 2616 [137]) instructing web entities to either blanking or replacing with inaccurate data the referrer field of HTTP GET requests (referrer hiding), if it refers to a parent HTTPS page. This way, possible exposure of the visited website is fully prevented. Yet, there is an absence of similar directives for the case or HTTP Redirect requests as discussed by the web

Role	**Domain**
- Visited website	https://example.com
- Cookie setter - SetCookie	https://tracker1.com **ade87e60-5336-4dd9-9a2a-763e85516f6d-tuct150ff6a**
- Cookie syncer	http://idsync.tracker2.com/389.gif?partner_uid= **ade87e60-5336-4dd9-9a2a-763e85516f6d-tuct150ff6a**&redirect=1 referrer: **example.com** Get-cookie: {VwkBRjV8XjsYsUgWqL4jEl4=}
- Cookie syncer	http://pixel.tracker3.com/idsync?partner_id= 227&partner_user_id= **ade87e60-5336-4dd9-9a2a-763e85516f6d-tuct150ff6a** referrer: **example.com** Get-cookie: {c57bd1f-8e2ac1hb0242ac-110005}

Table 4.6: Example of leak in our dataset. Our crawler visited over TLS the *https://example.com*. 2 Cookie Synchronizations appear, where *https://tracker1.com* advertiser shares with *tracker2.com* and *tracker3.com* the ID it assigned to the user. By doing that over plain HTTP, the visited website is leaked through the referrer field to a monitoring ISP or other entity.

community. [5] A consequence of this aspect, is that unstripped referrer fields of redirections from HTTPS domains to HTTP domains harm the confidentiality of the secure session, by leaking the TLS-visited website to any monitoring body. Although, at first glance, this leak seems generic and not directly connected with Cookie Synchronization redirections, a careful reader may have observed that in our above scenario, the exposed synced userID of the Cookie Synchronization is the actual identifier that makes this referrer leak persistent. Specifically, Cookie Synchronization amplifies the traditional referrer leak by allowing the snooping ISP to bind the leaked visited website with a persistent ID, which will identify the user even after an IP address or Tor circuit change.

Given that our proxy is monitoring only HTTP traffic, one would expect that no information from secure TLS sessions would be captured. To our surprise, in our dataset we see userIDs originated from TLS sessions getting leaked to unsecured HTTP third parties requests, and as a result, anybody could monitor them[6].

After a deeper inspection, we see that this ID-spilling was caused by CSync redirections appearing in TLS protected websites that synced a userID from a set (over TLS) cookie with non-TLS third parties. An example of such leak is depicted in Table 4.6. In this example, a user visited over TLS the page *https://example.com*. In this website, 2 CSyncs are performed, where *https://tracker1.com* advertiser shares with *http://tracker2.com* and *http:// tracker3.com* the ID it assigned to the user. This way, the two tracking entities sync

[5] https://stackoverflow.com/questions/2158283/will-a-302-redirect-maintain-the-referer-string/5441932#5441932

[6] As soon as we verified this leak, we notified our volunteering users updating the consent they had signed.

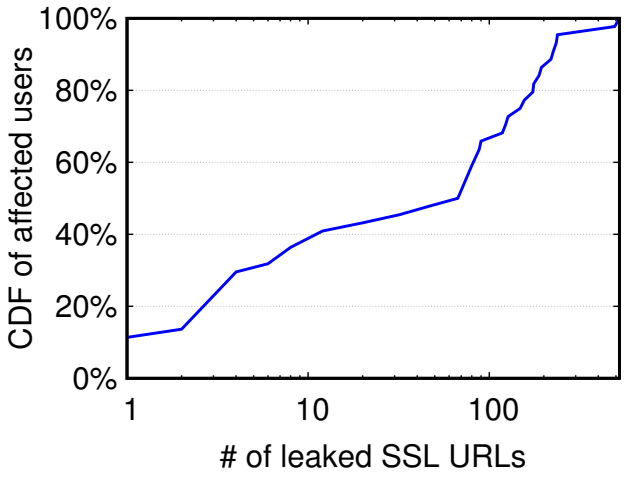


Figure 4.16: Distribution of the leaked TLS URLs per affected user. The median of those users has 70 TLS URLs leaked through Cookie Synchronization, when the 90th percentile has up to 226 TLS URLs leaked.

their set-cookies with the one of *https://example.com*. However, by doing that over plain HTTP, the visited website gets leaked through the referrer field to a possible monitoring entity (e.g., ISP). In addition, this snooping entity from now on can skip the privacy provided by TLS and re-identify the user in the web just by monitoring the cookies of the HTTP traffic of *http://tracker2.com* and *http://tracker3.com*, even if the user browses through proxies.

We measured such cases in our dataset and found **44 users (5%) affected by the ID-spilling of Cookie Synchronization**. The majority of the leaked domains regard top content providers, where an eavesdropper from the referrer field, apart from the domain, can also see sensitive information like the user's search queries. As can be seen in Figure 4.16, the median of those users has 70 TLS URLs leaked through Cookie Synchronization, when the 90th percentile has up to 226 TLS URLs leaked.

4.5 Measuring ID-Spilling in the wild

To assess the feasibility of the leak we described earlier, we collected and analyzed a dataset from the most popular websites worldwide during December 2017. By using Selenium, we deployed a headless Chrome browser and crawled the landing pages of the top 12K Alexa websites, through the secure VPN of our institution (FORTH). In this process, we fetch each website once, and after each probe, we erase the state of our browser. The overall volume of our dataset includes 440K HTTP(S) requests.

Type	**Amount**
- Websites crawled	12000
- HTTP(S) requests	440000
- Websites with CSync	3878
- Total CSync redirections	89479
- Total unique synced IDs	17171
- Total unique 3rd party CSync domains	773
- SSL Syncing companies	475/733 (64.2%)
- non-SSL Syncing companies	258/733 (35.2%)
- TLS websites	8398/12000
- TLS websites with CSync	2317/8398
- TLS CSync redirections	58831
- Unique synced IDs in TLS websites	9045
- **Non-TLS syncings in TLS websites**	2879
- **Unique UserIDs leaked**	609/9045
- **Leaking TLS-protected websites**	174/2317

Table 4.7: Summary of results

4.5.1 Data analysis

We examine our dataset by building an analyzer which implements our Cookie Synchronization detection technique. We detect 89479 HTTP/HTTPS syncing requests that appear in 3878 websites (32% of the overall crawled websites) and synchronize with 733 different third-party domains, a number of 17171 unique identifiers (UserIDs). From these 733 third parties, 35.2% does not support HTTPS. Table 4.7 summarizes our findings in this dataset.

We note the following caveats in our data collection process: First, the top Alexa list includes domains from supporting web services, such as CDNs, DDoS protection, or comment hosting services, etc. These are sites the user indirectly visits, while visiting the website they are actually interested in. Thus, it is highly unlikely that a real user in their everyday browsing behavior will visit such web services directly as 1st party websites. Second, due to the automated nature of our data collection, we had a few cases where the fetched domains denied to serve our crawler due to their anti-bot policy. Third, there are cases where third parties may avoid to perform ad-auctions [185] and Cookie Synchronizations whenever they detect requests from a headless browser, as the ad-ecosystem is not interested in serving targeted ads to bots. Considering all the aforementioned, our findings in this dataset are a lower bound of the problem. In fact, we believe that the portion of affected websites a user may face this privacy leakage while browsing the web, can be higher than reported here.

Cookie Synchronization vs. Websites: Given that in this study, we investigate leaks from TLS-protected websites, we extract a subset of these websites which included 8398 distinct domains. From these domains, 2317 (27.5%) have at least one Cookie Synchronization. In total, the Cookie Synchronizations we detected in these TLS-protected websites is 58831 and 9045 unique userIDs, as reported in Table 7.1. In Figure 4.17, we plot the distribution

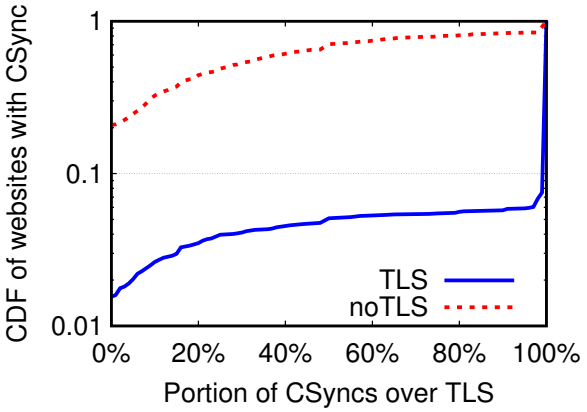


Figure 4.17:
Distribution of the portion of TLS-based synchronizations per website for both TLS and non-TLS websites. As we see, the median non TLS website has around 27% of its Cookie Synchronizations over TLS when, most of the TLS-protected websites have 92% of their synchronizations over TLS.

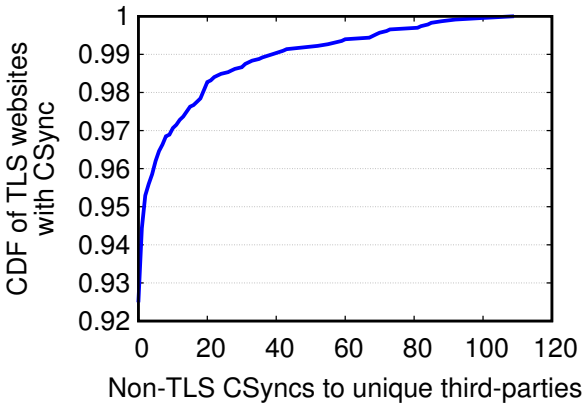


Figure 4.18:
Distribution of non-TLS synchronizations per TLS website. Few websites (1 in 13) include quite a lot (up to 100!) of plain-HTTP CSync redirections.

of the portion of TLS-based synchronizations per website for TLS and non-TLS websites. As we see, the vast majority (92%) of the Cookie Synchronizations of TLS supported websites happen over TLS. However, we see a quite respectable 7.6% of synchronizations initiated over plain HTTP, risking the confidentiality of the entire browsing session.

TLS browsing session leak: To examine further the TLS websites that use non-TLS synchronizations, we plot the distribution of the plain-HTTP synchronizations per TLS website. As we see in Figure 4.18, some of these websites include synchronizations of userIDs over plain HTTP with up to 100 different third parties. In fact, 1 in 13 TLS-supported websites perform Cookie Synchronization via plain HTTP. This means that a snooping ISP has 100 times more chances to find, in the future, a HTTP request to one of these synced third parties and re-identify the user.

Next, we examine the referrer fields of the syncing redirections from TLS websites, and check if the domain in the referrer field matches with the visited Alexa website. This check enables us to filter-out cases where the Cookie Synchronization was triggered by an embedded iframe of the website. In this case, the referrer field links to the iframe's domain.

We find 174 cases of websites, where referrer fields from HTTP Cookie Synchronizations leak the visited webpage along with full URL parameters. Besides that, through the same synchronizations, 610 unique userID were exposed to a monitoring ISP. Using this user infor-

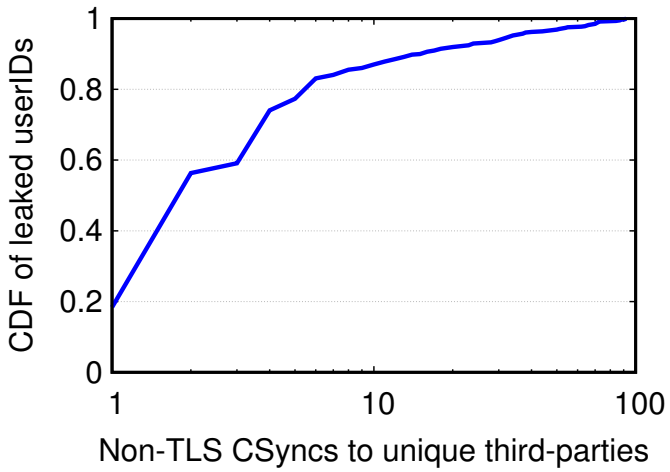


Figure 4.19: Distribution of the non-TLS synchronizations per leaked userID. There is a 10% of IDs that gets synced with more than 17 third parties.

mation, the ISP can re-identify the user in the web and through the referrer-leaked webpages it can reconstruct the browsing history of the user. In Figure 4.19 we plot the distribution of the leaking synchronizations for each of these 610 userIDs. As we see, the median userID gets leaked by synchronizations with 2 different third parties. However, there is a 10% of user IDs that gets synced, and thus leaked, with more than 17 different third parties.

Countermeasures: Nowadays, a lot of users deploy ad-blockers [169] to remove the annoying or resource consuming [183] ads. Indeed, ad-blockers can eliminate the privacy leak we present here by killing all third party domains. However, they would also kill the funding model of contemporary web, making content providers block ad-blocking visitors [159]. In addition, according to Englehardt et al. [67], less than half of the third parties (46%) in top websites use HTTPS, thus, impeding the overall adoption of HTTPS and generating lots of cases of mixed content in TLS supporting websites.

In effect, the most important countermeasure against this leak is to increase the adoption of HTTPS (in both web and mobile apps [179]), as major Non-Profit Organizations and Internet stakeholders promote [78]. This way, every single connection the user establishes with the remote servers to fetch a component in a mashup, will be secured. Of course, supporting HTTPS does not come cheaply. Despite of its huge advantages, one may argue that there are specific latency and maintenance overheads, such as key and certificate maintenance, trust revocation handling, etc. [33, 34, 168]. Therefore, tracking entities may lack the incentives to deploy and deal with such overheads.

For this reason, we believe that it is the browser vendors that must (i) force the use of TLS by everyone and forbid *any* susceptible use of mixed content, and (ii) during request

Feature subset	Features	TP rate	FP rate	Precision	Recall	F-Measure	AUCROC
NoOfParams*	1	0.639	0.639	0.408	0.639	0.498	0.500
WhereFound+	1	0.643	0.610	0.612	0.643	0.535	0.602
StatusCode*+	1	0.648	0.619	0.723	0.648	0.523	0.633
TypeOfEntity*+	1	0.735	0.432	0.752	0.735	0.701	0.661
Browser*+	1	0.700	0.492	0.710	0.700	0.651	0.628
ParamName+	1	0.815	0.295	0.828	0.815	0.803	0.834
EntityName*+	1	0.803	0.295	0.806	0.803	0.793	0.854
{no id-related}*	5	0.840	0.242	0.845	0.840	0.834	0.887
{high importance}+	6	0.870	0.206	0.877	0.870	0.865	0.919
ALL	9	0.900	0.144	0.901	0.900	0.898	0.946

Table 4.8: Performance of decision tree model trained on different subsets of features available at runtime for classification, given already identified id-sharing entries, and 10 cross-fold validation.

marshaling, strip *any* information (the referrer field in our case) may link together different type of traffic (HTTPS and HTTP). In fact, some privacy-sensitive browsers have already started providing such alternatives [25, 39]. By applying the above, not only the privacy of the users will be preserved, but the content providers will be fortified against visitor data and revenue loss [16].

Future Work: We plan to investigate further the characteristics of the TLS protected websites that are more prone to expose the privacy of the users. Specifically, by crawling a larger dataset of websites, we will conduct a deeper analysis of the content category and top-level domains of these websites. In addition, we plan to investigate to what extend the phenomenon appears in websites with lower popularity, considering that these websites may draw more 'sloppy' trackers that do not care about supporting TLS, and explore if there is an association with the popularity of such trackers.

4.6 Evaluation of Cookie-less Detection

In this section, we explore two different scenarios outlined in Section 4.2.2 regarding the detection of CSync via ID sharing, while such IDs may be obfuscated to remove the possibility of matching them with past IDs shared between entities. First, we explore the scenario where IDs have been shared, detected by the heuristic-based approach, but have not yet been confirmed as CSync events. That is, we consider an already identified set of id-sharings, which are candidate CSync events, but cannot be validated as CSyncs because of the cookie ID being encrypted or unavailable (Section 4.6.1). Second, we take a step back and consider the more general case where various HTTP connections are ingested by the method, and

Feature subset	Features	TP rate	FP rate	Precision	Recall	F-Measure	AUCROC
NoOfParams*	1	0.541	0.314	0.584	0.541	0.495	0.706
StatusCode*+	1	0.666	0.229	0.673	0.666	0.598	0.764
TypeOfEntity*+	1	0.760	0.162	0.724	0.760	0.695	0.834
EntityName*+	1	0.865	0.075	0.863	0.865	0.860	0.962
ParamName+	1	0.870	0.083	0.878	0.870	0.859	0.953
{no id-related}*	4	0.904	0.057	0.904	0.904	0.898	0.973
{high importance}+	4	0.919	0.051	0.923	0.919	0.914	0.978
ALL	5	0.920	0.051	0.925	0.920	0.916	0.978
Unbalanced test-set	5	0.981	0.004	0.989	0.981	0.984	0.999

Table 4.9: Performance of decision tree model trained on different subsets of features available at runtime for classification, given a pre-filter for ID-looking strings. All results besides the last row are with balanced dataset across the three classes, and 10-cross fold validation. The last row's results are computed given an unseen, and unbalanced test set, maintaining the original ratio of classes.

it needs to decide at run-time which are CSync events and which are not based on given features (Section 4.6.2).

Towards this end, we train and test the classifier in these two experiments. We remind the reader of the assumption made earlier, that the distributions of the features describing the CSync events with unencrypted IDs have the same variability in the cases of encrypted IDs, and therefore can be used for the detection of such cases. This assumption allows us to handle the problem as an out-of-sample estimation, leaving as future work the final validation with a set of ground-truth data of encrypted IDs that we also know their unencrypted versions.

Data and Features: Based on the ground truth data presented earlier with the heuristic-based technique, we have a total of 412.8k *id-sharing* events, from which 263.6k are confirmed CSync, and 149.2k are identified as *non-CSync*. The features available for these events can be various as already explained in 4.2.2. The ones we use are features available at run time, and during the user's browsing to websites.

Algorithms: The final machine learning classifier used is decision tree-based. Others like Random Forest, Support Vector Machines and Naive Bayes were tested, but the decision tree algorithm outperformed them, with significantly less computation and memory overhead.

Metrics: To evaluate the performance of the classifier on the different classes and available features, standard machine learning metrics were used such as Precision, Recall, F-measure, True Positive rate (TP), False Positive rate (FP), and area under the receiver operating curve (AUCROC).

TP Rate	FP Rate	Precision	Recall	F-Measure	AUCROC	Class
0.988	0.014	0.534	0.988	0.693	0.998	Csync
0.603	0.005	0.458	0.603	0.521	0.990	*id-sharing but non-CSync*
0.984	0.004	1.000	0.984	0.992	0.999	other
0.981	0.004	0.989	0.981	0.984	0.999	weighted average

Table 4.10: Detailed performance of decision tree model trained on different subsets of features in a balanced dataset, and tested on an unseen, and unbalanced test set, which maintains the original ratio of classes (last row of Table 4.9.

4.6.1 Detecting CSync in ID-sharing HTTP

In this experiment, we assume there is already in place an existing technique for analysis of the HTTP traffic of the user, similar to the method outlined in Figure 4.3. However, there are candidate CSync events that cannot be confirmed, as the IDs cannot be matched with SET cookie IDs, either because these actions are not available to the method, or because the IDs are encrypted.

In this case, a machine learning classifier can be trained to detect if an id-sharing HTTP request is a true CSync event, by matching its pattern to past verified CSync events. For this experiment, we use two classes: the CSync events and the *id-sharing but non-CSync* events, to train and test a decision tree classifier under different subsets of features. The training and testing was performed using 10 cross-fold validation process. The results are shown in Table 4.8. We observe that independently, each of the features considered have some predictive power, except from the *NoOfParams* feature. When the most important[7] features are combined, a weighted AUC of 0.919 is achieved. Furthermore, when we select features that are non-ID related, a respectable weighted AUC=0.887 is achieved. When all features are used, the classifier can reach a weighted AUC=0.946.

4.6.2 Detecting CSync in HTTP with ID looking strings

In this experiment, we assume there is a simple HTTP pre-filter, which keeps connections with ID looking strings for further investigation. This is a necessary step to reduce the run-time workload of the classifier, as connections relevant to the task are only $\sim 20\%$ of the overall HTTP workload. Then, the classifier has to decide which of the following classes match for each one of the selected HTTP requests: 1) CSync, 2) *id-sharing but non-CSync*, and 3) other. In this case, *other* refers to HTTP entries that contain an ID-looking string, but are *not id-sharing*.

We perform two rounds of tests on one month's worth of data: 1) train and test the

[7] using information gain as metric

algorithm using balanced data from the three classes, in a 10 cross-fold validation process. 2) train the algorithm on balanced data from the three classes (as in (1)), but test it on an unseen and unbalanced dataset which maintains the ratio of the three classes: CSync: 1.6%, *id-sharing but non-CSync*: 0.73% other: 97.67%.

As seen from the classification results (Table 4.9), the company name and parameter used are among the most important features, and number of parameters is the worst. Beyond that, non-ID related features allow the classifier to reach weighted AUC = 0.973, with a high weighted Precision and Recall across all classes. When all features are used, a weighted AUC = 0.978 is reached, similarly to the high importance feature set that disregards the number of parameters. Interestingly, when the classifier is trained on the balanced dataset, and tested on the unbalanced test set (last row of Table 4.9), the classifier can distinguish very well the three classes, with low error rates across all three classes, even though there is high imbalance in the classes. These results are further validated by the breakdown of performance per class, demonstrated in Table 4.10, which show high TP rate and low FP rate for all three classes independently.

Overall, the results show that it is possible to understand and model the patterns of CSync, as they are driven by particular types of companies, using specific parameters, etc. Therefore, an online classifier could be trained to provide insights as to what each HTTP connection is and how likely it is to be performing CSync, without the need to match the IDs to SET cookie actions.

4.7 Discussion

User data have become a precious asset of web entities, which invest a lot in elaborate tracking mechanisms. These mechanisms, by monitoring the users' actions, collect behavioral patterns, interests and preferences, PII data, geolocation, age, etc. These data are then sold to advertisers in data markets, typically for the purpose of delivering highly targeted ads to their owners, based on online (and offline) preferences. However, each tracker uses a different ID for the same user, and in order for all these data to matter, a universal identification must be applied. The most popular such technique is Cookie Synchronization, with which different entities can synchronize the userIDs they use for a specific user and merge their databases on the background. Syncing the set of userIDs of a given user, increases the user identifiability while browsing, thus reducing their overall anonymity on the web.

In effect, such data collection and sharing activities, done without users' explicit consent, are illegal and punishable with hefty penalties imposed to the companies performing these activities. This is especially true since May 2018, with the introduction of new EU regulations for the protection of user personal data and online privacy (GDPR and e-Privacy). Therefore, it is important for the design and development of practical, web transparency tools, which will be readily available to privacy researchers, regulators and end-users. Both advanced

or tech-savvy users, as well as average users, should be able to utilize these tools for the investigation of personal data leakage and anonymity loss they experience while browsing the web, due to third parties' activities such as Cookie Synchronization.

In this chapter, we build CONRAD: a holistic system to detect events of Cookie Synchronization, either when the synced IDs are available in plaintext, or even when they are obfuscated (i.e., hashed, encrypted). Using our detection mechanism, we are the first to explore Cookie Synchronization in the mobile ecosystem and the first to analyze it in depth, using a year-long dataset of real mobile users. CONRAD is able to capture 3.771% more cases of Cookie Synchronization than related work.

4.7.1 Lessons Learned

Our results in this chapter can be summarized as follows:

- 97% of the users are exposed to CSync at least once. The median user experiences at least one CSync within the first week of browsing.
- Ad-related entities participate in more than 75% of the overall synchronizations through the year, learning as much as 90% of all synced userIDs.
- No more than 3 companies learn more than 30% of the all userIDs each.
- The median userID gets leaked, on average, to 3.5 different entities.
- The average user receives around 1 synchronization per 68 GET requests, and gets up to 6.5 of their userIDs synced.
- The number entities that learn about the median user after Cookie Synchronizations grows by a factor of 6.7x.
- We find 63 cases, where domains set cookies on the users' browsers with userIDs previously set by other domains. This universal identification model enables collaborating entities to share data without background database merges.
- We find 131 cases of domains storing in cookies their Cookie Synchronizations results forming ID Summaries.
- 5% of the users suffer from ID-spilling in their secure TLS connections. Together with the userID, the visited website gets also leaked.
- There are cases where several privacy-sensitive information is passed to the syncing entity along with the userID like gender, birth dates, email addresses, etc.

In addition to the classic, heuristics-based method applied to collect the above findings, in this work, we proposed a novel, Cookie Synchronization detection mechanism that is able to detect at real time Cookie Synchronization events, even if the synced IDs are obfuscated. In particular, this online, machine learning classifier can be trained to provide insights as to what each HTTP connection is, and how likely it is to be performing CSync, without the need to match the IDs to SET cookie actions. We use the set of detected Cookie Synchronizations from the heuristics-based method as ground truth, to train our machine learning-based, cookie-less detection algorithm. We were able to achieve high accuracy (84%-90%) and high AUC (0.89 - 0.97), when non-ID related features were used.

4.8 Countermeasures

Nowadays, the most popular defense mechanism of Cookie Synchronization is the use of the traditional ad-blockers. Indeed, since the vast majority (75%) of CSync takes place among ad-related domains (see Figure 4.13), it is easy to anticipate that by blocking ad-related requests, one can eliminate a large portion of the privacy leak that Cookie Synchronization causes. However, the all-out approach of ad-blockers causes significant harm on the publishers' content monetization models, forcing some of them to deploy anti-adblocking mechanisms [115, 161, 162, 173] and deny serving ad-blocking users [159, 216, 223].

Consequently, we believe that mitigating mechanisms against Cookie Synchronization require a more targeted blocking strategy, that would not harm the current ad-ecosystem blindly. Instead, by applying detection techniques such as CONRAD, and blocking the specific traffic which has been found to facilitate CSync, we believe that the harmful privacy leakage and loss of anonymity of users due to CSync can be avoided, and without the dire consequences on publishers' business models. Also, by applying such detection techniques, instead of general ad-blocking, CSync that is initiated beyond advertising and tracking domains can also be stopped.

Indeed, blocking CSync may have consequences on the effectiveness of ads, since fewer highly targeted ads will be matched to users and, therefore, delivered to them. However, we believe it is imperative for a balance to be found between the need for highly targeted ads, which are primarily depended on aggressive user personal data collection, and the utility these ads offer to the users, to the hosting publishers and the advertisers involved, while taking into consideration the user privacy and anonymity lost during this data collection.

Chapter 5
The Impact of User Data on Ad-pricing dynamics

In the previous chapters, we explored the techniques used for user tracking by web entities in an attempt to collect as much detailed user information possible. It is apparent, that digital advertising is the most important means of monetizing such collected data. It grew to $194.6 billion in 2016 [218] of which $108 billion were due to mobile advertising. In fact, more and more companies rush to participate in this rapidly growing advertising business either as advertisers, ad-exchanges (ADXs), demand-side platforms (DSPs), data management platforms (DMPs), or all of the above. For these companies to increase their market share, they need to deliver more effective and highly targeted advertisements. A way to achieve this is through programmatic instantaneous auctions. An important enabler for this kind of auctions is the Real-Time Bidding (RTB) protocol for transacting digital display ads in real time. RTB has been growing with an annual rate of 128% [243], and currently accounts for 74% of programmatically purchased advertising. In US alone it created a revenue of $8.7 billion in 2016 [19].

Besides the obvious privacy implications the aggressive user tracking mechanisms we presented may cause, still, there is an outstanding question that remains unaddressed by the related work in the area. This question concerns transparency and is the following: *Based on the exposed user personal data, how much do advertisers pay to reach an individual?*

Despite the importance of this question, it is surprising how little is known about it. There exist several reports about the *average* revenue per user (ARPU) from online advertising [43, 99,188], but ARPU, as its name suggests, is just an average. It can be calculated by dividing the total revenue of a company by the number of its monthly active users. Computing the revenue per *individual* user is a completely different matter for which very limited work is available.

In particular, the FDVT [48] browser extension can estimate the value of an individual user for Facebook, by tapping on the platform's ad-planner. Another important prior work [176] leverages similarly the RTB protocol and specifically its final stage, where the

winning bidder (advertiser) gets notified about the auction's charge price per delivered impression. These charge prices were initially transmitted in cleartext and focused solely on them. However, more and more advertising companies use encryption to reduce the risk of tampering, falsification or monitoring from competitors. This trend renders that method inapplicable for the current and future ad ecosystem, whose majority of companies will deploy charge price encryption. In contrast to these works, our present method takes into account all the web activity of a user (not only on Facebook), and all RTB traffic, i.e., both cleartext and encrypted prices.

In this chapter, our motivation is to enhance *transparency* in digital advertising and shed light on pricing dynamics in its personal data-driven ecosystem. Therefore, we develop and evaluate a first of its kind methodology for enabling end-users to estimate in real time their actual cost for advertisers, even when the latter encrypt the prices they pay. Designed as a browser extension, our method can tally winning bids for ads shown to a user and display the resulting amount as she moves from site to site in real time.

In summary, in this chapter we propose the first to our knowledge holistic methodology to calculate the overall cost of a user for the RTB ad ecosystem, using both encrypted and cleartext price notifications from RTB-based auctions. We study the feasibility and efficiency of our proposed method by analyzing a year-long weblog of 1600 real mobile users. Additionally, we design and perform an affordable (a few hundred dollars cost) 2-phase real world ad-campaign targeting ad-exchanges delivering cleartext and encrypted prices in order to enhance the real-users' extracted prices. We show that even with a handful of features extracted from the ad-campaign, our methodology achieves an accuracy $> 82\%$. The resulting ARPU is $\sim$55% higher than that computed based on cleartext RTB prices alone. Our findings challenge the related work's basic assumption [176] that encrypted and plain text prices are similar (we found encrypted prices to be $\sim 1.7\times$ higher). Finally, we validate our methodology by comparing our average estimated user cost with the reported per user revenue of major advertising companies. Using lessons from this study, we implemented a prototype of our approach, where the users, by using our Chrome browser extension, can estimate in real-time, in a privacy-preserving fashion on the client side, the overall cost of their exposed private information for the advertisers. In addition, they can also contribute anonymously their impression charge prices to a centralized platform for further research.

5.1 Technical Challenges

5.1.1 Encrypted vs. cleartext prices

Although in the early years of RTB, all charge prices in nURLs were in cleartext, we see that nowadays more and more companies deliver charge prices in encrypted form (see examples in Table 2.1). While cleartext prices captured at the user's browser can be easily tallied

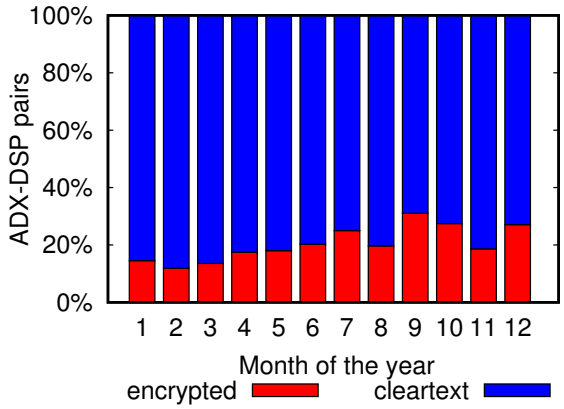


Figure 5.1:
Portion of encrypted and cleartext pairs of ADX-DSP over time (2015).

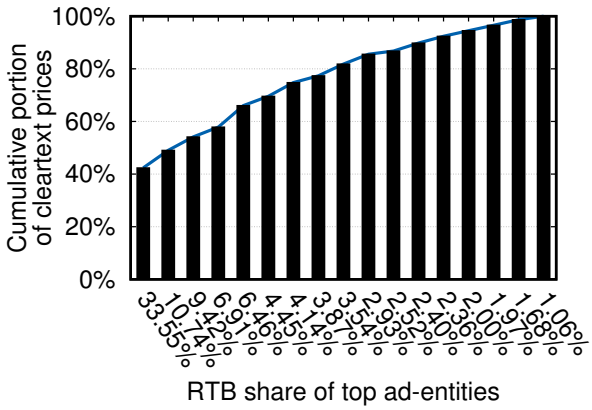


Figure 5.2:
Cumulative portion of cleartext prices vs. ad-entities' portion of RTB.

to estimate the total cleartext cost, the same does not apply for the encrypted prices. The popular 28-byte encryption scheme companies use cannot be easily broken [90].

Previous studies [176] assumed that encrypted prices follow the same distribution as cleartext ones. Indeed, one may argue that the price encryption is just to avoid tampering of reported prices, so encrypted prices probably follow the cleartext price distribution. However, encryption provides also confidentiality to the bidding strategy. Thus, possible use of encryption in charge prices may be also a sign of a higher value that the bidder wants to hide: aggressive re-targeting because of user's previous incomplete purchases, targeting users with higher spending habits, or users with specialized needs (e.g., sensitive products, expensive drugs, etc.). Hence, a bidder (e.g. a DSP) may choose encryption to reduce transparency over its bidding strategies, or possible special knowledge it may have about a specific user, thus preventing an external observer or competitor from assessing its bidding methods and ad-campaigns.

We should note that encryption is not a feature that comes for free. There are significant costs for the participating parties such as more computation and storage overhead, energy consumption and higher imposed latency. Therefore, these costs alone could be a reason for an ADX to charge more for providing the benefits of encryption to a DSP. Considering all the aforementioned, in our study, we remove the need for making any assumptions regarding encrypted prices and allows us to account for any potential differences between cleartext and encrypted prices.

Notation	Definition
V_u	Total cost of user u
C_u, E_u	Sum of cleartext, encrypted prices of user u
SC_u, SE_u	Set of cleartext, encrypted price nURLs of user u
F_i	Vector of features for a price nURL i
$S_i \subseteq F_i$	Core features selected to represent nURL i
$ESe(S_i)$	Estimated encrypted price based on vector of features S of price nURL i

Table 5.1: Summary of notations.

5.1.2 Encrypted prices on the rise

Encryption is a regular practice in desktop RTB ads (∼68% as reported in [175] with major supporters being DoubleClick, RubiconProject and OpenX). By analyzing a weblog of 1600 real mobile users (see Section 7.2), we detected a smaller portion in mobile RTB ads (∼26%). However, we found that the percentage of ADX-DSP pairs using encrypted price nURLs was steadily increasing through time (Figure 5.1), which means that more and more mobile advertising entities have started using nURLs with encrypted prices.

In fact, we found that the mobile advertising entities with the larger RTB shares deliver the highest portion of cleartext prices as well (Figure 5.2). For example, MoPub and Adnxs, the two leading ad-entities in our dataset, are responsible for 33.55% and 10.74% of the overall RTB ads detected, respectively (x-axis). They are also responsible for 45.40% and 5.45% for the cleartext prices detected, respectively (cumulatively in y-axis). If these two (or more) companies flipped their strategy from cleartext to encrypted, it would dramatically impact the RTB-ecosystem's transparency and hinder price information exposed to an external auditor or the involved user.

Given these trends in mobile and desktop, we expect that in the near future RTB auctions will dominate, and many of the ad-entities will use encryption to deliver their charge prices. Our methodology anticipates these trends and promotes better transparency in online advertising and usage of user personal data. This methodology allows end-users to accurately estimate on their browser, at real-time, their average ad-related cumulative cost, even when the charge prices are encrypted.

5.2 Methodology

In this section, we describe our proposed methodology, with which a user u can estimate in real-time the accumulated cost V_u for the ads she was delivered while browsing the web (§ 5.2.1) (notations used are summarized in Table 5.1). Following this methodology, we design our system based on two main components: (i) a remote *Price Modeling Engine* (§ 5.2.2)

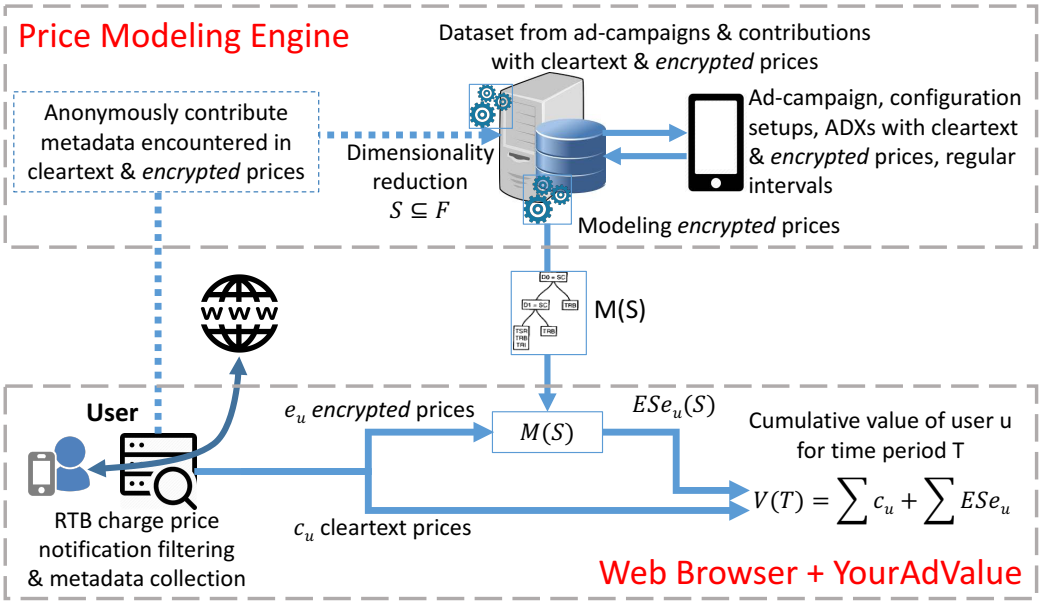


Figure 5.3: High level overview of our method. The user deploys YourAdValue on her device, which calculates in real-time the total cost paid for her by advertisers. In case of encrypted prices, it applies a decision tree model derived from the PME.

and (ii) a user-side tool, namely *YourAdValue* (§ 5.2.3). Figure 5.3 presents an overview of our proposed methodology.

5.2.1 Overall cost of the user's data

The overall ad-cost of the user for time period T is the sum of charge prices the advertisers have paid after evaluating her personal data they have collected and delivering ads to her device. Specifically, this overall value is the sum of both her cleartext $C_u(T)$ and encrypted $E_u(T)$ prices and can be stated as:

$$V_u(T) = C_u(T) + E_u(T) \tag{5.1}$$

The cleartext prices of a user can be aggregated in a straightforward fashion, thus producing the ad-cost for user u over such prices:

$$C_u(T) = \sum_i c_i, \ where \ \ i \in SC_u(T) \tag{5.2}$$

On the other hand, the calculation of the aggregated $E_u(T)$ of the encrypted prices for

the same user is not easy. The actual price values e_i are hidden and therefore need to be estimated. To achieve that, we leverage the metadata of each charge price in the user's set SE_u(T) of encrypted price notifications. Such metadata may include: time of day, day of week, size of ad, DSP/ADX involved, location, type of device, associated IAB, type of OS, user's interests, etc. All these metadata are collected in a feature vector F_i that captures the context of a specific charge price e_i in nURLi.

In order to estimate each encrypted notification price i, we built a machine learning model, which receives as input the feature vector F_i (or a subset $S_i \subseteq F_i$), extracted from the nURLi, and estimates a charge price $ESe(S_i)$ for the encrypted price e_i. This permits us then to aggregate the estimated encrypted prices for user u as we have done for the cleartext ones:

$$E_u(T) = \sum_i ESe(S_i),\ where\ \ i \in SE_u(T) \tag{5.3}$$

5.2.2 Price Modeling Engine

The core element of our solution, the Price Modeling Engine (PME), is a centralized repository responsible for the estimation of encrypted prices. To achieve this, the PME requires a sample of charge price data and associated features to train a machine learning model. This component is designed to incorporate data such as offline weblogs (see Section 7.2), or online anonymous contributions (anonymized features and charge prices) from participating users, similarly to other systems that depend on crowd-sourcing (e.g., Floodwatch [77]). Using such data, the PME can re-train the computed model at any time. To assess the difference between cleartext and encrypted price distributions in the wild and fine-tune the training model, the PME runs small "probing ad-campaigns" to collect ground truth of real charge prices from both encrypted and cleartext formats.

Feeding the PME with all possible metadata available, i.e. auctions' metadata and users' personal data, is clearly not practical. There exist hundreds of data points per individual price. Passing all of them to the modeling engine would make the computational cost excessive. Additionally, if all data points were to be used in the probing ad-campaigns, they would render such campaigns too expensive for their purpose. In order to run effective and efficient ad-campaigns, and allow the training of a price model without high computation overhead, the PME performs careful dimensionality reduction on the extracted metadata (F) to derive a subset $S \subseteq F$ of core features capable to capture the value of an impression. This dimensionality reduction makes the probing ad-campaigns feasible by reducing by many orders of magnitude the needed features of each testing setup, and effectively the number of setups to be tested (see Section 5.4).

Using the collected ground truth of encrypted prices from ad-campaigns, the PME trains a machine learning model M to infer encrypted prices based on their associated subset of features S. Then, each user can apply the model M (in the form of a decision tree) locally on

their device to estimate each of her encrypted charge prices based on the matching metadata S.

In case the availability of cleartext prices is limited, the reduction step to identify important features could be hindered, but not obstructed. To mitigate this, the PME can run more probing ad-campaigns to cover extra features found in users' anonymous contributions, or that are available in professional ad-campaign planners (as in FDVT [48]). Then, the most important features can be selected based on their contribution to model the encrypted prices extracted from these campaigns.

5.2.3 YourAdValue

YourAdValue is a user-side tool responsible for monitoring the user's nURLs and calculating locally the cumulative cost paid for her in real-time. To achieve this, it filters nURLs from her network traffic and extracts (i) the RTB auction's charge prices (both encrypted and cleartext), and (ii) metadata from each specific auction (e.g. time of day, day of week, size of ad, involved DSP and ADX, etc.) along with the personal data the user leaks while using online services (e.g. location, type of device and browser, type of OS, browsing history, etc.).

As we mentioned earlier, cleartext prices can be aggregated directly, but encrypted prices must be estimated. Therefore, YourAdValue retrieves from the PME a model $M(S_i)$ that (i) includes the features S_i that need to be extracted from the collected metadata, and (ii) provides a decision tree for the estimation of an encrypted price based on these features.

Using this model, YourAdValue can estimate locally on the client side, the value $ESe(S_i)$ of the encrypted charge prices based on the features S_i of the given nURL. After estimating each encrypted price, YourAdValue presents to the user the calculated sums $C_u(T)$ and $E_u(T)$ along with relevant statistics and the total amount $V_u(T)$ paid by advertisers (see Section 6.2).

YourAdValue can be implemented in the same manner, either as a browser extension for desktops or as a module for mobile devices. In the latter case, YourAdValue can monitor traffic of both browsers and apps similar to existing approaches [179]. For simplicity, in this work we design YourAdValue as a browser extension; its mobile counterpart is part of our future work.

Our tool, built as an extension for Chrome browser, monitors both HTTP and HTTPS traffic of the user and detects the RTB nURLs. Additionally, it stores in the browser's local storage the filtered charge prices, the personal and auctions metadata and the estimation of the encrypted prices. The extension, through toolbar notifications, informs the user about newly detected RTB charge prices. Upon request, it reports the cumulative cost along with the previous individual charge prices. Finally, the extension periodically issues requests to PME to check for new versions of the model.

Metric	D	A1	A2
Time period	12 months	13 days	8 days
Impressions	78560	632667	318964
RTB publishers	∼5.6k/month	∼0.2k	∼0.3k
IAB categories	18	16	7
Users	1594	-	-

Table 5.2: Summary of dataset and ad-campaigns.

5.3 Bootstrapping PME

We assess the feasibility and effectiveness of our methodology by bootstrapping the PME to train our model on real data by collecting a year long dataset containing weblogs from 1594 volunteering mobile users from the same country. Our users agreed to use a server of our control as a proxy, allowing us to monitor their outgoing HTTP traffic.[1] As a result, we were able to collect a large dataset D of 373M HTTP requests spanning the entire year of 2015. Note that though our dataset consists of HTTP-only traffic, in principle our approach works with HTTPS as well, using as input the users' contributed data as can be seen in Figure 5.3. Table 5.2 presents a summary of our dataset D. Next, we present the data collection and analysis to extract features used in the price modeling and ad-campaign planning.

5.3.1 Dataset analysis

Weblog Ads Analyzer. To process our dataset, we implemented a weblog advertisements analyzer, capable of detecting and extracting RTB-related ad traffic. First, the analyzer uses a traffic classification module to categorize HTTP requests based on an integrated blacklist of the popular browser adblocker Disconnect [55].[2] Using this blacklist, the analyzer categorizes domains in 5 groups based on the content they deliver: (i) Advertising, (ii) Analytics, (iii) Social, (iv) 3rd party content, (v) Rest. It consequently applies a second-level filtering on the advertising traffic by parsing each URL for any RTB-related parameters (like nURL). The analyzer detects nURLs by applying pattern matching against a list of macros we collected after (i) manual inspection and past papers [146, 176], and (ii) studying the existing RTB APIs [58,109,157,177,193] used by the current dominant advertising companies. From these detected nURLs, it extracts the charge prices which we assume in this study that are in US dollars[3] paid by the winning bidders, after filtering out any bidding prices that may co-exist in each nURL. It also extracts additional ad-related parameters such as ad impression ID, bidder's name, ad campaign ID, auction's ad-slot size, carrier, etc.

[1] Data were treated anonymously although users signed a consent form allowing us to collect and analyze their data.

[2] Our analyzer can also integrate more than one blacklists (e.g., Adblock Plus' Easylist, Ghostery's blacklist, etc.)

[3] Given that the majority of ADXs are located in US and following previous works [176], we assume every charge price to be in US Dollars (so $1CPM$ =1/1000 impressions).

Type	Feature
Geo-temporal	Time of day, Day of week
	Location of user based on IP, # of unique locations of the user, location history
User	Interest categories of the user, Type of mobile device, # of total web beacons detected for the user, # of cookie syncs detected of the user up to now, # of publishers visited by the user, # of total bytes consumed by the user,
	Avg. number of reqs per user for the advertiser, # of HTTP reqs of the user, Avg. number of bytes per req of user, Total duration of reqs of the user, Avg. duration per req of the user
Ad	Size of ad, ADX of nURL, DSP of nURL, IAB category of the publisher, popularity of particular ad-campaign,
	# of total HTTP reqs of the advertiser, # of bytes of HTTP req, Avg. duration of the reqs for the advertiser, # of URL parameters, Number of total bytes delivered for the advertiser

Table 5.3: Features extracted by summarizing data from parameters embedded in each price notification detected in the dataset for users and advertisers.

Other operations carried out by our analyzer include: (i) user localization based on reverse IP geo-coding, (ii) separation of mobile web browser and application originated traffic based on the *user-agent* field of each HTTP request, (iii) extraction of device-related attributes from the *user-agent* field (type of device, screen size, OS etc.), (iv) identification of cooperating ADXs - DSPs pairs, leveraging the nURL used by the ADX to inform the bidder (i.e. DSP) about its auction win, (v) user interest profile based on web browsing history.

Feature extraction. DSPs use different machine learning algorithms for their decision engines, taking various features as input, each affecting differently the bidding price and, consequently, the charge price of an ad-slot. To identify such important parameters, we extracted several features from the nURLs of our dataset such as user mobility patterns, temporal features, user interests, device characteristics, ad-slot sizes, cookie synchronizations [2], publisher ranking, etc. Next, we present the analysis of the most interesting features (Table 5.3 presents a summary). We group them into 3 categories: geo-temporal state of the auction (§ 5.3.2), user's characteristics (§ 5.3.3), and ad-related (§ 5.3.4).

5.3.2 Geo-temporal features

An important parameter that affects the price of an RTB ad is the user's current location [100], information which is broadly available to publishers and trackers. Thus, in our dataset we extract user IP address and using the publicly accessible MaxMind geoIP database [151], we map each IP to its city level. In Figure 5.4, which presents the 5th, 10th, 50th, 90th and 95th percentile of the charge prices, we see that although the median values are relatively lower in large cities, the fluctuation of their price values is higher.

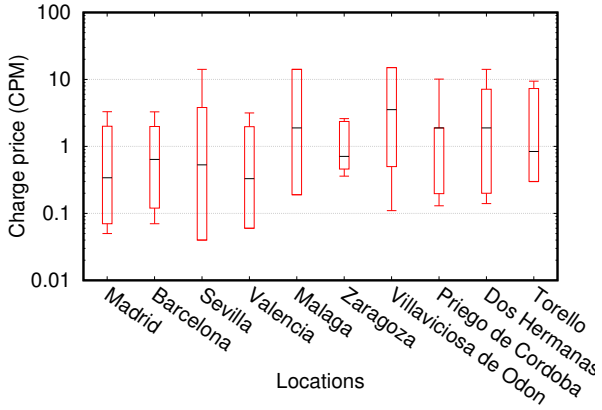


Figure 5.4:
Distribution of charge prices per city (sorted by city size).

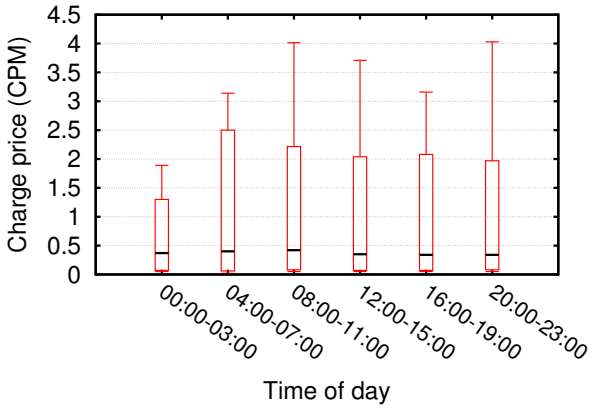


Figure 5.5:
Distribution of charge prices for different times of day.

Another important feature is time, and specifically the time of day and day of week. This is important due to the different level of attention a user may give to an ad impression and the amount of time she has to purchase an advertised product (e.g., working hours vs. afternoon's free time, or weekdays vs. weekends). In Figure 5.5, although the median charge prices are of similar range, we see that the early morning hours until noon tend to have more charge prices with increased values. In Figure 5.6, we see a periodic phenomenon, where although in median values the charge prices are quite close, during weekdays the max prices are relatively higher than on weekends.[4]

5.3.3 User-related features

Device type. By parsing the *user-agent* (UA) header information, our analyzer classifies traffic and inspects the different fingerprints the UA leaks (specifications of process virtual machine (e.g., Dalvik or ART) or kernel (e.g., Darwin), operating system, browser vendor etc.) Thus, we are able to identify the type of device (PC or mobile), the different types of mobile operating systems (Android, iOS, Windows) and if the traffic was generated from a mobile app or a mobile web browser.

In Figure 5.7, we see the percentage of RTB traffic for the different OSes over time. As expected, Android and iOS dominate, owning the larger portions of the market through the entire year, with Android-based devices appearing in 2x times more RTB auctions. However, when normalizing this RTB share per mobile OS (Figure 5.8), we find that Android and iOS

[4]For time-of-day and day-of-week distributions, which visually appear to be similar, we confirmed that they are, in fact, statistically different with non-parametric, two-sample Kolmorogov-Smirnoff tests at p-value levels of $p_{tod}<0.0002$ and $p_{dow}<0.002$.

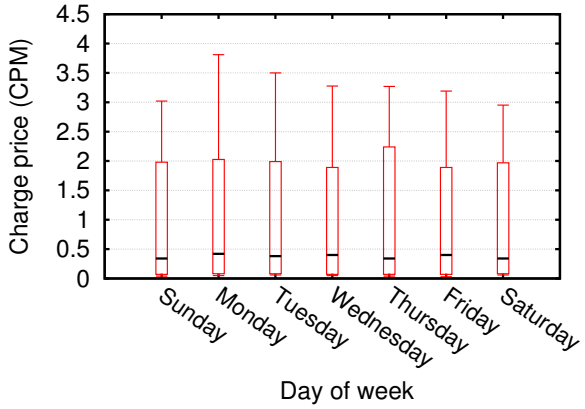


Figure 5.6:
Distribution of charge prices for different days of week.

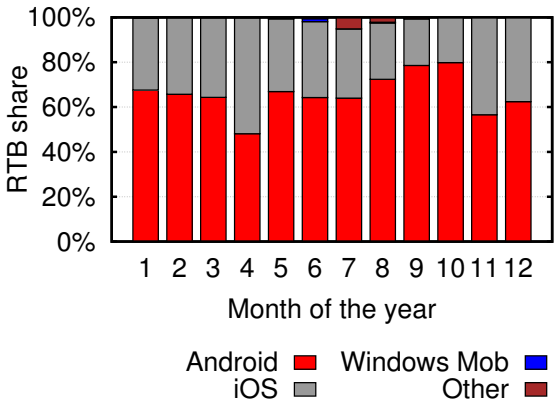


Figure 5.7:
Portion of RTB traffic for top mobile OSes.

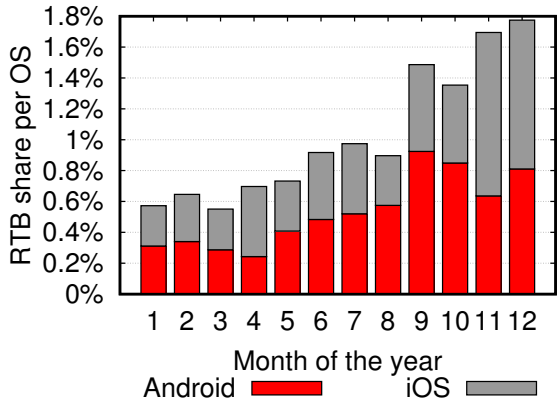


Figure 5.8:
Portion of RTB traffic normalized by OS.

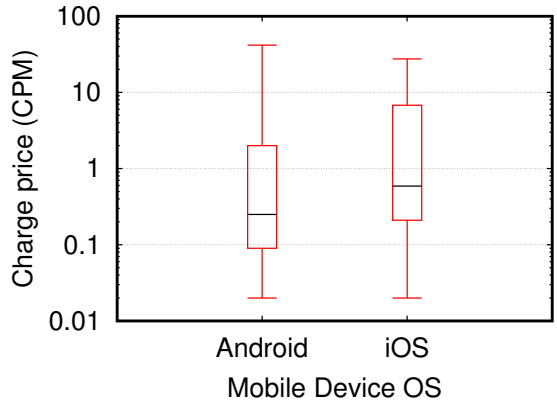


Figure 5.9:
Distribution of charge prices per mobile OS.

devices are delivered mostly equal RTB impressions, with some months Android surpassing iOS and vice-versa. Then, we extract the traffic originated from the most popular ad-entity, MoPub [158], and analyze the charge prices of the impressions rendered in the different OSes. Surprisingly, although Android-based devices are more popular, we see in Figure 5.9 that iOS-based devices tend to receive higher RTB prices, in median values.

Inference of the user's interest. The browsing history of a user is used by the advertising ecosystem as a proxy of her interests. By monitoring the websites a user visits through time, a tracker can infer her interests, political or sexual preferences, hobbies, etc., quite accurately [18]. To enrich our set of features with the users' interests, we collect all the

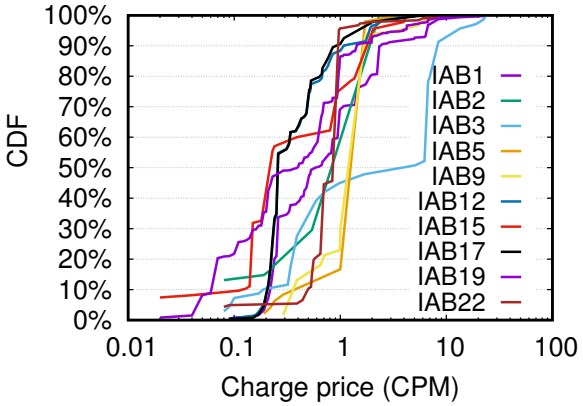


Figure 5.10:
CDF of the generated cost per IAB category.

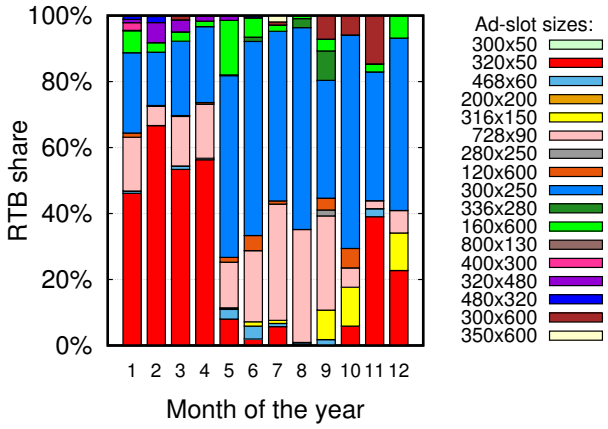


Figure 5.11:
Ad-slot size popularity through time (sorted by area size).

websites each user visits across her whole network activity. Such information is available to the RTB ecosystem as well, by using cookie synchronization [2] or web beacons [106]. To extract the interests from the visited websites, similar to existing approaches [10], we retrieve the associated categories of content for each website according to Google AdWords [92]. Then, we aggregate across groups of categories for each user and get the final weighted group of interests for each user in the form of IAB categories [231]. Figure 5.10 presents for the top mobile ADX (MoPub) a distribution of the generated ad revenue for the different IAB content categories in a 2 month subset of our dataset. As expected, not all IAB categories cost the same. Indeed, there are categories that are associated with products which attract higher bid prices in auctions, like IAB-3 (Business & Marketing), with an average charge price of up to 5 CPM for the 50% of the cases. Alternatively, there are categories like IAB-15 (Science), which are unable to draw prices higher than 0.2 CPM for the 50% of the cases.

5.3.4 Ad-related features

Web Vs. Apps Advertisers bid for ad-space in both webpages and mobile apps. After studying the cost per ad in both counterparts in our dataset, we see that apps draw on average 2.6× higher prices (0.712 CPM vs. 0.273 CPM). This is expected; studies have shown that more advertising budget is spent on mobile application ads instead of mobile web, driving higher prices per ad [167]: (i) Users pay more attention to app ads as they typically occupy fixed places in the screen, with no opportunity to scroll them out of sight

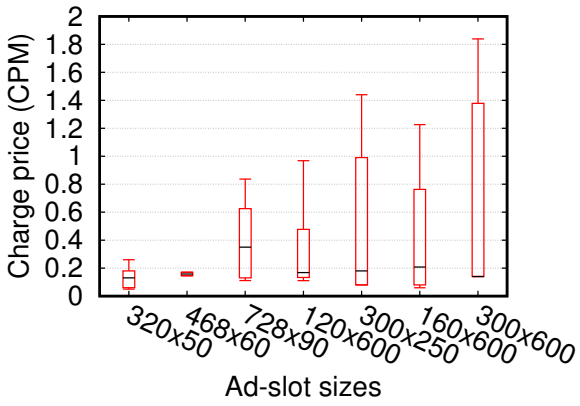


Figure 5.12:
Distribution of the charge prices per ad-slot size (sorted by area size).

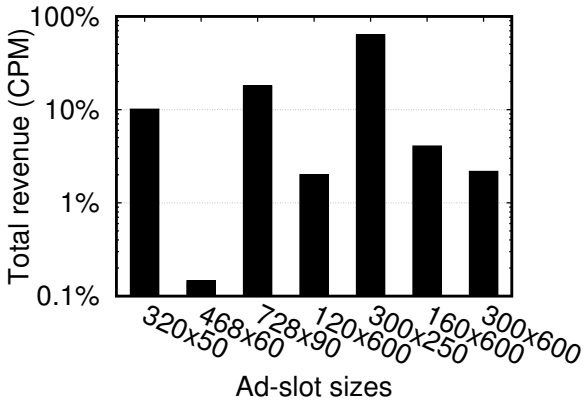


Figure 5.13:
Accumulated revenue per ad-slot size (sorted by area size).

as in web ads. In addition, (ii) studies [179] have shown that apps leak more personal data to advertisers, enabling them to deliver more targeted ads.

Ad-slot sizes. Some ad-entities carry in their nURLs a parameter with the size of the auctioned ad-slot. In Figure 5.11 we plot the popularity of each of the ad-slot sizes through time. It's interesting to see that 300x250 ad-slots (known as "MPUs" or "Medium Rectangles") dominate the dataset from May'15 on, taking the place of 320x50 ad-slots (known as "large mobile banners"). In fact, 300x250 ad-slots have more ad content available from advertisers, so they can increase earnings when both text and image ads are enabled. In addition, we see that the 728x90 ad-slot ("leaderboard" or "banner") is also popular. This ad-slot, usually placed at the top of a page, is seen by users immediately upon page load.

It is easy to anticipate, that the more space an ad-slot covers in the user's display, the higher the price will be. To verify this intuition, we isolated the traffic of an ad-entity (i.e. Turn [63]), which carries the ad-slot size in its nURLs along with the associated charge prices. Surprisingly, in Figure 5.12, we see that this intuition is wrong since the most expensive ad-slots for an advertiser are in fact, not the largest ones. In our dataset, we see that the two most expensive ad-slots are the MPU (300x250) and Monster MPU (300x600), with median prices of 0.47 and 0.39 CPM, respectively. However, from Figure 5.13, the increased popularity of MPU and Leaderboard ad-slots, allows them to accumulate 64.3% and 20.6% of the total RTB revenue of Turn in our dataset, respectively. Finally, it is worth to note that our results verify past resources [59, 88] regarding the more expensive ad-slots.

5.3.5 Summary

In summary, by analyzing the features extracted from our offline dataset, we find that a user's location (at city level) affects the median price that advertisers pay as well as its variability. However, such price differences are expected to be more evident at the country-level, as shown in [176]. In addition, the days and hours that a user may not be busy (Sundays), or may offer more attention (e.g., early mornings, Mondays) lead to higher charge prices. The type of user's device also affects the charge prices but in a rather contradicting fashion: though there are more Android devices, iOS-based devices draw higher median prices. As expected, the total revenue per category of user interest (through IABs) varies a lot, with some IABs being more costly than others. Finally, the display's real-estate occupied by an ad-slot does not correlate well with price. In fact, larger ad-slot sizes do not mean higher prices. As shown in the next section, these extracted features are used to plan effective ad-campaigns and model encrypted charge prices.

5.4 Charge Price Estimation

In order to create a model that estimates the encrypted prices detected on the user's browser and computes the total cost advertisers pay for her personal data, we need to have ground truth on charge encrypted prices. However, such dataset is not easy to acquire. One way to obtain this information is to collaborate directly with an ADX that sends such encrypted prices.[5] We assume this to be the rare case, since ADXs are generally unwilling to share such kind of data that may reveal bidding strategies and revenues.

In order to collect ground-truth data on encrypted prices, our system conducts small probing ad-campaigns on ADX(s)-DSP pairs that encrypt the winning prices. Such ad-campaigns can be designed and executed with the help of a single or few DSPs, with little overhead and a small budget of a few hundred dollars. In addition, they can be optimized by using a specific set of experimental setups, which cover all possible scenarios from the small parameter vector S to be kept short, efficient and cheap. Given that the prices do not change drastically over time, these campaigns can be executed every few months to collect probing data for *time-shift correction* and increased coverage of more ADXs. Besides, they can be automated and re-launched as frequently as needed, e.g., every few months or when the detected cleartext prices deviate from historical data. Having such campaigns launched from a specific location allows for more accurate and cost efficient price modeling that can be shared across all participating platform users in the same area or country.

We envision that such campaigns can be crowd-funded (like Tor Project [226], Wikipedia, WiGLE [23], etc.), thus, contributing to an independent and sustainable platform that can

[5] We considered top ADXs for encrypted prices (DoubleClick, OpenX, RubiconProject, PulsePoint), and ADXs for cleartext prices like MoPub (top mobile ADX).

scale better across users, countries, and ADXs covered. One may argue that these probing campaigns could pollute users' browsing with non-useful ad impressions. Thus, they need to comply with the current standards, and if possible, consider an actual product or service. Of course, ADXs could in principle fight back and try to identify and block such campaigns, but their huge clientele combined with the low volume of such campaigns makes the detection very difficult. Next, we describe the effort to select a subset of core features important for price modeling (§ 5.4.1) and how they allow us to design efficient and effective ad-campaigns (§ 5.4.2). Then, we provide an analysis of the data collected by two such campaigns (§ 5.4.3) and we describe the model that estimates encrypted prices, which can be used by end-users (§ 5.4.4).

5.4.1 Dimensionality reduction of features

The cost of testing all possible combinations of parameters and their values from the available feature set F (with one probing ad-campaign each), would constitute the budget for the ad-campaigns impossible (1000s of setups x 10s Euros/setup). Therefore, to perform ad-campaigns that are both effective and cost efficient, we need to select a subset of features $S \subseteq F$ that best describe the RTB prices found in weblogs such as the historic dataset D. This subset of features should explain as much of the variability of prices as possible. Assuming both encrypted and cleartext prices are affected by the same set of important features, this set should be small. The fewer features we select as important, the smaller the cost of running ad-campaigns to collect representative RTB prices using these features (e.g., 10s-100s of setups).

To achieve this selection, we performed dimensionality reduction using all the available features (288) described in Section 7.2 and Table 5.3, using the cleartext prices as the target variable for optimization. Some of these features are dense, i.e., they have an actual value in each price (e.g., time of day, day of week, size of ad, etc.) and others are sparse (e.g., interest categories of the user through time, publishers visited by user through time, etc.). First, for normalization, we applied a log transformation on the extracted cleartext prices found in D. Then, we applied a clustering of the prices into 4 classes, using an unsupervised equidistance model that finds the optimal splits between given prices using a method of leave-one-out estimate of the entropy of values in each class. Next, we filtered out features that did not vary at all (i.e., constants) or had very high variance (99%) (i.e., likely to be noise).

As a final step, dimensionality reduction (or feature selection) techniques such as PCA or Random Forests (RF) can be used [126]. We chose the RF model[6] because it takes into account the target variable (cleartext price), it can be trained quickly on large datasets, it maintains interpretability of features and generally does not overfit the given data. In case the availability of cleartext prices is limited, the reduction step to identify important

[6] An ensemble of decision trees built using a random subset from the available features.

features to be used in ad-campaigns could be hindered. To mitigate this, the PME can use intermediate techniques such as high correlation filters that do not require a target variable, to eliminate features carrying similar information.

We trained various RF models using subsets of semantically related features from the available feature set and the best features from each subset were selected based on their power to describe the cleartext price distribution. In summary, we grouped features in the following sets: A) time, B) http-related, C) advertisement-related, D) DSP-related, E) publisher/host interests, F) user http statistics (historical), G) user interests (historical), and H) user locations (historical). We also tried selecting representative features out of each set to create minimal combinations covering all aspects of the http-available information.

In total, we tried tens of feature subsets and combinations and evaluated them using standard machine learning metrics such as precision, recall, weighted area under the receiver operating characteristic curve (AUCROC) and out-of-bag error. Dimensionality reduction could, in principle, lead to loss of accuracy in the effort to explain price classes. However, our experimentation lead to a small subset of features with minimal loss of precision ($< 2\%$) and recall ($< 6\%$). In fact, we conclude that an optimal subset that performs very well and is small enough to allow cost efficient ad-campaigns is a set that *combines features from different groups*. In particular, also confirmed with an ad-campaign expert, we select the following features to be used for the probing ad-campaigns described next: S={application/web-browsing, device type, user location, time of day, day of week, ad format (size), type of website, ad-exchange}.

5.4.2 Ad-campaigns setup

Using the most important parameters extracted in set S, we construct various experimental setups $s \in S \subseteq F$ that can be used to deploy such ad-campaigns over a short period of time T' in selected ADXs to match top ADXs found in D. These setups combine different values of control variables that are important for an ad-campaign:<user location, web-interaction type, time of day, day of week, device type, OS, ad-size, ADX>. For example, an experimental setup could be this: <Madrid, app, 12am-9am, weekday, smartphone, iOS, 320x50, MoPub> (144 setups, Table 5.4). Clearly, using more features would increase coverage of different types of ads, but also the campaigns' cost. Instead, by running controlled ad-campaigns with a small feature set, we can receive ground truth data about encrypted prices, thereby allowing us to train a model for such prices, in a reasonable ad-campaign cost.

Campaigns with ADXs that deliver cleartext prices also allow us to compare prices in different times and compute shifts in the price distribution due to time passed between the collection of dataset D and present time. To compensate for the loss of information from cleartext prices becoming less abundant, additional features available in professional ad-campaign planners (as in FDVT [48]) could be used in the future to enhance the setups

Filter name	Range of values (type)
Cities	Madrid, Barcelona, Valencia, Seville
Type of interaction	Mobile in-app, Mobile web
Time of day	12am-9am, 9am-6pm, 6pm-12am
Day of week	Weekday, Weekend
Type of device	Smartphone, Tablet
Type of OS	iOS, Android
Ad-format (smartphone)	320x50, 300x250, 320x480 or 480x320
Ad-format (tablet)	728x90, 300x250, 768x1024 or 1024x768
Ad-exchange	MoPub, OpenX, Rubicon, DoubleClick, PulsePoint
Categories of targeting	all IABs possible

Table 5.4: Basic filters used in controlled ad-campaigns in Spain. In total, 144 experimental setups were attempted.

tested. With the results of these campaigns (in essence, charge prices for RTB ads that fulfil a given setup s), the PME can train a model to estimate the cost of new ads with a given setup s' close, or equal, to one tested, i.e. $s' \sim s \in S$.

Number of required ad-campaigns. An important decision in running probing campaigns is how many of them to launch, and with how many impressions in each one, in order to obtain a good approximation of the underlying distribution of prices. For this, we analyzed the ad-campaigns found for MoPub in D. We identified 280 such campaigns in 2015, with mean and standard deviation of charge price of $m = 1.84$ and $std = 2.15$ CPM, respectively. We use the process described in [116] and the next formulation to compute d, the expected error on the mean, assuming a suggested number of setups n, and ignoring the finite population correction adjustment (thus assuming a more conservative approximation of n) $d = \frac{Z_{\alpha/2} \times std}{\sqrt{n}}$, where Z is the z-score of normal distribution. Using the 144 setups proposed, we can approximate to more than 95% CI (i.e., α=0.05) the mean price of campaigns observed in the wild, assuming a margin of error 0.35 CPM. Also, considering the distribution of prices within the largest of ad-campaigns detected for MoPub with 1.8k impressions, we can approximate to 95% CI the mean price of a campaign, assuming an error 0.1 CPM and minimum of 185 impressions per campaign.

5.4.3 Ad-campaigns analysis

Using the above as guideline, we executed two different ad-campaigns to collect data on prices (Table 5.2). Our ad-campaigns advertised a real non-for-profit NGO in the area of data transparency, in an attempt to avoid polluting users with meaningless impressions, and trying to do something useful with the allocated budget.

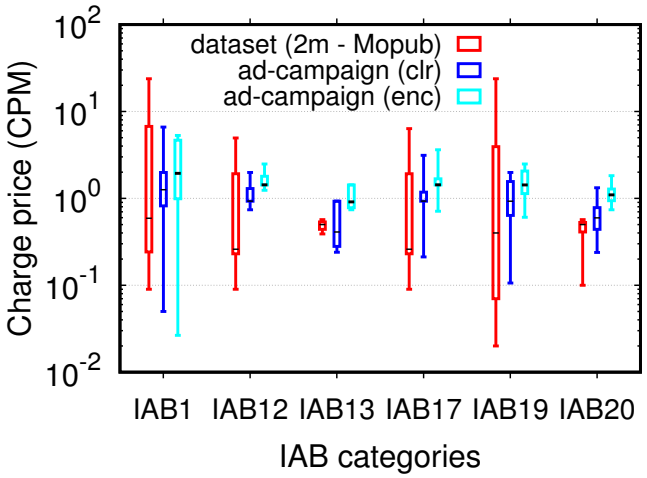


Figure 5.14: Comparison of CPM costs for the different IAB categories in our dataset and the 2 probing ad-campaigns.

Dataset collected. The first round ($A1$) was executed for 2 weeks in May 2016 and utilized the 4 ADXs mentioned earlier (also found in D) that encrypt price notifications and targeted publishers of many IAB categories. The second round ($A2$) was executed with the same experimental setups as $A1$ during June 2016, but in this case the DSP was instructed to use only MoPub, while still targeting similar IAB categories of publishers. These constraints allowed us to directly compare encrypted with cleartext prices in the same period, and time-shift all prices detected in D from 2015 to 2016.

In both campaigns, the DSP was given an upper bound on the bidding CPM price to safeguard that the allocated budget will not be consumed quickly. Because studying the effects of retargeting is beyond the scope of this study, we did not ask the DSP to perform such campaigns. However, the DSP was instructed to bid in a dynamic manner, as low or high as needed to get the minimum of impressions delivered for the various experimental setups we requested. We plan to investigate the effects of retargeting in a separate and dedicated future study. Overall, we managed to receive across all setups, over $600k$ impressions displayed with encrypted price notifications to more than 200 publishers, and over $300k$ impressions with cleartext price notifications to more than 300 publishers, reaching audiences of 6 IAB categories common to both notification types.

Cost paid vs. IAB category. In Figure 5.14, we compare the overlapping IAB categories of the RTB impressions we took from (i) the set of encrypted prices from the ad-campaign on four ADXs in $A1$, (ii) the set of cleartext prices from the ad-campaign on MoPub ($A2$), (iii) the 2 months MoPub subset of D. Note that in some cases, the results from D vary more than in the ad-campaigns. This is to be expected, as the dataset includes prices from

numerous DSP-ADX pairs for many ad-campaigns running in parallel in the duration of a year, whereas our two ad-campaigns are more targeted to specific DSP-ADX pairs.

Regarding the cleartext prices of different IAB categories, although the median prices are usually in the same order of magnitude, they are higher in the case of the recent ad-campaign contrary to the 2 month dataset. We believe that this difference is due to the time shift between the dataset collected in 2015 and the ad campaigns performed in 2016. In addition, we see that the median price is always higher in case of encrypted prices ($A1$), compared to the cleartext prices of the second ad-campaign ($A2$) and dataset D.

5.4.4 Encrypted price modeling

Using the ground truth data collected from the first round of ad-campaigns (encrypted prices) with various parameters within the subset of features S, we trained a machine learning classifier to predict values of encrypted prices. We note that given the problem of modeling real values, we first applied regression models with different combinations of dependent variables (S). However, the high variability of charge prices lead to low performance (high error) of the regression models. Therefore, we proceeded to split the prices into groups for classification. As a first step, we performed similar preprocessing for the encrypted prices as we did earlier for the cleartext prices (normalization and clustering to 4 classes of well balanced groups). Next, we trained a RF model to predict the class of an encrypted price, based on the available parameters S. For the training and testing, we applied 10-fold cross validation, and averaged results over 10 runs. Using features such as city of user, day of week and the time the ad was delivered, ad size, mobile OS of the user's device, IAB category of the publisher, ADX used and device type, our classifier can achieve a very good performance: TP=82.9%, FP=6.8%, Precision=83.5%, Recall=82.9%, 0.964 AUCROC. These scores are weighted averages across all classes, with no class performing worse than 5% from the average. We repeated this process with more price classes (i.e., 5-10 groups) for higher granularity of price prediction, but the results with 4 classes outperformed them.

When the exact publisher used is also taken into account in the model, the performance of the classifier increases to 95% accuracy, and 0.99 AUCROC. However, this is classic overfitting and we should caution that the publishers used in the ad-campaigns are just a subset of the thousands of possible publishers that can be found in real weblogs. Therefore, we chose to use the model with the IAB category but without the exact publisher as part of its input features. Next, this model was used for the estimation of the encrypted prices of nURLs found in the weblogs of each user in D, given the matching parameter values from $S \subseteq F$.

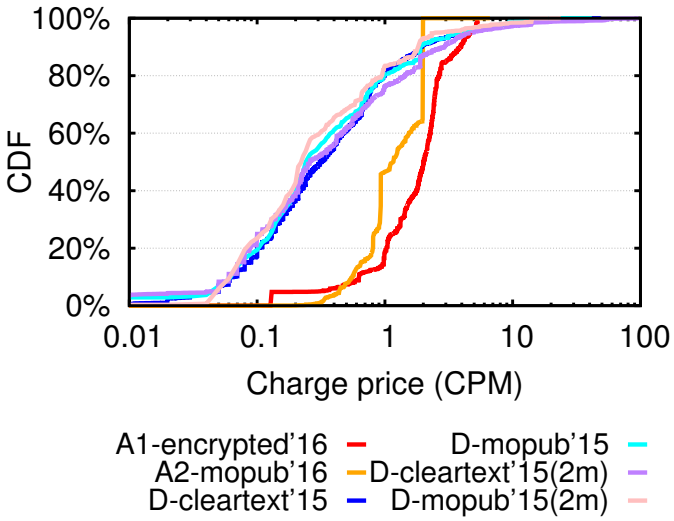


Figure 5.15: Comparison of price distributions between cleartext and encrypted, for different time periods and datasets (D vs. $A1$ and $A2$).

5.5 User Cost for Advertisers

The previous sections allowed us to: (1) bootstrap our price modeling engine from existing user weblogs, so that we find the important features describing well the observed RTB cleartext prices, (2) using these important features, run probing ad-campaigns with ADXs that send encrypted price notifications, so that we collect ground truth on such prices from performance reports delivered to us, (3) using such ground truth, train a machine learning model to estimate the price of new RTB notifications sent in encrypted form. We are now ready to study the overall cost advertisers paid for each of the users in our dataset D, who received cleartext and/or encrypted prices in nURLs of delivered ads.

5.5.1 Encrypted vs. cleartext price distributions

The work in [176] assumed that encrypted prices follow the same distribution with cleartext prices. To examine the validity of such assumption, we plot the distributions of both encrypted and cleartext charge prices we got from the two ad-campaigns we performed. Interestingly, from Figure 5.15, the distribution of *encrypted prices* in $A1$ is distinctly different and of *higher median value* ($\sim$1.7$\times$) *than cleartext prices* of $A2$.

In addition, we study the distributions between different time periods and ADXs to extract important lessons. First, we see that the cleartext price distribution of MoPub (2015) is similar to all ADXs sending cleartext prices, either when considering a 2 month period or a full year. Hence, we can study MoPub as a representative example and extrapolate lessons

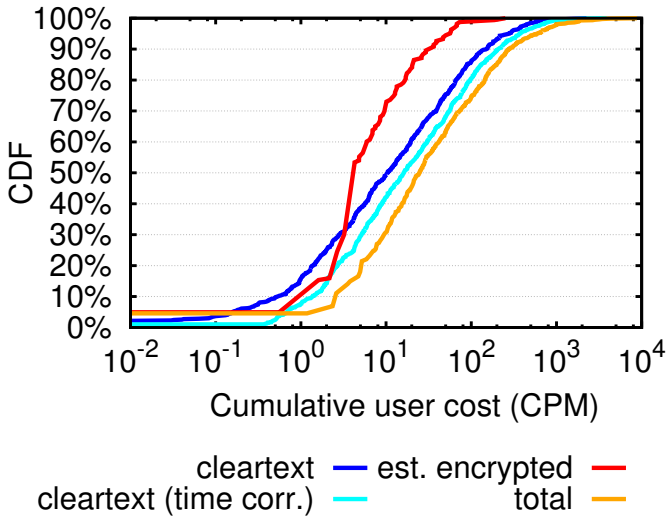


Figure 5.16: Cumulative CPM paid per user in our year long dataset.

for the rest of the ADXs that send cleartext prices. Second, the distribution of cleartext prices from $A2$ (MoPub) are of higher median value and can be used to establish the price shift due to time difference between the time T the dataset was collected, and T' when the campaigns were executed. In reality, this price shift can be detected evenly across multiple probing ad-campaigns (e.g., once per quarter of year).

5.5.2 How much do advertisers pay to reach a user?

Equipped with our presented methodology for estimating encrypted prices, we are now ready to respond to our motivating question. Specifically, we utilize our method and compute the overall cost advertisers paid for each user in the dataset D, i.e., across a whole year of mobile web transactions. We also apply a time-correction coefficient on the cleartext prices using the prices from the second round of ad-campaigns. This allows us to consider the increase in cleartext prices due to time difference from the weblog collection (2015) and the ad-campaigns execution (2016).

Figure 5.16 presents these cumulative costs in the form of CDFs of the price distributions. As expected, we observe that the cumulative cost due to encrypted prices is still not surpassing the cleartext, since the latter is still the dominant price delivery mechanism in mobile RTB. We also note that some users are more costly than others. Specifically, the median user costs $\sim$25 CPM, and up to 73% of the users cost $<$ 100 CPM through the year for the mobile ad ecosystem in the given dataset. This means that the ad-ecosystem reaches the average user very cheaply and multiple times below what users estimate this cost to be (e.g.,10s of dollars [31]).

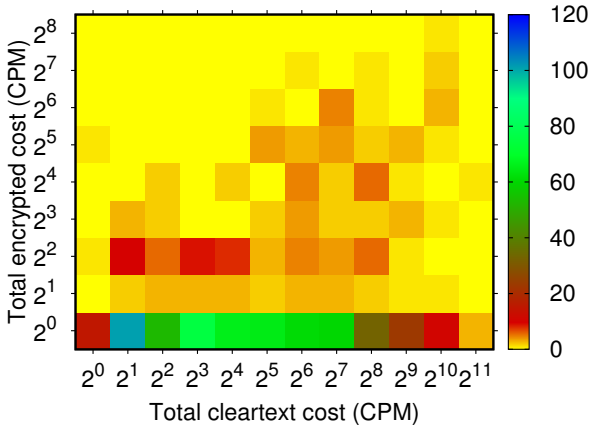


Figure 5.17:
Total cleartext vs. total estimated encrypted cost of each user in D (color indicates number of users).

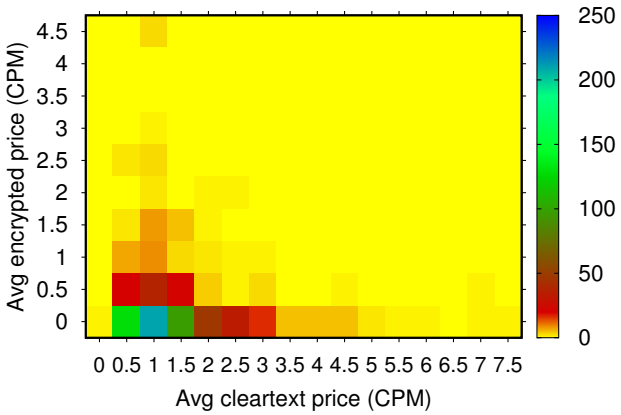


Figure 5.18:
Average cleartext vs. average estimated encrypted price per impression of each user in D.

On the other hand, for $\sim 2\%$ of users, the advertising ecosystem spent 1000-10000 CPM for the same time period. Finally, about 60% of users had an increased average cumulative cost of $\sim 55\%$ on top of their cleartext cost, due to the estimated encrypted prices. These users had a median of 14.3 CPM added to their total cost, with some extreme cases of 1000-5000 CPM.

In the previous result, we compared the distributions of encrypted and cleartext prices, while disregarding the targeted user. In order to identify if the cost paid through encrypted prices is the same with cleartext for a specific user, we compare for each user the total costs in Figure 5.17 and average cost per impression in Figure 5.18, for each type of price. We observe that a significant portion of users ($\sim$20-25%) cost similarly for ads embedded with encrypted or cleartext prices. As expected, due to the current majority of cleartext prices in the mobile ad market, a large portion of users ($\sim$75%) have higher cumulative cost from cleartext than encrypted prices. However, a small portion ($\sim$2%) costs more (2-32$\times$) in encrypted than in cleartext form, because they were delivered mostly ads with encrypted prices. When we normalize the cumulative ad cost of user per impression delivered (Figure 5.18), we find that for small prices of $\leq$3 CPM/impression, cleartext is more dominant across users. We also find a small portion ($\sim$2%) of users who cost up to 5$\times$ more CPM/impression for the delivered ads in encrypted than in cleartext form. We anticipate this portion to increase as the encrypted notification becomes the dominant delivery of RTB prices in mobile.

5.5.3 Summary

By studying the overall RTB advertising cost for users in our dataset, and distinguishing the encrypted from the cleartext prices, we found that the basic assumption of related work [176] that encrypted and cleartext prices are similar, is not valid (encrypted prices are around 1.7× higher). Furthermore, advertisers, based on users' personal data, paid ∼25 CPM for delivering ads to an average user, and less than ∼100 CPM for delivering ads to 3/4 of users during a year. We also identified a small portion of outlier users (∼2%) who cost 10-100× more to the ad-ecosystem than the average user, and a similar portion that costs up to 32× more in encrypted than cleartext prices, even though encrypted prices are only a quarter of the mobile RTB ecosystem.

Validation. As an effort to validate our methodology, we can extrapolate how much users cost for the ad-ecosystem and if this estimation compares with current market numbers. For this extrapolation, we make some assumptions on how our dataset represents the overall ecosystem of users and advertisers. In particular, we assume that our average mobile user, whose annual ad-cost is in the 8-102 CPM range (25th-75th perc.), has: (1) performed 2.65 hours online daily, which is ∼83% of the average daily mobile internet usage, when considering average tablet and other mobile device usage [138], (2) performed internet activity from both mobile and laptop/desktop devices, the former traffic type being ∼51% of total internet time [246], (3) received ads in a similar fashion in both HTTP and HTTPS, the former being ∼40% of the total traffic delivered to a user [66, 207], (4) received ads over RTB, which has an overhead management and intermediaries cost of ∼55% [192], and (5) received ads in a similar fashion over RTB and traditional and other online advertising, the former being ∼20% of the total online advertising [111]. Considering these factors, the overall average user ad-cost (25th-75th perc.) would be in the range of $0.54-6.85, which is in the order of magnitude reported by major online advertising platforms such as Twitter (owner of MoPub, ARPU: $7-8 [99]) and Facebook (ARPU: $14-17 [43]) during the period 2015-2016.

5.6 Discussion

5.6.1 Limitations

Our approach, through YourAdValue plugin (a screenshot of which can be seen in Figure 5.19), monitors the charge prices for each auctioned ad-slot. However, there are several cost models in digital ad-buying. For example, Cost-Per-Impression is where the advertiser pays when an impression is rendered, and Cost-Per-Click is where the advertiser pays only if the impression is rendered and clicked, etc. Given that our study is based on passive measurements, we currently unable to determine the cost model of each auctioned ad-slot. Therefore, we assume all charge prices are under the Cost-Per-Impression model, thus computing the maximum cost advertisers pay for a user.

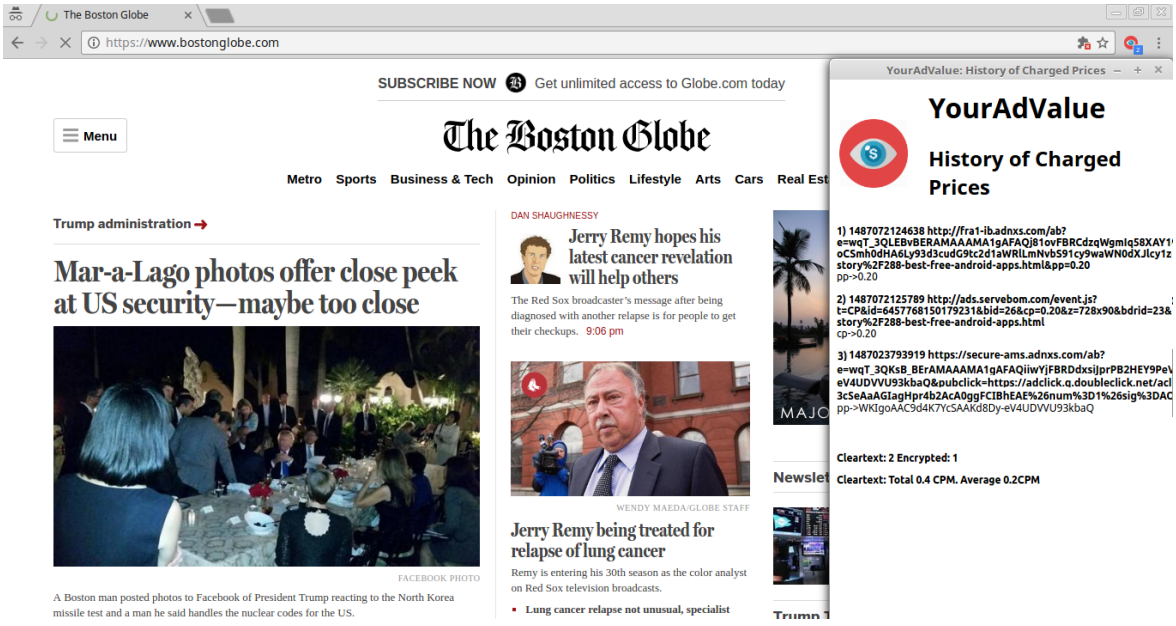


Figure 5.19: Preliminary implementation of YourAdValue Chrome extension in use.

5.6.2 Computing The financial worth of individuals

Via our methodology, users can estimate, at real time, the cost advertisers pay to reach them. However, this work's important technical contribution, i.e., *how to compute the financial worth of individuals* with a passive measurement method has several applications. Our methodology could provide more transparency on what each type of the users' personal data is worth, and allow users to take advantage of, and (re)negotiate their online value with data hub companies who are interested in investing and innovating in the area of targeted advertising. Also, such companies can use our methodology to assess the costs implied in this area, how to allocated appropriate resources and, even, estimating bidding strategies of competitors. In addition, regulators and policy makers could provide guidelines and laws to users and companies for containing the leakage of users' personal data. Finally, tax auditors could estimate ad-companies' revenues, and detect discrepancies from their tax declarations in an independent and transparent way.

Chapter 6
Costs of Advertising on Users and Advertisers

The vast majority of the content providers nowadays offer their websites or their sophisticated services free-of-charge (e.g. Google Docs, Facebook, Twitter, Gmail) in exchange for allowing third parties to access and display advertisements to their users. As presented in this dissertation, in this model, advertisers buy available ad-slots in the user's display in an automated fashion based on how well the advertised product matches the profile of the user. As a consequence, when a user visits a website, each of the available ad-slots is auctioned, and advertisers decide if they will bid and how much, based on the information (interests, income, gender etc.) they have about the current user. Following this process, a careful reader identifies 3 key role players: (i) the website provider who earns money from advertisers through the auctions, (ii) the advertisers that pay to promote, and eventually sell their products by delivering effective advertisements to the proper eyeballs, and (iii) the user that receives from the website the content of his interest, for free. Seemingly, everyone benefits from this model. *But are the users indeed receiving the content they want free of charge?*

Contrary to the traditional advertising (i.e. in newspapers, TV, radio), in the digital world, it is not only the advertiser that pays the cost of advertisement delivery, but the user as well! Indeed, it is the user's data plan that is being charged to download the additional ad-related KBytes. To make matters worse, there are several other bytes the user downloads regarding analytics and user tracking, totally unassociated with the actual content of the visited website. Of course, the cost is not only monetary, since the privacy loss of the above operation has proven significant [132].

In this chapter, we examine the hidden costs of mobile advertising for both the transmitter (advertiser), and the receiver (user) of the advertisments. In fact, we compare them for the same user profiles and investigate how fairly they are shared among the two sides. Our motivation is to enhance transparency regarding the overall costs of online advertising, and increase awareness of users regarding hidden costs they pay while using ad-supported online services.

Past works in the area already attempted to reveal the hidden costs of advertising in the mobile ecosystem. For example, Gui et al. [103] analyze free and paid version of apps to compare the advertising costs from the developers' side. They actively analyzed mobile apps to measure costs related to memory, power consumption and CPU usage. Similar to the study of Gao et al. [80], they compared these costs with the users' feedback from app reviews.

This work is the first to our knowledge that measures the hidden cost of advertising when mobile users browse the web. Contrary to the above inspiring approaches, our more user-centric study attempts to examine these costs, not from a developer perspective but from the side of the end-user. Towards this goal, we design a methodology and we implement it in OpenDAMP: a tool to estimate the costs of advertising for both advertisers and users, by passively analyzing a dataset of user HTTP traffic. We collected a dataset consisting of mobile traffic from 1270 volunteering users that spans over an entire year, and use OpenDAMP to analyze it. Finally, we compare the costs of both sides to assess how fair they are shared across the two ends.

In this chapter, we design a methodology to measure the costs a user pays when receiving ad-related traffic. These costs may be either directly quantifiable (e.g., requests, bytes, energy), or qualitative such as loss of privacy. In addition, and building on our previous Chapter [**?**], our methodology estimates the costs advertisers pay to display each of the advertisements a user receives through the contemporary programmatic RTB auctions [53]. We implement our methodology in OpenDAMP (open Digital Advertising Measurement Platform): a framework for passive weblog analysis oriented to digital advertising. OpenDAMP can analyze user HTTP traffic and detect ID sharing incidents among third parties (known as Cookie Synchronizations). In addition, by incorporating information from external resources and blacklists, OpenDAMP can classify the traffic based on the content the domains deliver, and extract metadata and charge prices from RTB ad-auctions. To assess the effectiveness of our methodology, we collected a year-long dataset with mobile browsing traffic from 1270 volunteering users. Our analysis shows that the costs advertisers and users pay are largely unbalanced, In fact, users pay $\sim$3 times more through their data plan to download ads, than what the advertisers pay to deliver them to these users. Furthermore, the majority of users sustains a significant loss of privacy to receive these personalized advertisements.

6.1 Cost Analysis with OpenDAMP

In this study, we measure the hidden costs of advertising for users, by passively monitoring their browsing traffic, while taking into account the advertisers' view. For our analysis, as previously, we use the same year-long dataset with weblogs from volunteering users.

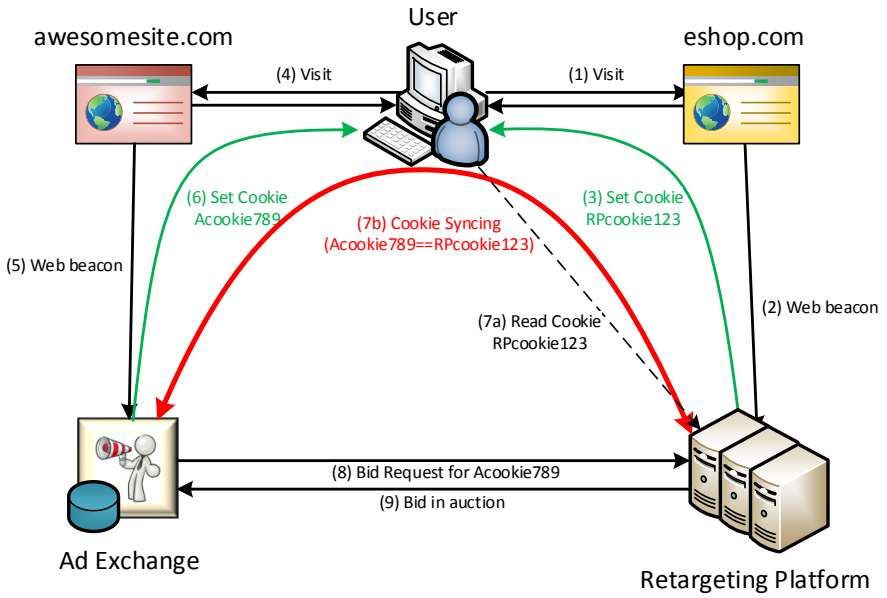


Figure 6.1: An example use of CSync in programmatic advertising. Advertisers can track and re-identify users while they surf the web.

6.1.1 Quantitative & Qualitative User Costs

Besides the quantitative costs a user may pay to receive advertisements, such as the additional network usage, there is also an important, qualitative cost for the user: the loss of privacy. It is well known that companies comprising the online advertising ecosystem collect several types of user data: location, behavior, preferences, interests, etc. Such data are used by these companies to deliver more personalized advertisements to online users.

Cookie Synchronization: In order for all this abundance of user data to be useful for the companies, there must be a *matching* process of all the userIDs that the third parties have assigned to the same user. Thus, it is easy to anticipate, that the synchronized userIDs of Cookie Synchronization is of paramount importance for tracking entities in order to (a) re-identify users across the different websites they browse, but also (b) participate in user data auctions and marketplaces [6], thus increasing the wealth and detail of the information they know about each user. Thereby, in this study, we use CSync as a proxy for privacy loss. In fact, assuming 1 CSync leaks 1 userID, we use performed CSyncs as a metric to quantify and compare users' privacy and anonymity loss in mobile web.

Cookie Synchronization & Personalized Advertising: Besides user tracking, CSync, is also a core component of personalized advertising, which allows advertisers to re-identify

(or retarget) users as they browse the web, and deliver them the proper ad. An example, as seen in Figure 6.1, is the following. Let's assume a publisher, e.g., a shoes-selling e-shop E, which collaborates with the Re-targeting Platform RP to improve the efficiency of its marketing strategy. In addition, let's also assume an Ad-Exchange A, with which RP is also collaborating. RP needs to be aware of the users visiting E at any time, as well as their movements: what other pages they visit, when and for how long. Therefore, RP asks E to tag each page of its website by embedding a *Web Beacon* [106, 149] pointing to the RP in each one of them: a 1-by-1 pixel image (also known as Pixel Tag or Web Bug). This way, the user will send this web beacon every time she browses the page, allowing RP to know her moves and also set a cookie (e.g. *UID_RP123*) on her side. Now, let's assume a user U who adds a pair of shoes in her shopping cart in E, but never makes it to the checkout. E would clearly want to re-target U and serve an ad, directing U back to E to try and finish the sale.

After a while, U surfs around the web, and lands on *awesomesite.com*, which is using A to monetize their ad inventory. Using a similar web beacon, *awesomesite.com* allows A to (i) learn about the visit of U and (ii) set a cookie *UID_A789*. Before A calls an auction for the available ad-slots of *awesomesite.com*, it trigger a Cookie Synchronization on U's browser to share ID *UID_A789* with it's associated bidders (including RP). After this synchronization, RP can re-identify the user by matching the two aliases: *UID_A789* == *UID_RP123* and will bid accordingly to place a retargeting ad about the shoes of E that U intended to buy.

6.1.2 The OpenDAMP framework

To analyze our traffic, we built *OpenDAMP* (open Digital Advertising Measurement Platform): a framework for weblog analysis oriented to digital advertising. OpenDAMP parses HTTP traffic and classifies it based on the content delivered by the domains. In addition, using metadata from public crowd-based resources[1], it can further categorize advertisers based on the products they provide (DMPs, ad platforms, DSPs, SSPs, etc.). Finally, leveraging the User-Agent field of the HTTP requests, OpenDAMP can identify the operating system of the device (iPhone, WindowsPhone, Android) based on the set hardware characteristics.

As we noted above, using OpenDAMP, we are able to classify the traffic into 5 categories (i) *Advertising*, (ii) *Analytics*, (iii) *Social*, which includes social widgets and plugins and (iv) *3rd party Content*, which includes content originated from 3rd party providers (for example content from CDNs, embedded Instagram photos, Captchas, blog comment hosting services like Disqus and many more) and (v) *Other*, which includes the rest of the content that is the useful content the user is actually interested in. To do such classification, OpenDAMP uses a popular browser adblock extension's blacklist [55]. This blacklist groups the different domains that belong to the same company (e.g. Google groups Doubleclick, AdMob and Adscape). It includes:

[1] Business Software and Services reviews: g2crowd.com

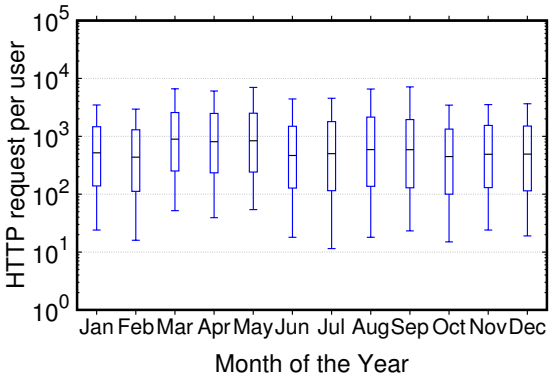


Figure 6.2:
HTTP requests produced per user, across the year. Users create a relatively stable HTTP traffic, increased during holiday periods.

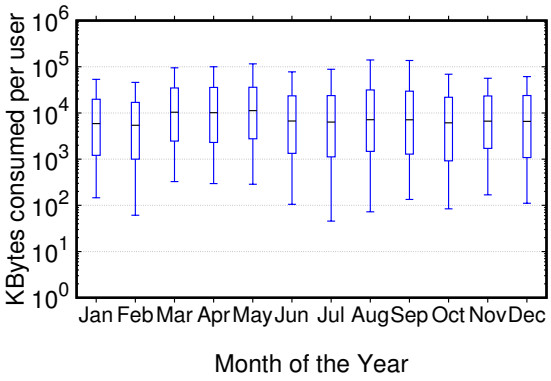


Figure 6.3:
Volume of total consumed KBytes per user, across the year. Users consume an average of 5.9 GBytes per month.

1) Advertising: 770 companies resulting in 1395 domains
2) Analytics: 150 companies resulting in 239 domains
3) Content: 111 companies resulting in 522 domains
4) Social: 17 companies resulting in 58 domains

To detect Cookie Synchronization processes in our dataset, we implement in OpenDAMPthe heuristics-based Cookie Synchronization detection algorithm as described earlier in Section 4.2.

6.2 The view of the User

In this section, we analyze the costs that users sustain to receive advertisements while browsing the web. In our dataset, we separate the web traffic of each user and we compose user timelines that describe the traffic characteristics of each one of them. The timelines include HTTP requests received, Bytes transferred, files received, impressions received etc.

All the above constitute quantifiable properties that we can measure to extract the final cost a user paid. However, while browsing the web, users also leak information that is useful for the advertising ecosystem and this is another cost of advertising. In this section, we also attempt to quantify this cost besides its qualitative properties.

6.2.1 Network resources consumption

How many HTTP requests are due to ads?
First, we conduct a brief analysis to explore the contents of the collected dataset regarding the

network traffic of the users. In Figure 6.2 and Figure 6.3, we see respectively the distribution of the overall HTTP requests produced and the KBytes consumed per user through the year in our dataset (percentiles: 10th, 25th, 50th, 75th, 90th). As we see, the median user has a relatively steady production of network traffic, thus consuming per month around 5891 KBytes, on average. In addition, we see an expected monthly behavior, where there is an increase of the produced web traffic during months that include long holidays (spring break, summer holidays etc.). A diurnal behavior can be also seen when measuring the time of day the traffic was produced. As shown in Figure 6.4, users produce web traffic in their mobile devices mostly from morning till early afternoon, and this repeats throughout the week.

In Figure 6.5, we use OpenDAMP to classify the HTTP requests the average user fetches, based on the content served by their domain, across the whole year. Considering that 3rd party content is an essential (external) component of a website and its absence could break the provided functionality and degrade the experience of the user, we consider it as part of the actual content of the website. On the other hand, the Analytics category includes services which aim to monitor performance and behaviorally track the audience of a website. Thus, we see that the percentage of requests bringing to the user's browser the actual content they are interested in is steadily around 77% across the whole year, and the percentage of ad- and analytics- related percentage is as high as 19%, on average.

Next, in Figure 6.6, we investigate what are the different resources a user retrieves for these two content categories through the year. In this plot, we present the distribution of the users (percentiles: 10th, 25th, 50th, 75th, 90th). For the median user, most of the advertising HTTP requests are animated and static images and scripts, besides the expected volume of HTML. Also, in analytics, the largest amount of requests are monitoring scripts.

How much of the downloaded volume is related to ads?

The cost for all of the above (additional) resources the user downloads is translated to consumed Bytes. This is the most important metric that not only monetarily affects the user's data plan, but affects also the device's battery by keeping its CPU and network card on, in order to marshal the received content. From Figure 6.7, it is evident that the volume of bytes for the downloaded static advertising images and scripts reaches around 700 KBytes and 850 KBytes, respectively; the 90th percentile peaks at almost 10 MByte for each one. It is easy to observe in these two Figures (Figure 6.6 and Figure 6.7) the large amount and size of the scripts that both Advertising and Analytics related domains instruct the user's browser to run. Note that these scripts, and the additional CPU cycles they require, are unrelated with the actual content the user is interested in, and therefore constitute a clear additional overhead for the user.

If we have a deeper look in the HTTP requests and the volume of bytes they deliver, in Figure 6.8 we observe an increasing trend across the year, with the HTTP requests for ads requiring to transfer double the volume, on average (from 4KB to 8KB). Taken in conjunction

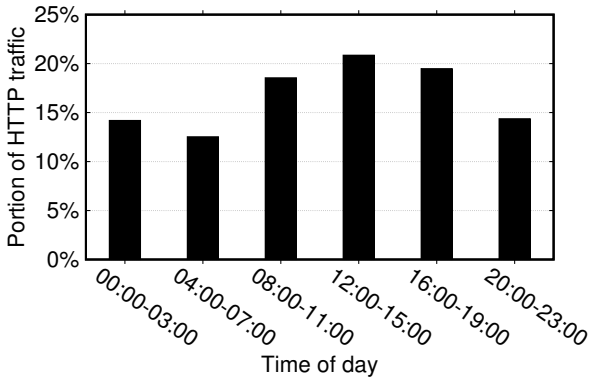


Figure 6.4:
Portion of HTTP requests produced across the day. As expected, users produce web traffic mostly from morning till early afternoon.

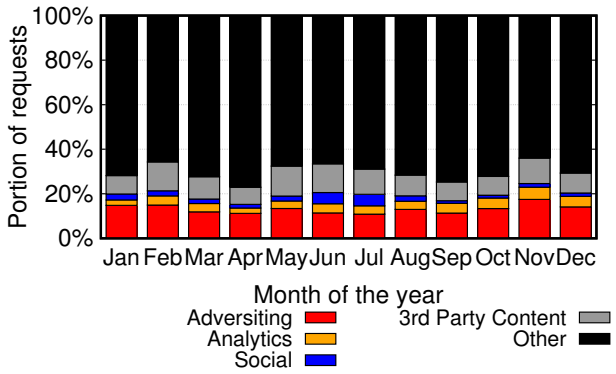


Figure 6.5:
Portion of HTTP requests per content category the average user fetches through the year. On average, 77% of the HTTP requests is associated with the content the user is actually interested in.

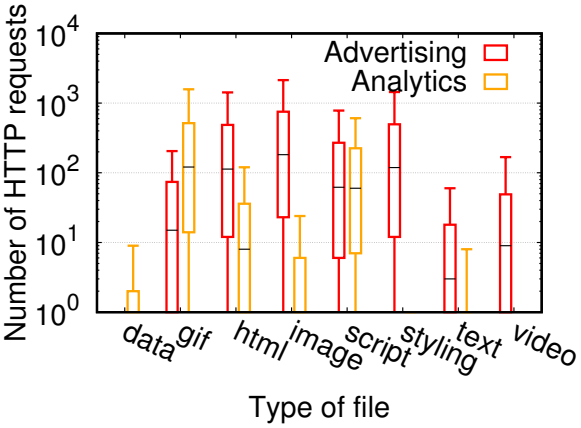


Figure 6.6:
HTTP requests received per user, per different resource type.

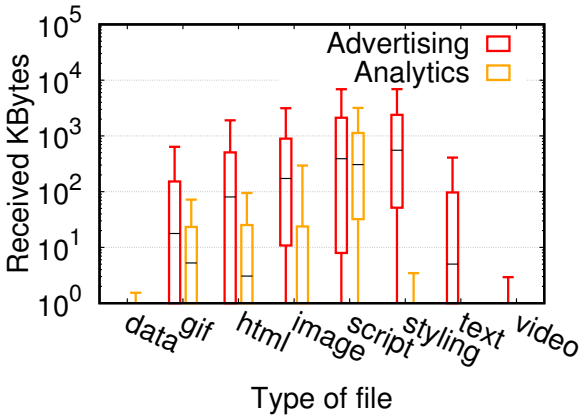


Figure 6.7:
Bytes received per user, per different resource type.

with Figure 6.5 which shows a steady portion of ad-related requests, delivering larger payloads in the same number of requests, although it may require more memory from the device, it gives the opportunity for the device to minimize the required latency to marshal/unmarshal each ad-related requests. However, we also suspect that advertisers take advantage of better mobile network speeds and device resources, as they become available through time. Consequently, they force each mobile device to download an ever-increasing amount of data displayed in the publishers' pages, at the users' expense.

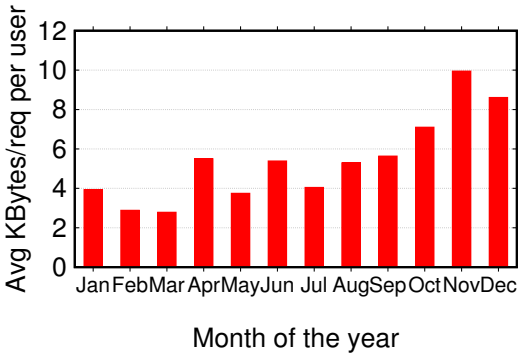


Figure 6.8:
KBytes per ad-related HTTP request per user, across the year.

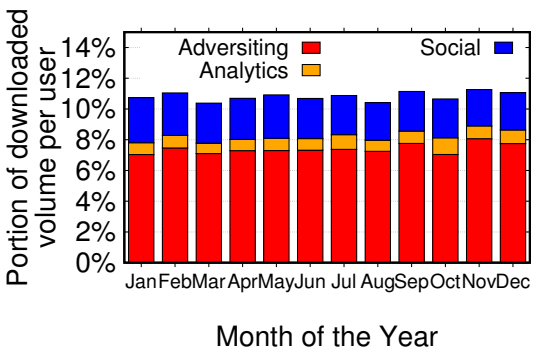


Figure 6.9:
Ad-related KBytes downloaded per user, through the year.

Finally, we measure the portion of the total downloaded volume per user that is associated to Advertising and Analytics. In Figure 6.9, we see that a user steadily downloads an average 8.2% of bytes (extra to the actual content they browse) across the year, which belongs solely to Advertising (7.3%) and Analytics (0.8%) related content. We see a small increase in the ad-related volume with previous studies (5 years ago) [234] measuring the same volume at 5.6%. If we also add the Social-related traffic, the total percentage of additional content the user has to download reaches as high as 11%, on average.

Using the results from [102, 168, 252], we also provide an estimation of the power the ad-related traffic consumes on the user side. Given the results in Figure 6.9, the network component of a mobile device alone consumes 7.98% more, due to the additional ad-related transmitted bytes, and 0.86% more, due to analytics-related bytes. This means that a mobile device, whose battery can sustain 10 hours of ad-free browsing, will last 9.2 hours due to the additional ad-related network volume received. In fact, and according to previous studies [103], if we also consider the energy consumption of the display, this cost may surpass 15%.

Unlimited data plans

Passively measuring the cost on the users' data plans, of course, comes with some limitations. First of all, there may be user devices connected to the Internet through WiFi. In addition, some ISPs recently offered *unlimited data plans*, providing a large volume of data (usually around 20 GB/month [107]) to their clients. Despite the current issues of such products (i.e., throttling [209], high prices (70-90$/month) [107], expensive Internet roaming), it is likely that in the future they will become cheap enough to become popular. Therefore, the respective monetary cost for users with unlimited data plans will become practically negligible. However, even in such cases, personalized advertisements do consume device resources (battery, network traffic, CPU, etc.), and still incur a high cost on user privacy and anonymity loss.

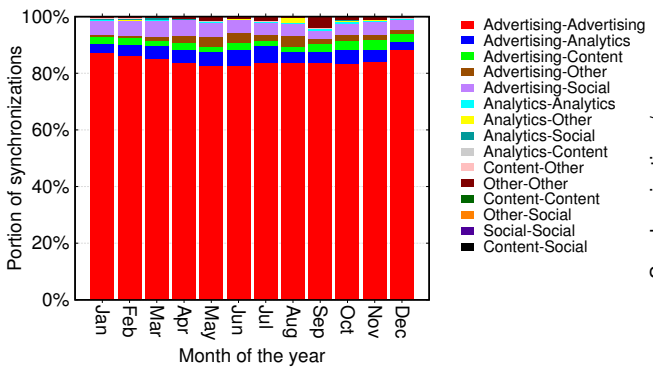


Figure 6.10:
Portion of CSyncs per content category pair, through the year.

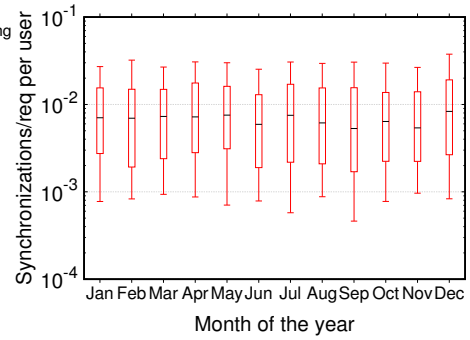


Figure 6.11:
Synchronizations per HTTP request users receive through the year. The median user is exposed to a steady number of CSyncs.

6.2.2 User privacy loss

What is the user's exposure to Cookie Synchronization?
By using OpenDAMP, we detect CSyncs in our dataset and we see that for users with regular activity on the web (> 10 HTTP requests per day), 97% of them were exposed to CSync at least once. Next, we separate and classify the pairs of entities that conduct CSync in our dataset through the year and in Figure 6.10 we show the portion of CSyncs performed by each type of pair. The majority (∼85%) of the CSync takes place within the different advertising entities, but there are also cases where advertising entities synchronize their userIDs with Social or Analytics related entities.

Next, we investigate if the synchronizations the users are exposed to change over time. Hence, we extract CSyncs per user, normalized by the user's total number of requests. In Figure 6.11, we plot these synchronizations across the year. The median user receives 1 synchronization per 140 HTTP requests, while the 90th percentile user is exposed to 1 synchronization per 50 requests! Considering the different userIDs that tracking entities may assign to a user, in Figure 6.12, we measure the number of unique userIDs that got synced per user. Evidently, a median user gets up to 63 different userIDs synced (at least once) through the year, and the 75th percentile user gets up to 195 of their userIDs synced.

How much do tracking entities know about a user?
Next, we measure the pervasiveness of the tracking entities. Specifically, in Figure 6.13, we measure the portion of the overall userIDs each (ad- and analytics- related) entity learned through CSync. Interestingly, ad and analytic entities follow similar distributions, and ap-

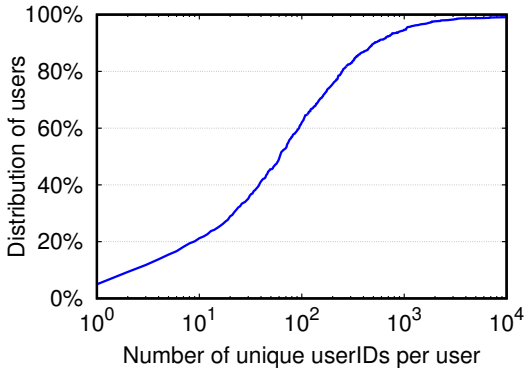


Figure 6.12:
Unique synced userIDs per user. The 50th (75th) percentile user gets up to 63 (195) unique IDs synced, at least once.

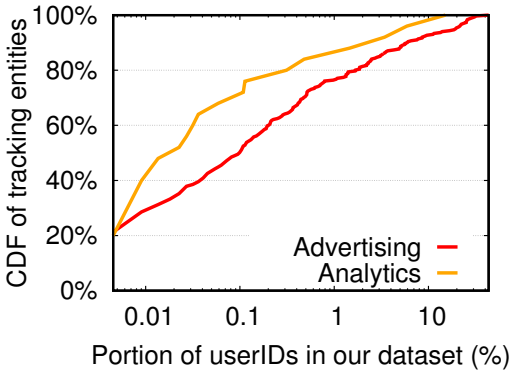


Figure 6.13:
Portion of the total userIDs in our dataset each tracking entity learned. Some entities have learned more than 10% of all userIDs.

parently, such entities tend to learn significant portions of userIDs. Therefore, although a median ad-related entity may learn around 0.03% of the overall userIDs, there is a portion of 5% of entities than learned more than 10%, and another 0.6% of entities that learned more than 25% of the overall userIDs in our dataset.

As we described earlier, CSync is a mechanism for trackers to increase the identifiability of a user in the web, by joining their assigned userIDs. In Figure 6.14, we plot the number of entities that gained access to the user's IDs. As we see, the median user loses up to 20% of her anonymity to 22 tracking entities and up to 40% to 3 tracking entities. Such an important leak enables a handful of entities to accurately re-identify the user on the web and construct a rich user profile by merging their collected data on the backend.

6.3 The view of the Advertiser

It is of no doubt that digital advertising moves towards a more personalized ad-delivery approach, where advertisements are matched to the interests of the individuals following a programmatic ad-buying model. The most popular one is the model of programmatic auctions of the Real-Time Bidding (RTB) [93], which has a five-year CAGR of 24% [243]. In RTB, ad-slots on the users' displays are being sold in auctions where the higher bidder delivers its impression.

More specifically, in RTB-based auctions, whenever a user visits a website with an available ad slot, an ad-request is sent to an Ad Exchange (ADX), which calls an auction and sends bid requests (along with user info) to ad-buyers (bidders). These bidders in RTB are usually Demand Side Platforms (DSPs), which are agencies that utilize sophisticated decision

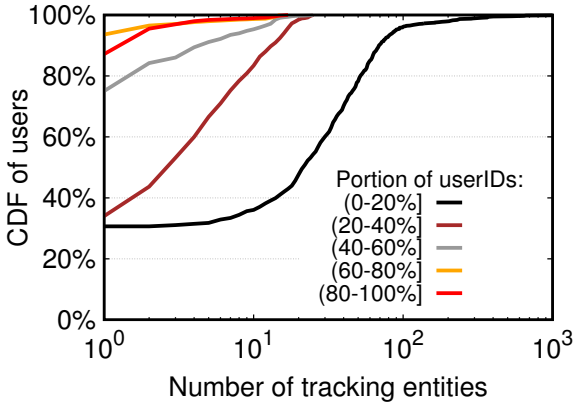


Figure 6.14:
Number of entities having access to a portion of a user's IDs. The median user loses up to 20% of its anonymity to 22 tracking entities.

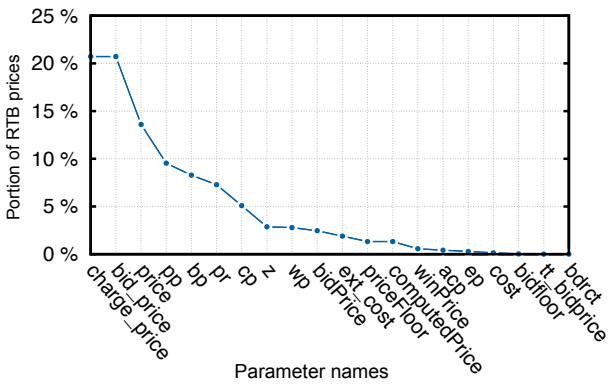


Figure 6.15:
Although there is an OpenRTB standard [110], every company follows its very own protocol with different parameter naming, making RTB price filtering a challenging task.

engines and aim to assist advertisers to decide at real time if they will bid at an auction and how much, based on the user info they receive and how close the advertised product is to the user's interests. The entire auction has a strict time constraint and usually takes 100 ms from the time that the user will visit the site till the winning impression is finally delivered.

In this study, we leverage mobile RTB as previously to assess the cost that advertisers pay, in order to deliver personalized ads to users. For this, we search for a specific step of the RTB where the ADX notifies, through the user's browser, the higher bidder about its win. Typically, this notification URL is parametrized with a keyword agreed between the two companies (ADX and DSP), and carries the RTB price to be paid by the winning DSP. The price can be cleartext or encrypted, as shown in two examples in Table 2.1.

Although the RTB protocol is well standardized by OpenRTB [110] since 2010, in Figure 6.15 we observe a large heterogeneity of keywords used to define the charge price. In fact, each ADX may use its very own parameter, making the RTB process less transparent, and more difficult for an external observer to detect and study the RTB parameters and values used.

In Figure 6.16, we present the RTB market share of each bidder in our dataset. As we can see, from the market share segmentation there is only a handful of big players winning the larger portion of auctions. Specifically, no more than 5 companies have won 67.7% of the overall RTB auctions. In addition, we see only 14 of the total number of bidders in our sample, winning a portion of auctions greater or equal to 1%.

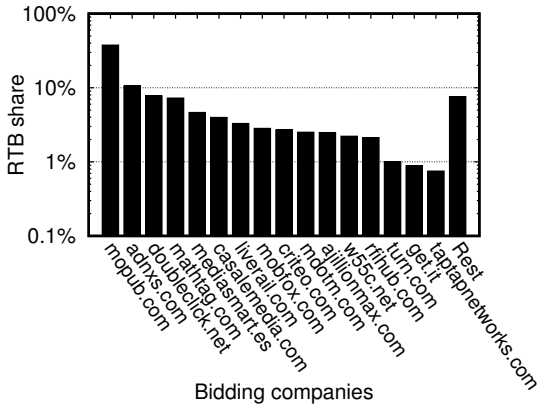


Figure 6.16:
The RTB market share of the different bidders in our dataset. As we see, the market share is mainly divided to a dozen of companies with the top 5 wining 67.7% of the RTB auctions.

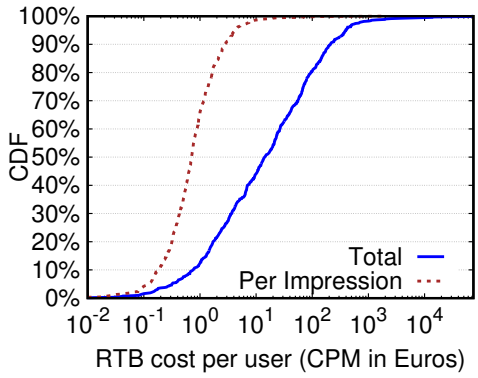


Figure 6.17:
Cost per user for advertisers to display ads across the year. The average cost per impression for the median user is 0.9 CPM. The total cost paid by advertisers for the median user is ∼22 CPM.

In Figure 6.17, we show the CDF of the total cost paid by advertisers to deliver and display ads to the mobile users of our dataset. These prices (in blue) represent what we have detected and computed as the total cost across the year for each user in our dataset, and expressed in CPM. As we can see, some users are orders of magnitude more costly to reach than the average user: advertisers paid for the 75th percentile user up to 100 CPM for the entire year, when they paid around 20 CPM for the median user.

In the same figure, we also plot the distribution of the costs per impression per user (in red). We see that an impression for the median user costs 0.9 CPM, but it is interesting to see that there are three classes of users: the users who are quite cheap to reach and are below average (<1CPM), the average users that can be reached with around 1 CPM, and the more expensive users (>1CPM) that advertisers paid up to 9 CPM per impression.

At this point, we must note that the above computed RTB charge prices regard only the value that a bidder paid for the specific ad-slot in a specific user's display. Commissions for possible intermediate agencies and platforms may appear, thus, increasing the actual cost that the advertised company may have paid.

6.4 Consolidating the two Views

Earlier, we showed how much advertisers paid to deliver ads to users, through various RTB ad-campaigns and companies. In this section, we use this RTB cost as a proxy for the monetary cost of the entire advertising process (e.g. user tracking, analytics and finally ad retrieval).

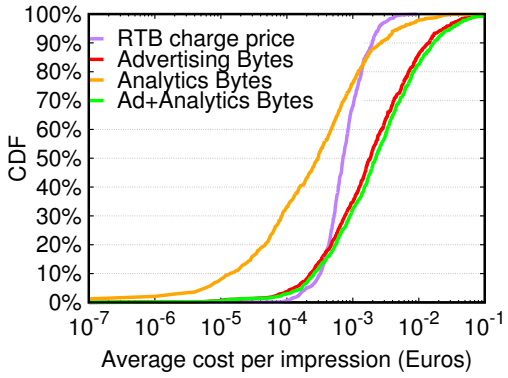


Figure 6.18:
CDF of the average cost on the users' data plan, and cost paid by advertisers to deliver personalized ads to the same users.

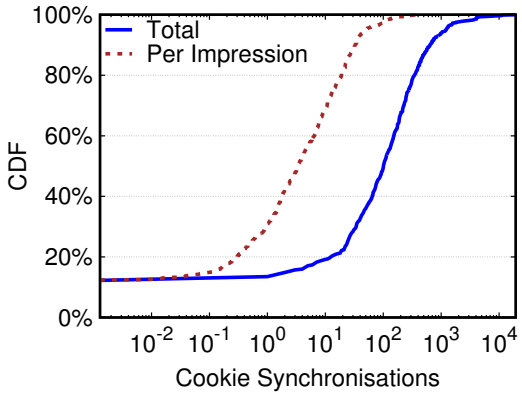


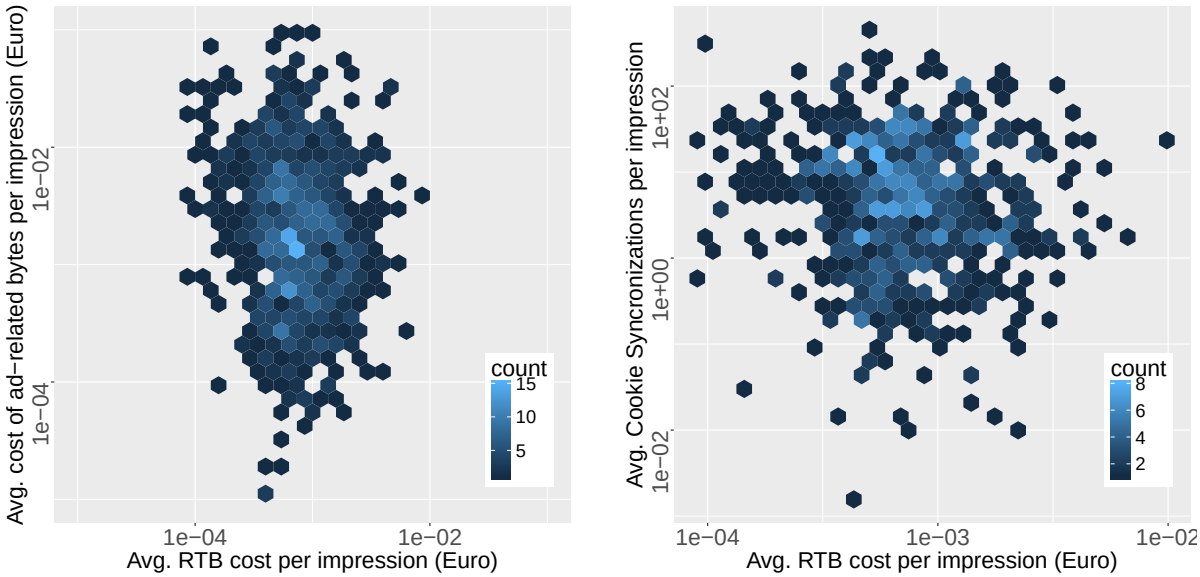
Figure 6.19:
CDF of the average CSyncs per impression retrieved per user, across the year.

We compare it with an estimated cost paid by the users to download these corresponding ads in their device. In particular, we use an estimation of the cost per byte that users paid in their data plans for the total bytes downloaded for these ads. We also look at the privacy cost of users via the CSync metric, and how that also compares with the advertisers' RTB cost.

6.4.1 Cost on data plan vs. Cost of RTB

For this comparison, we use currently available prices [9, 73], for various data plans in the country the users were located, while the dataset was collected. Using prices for 20 different data plans from 6 different ISPs and subsidiaries, we computed an average cost of Euros per Byte. Historically, the data plan prices have been dropping, thus, our estimation of the Byte cost can be considered a lower bound of the actual cost users paid during the data collection.

In Figure 6.18, we plot the CDF of average cost per impression paid by the two parties considered: (i) the end-users for Bytes consumed by their phones for downloading advertising and analytics requests, and (ii) the advertisers for ads they delivered to these devices through the RTB mechanism. These average scores reflect the traffic across the year. We make the following observations. Surprisingly, the cost on advertising bytes for the majority (about 80%) of users is higher than the RTB cost paid by the advertisers. Specifically, we see that the median user paid an average cost of 0.0022 Euros per ad for advertising and analytics bytes, whereas the median advertiser paid 0.00071 Euros per ad. This means that for each delivered ad impression, **users are charged 3 times more than advertisers** who benefit from the ad delivery!



(a) Cost on data plan vs. Cost of RTB

(b) Cost of CSync vs. Cost of RTB

Figure 6.20: Heatmaps of (a) average cost per impression for Bytes consumed by users in advertising requests, (b) average Cookie Synchronizations per impression, both compared against the average cost paid by advertisers to deliver RTB ads to the same users (1-1 mapping), across the year.

Furthermore, we look at the average cost users pay for being delivered ads vs. the corresponding average cost advertisers paid for the exact same ads, for each user via a heatmap in Figure 6.20(a). We observe that the counts are skewed towards the upper left triangle for many of the users. In total, 67.4% of users paid more in bytes than what the advertisers paid for the same ads to be delivered. This means that the majority of mobile users pay more in data plan cost to download each impression (or even in total through the year), than the corresponding cost that advertisers pay to send the given ads displayed.

6.4.2 Cost of Privacy vs. Cost of RTB

In section 6.2.2, we analyzed the cost of privacy for mobile users given the CSyncs performed by the advertising ecosystem. We measured how prevalent this practice is across users and through time. Here, we compare this privacy loss with the cost paid by advertisers in RTB ads delivered to users during the same time period.

In Figure 6.19 we show the CDF of the average CSyncs per impression (total CSyncs through the year in Figure 6.19 (line in blue)) that were performed through each user's device. We notice that the median user had about 3.4 synchronizations per impression, and

101 in total through the year. As explained earlier, this leads to loss of privacy to multiple third party companies.

We compare this cost on user privacy to the cost paid by advertisers with a heatmap in Figure 6.20(b). We notice that the main mass of the distribution of users cluster between 1 and 100 synchronizations per impression delivered (as also evident from Figure 6.19 (line in red)) and cost for the advertisers between 0.0005 and 0.001 Euros, per impression delivered. Also, in totals across the year, users have been exposed to 10-1000 synchronizations for all the ads they received, and these delivered ads cost between 0.005 and 0.05 Euros to the advertisers.

6.5 Discussion

6.5.1 Learnings

Unlike traditional advertising, in online mediums advertising imposes costs not only to the one who wants its message to be spread (the advertiser), but also to the one that receives it (the user). To make matters worse, the growth of personalized advertisement, where the advertisements are matched to the interests of the individuals, impose an additional cost for the users: the cost on their privacy and loss of anonymity.

In this study, we compare the costs on digital advertising for the advertiser and the user, in an attempt to identify how equal, or even comparable these costs are. Surprisingly, our results show that these costs are unbalanced, with the majority of users sustaining a significant loss of privacy, when the monetary cost they pay is, on average, 3 times more than what the advertisers are charged to deliver the given ads. Our findings can be summarized as follows:

- Ad- and analytics- related traffic is 19% of the total requests, and 8.2% of data plan volume of an average mobile user.
- Ad-related volume has been steadily increasing through the year, doubling from 4 KBytes to 8 KBytes per ad-request.
- Ad- and analytics- related traffic can potentially consume up to 9% of the phone's power, considering only the additional network overhead.
- 97% of regular mobile users are exposed to Cookie Synchronization at least once in a year.
- The 50th (75th) percentile user is exposed to one CSync every 140 (50) traffic requests, or every 3-4(1-2) website visits.
- The 50th (75th) percentile user gets up to 63 (195) of their unique user IDs synced in a year, at least once.

- Top 5% (0.6%) of ad-companies learn more than 10% (25%) of all user IDs, through the year.

- The median user loses up to 20% of their anonymity to 22 tracking entities, and up to 40% to 3 tracking entities.

- The top 5 ad-companies dominate 68% of RTB auctions.

- Mobile users are exposed to 10-1000 synchronizations for ads received through the year, which cost to the advertisers 0.005-0.05 Euros.

- The median advertiser paid 0.00071 Euro per delivered ad, but the median user paid 0.0022 Euro per ad in downloaded bytes.

6.5.2 Impact of Advertising Cost

Our results showed that in aggregate, and monetarily, over 2/3rds of users pay more through their data plan for downloading bytes related to ads and analytics, than the advertisers who sent the ads in the first place. In addition, given that: 1) the median user loses up to 20% of its anonymity to 22 tracking entities, 2) the top 5 ad-companies win the great majority of RTB auctions, and 3) these companies can sell the acquired data to 4th party companies in a non-transparent and backend fashion [135,189], the loss of privacy experienced by an average user can be multiple times higher than that conservatively measured so far. Unfortunately, this pervasive user tracking effort to deliver more targeted impressions, fails to increase the effectiveness of the delivered ads. In fact, and according to [205], the average person is served over 1700 ads per month, but only half of them are ever viewed, and click rates for display ad campaigns reach 0.1% on average (i.e., one in a thousand impressions in a campaign is ever clicked). Furthermore, Budak et al. in [28] show that retailers attract only 3% of their customers through digital ads. Therefore, even though someone could argue that the user receives value from free access to the websites supported by advertisers, the amount of ineffective ads delivered to user devices is currently extreme, and costly for the end-user.

Considering all the above, the cost on the user's side with respect to 1) device resources spent for processing and displaying ads, 2) bytes downloaded and paid to the user's data plan, 3) loss of privacy experienced by the average user, all significantly outweigh both the efficiency of the received ads, and the cost paid by the ad ecosystem to deliver them to the user's device. Thus, it remains unclear whom the current advertising model benefits, apart from the ad-delivery and targeting companies.

6.5.3 Reducing or rebalancing the costs

Evidently, the annoyance, the inefficiency and the increased cost of advertisements have made users take measures to reduce the unbalanced costs they pay. The most popular of such actions is the use of mobile [56, 179] or desktop based [29, 55] ad-blockers. However, there are concerns [30, 112] that such all-out approaches are non-vital for the free Internet, as they significantly reduce the income of the ad-supported content providers, making them stop serving ad-blocking users [61, 159].

Approaches able to strike a vital middle-ground and rebalance the costs between advertisers and users, include Personal Information Management Systems (PIMS) [35, 130, 131, 160]. In PIMS, the user controls the privacy they expose to the online world, in return for a free service. A different approach is third-party ad-replacement systems [26] such as the Brave Browser [25], where the user gets compensated for each ad they receive. In addition, there is the CAMEO middleware [121], which aims to pre-fetch context-sensitive advertisement by predicting user context and pro-actively identifying relevant advertising content. This way, it can opportunistically use inexpensive wireless networks (e.g., WiFi) to predictively cache advertisement content on the mobile device.

The contribution of our work is to shed light upon the actual costs of ad-supported web. This way, we enhance the awareness of users regarding costs that they can easily measure (e.g., on their data plan), or cannot measure (e.g., privacy loss), in an attempt to help them choose between a seemingly free, ad-supported website and its paid ad-free counterpart [256].

Our future steps include active analysis of the user devices in order to measure additional hidden costs of advertising, that appear in power consumption, main memory, CPU. We will also study the impact advertising has on user experience by measuring the imposed latency due to the rendering time of digital ad impressions. In addition, active analysis on crafted user personas will allow us to determine the user data that get leaked together with the userIDs and if this is compliant with COPPA [74] rules and DAA's AdChoices [54] program.

Chapter 7
Web-Mining as an Alternative Monetization Model

It is of no doubt that digital advertising is the dominant monetization model of the free Internet. It constitutes the driving force of the web, leading to the provision and support of new web services and applications [83, 234]. However, in the recent years, either due to the roaring privacy implications of targeted advertising [32, 195, 235] or the irritation dodgy ads may cause [85], a growing number of users (615 million devices – 30% growth since last year [46]) decided to abdicate from receiving ads by adopting all-out approaches (like deploying ad-blocking mechanisms [38, 179, 245] or ad-stripping browsers [25, 39, 75]). This increasing ad-blocking trend made some major web publishers, after seeing their income significantly shrinking (total losses of $22 billion [162]), to deploy ad-blocker detection techniques [115, 161, 173] and deny serving content to ad-blocking users [42, 159, 216, 223]. Such aggressive actions from both sides escalated an inevitable arms race between the ad-ecosystem on the one side, and the ad-blockers and privacy advocates on the other side [20, 163, 173].

In such a dispute, evidently, publishers were trapped in the crossfire being unable to effectively monetize their services. To that end, it did not take long for some of them to look for effective and reliable alternative schemes to support their websites. Some of these schemes include paid website versions, user compensation (e.g., Basic Attention Token [12]) and cryptomining. Especially the latter, given its privacy protecting nature (no user tracking and personal data collection required, thus making cryptocurrency mining GDPR compliant) and the frenetic increase of the market value of cryprocoins, gains an ever increasing popularity.

Of course, in-browser based mining is not a new idea. The compatibility of Javascript miners with all modern browsers gave motivation for web mining attempts since the very early days of Bitcoin, back in 2011 [165]. To that end, web miners "borrow" spare CPU cycles of the visiting users' devices for performing their Proof-of-Work (PoW) computations [222] for as long as the user is browsing the website's content. However, the increased mining difficulty of Bitcoin was the primary factor that led such approaches to failure. Yet, the

rapid growth of Bitcoin lured several initiatives to construct their own derivatives (more than 1638 nowadays [114]) providing specific extra features e.g., transaction speed, proof-of-stake. One of the provided features: mining speed, became the growth factor for some coins (Monero [41, 203] grew from 13$ to 300$ within 2017 [84]) and was the catalyst a for the incarnation of web cryptomining [143].

Indeed, since the release of the first JavaScript miner (i.e., September 2017) by Coinhive [40], we observe a rapidly increasing [5, 52, 69] number of content providers deploying web-based cryptomining libraries in their websites, monetizing their content either by using both ads and web-mining or by fully replacing ads (e.g. PirateBay [51]). So the important question that arises at this point is the following: *Can web-mining become the next business model of the post ad-supported era of Internet?*

There are numerous opinions about this subject [21, 136, 208], but it is apparent that in order to accurately respond to such a question we first need to investigate all aspects of both advertising and web-mining. These aspects include, first of all, the profitability that cryptominers provide to publishers and also the costs that users have to sustain from the utilization of their resources: let us not forget that the unsustainable costs [103, 183] of advertising made ad-blocking popular.

In this study, we aim to address exactly that; we conduct the first full-scale analysis of the profitability and costs of web-mining, in an attempt to shed light in the newly emerged technology of in-browser cryptomining and explore if it can replace ads on the web. Specifically, in this study we estimate the possible revenues for the different monetization strategies: advertising and web-mining, aiming to determine under what circumstances a miner-supported website can surpass the profits from digital advertising.

Additionally, we collect a large dataset of miner- and ad- supported websites and by designing and developing WebTestbench, a sensor-based testbed, we measure the resource utilization of both models in an attempt to compare their imposed user-side costs. In particular, WebTestbench is capable of measuring (i) the utilization of mining regarding system resources such as CPU and main memory, (ii) the degradation of the user experience due to the increased mining workload, (iii) the energy consumption and how this affects battery-operated devices (e.g., laptops, tablets, smartphones), (iv) system temperature and how overheating affects the user's device and (v) network and how this can affect a possible mobile dataplan.

To summarize, in this chapter we conduct the first study on the profitability of web-based cryptocurrency mining, questioning the ability of mining to become a reliable monetization method for future web services. Our results show that for the average duration of a website visit, ads are 5.5x more profitable than cryptomining. However, a miner-supported website can produce higher revenues if the visitor remains in the website for longer than 5.3 minutes. We design a methodology to assess the resource utilization patterns of ad- and miner-supported websites on the visitor's device. We implement our approach in WebTestbench

framework and we investigate what costs these utilization patterns impose on the visitor's side with regards to the user experience, the system's temperature, and energy consumption and battery autonomy. We collect a large dataset of around 200K ad- and miner-supported websites that include different web-mining libraries and cryprocurrencies. We use this dataset as input for the WebTestbench framework and we compare the resource utilization and costs of the two web monetization models. Our results show that while browsing a miner-supported website, the visitor's CPU gets utilized 59 times more than while visiting an ad-supported website, thus increasing the temperature (52.8%) and power consumption (2x) of her device.

7.1 Background

7.1.1 Web-based cryptocurrency mining

Web-based mining is a method of cryptocurrency mining that happens inside a browser, using a script delivered through a website. Additionally, the rise of alternative cryptocoins that provide distributed mining, increased mining speed and ASIC (Application-Specific Integrated Circuit) resistance, made distributed CPU (i.e, x86, x86-64, ARM) based mining very effective [1], even when using commodity hardware. This way, all these new coins e.g., Electroneum, Bytecoin and Monero opened new funding avenues for web publishers.

The motivation behind this new business model is simple: users visit a website and pay for the received content indirectly by mining cryptocurrency coins, without being polluted with (possibly annoying [85]) ads. Furthermore, publishers do not have to bother collecting behavioral data to get higher prices (as seen in Chapter 5) for their ad-slots. As a consequence, users get a cleaner, faster, and potentially less risky website.

7.1.2 How does web mining work?

The large growth of web-mining started with the release of Coinhive's JavaScript implementation of a Monero miner in September 2017 [40]. This JavaScript-based miner, which computes hashes as a Proof-of-Work, could be easily included in any website for enabling publishers to utilize visiting users' CPUs as a way to monetize the visits to their websites.

Upon visiting a miner-supported website, the user receives a mining library along with the website's content. Usually these libraries are provided by third parties, which we will refer to as Mining Service Providers (MSP), who are responsible for maintaining the source code, controlling the synchronization of computations, collecting the computed hashes and sharing the profits with the publishers. Upon rendering, a miner establishes a persistent connection with the MSP (e.g., coinhive.com) to communicate with the service/mining pool. Through this channel the miner receives periodically PoW tasks and reports the successfully computed hashes.

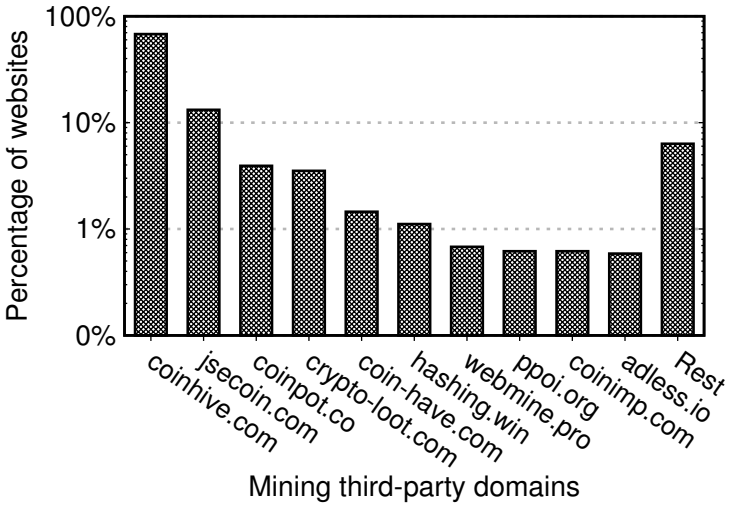


Figure 7.1: Cryptomining market share per third party library in our dataset. Coinhive owns the dominant share (69%) when JSEcoin follows with 13%.

7.1.3 Cryptojacking

The increasing growth of web-based miners does not create opportunities only for legitimate publishers, but cyber-attackers as well. Soon after the release of the first mining library by Coinhive, numerous incidents have been reported [164] of attackers injecting mining code snippets in websites with increased audience. This so-called Drive-by Mining, or *cryptojacking*, takes place either by compromising embedded third party libraries or by delivering malicious mining code through the ad ecosystem [86]. For example, the compromisation of a single screen reader third party (i.e., Browsealoud [178]) resulted in infecting more than 4000 websites that were using it. Victims of cryptojacking have been popular and prestigious websites [79, 97, 144, 145, 153].

Of course, the notion of cryptojacking does not only include compromised websites but also websites that use mining as a method for monetization but abstain from informing the users about the existence of cryptominers. Indeed, contrary to digital advertising were the visitors can identify the ad-impressions, in web mining it is not easy for the visitors to perceive the existence of an included miner. To that end, cryptojacking is a malicious action that abuses the user's processing power, and includes any web-mining attempt without the user's consent irrespectively whether the mining code has been deployed by the website's publisher or an attacker that hijacked the website.

Type	Amount
Blacklist entries	3610
Miner-supported websites	107511
Ad-supported websites	100000
Unique third-party miners	27
Crypto-coins	Monero, JSEcoin

Table 7.1: Summary of our dataset

7.2 Data collection and analysis

To gather the necessary data for our study we collect several coin-blocking blacklists[1] including the ones used by the 5 most popular mine-blocking browser plugins.[2] By merging these blacklists we compose a list of 3610 unique entries of mining libraries and keywords. Then, we use these entries to query PublicWWW,[3] and we identify a total of 107511 mine-including domains. It should be noted that the domains we collected are ranked in the range from 1353 to 960540 in the Alexa rank of popular websites, and that the majority of them are based in the USA, Russia and Brazil.

The mining websites we collected, include more than 27 different third party miners, such as Coinhive, CryptoLoot and CoinHave. In Figure 7.1, we present the portion of websites in our dataset that use each one of these libraries. As seen, besides the large variety of mining libraries, there is a monopolistic phenomenon in the market of cryptominers, with Coinhive owning the dominant share (69%), when from the rest of its competitors only JSEcoin miner surpasses 10%.

Apart from these miner-supported websites, we also collected an equal number of ad-supported ones, which are among the same popularity ranking range. We process each of these domains and by using the blacklist of Ghostery open-source adblocker, we enumerated all the ad-slots in the landing page. The average number of ad-slots per website was 3.4. Finally, Table 7.1 summarizes the contents of our dataset.

7.2.1 WebTestbench framework for utilization analysis

To measure the costs each domain in our dataset imposes on the user, we designed and developed WebTestbench: a web measuring testbed. A high-level overview of the architecture of WebTestbench is presented in Figure 7.2. The WebTestbench framework follows an extensible modular design, and consists of several measuring components that work in a plug-and-play manner. Each such component is able to monitor usage patterns in different system resources (memory, CPU etc.). The main components of our platform currently include:

[1] https://zerodot1.gitlab.io/CoinBlockerListsWeb/index.html
[2] Coin-Blocker, No Mining, MinerBlock, noMiner and CoinBlock
[3] https://publicwww.com/

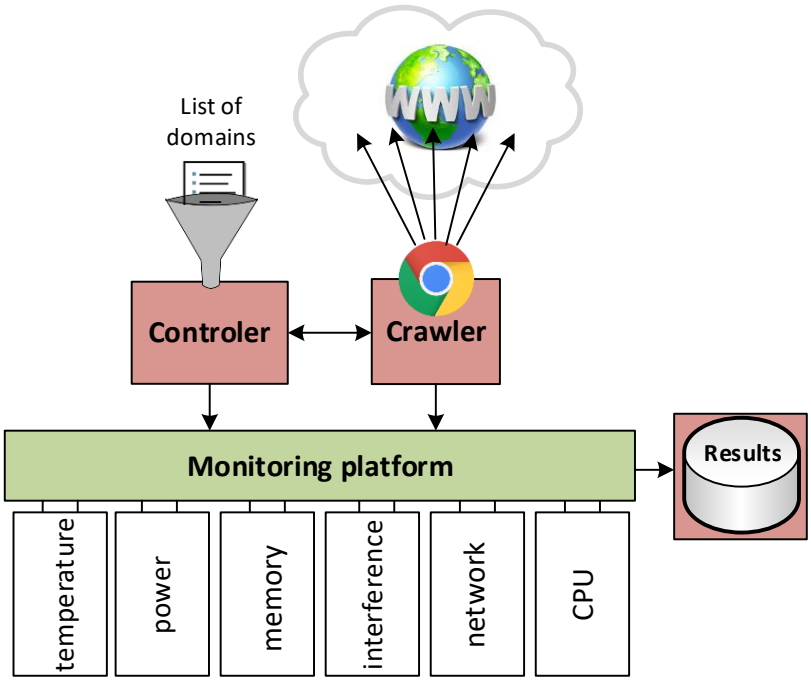


Figure 7.2: High level overview of our measurement testbed. A Chrome-based platform fetches each website for a specific time and its different components measure the resources.

A. **crawler** component, which runs the browser (i.e., Google Chrome) in a headless mode. The crawling is responsible of stopping and purging *any* state after a website probe (e.g., set cookies, cache, local storage, registered service workers, etc.), and listening to the commands of the main controller (i.e., next visiting website, time interval, etc.).

B. **main controller**, which takes as input a list of domains and the visiting time per website. It is responsible for scheduling the execution of the monitoring components.

C. **monitoring platform**, which is responsible for the per time interval execution of the monitoring modules. This platform was build in order to be easily expandable in case of future additional modules.

For the scope of this analysis, we developed 6 different modules to measure the utilization that miners perform in 6 different system resources:

1. memory activity (physical and virtual), by using the psrecord utility [200] and attaching to the crawling browser tab's pid.

2. CPU utilization per core, by using the dedicated linux tool process status (ps [225]).

3. system temperature (overall and per core), by leveraging the Linux monitoring sensors (`lm_sensors` [201]).

4. network traffic, by capturing (i) the network packets through `tcpdump` and (ii) the HTTP requests in the application layer along with their metadata (e.g., timing, initiator, transferred bytes, type, status code).

5. process interference, to infer the degradation of user experience caused by the heavy CPU utilization of mining processes. Specifically, this module consists of a CPU intensive benchmarking that includes multi-threaded MD5 hash calculations.

6. energy consumption, by installing in our machine an external Phidget21 power sensing package [190]. Phidget enables us to accurately measure the energy consumption of the 3 ATX power-supply lines (+12.0a, +12.0b +5.0, +3.3 Volts). The 12.0 Va line powers the processor, the 5.0V line powers the memory, and the 3.3V line powers the rest of the peripherals on the motherboard.

Methodology: In order to explore the different resource utilization patterns for miner- and ad- supported websites, we load our domain dataset in WebTestbench and we fetch each landing page for a certain amount of time. During this period the network monitoring module captures all outgoing HTTP(S) requests of the analyzed website. Additionally, the modules responsible for measuring the energy consumption, the CPU and memory utilization and the temperature report the sensor values in a per second interval. By the end of this first phase, WebTestbench erases any existing browser state and re-fetches the same website. This time, the only simultaneously running process is the interference measuring module which reports its progress at the end of the second phase.

7.3 Analysis

In this section, we aim to explore the profitability of the cryptomining web monetization model for the publishers, and to compare it with the current dominant monetization model of the web: digital advertising. Towards that direction, we assess the costs imposed on the user in an attempt to determine the overheads a website's visitor sustains while visiting a miner-supported website. For the following experiments, we use a Linux desktop equipped with a Hyper-Threading Quad-core Intel I7-3770 operating at 3.90 GHz, with 8 MB SmartCache, 8 GB RAM and an Intel 82567 1GbE network interface.

7.3.1 Profitability of publishers

In the first set of experiments, we set out to explore the profitability of cryptominers and compare it to the current digital advertising model. Thereby, in the first experiment we

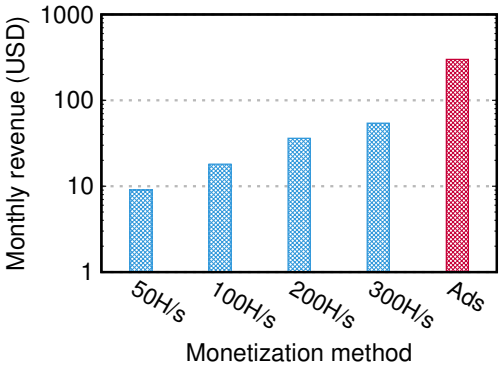


Figure 7.3:
Estimation of monthly profit for the different monetization methods for a website with 100K visitors and average visit duration of 1 minute. Even for visitors with powerful devices (300Hashes/sec), a publisher gains 5.5× more revenue by including 3 ads in its website.

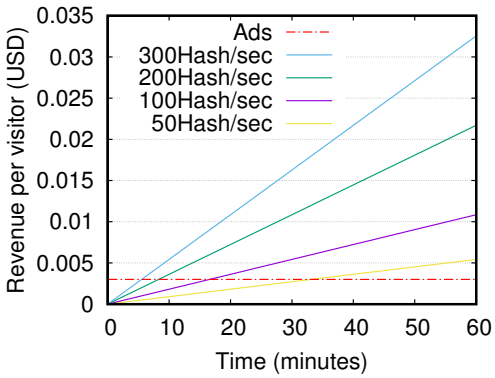


Figure 7.4:
Revenue per visitor for a website running in a background tab. In order for a publisher to gain higher profit from mining than using ads (3 ad-slots), a visitor must keep his tab open for duration > 5.3 minutes (depending on the their device).

simulate the monthly profit of the two strategies for a website of moderate popularity: 100,000 visitors per month. Studies have measured the average duration of a website visit being around 1 minute [170]. For this experiment, we use the popular Monero mining library of Coinhive, which currently provides a rate of 0.0001468/1M hashes. This means that the publisher gets 0.0001468 Monero (XMR) (at the time of the experiment: 1 Monero=205 USD) per 1 million calculated hashes. Apart from the visit duration, the amount of total calculated hashes of a publisher depends on the computation power of the visitors' devices. Thus, in this experiment, in order to cover a wider range of CPU hashrate capabilities [47], we use 4 different levels of computation rates: the rate of iPhone7: 50 Hash/sec, iPhoneX: 100 Hash/sec, 4-core PC: 200 Hash/sec and 8-core PC: 300 Hash/sec).

Apart from the profit from cryptomining, in this experiment we also compute the monthly revenue of the same website in the case of following the traditional advertising model. The most popular medium for personalized ad-buying nowadays [64, 229] is the programmatic instantaneous auctions. In this model, advertisers bid in real time auctions for each available ad-slot of a publisher's inventory based on how well the visitor's interests match their advertised product. As described in Section 7.2, the average number of ad-slots in an ad-supported website is 3 and the median charge price per ad impression as measured in previous Chapter [**?**] is 1 CPM. As can be seen in Figure 7.3, for the average duration of a user's visit, the publisher when achieving an average computation rate from visitors of as high as 300Hash/sec,

gains 5.5x more revenue when using ads instead of cryptomining[4]. In addition, we see that as the visitor's hardware improves, the distance between these two monetization methods becomes smaller. This means that in the near future web mining can be capable of providing comparable profits for the publishers.

It is apparent that for a miner-supported website *time matters*. Indeed, recent studies [5] show that the majority of miner-supported websites provide content that can keep the visitor on the website for a long time. Such content includes TV, video or movie streaming, flash games, etc. Of course in cryptomining, the user does not need to interact with the website's content per se. As a consequence, numerous deceiving methods (e.g., pop-unders [230]) are currently used, aiming to allow the embedded miner to work in the background for as long as possible.

In the next experiment, we set out to identify the minimum time the publisher's website needs to remain open inside a tab of a visitor's browser in order to gain profit higher than when using ads. In Figure 7.4 we simulate the revenue per visitor for a website running in the background and we use the same hash-rate levels as above. As shown, the miner-using publisher, in order to produce revenues higher than when ads are delivered, must keep its website open in the user's browser for a duration longer than 5.53 minutes. When on background, the website do not receive fresh ads so publishers do not have more revenue. So we see that on the left of the break-even point using ads is more profitable but when moving on the right of break-even point, web-mining generate higher income.

To mitigate the above issues, we propose a Hybrid model to combine both. Specifically, as shown in Figure 7.5, publishers utilize ads to get a basic revenue from the landed visitor and move to web-mining when the visitor switches to different browser tab (e.g., after 1 minute). This way, publishers can continue gaining profit when their websites become idle. So a publisher's revenue when using ads is given by R_A (1), when using web-mining by R_M (2), and when using the Hybrid model by R_H (3), where t_0 is the average duration of a visit.

$$for\ t \in (0, t_0):\ R_A(t) = C_1 \tag{7.1}$$

$$for\ t \in [t_0, oo):\ R_M(t) = C_2 * t \tag{7.2}$$

$$R_H(t) = C_1 + C_2 * (t - t_0) \tag{7.3}$$

As we can see in Figure 7.5, the revenue produced by our Hybrid approach either before or after the break-even point, is always higher or equal to both ads and web-mining.

7.3.2 Costs imposed on the user side

After estimating the revenues of a publisher for the different monetization methods, it is time to measure the costs each of this method imposes on the user.

[4]Our simulation results are verified by the real world experiments [44]

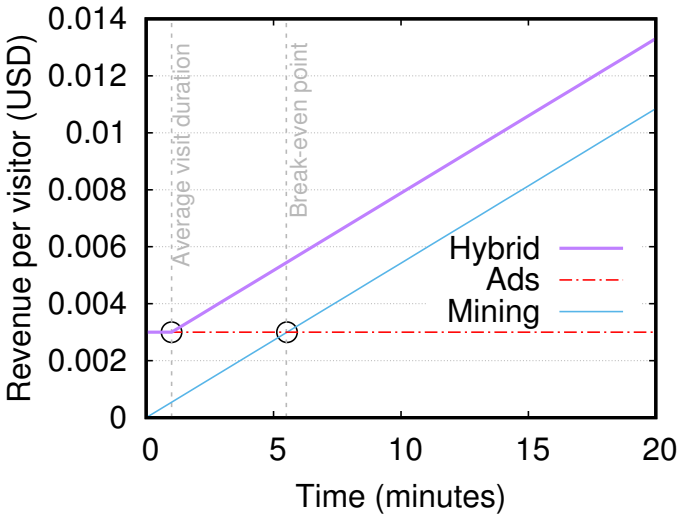


Figure 7.5: Revenue per visitor. In our Hybrid approach the revenue is bounded either before or after the break-even point, to be always higher or equal to both ads and web-mining

CPU and Memory Utilization

In the first set of experiments, we explore the average CPU and memory utilization by mining-supported websites. Note at this point, that the intense of mining is tunable. The majority of mining libraries allow the publishers to fine tune the number of threads and the throttling of their miner. In this experiment we fetch each website in our two subsets for 3 minutes using WebTestbench and we extract the distribution of its CPU utilization through time. In Table 7.2 we report the average values for the median, the 10th and 90th percentiles. As we see, the median miner-supported website utilizes the visitor's CPU up to 59 times more than an ad-supported website.

In the same way, we measure the utilization of the visitors main memory and in Figure 7.6 we plot the average values for both real and virtual memory activity. As expected, miners

Type	10^{th} **Perc.**	**Median**	90^{th} **Perc.**
Advertising	3.33%	9.71%	17.19%
Mining	560.11%	574.01%	580.71%

Table 7.2: Distribution of the average CPU Utilization for the different monetization methods. The median miner-supported website utilizes 59x more the user's CPU than the median ad-supported website.

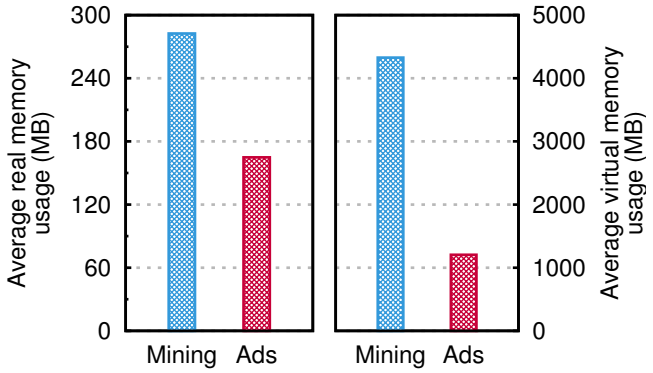


Figure 7.6: Distribution of average real and virtual memory utilization through time. Miner-supported websites although reserve (3.59x) larger chunks of virtual memory, require 1.7x more MBytes of real memory than ad-supported websites.

do not utilize memory as heavy as CPU. In particular, we see that on average the miner-supported websites require 1.7x more space in real memory than the ad-supported websites.

Network Activity

Next, we measure the network utilization of the average mining-supported website. As discussed in Section 7.1, a mining library needs to periodically communicate with a remote third party server (i.e., the MSP's server) in order to report the calculated hashes but also to obtain the next PoW. This communication in the vast majority of the libraries in our dataset takes place through a special persistent channel that allows bidirectional communication. To assess the network activity of web miners, we use the network capturing module of WebTestbench and we monitor the traffic of each (ad- and miner-supported) website for 3 minutes.

Based on the detected third-party mining library, we isolate the web socket communication between its in-browser mining module and the remote MSP server. In order to compare this PoW-related communication of miners with the corresponding ad-related traffic of ad-supported websites, we utilize the open-source blacklist of the Disconnect browser extensionto isolate all advertising related content. In Figure 7.7, we plot the distribution of the total transmitted volume of bytes per website for the visit duration of 3 minutes. Although the web socket communication of miners consists of small packets of 186 Bytes on average, we see that in total the median PoW-related communication of miner-supported websites transmitted 22.8 KBytes, when the median ad-traffic volume of ad-supported websites was

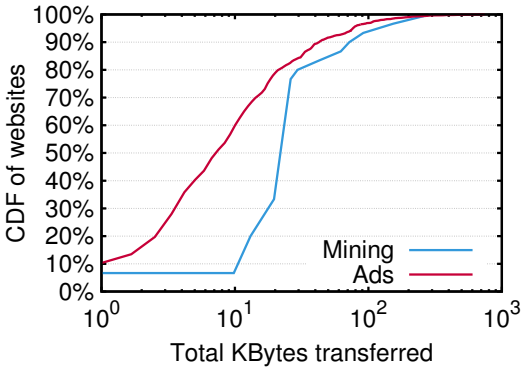


Figure 7.7:
Distribution of the total transmitted volume of bytes per website for a visit duration of 3 minutes. The median miner-generated traffic volume is 3.4x larger than the median ad-generated. In 20% of the websites the difference reduces significantly (less than 2x).

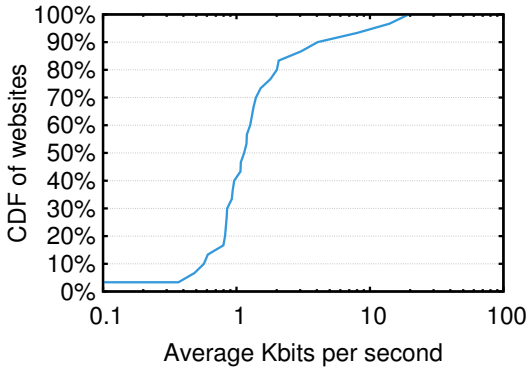


Figure 7.8:
Distribution of the transmitted bit rate per miner-supported website in our dataset. The median in-browser miner communicates with its remote MSP by transmitting 1.17 Kbits per second.

6.7 KBytes. This means that the median miner-generated traffic volume is 3.4x larger than the median ad-generated. In this experiment, we see that the network utilization patterns depend not only on the throttling of the miner but also on the different implementations. For example, while using the same portion of CPU, the miner of coinhive.com transmits on average 0.6 packets/sec, webmine.cz: 2.2 packets/sec, cryptoloot.com: 4.7 packets/sec and jsecoin.com: 1.3 packets/sec.

In Figure 7.8 we plot the distribution of the average data transfer rate per miner-supported website in our dataset. As shown, the median communication between the miner and the MSP has a transfer rate of 1 Kbit per second (or 146 Bytes/sec). As in the previous experiment, the rate highly depends on the mining library, with some of them reaching up to 14 Kbit per second. At this point, recall that the PoW-related communication between the miner and the MSP holds for as long as the miner is running, and as shown in Figure 7.4, a miner must run for longer than 5.53 minutes to produce revenues higher than ads. This means that for the median case, the total volume of bytes transferred will exceed 46 KBytes.

In the case of a user that browses through a cellular (4G) network,[5] the monetary cost imposed is 0.000219$ per minute on average when browsing a miner-supported website. On the other hand, a publisher including a coinhive miner in its website earns 0.000409$ per minute from that user (considering that the user provides a hash rate of 227 Hash/sec as

[5]Considering the average prices per byte in USA and Europe [9, 73, 242]

Component	Type	10^{th} Percentile	Median	90^{th} Percentile
CPU & Network adapter	Advertising	31.88 Watt	32.39 Watt	34.17 Watt
	Mining	63.35 Watt	67.60 Watt	71.22 Watt
Main Memory	Advertising	4.37 Watt	4.46 Watt	5.35 Watt
	Mining	4.76 Watt	4.99 Watt	5.67 Watt

Table 7.3: Distribution of the average consumption of power for the different monetization methods. The median miner-supported website forces the user's device to consume more power than the median ad-supported website: 2.08x and 1.14x more power for the CPU and the memory component, respectively.

in [44]). Hence, we see that cellular users, among other costs while visiting miner-related websites, pay a monetization cost that is only 53% less than the revenue of the publisher.

Power Efficiency

Of course the intensive resource utilization of cryptominers affects also the power consumption of the visitor's device, which has a direct impact on its battery autonomy. In the next experiment, we measure the power consumed by (i) main memory and (ii) CPU and network adapter components of the user's device while visiting miner- and ad- supported websites for a 3 minute duration. In Table 7.3, we report the average median, 10th and 90th percentile values for all websites in our dataset. As shown, there is a slightly increased (1.14x more than ad-supported websites) consumption of the memory component in miner-supported websites. However, we see that the heavy computation load of cryptominers significantly increases the CPUs and network adapters consumption, making miner-supported websites consume 2.08x more energy than ad-supported websites! This means that a laptop able to support 7 hours of consecutive traditional ad supported browsing, would support 3.36 hours of mining-supported browsing.

System Temperature

The increased electricity powering of the visitor's system results to an increased thermal radiation. During the above experiment, we measure the distribution of the per-core temperatures while visiting each website in our dataset for 3 minutes. In Figure 7.9, we present the average results for the percentiles: 10th, 25th, 50th, 75th, 90th. As we can observe, the core temperatures for miner-supported websites are constantly above the optimal range of $45-60^{\circ}$ Celsius [150, 191]. In particular, the visitor's system operates for most of the time in the range of $43-50^{\circ}$ Celsius while visiting ad-supported websites. When the visited website includes a miner, the average temperature of the cores reaches up to 52.8% higher, in the range of $73-77^{\circ}$ Celsius, when in 10% of the cases it may reach higher than 84° Celsius.

To that end, with regards to the costs imposed to the user, high temperatures may lead to

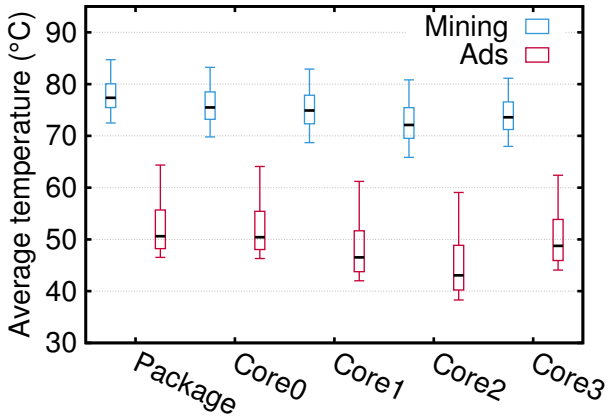


Figure 7.9: Distribution of average temperatures per system's core. When the visited website includes miner, the average temperature of the cores may reach up to 52.8% higher ($73 - 77°$ Celsius) than when with ads.

degraded system performance and poor user experience. Apart from that, constantly running a commodity device (e.g., mobile phone, laptop or desktop PC) at high temperatures, without a proper cooling mechanism, may significantly decrease the hardware's lifespan in the long term or even cause physical damage by thermal expansion.

Effects on Parallel Running Applications

It is apparent that the heavy utilization of the visitor's CPU is capable of affecting the overall user's experience not only in the visited website, but in parallel processes and browser tabs, too. Indeed, for as long as the browser tab of a mining-supported website is open, the multi-threaded computations of the miner leaves limited processing power for the rest of the running applications. To make matters worse, as part of a PC's own cooling system, the motherboard in case of increased temperatures may instruct the CPU component to slow down, or even force the system to turn off without warning [13].

To assess how these factors may affect parallel running processes in the visitor's device, we use the interference measuring module of WebTestbench and we measure the performance overhead caused by background running miners. This module, introduces computation workloads to the system to emulate a parallel running process of the user. Specifically, WebTestbench fetches each website in our dataset for the average visit duration (i.e., 1 minute), in parallel conducts multi-threaded MD5 hash calculations, and in the end reports the number of calculated hashes. In order to test the performance of the user's parallel process in different computation intensity levels, we visit each website using 3 setups for the MD5 process,

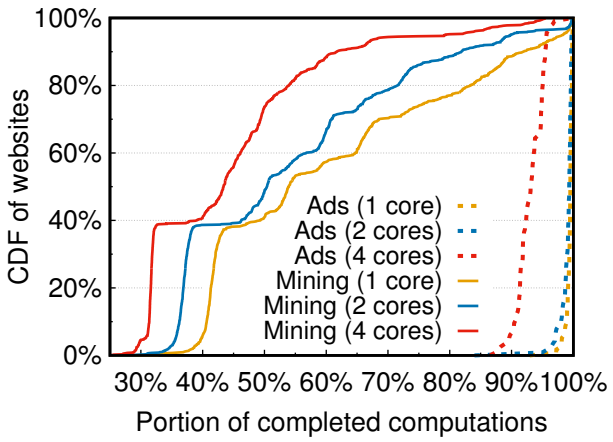


Figure 7.10: Impact of background running miner- and ad-supported websites to a user's process. When the majority of ad-supported websites have negligible effect in other processes, the median embedded miner in our dataset through its heavy CPU utilization may cause a performance degradation of higher than 46% to a parallel running process.

utilizing in parallel 1, 2, and 4 cores of the CPU. In addition, we run the MD5 process alone for 1 minute to measure the maximum completed operations.

In Figure 7.10, we plot the distribution of completed operations per website. As expected, when there is a miner-supported website running in the user's browser, the performance of the user's processes that run in parallel is severely affected. In particular, we see that the median miner-supported website forces the parallel process (depending on its computation intensity) to run in 54%, 50% or even 43% of its optimal performance, thus causing an performance degradation that ranges from 46% to 57%! Additionally, we see a 39% of miners greedily utilizing all the system's CPU resources causing a performance reduction of 67% to the parallel process.

Moreover, in this figure we plot the interference that ad-supported websites impose to a parallel process. As expected, the impact is minimal and practically only processes with full CPU utilization are affected, facing a performance degradation of less than 10% for the majority of such websites. This slight performance degradation is caused by the JavaScript code responsible for ad serving, user tracking, analytics, etc.

Such severe performance degradation when the user is visiting a mining-supported website can cause glitches, or even crushes in other, parallel, CPU utilizing applications (like movie playback, video calling, video games, etc.), thus ravaging the user's experience. Of course, this performance degradation does not only affect parallel running applications but also mining operations from other open browsing tabs.

Indeed, a miner can achieve full utilization when the user has visited the miner-supported *website*1. However, when the user opens a second miner-supported *website*2 the maximum utilization for both, as well as the revenues for *publisher*1 and *publisher*2, drop to a half. It is easy thus to anticipate, that **the scalability of cryptomining is limited since the more websites rely on web-mining for funding, the less revenues will be for their publishers.** While this monetization model has that apparent drawback, in digital advertising each ad-supported website is totally independent from any parallel open browser tabs.

7.4 Discussion

7.4.1 User awareness

The lack of adequate policies and directives regarding the proper use of cryptomining has raised a big controversy regarding the lack of transparency in miner-supported websites. Many miner-supported websites do not inform the user about the existence of a miner, neither ask for the visitor's consent to utilize their system's resources for cryptocurrency mining.

In one of the first law cases about web-based mining, the Attorney General John J. Hoffman stated that "*no website should tap into a person's computer processing power without clearly notifying the person and giving them the chance to opt out*" [117]. As a result, whenever a user visits a website and she is not aware about the background web-mining p, irrespectively whether the mining code has been legitimately deployed by the publisher or a malicious actor that hijacked the website, this is considered as a *cryptojacking* attempt.

7.4.2 Letting users choose

Since both digital advertising and web-mining impose a hidden cost on the user, each one in a different way, a new paradigm could be to inform the user about these costs and give them the option to choose which of the two monetization schemes is more suitable for them. In the case of advertising the main cost to the user is related to the lack of privacy, while the cost of web-mining is associated with higher energy consumption (and battery drainage, overheating etc). It seems that a viable option for publishers would be to inform the users about these costs, and provide two different versions of their website (i.e., one that serve ads and one that uses cryptoming), thus allowing the user to choose between the two schemes.

7.4.3 Web-miner detection

The emergence of web-mining, and especially the many reported cases of cryptojacking, pushed towards considering web-mining by default as a malicious operation. To that regard, many major Antivirus vendors recently launched software products [236] for detecting

and blocking in-browser miners. The vast majority of these approaches are mainly based on detecting outgoing requests to MSP and mining pools. Chrome removes from the Chrome Web Store all extensions that perform mining [238] and implements a throttling mechanism that will limit CPU utilization for JavaScript code running in the background [37].

However, even though these approaches currently seem reduce the extend of web-mining and cryptojacking, they are not very robust against determined publishers/attackers, especially the ones based on the use of blacklists for detecting domains associated to miners. Recently we have seen miners that try to avoid detection by only utilizing a percentage of the users' CPU processing power and by employing cloud-based proxy servers to handle all communication with the MSP [210]. Also, in many cases the mining code is highly obfuscated [215] to prevent pattern matching tools from detecting snippets of such suspicious code.

7.5 Summary

In this chapter, we investigate the ability of web-mining to become a reliable alternative monetization model for the web. Particularly, we estimate the monthly revenue a publisher may gain by using cryptominers to monetize its content, and we compare our results with the estimated revenue for the same publisher when using the traditional personalized advertising model. We compute the duration threshold for a website visit, after which a publisher can earn more revenue when using a cryptominer instead of ads. Additionaly, we propose a Hybrid model, where publishers can gain a basic profit from landed visitors and continue generating revenue by using web-mining when the browser tabs their websites reside become idle.

Next, we measure the costs cryptominers impose on the user side by analyzing the utilization patterns of miner-supported websites in the visitor's system resources (like CPU, memory, network). We study the impact of these utilization patterns (i) on the visitor's device by measuring the system's power consumption and temperature, and also (ii) on the visitor's experience while running other applications in parallel.

7.5.1 Lessons Learned

The findings of our analysis can be summarized as follows:

- for the average duration of a website visit, a publisher gains 5.5x more revenue by including 3 ad impressions in its website than by including a cryptominer.

- to produce higher revenues with a miner than with ads, the publisher must keep the user's browser tab open for a duration longer than 5.53 minutes or use a hybrid approach.

- the median miner-supported website utilizes up to 59x more of the visitor's CPU and require 1.7x more space in real memory than ad-supported websites.
- the transfer rate of the median miner-MSP communication is 1 Kbit/sec. For a user with a mobile data-plan the monetary cost imposed is on average 0.000219$ per minute, when the publisher from the same user earns 0.000409$ per minute.
- the median miner-generated traffic volume is 3.4x larger than the corresponding ad-generated.
- a visit to a miner-supported website consumes on average 2.08x more energy than to an ad-supported website.
- a visitor's system operates in up to 52.8% higher temperatures when visiting a website with miner than when with ads.
- web-miners severely affect parallel running processes. The median miner-supported website when running in the background may degrade even 57% of the performance of parallel running applications.

7.5.2 Can web-mining become the next web monetization model?

After completing our analysis, we see that web-mining can indeed constitute a reliable source of income for specific categories of publishers. By using the Hybrid approach we propose, publishers can even increase their profits by exploiting the time when their websites reside in idle tabs. What is more, in these days, where EU regulators [174] aim to reform the way user data are being collected and processed for targeted advertising, cryptomining provides a privacy-preserving monetization model that requires zero data from the users. However, this study shows that the intensive resource utilization of web-mining libraries imposes a significant cost on the user's device, accelerating the deterioration of its hardware. To make matters worse, this utilization also limits the scalability of web-mining, since the more websites adopting miners the less portion of resources each of them acquire from a user with multiple open tabs. To conclude, cryptomining has indeed the potential to become a reliable alternative for publishers, but it cannot replace the current ad-driven monetization model of the web.

Chapter 8
Related Work

8.1 User tracking and Device Fingerprinting

There is a plethora of papers studying privacy loss and tracking techniques in the wild [2, 60, 67, 132, 142, 172, 202]. There are also others proposing privacy preserving countermeasures based on either (i) randomization-/obfuscation- based techniques [171,186], where the authors aim to pollute the information trackers retrieve in order to hide the users' data and interests, or (ii) anti-tracking mechanisms [129], where requests to trackers are avoided or blocked. All the above studies, highlight the voracity of web entities to collect data about the user and her online behavior, and an arms race between the privacy-aware users and trackers.

Leontiadis et al. [140], analyze a large dataset of 250,000 Android apps and their results reveal the ineffectiveness of current privacy protection mechanisms. Finally, they propose a market-aware privacy protection framework aiming to achieve an equilibrium between the developer's revenue and the user's privacy. Senevirante et al. [213] conduct a large-scale study comparing the presence of tracking libraries in paid and free apps. Their key findings denote that almost 60% of paid apps, contain trackers that collect sensitive information, compared to 85%–95% found for free apps.

Han et al. [105], performed a real-world tracking study of mobile apps running on the devices of 20 participants instrumented with dynamic taint tracking of specific sensitive information. They found that 36% of the domains perform user tracking and that 37% of them use persistent identifiers that allow cross-application and cross-site profiling of the user. Demetriou et al. [50], explored the capabilities of various ad libraries and showed that ad networks are prone to leak more of the user's data than before. Therefore, they propose Pluto, a mobile risk assesment framework to discover personal data leaks.

In [142], authors compare mobile apps and web browsers based on the amount of demographic information they leak. The authors manually examined a small group of 50 online services and monitored the PII each of them shares. In our paper, we extend the investigated privacy leaks to include device specific identifiers as well. Our results show that by broadening the spectrum of leaks, web facilitates less leaks than apps.

There are also studies focusing solely on privacy preservation in the advertising ecosystem.

For example, Privad [101] is designed to conceal user activities from an ad-network, by interposing an anonymizing proxy between the browser and the ad-network, allowing a trusted client software to select relevant ads locally. Unfortunately, it requires broad adoption of high-performance anonymizing proxies. Alternatively, Adnostic [227] is an architecture for interest-targeted advertising without tracking. Like Privad, Adnostic uses client-based functionality to perform ad selection, but eliminates anonymizing proxies at the cost of less precise ad targeting. In [187], authors propose obfuscation of the user's full identity while browsing the web. This was achieved by introducing Web Identity Translator (WIT) in-between the user's client and the visited websites.

In [214], Shekar et al. present an extension that splits application's functionality code and ad code to run in separate processes, thus eliminating the ability of applications to request extra permissions on behalf of their ad libraries.

Recon [198], is a VPN-based system capable of identifying PII leaks through the network. By combining machine learning and network trace analysis, Recon is able to completely block or add noise to PII leaking requests. Contrary to Recon, our implementation does not need any trusted entity (e.g. VPN or Proxy), to monitor the users traffic. To make matters worse, there are countries banning or blocking VPN-related traffic [134].

In [253], Yu et al. before proposing their own anti-tracking mechanism, conduct a large scale study regarding the prevalence and reach of trackers, by analyzing over 21 million pages of 350,000 unique sites. They show that 95% of the pages visited contain 3rd party requests to potential trackers, and 78% attempt to transfer unsafe data. In addition, the authors discuss how unsafe data can be transmitted through cookie values, something that happens for the 58% of the tracking cases. Finally, they rank tracker organizations, showing that a single organization can reach up to 42% of all page visits in Germany.

8.1.1 User ID Sharing

Cookie Synchronization is currently a commonplace on the web, with a number of sites sharing IDs with each other, constituting a mechanism able to severely harm the user privacy. However, cookie-based tracking is only the tip of the iceberg. One of the first academic works that discussed this mechanism is the paper of Olejnik et al. [176], which studies programmatic auctions from a privacy perspective and presents Cookie Synchronization (that they call Cookie Matching) as an integral part of communication between the participating entities. Their results indicate that over 100 cookie syncing events were found while crawling the top 100 sites. In our study, we extend their detection mechanism to detect CSync when cookieID is piggybacked in either URL parameters or URL path.

Additionally, in Acar et al. [2], the authors conduct a CSync privacy analysis by studying a small dataset of 3000 crawled sites. The authors study CSync in conjunction with respawning cookies and how, together, they affect the reconstruction of the user's browsing

history by the trackers. They highlight the inadequacy of current anti-tracking policies. Specifically, enabling Do Not Track in their crawling browser only reduced the number of domains involved in synchronization by 2.9% and the number of synced IDs by 2.6%.

In a recent census by Englehardt et al. [67], authors measure CSync and its adoption in a small subset of 100,000 crawled sites, before highlighting the need of further investigation given its increased privacy implications. Their results show that 157 of top 200 (i.e. 78%) third parties synchronize cookies with at least one other party. Contrary to these studies, we aim to conduct a full scale study of CSync and its characteristics by using a large, year-long dataset of real users.

Ghosh et al. [81] study the economics and the revenue implications of Cookie Synchronization from the point of view of an informed seller of advertising space, uncovering a trade-off between targeting and information leakage. Towards a similar direction, in [17], the authors explore the role of data providers on the price and allocation of consumer-level information and develop a simple model of data pricing that captures the key trade-offs involved in selling information encoded in third-party cookies.

Falahrastegar et al. [71] investigate tracking groups that share user-specific identifiers in a dataset they collected after recording the browsing history of 100 users for two weeks. In this dataset, they detect 660 ID-sharing groups, in which they found domains with sensitive content (such as health-related) that shared IDs with domains related to advertisement trackers.

Englehardt et al. in [68] measure the information that can be inferred through web traffic surveillance in conjunction with third-party tracking cookies. Results show that the adversary can reconstruct 62-73% of a user's browsing history. The authors also analyze the effect of the geolocation of the wiretap, as well as legal restrictions. Using measurement units in various geolocations (Asia, Europe, USA), they show that foreign users are more vulnerable to NSA's surveillance due to the concentration of third-party trackers in the U.S.

Bashir et al. in [11] aiming to enhance the transparency in ad ecosystem with regards to information sharing, develop a content-agnostic methodology to detect client- and server-side flows of information between ad exchanges by leveraging retargeted ads. By using crawled data, authors collected 35448 ad impressions and identified four different kinds of information sharing behavior between ad exchanges.

8.2 User data and the Ad-Ecosystem

The economics of private data have long been an interesting topic and attracted a considerable body of research either from the user's perspective [4, 31, 199, 217], or the advertiser's perspective [48, 65, 83, 176]. In [4] authors discuss the value of privacy after defining two concepts (i) *Willingness To Pay*: the monetary amount users are willing to pay to protect their privacy, and (ii) *Willingness To Accept*: the compensation that users are willing to accept

for their privacy loss. In two user-studies [31, 217] authors measure how much users value their own offline and online personal data, and consequently how much they would sell them to advertisers. In [199], the authors propose "transactional" privacy to allow users to decide what personal information can be released and receive compensation from selling them.

In [176], the authors perform an analysis of cookie matching in association with the RTB advertising. They leverage the RTB nURL to observe the charge prices and they conduct a basic study to provide some insights into these prices, by analyzing different user profiles and visiting contexts. Their results confirm that when the users' browsing histories are leaked, the charge prices tend to be increased. Similarly, in [175], the authors propose a transparency enhancing tool showing to the users the RTB charge price every time a RTB auction is performed. Furthermore, they collect profiled and un-profiled data from a browser extension and a crawler respectively, and they compare the RTB prices, the bidding frequency and the inter-relations among ADXs and DSPs.

In [83], authors use a dataset of users' HTTP traces and provide rough estimates of the relative value of users by leveraging the suggested bid amounts for the visited websites, based on categories provided by the Google AdWords. FDTV [48] is a plugin to inform users in real-time about the economic value of the personal information associated to their Facebook activity. Although similar to ours, our approach works for all HTTP activity of mobile users. Furthermore, journalists created an interactive calculator [65] to explore how valuable specific pieces of user data are for the ad-companies. This calculator is based on the analysis of industry pricing data from a range of sources in the US.

Finally, the rapid growth of RTB auctions has drawn the attention of the research community, which aims to explore the economics of the RTB ad ecosystem. In [254], the authors provide an insight to pricing and an empirical analysis of the technologies involved. They use internal data of an ADX and they study its bidding behaviors and strategies. In [249], the authors propose a winning price predicting mechanism by leveraging machine learning and statistical methods to train a model using the bidding history. Their predicting approach aims to help DSPs fine-tune their bids accordingly. Though such studies help us understand some internal mechanisms of ADXs and DSPs, they are not applicable to our setting as we try to infer the cumulative ad-cost of each user based on user-related features that are measurable from the user's device over time.

8.2.1 Costs of Advertising

There are several studies aiming to measure different aspects and hidden costs of the advertising ecosystem. Gui et al. in [103] measure the cost of mobile advertisements to the mobile application developer by performing an empirical analysis of 21 apps. The authors consider several types of costs: (i) app performance, (ii) energy consumption, (iii) network usage, (iv) maintenance effort for ad-related code and (v) the user' feedback from app reviews. Their

results show that apps with ads consume, on average: 48% more CPU time, 16% more energy, and 79% more network data. In addition, they found that the presence of ads in the apps affected the users' overall opinion leading to reduced ratings for the app.

Towards the same direction, Gao et al., propose IntelliAd [80], a tool to automatically measure the ad costs based on the different ad integration schemes. Similar to the above work, IntelliAd aims to provide developers with suggestions on how to better integrate ads into apps based on the costs the users are concerned. To identify the opinion of the users, the authors utilize several user reviews from 104 popular apps of Google Play. The types of the ad costs the users were concerned more include: number of ads, memory/CPU overhead, traffic usage, and battery consumption.

In [240] the authors quantify the network usage and system calls related to mobile ads, based on specific rules, aiming to quantify the difference between free and paid versions. In particular, they built a tool to profile apps at four different layers: (i) static, or app specification, (ii) user interaction, (iii) operating system, and (iv) network. They evaluate their approach by analyzing 27 free and paid Android apps. Their results show discrepancies between the app specification and app execution, as well as cases were free versions of apps were more costly than their paid counterparts due to their important increase in traffic. Finally, they observe that most network traffic is not encrypted and, even worse, apps communicate with many more sources than users might expect (as many as 13).

In [234], they analyze the characteristics of mobile ads by collecting a large volume of traffic of a major European ISP with over 3 million subscribers. Their results show that ad-related traffic is a significant portion of the overall traffic, and the associated market share is dominated by no more than 3 big ad-networks. In addition, they evaluate the energy consumption of three popular ad networks using a custom-built app with an ad slot at the bottom of the screen. In [28], they analyze the browsing activity of a large sample of Internet users aiming to assess the impact of ad-blockers and regulatory policies which limit the use of third-party data for targeted advertising. Their results show that retailers attract only a small percentage (3%) of their customers through display ads. Although many publishers use ads as their main source of income, which makes them vulnerable to ad-blockers, browsing patterns suggest that ad revenue can generally be replaced by a small fraction of loyal visitors paying a modest subscription fee (e.g. $2 per month).

Apart from the academic studies, there is also an increased interest regarding the cost of the advertising ecosystem from the side of journalists and major news sites. For example, in [98] the editorial team conducted a small study measuring the estimated load time and data usage before and after blocking ad-related content on 50 popular news websites. Their results show that more than 50% of all data came from ads and other content.

Contrary to the above studies, our more user-centric approach provides a methodology to measure the hidden costs of advertising through passive monitoring of the users' traffic. We compare the cost users sustain, with the cost the advertisers pay for the ad delivery.

Finally, we not only measure the monetary and network costs of digital advertising, but also the implications in privacy and anonymity of the users on the Internet via Cookie Synchronization.

8.3 Web-Mining and Monetization

Although recent, the technology of user-side cryptomining has drawn a lot of attention in the research community. At the beginning, cryptomining was used mostly illegally as payload of malwares. Indeed, Wyke [250], back in 2012, attempted to increase the awareness regarding the possibility of existing malwares delivering cryptomining payloads to infected user devices. Botnets are examples of such malwares, which adopted mining to directly monetize the computational ability of a compromised computer. Huang et al. [108] conduct a comprehensive study of existing Bitcoin mining malware, and present the infrastructure and miner-bot orchestration mechanisms deployed in the wild.

The advances of JavaScript, which provide developers with parallel execution of their operations, and the development of more lightweight altcoins like Monero and Litecoin, enabled the browser-based miners to grow. As a consequence, content providers can deploy mining-supported websites without affecting the user experience. Eskandari et al. [69], in one of the first web mining related studies, analyze the existing in-browser mining approaches and their profitability.

AdGuard Research which produces an ad-blocking software, in [5] analyze the Alexa top 100,000 websites for cryptocurrency mining scripts in an attempt to measure the adoption of cryptominers in contemptorary web. The analysis revealed 220 of these websites using cryptomining scripts with their aggregated audience being around 500 million people. The content of these hosting websites were usually movie/video/tv streaming (22.27%), file sharing (17.73%), Adult (10%) and News & Media (7.73%) with the majority of them based in U.S., India, Russia, and Brazil.

This rapid growth of web miners along with the frenzy increase of the cryptocurrency values, caused a serious debate over the Internet regarding the ability of cryptomining to become an alternative to the current ad-supported model of Internet [21, 208]. In accordance with this debate, in this study, we compare the profitability of ad and cryptomining supported Internet services, and we also measure the cost of cryptomining for the visitors. Of course, the advertising ecosystem also imposes costs on the user side.

Of course, the same advancements that allowed the growth of cryptomining for website monetization, enabled also attackers to perform cryptojacking. Recent reports from cybersecurity agencies [70] aim to warn Internet users about the emerging threat of cryptojacking and there are several incidents already reported, where websites [144, 153] got infected with malware either by malvertising [224] or server compromisation that abused visitors' devices.

Dorsey in [57] exploit the ad ecosystem to widely deliver malware which upon browser infection could perform computation on the user side like cryptomining.

Chapter 9
Conclusion

9.1 Synopsis of Contributions

In this dissertation, we explored the privacy violations of contemporary websites and apps in an attempt to identify which harms the least the user's privacy: mobile friendly websites or apps? Specifically, we conducted an extensive study of a broad spectrum of privacy- related leaks able to expose not only PII but also device specific identifiers (Chapter 3). These identifiers allow third parties to deploy fingerprinting mechanisms, that (i) track the user into the network (ii) link web with app session and (iii) correlate anonymous (such as Tor) sessions with eponymous ones. Our study suggests that apps leak more information than web browsers. To help users improve their privacy even when using a mobile app, we propose antiTrackDroid: an anti-tracking mechanism for Android apps, similar to the state-of-the-art ad-blockers of mobile browsers. Evaluation results show that our approach is able to reduce the privacy leaks by 27.41%, when it imposes a negligible overhead of less than 1 millisecond per request.

We then investigated how all these independent pieces of data get attributed to specific user profiles. We explain how the technique of Cookie Synchronization achieves exactly that (Chapter 4) and by bypassing same-origin policy form a universal user identification, which enables collaborating entities to share data by performing background database merges. We conduct the first in depth analysis of Cookie Synchronization using real year-long data from a large number of volunteering web users. In addition, this study is the first to explore the Cookie Synchronization in mobile devices. Based on the detection approaches of previous works, we propose an enhanced methodology able to capture 3.771% more cases of Cookie Synchronization. By applying this methodology, we collected a dataset of 263K requests syncing 22K unique userIDs, which we analyzed thoroughly. Our results show that 97% of the users are exposed to CSync at least once. The median user experiences at least one CSync within the first week of browsing. The median userID gets leaked, on average, to 3.5 different entities. The number entities that learn about the median user after Cookie Synchronizations grows by a factor of 6.7. In addition, we show that the tracking-related mechanism of Cookie Synchronization, can break the secure TLS session and (i) spill user

unique identifiers (userIDs) along with (ii) the full URL that the user has visited over TLS. This way, a snooping ISP can re- identify the user in the web, even if they use VPN or Tor circuits, as well as reconstruct their browsing history. To assess the feasibility of this threat in the real world, we crawled the top 12K Alexa websites and after extracting the performed Cookie Synchronizations, we found that 1 out of 13 TLS-protected websites expose the privacy of their users.

After that, aiming to enhance transparency in the ad ecosystem, we proposed a methodology to estimate at real time how much do advertisers pay to reach a user and how the above user's collected data affect the pricing dynamics (Chapter 5). In particular, we developed a first of its kind methodology to compute the financial worth of individuals at real time. Our methodology leverages the rapidly growing RTB protocol and the new advertising model of programmatic instantaneous auctions, where the advertisers evaluate the users' collected data at real time and bid for an ad-slot in their display. Our study analyzes the RTB price notifications sent to winning advertising bidders and focuses on the distinction between cleartext and encrypted price notifications and how to estimate the latter. Towards this end, we train a model using as ground truth prices obtained by running our own probing ad-campaigns. We bootstrap and validate our methodology using a year long trace of real user browsing data, as well as two real world ad-campaigns.

As a next step, in an attempt to shed light upon the actual costs of ad-supported web, we compared the above monetary costs advertisers pay with the costs the users pay for the same ad delivery (Chapter 6). These costs may be either directly quantifiable (e.g., requests, bytes, energy), or qualitative such as loss of privacy. To estimate the latter, we measure the anonymity loss caused by userIDs that get synced through ad-related Cookie Synchronizations. Surprisingly, our results show that the costs imposed on advertisers and users for the same ads are totally unbalanced, with the majority of users sustaining a significant loss of privacy, when the monetary cost they pay is, on average, 3 times more than what the advertisers are charged to deliver these ads. Specifically, mobile users are exposed to 10-1000 synchronizations for ads received through the year, which cost to the advertisers 0.005-0.05 Euros. Additionally, the median advertiser paid 0.00071 Euro per delivered ad, but the median user paid 0.0022 Euro per ad in extra downloaded Bytes.

The above results motivated us to explore the possibility for publishers to use alternative sources of content monetization, able to preserve the privacy of the users but at the same time generate enough profit for the free Internet as we know it today. There is already such an alternative gaining popularity on the web: user side in-browser cryptomining. Indeed, binded with the whopping values of crypto-coins, web-based cryptomining enjoys nowadays a steadily increasing adoption by service providers. But can cryptomining become a reliable alternative for the next day of the free Internet? To respond to this exact question, we estimated the monthly revenue a publisher may gain by using cryptominers to monetize its content, and we compared our results with the estimated revenue for the same publisher

when using the traditional personalized advertising model (Chapter 7). After exploring the profitability for the side of the publisher, we measured the costs cryptominers impose on the side of the user. Specifically, we analyzed the utilization patterns of miner-supported websites in the visitor's system resources like CPU, main memory and network. Then, we studied the impact of these utilization patterns (i) on the visitor's device by measuring the system's power consumption and temperature, and also (ii) on the visitor's experience while running other applications in parallel. Our results show that cryptomining indeed has the potential to become a reliable alternative for some content providers, but due to the scalability issues and increased user-side cost (e.g., 2.08x more energy, 52.8% higher temperatures), it is not capable of replacing the current ad-driven monetization model of the web.

9.2 Lessons Learned

After completing our analysis, we are now able to respond to the research questions we pondered at the beginning (see Section 1) of this dissertation:

1. *What kind of personal information gets leaked while using websites and apps on a mobile phone?*

   Both web and apps leak important fingerprinting information about the user's device. This allows third parties to not only cross-channel track the users by linking web with app sessions but also correlate eponymous with anonymous sessions. In addition, we see apps leaking information (e.g. installed and running apps, nearby APs, etc.) that allowing tracking domains to infer user interests, gender, age range, even behavioral patterns.

2. *Which of the two: browser accessed website or mobile app facilitates the most privacy leaks considering the same online service?*

   Our results prove that both versions of the online service leak information that can be used beyond the control of users (e.g., for targeted advertising). However, apps leak significantly more both device-specific information and PII.

3. *How trackers and advertisers synchronize the user IDs they have set for the same user profiles before they participate in data markets?*

   To overcome restrictions like same-origin policy and also create unified identifiers for each user, the ad-industry invented Cookie Synchronization: a mechanism that allows web entities to share (synchronize) cookies, and match the different IDs they assign for

the same user while they browse the web. The average user receives ∼1 synchronization per 68 GET requests. Cookie Synchronization increases the number of entities that track the user by a factor of 6.7.

4. *In the era of personalized advertising how much do advertisers pay to reach an individual? And how the personal data affect the pricing dynamics in programmatic ad-buying?*

   Advertisers pay ∼25 CPM for delivering ads to an average user based on their personal data, and less than ∼100 CPM for delivering ads to 3/4 of users during a year. A small portion of outlier users (∼2%) cost 10-100x more to the ad-ecosystem than the average user. Some advertisers use encryption to conceal the price they pay for an ad-slot. Contrary to related work [176], we prove that encrypted and cleartext prices do not follow the same distribution (encrypted prices are around 1.7x higher).

5. *Are the users indeed receiving the content they want free of charge in the ad-supported web? If not, what are the costs the median user has to sustain and how comparable these costs are with the monetary cost advertisers pay to deliver them ads?*

   Users, when receiving free, ad-supported conten, have to sustain costs imposed by the ad ecosystem. These costs may be either directly quantifiable (e.g., requests, bytes, energy), or qualitative such as loss of privacy. Specifically, users may pay 3x more money to receive ads than what advertisers pay to deliver them. In addition, the median user may lose up to 20% of their anonymity to 22 tracking entities, and up to 40% to 3 tracking entities.

6. *Are there alternatives to the data-centric model of digital advertising nowadays? Can the resource-borrowing model of cryptomining constitute the next primary monetization model for the post-advertising era of free Internet?*

   Web mining has the potential to become a reliable alternative for publishers, but it cannot replace the current ad-driven model of web. To produce higher revenues with a miner than with ads, the publisher must keep the user's browser tab open for longer than 5.53 minutes or use a hybrid approach that starts with ads rendering before switching to mining and produce this way more revenue than both approaches.

9.3 Directions for Future Work and Research

There are several aspects for further work and research that this dissertation leaves open:

1. **Sensitive User Groups and Interests.** Several organizations and regulators promote directives such as COPPA [74] ruling and DAA's AdChoices [54] regarding the proper delivery of ads to individuals respecting sensitive groups (i.e., children). It is important for the research community to investigate the adoption of such regulations by the ad-ecosystem and in what extend advertisers respect the choices of users regarding the category of advertisements they prefer to receive. In addition, further investigation is needed to determine how collected user information considered as sensitive is being used: e.g., health issues (e.g., when based tracked geolocation user visited a clinic specializing in AIDS or cancer), religious beliefs (e.g., when based on tracked geolocation user visited a catholic church), political inclinations (e.g., when user attends a political party's rallies), etc.

2. **Client Side Ad-Personalization.** As shown in this dissertation, the current model of personalized advertising imposes heavy costs on user's privacy leading more users to deploy client side ad-blocking mechanisms. A way to mitigate these issues is to promote a client side ad personalization model. Specifically, users will retrieve a bundle of ad creatives from the ad agency and the final decision about which of them will end up on the device's display will be taken locally based on the interests of the user. Apart from the privacy preservation, this approach will exclude middlemen, data brokers and trackers from digital advertising, thus preventing annoying ads, malvertising and fraud.

   Some of the problems requiring careful research in this approach include the (i) *ROI counting*: advertisers must be able to learn how many times their ad creatives were rendered but without revealing in which users, (ii) *publisher attribution*: the rendered creatives must be attributed to the visited publishers but without revealing the browsing history of the user.

3. **Digital Advertising and Spread of Fake News.** Nowadays, we observe a massive growth of fake news that pollute online social networks and microblogging services, aiming to shape the public opinion or earn clicks. But what is the impact of the ad ecosystem on the production of fake news? To respond to this question, it would be very interesting to investigate in what extend the profitability of digital advertising can work as motivation for a publisher to generate and spread fake news (as with click-baiting) and what the advertisers pay to have their ad rendered in a website that delivers fake or misleading news.

Bibliography

[1] Cpu coin list. http://cpucoinlist.com/.

[2] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, 2014.

[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*, CCS '13, pages 1129–1140, New York, NY, USA, 2013. ACM.

[4] Alessandro Acquisti, Leslie K John, and George Loewenstein. What is privacy worth? *The Journal of Legal Studies*, 2013.

[5] AdGuard Research. Cryptocurrency mining affects over 500 million people. and they have no idea it is happening. https://adguard.com/en/blog/crypto-mining-fever/, 2017.

[6] Adthink S.A. Big, the user data exchange. https://big.exchange/, 2018.

[7] Android Developers. Android Debug Bridge. http://developer.android.com/tools/help/adb.html.

[8] Android Developers. Class Overview: BroadcastReceiver. `http://developer.android.com/reference/android/content/BroadcastReceiver.html`.

[9] AT&T. Create your mobile share advantage plan. https://www.att.com/shop/wireless/data-plans.html, 2018.

[10] Paul Barford, Igor Canadi, Darja Krushevskaja, Qiang Ma, and S. Muthukrishnan. Adscape: Harvesting and analyzing online display ads. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 597–608, New York, NY, USA, 2014. ACM.

[11] Muhammad Ahmad Bashir, Sajjad Arshad, William Robertson, and Christo Wilson. Tracing information flows between ad exchanges using retargeted ads. In *25th USENIX*

Security Symposium (USENIX Security 16), pages 481–496, Austin, TX, August 2016. USENIX Association.

[12] BAT team. Basic attention token. https://basicattentiontoken.org/.

[13] Philip Bates. How heat affects your computer, and should you be worried? https://www.makeuseof.com/tag/how-heat-affects-your-computer-and-should-you-be-worried/.

[14] BDEX - Big Data Exchange. Dmp 2.0 - introduction of the dxp. http://www.bigdataexchange.com/dmp-2-0-introduction-of-the-dxp/, 2015.

[15] Howard Beales. The value of behavioral targeting. *Network Advertising Initiative*, 2010.

[16] Ross Benes. "we get audience data at virtually no cost" confessions of a programmatic ad buyer. https://digiday.com/marketing/get-audience-data-virtually-no-cost-confessions-programmatic-ad-buyer/, 2018.

[17] Dirk Bergemann and Alessandro Bonatti. Selling cookies. *American Economic Journal: Microeconomics*, 7(3):259–294, 2015.

[18] Paul Bernal. Our web history reveals what we think and do. shouldn't that remain private? https://theconversation.com/our-web-history-reveals-what-we-think-and-do-shouldnt-that-remain-private-50289, 2015.

[19] BI Intelligence. Programmatic advertising is under review. http://www.businessinsider.com/programmatic-advertising-under-review-2017-1, 2017.

[20] Jason Bloomberg. Ad blocking battle drives disruptive innovation. https://www.forbes.com/sites/jasonbloomberg/ 2017/02/18/ad-blocking-battle-drives-disruptive-innovation/, 2017.

[21] Violet Blue. As online ads fail, sites mine cryptocurrency. https://www.engadget.com/2017/12/15/as-online-ads-fail-sites-mine-cryptocurrency/, 2017.

[22] BlueKai. Data management platforms demystified. `http://www.bluekai.com/files/DMP_Demystified_Whitepaper_BlueKai.pdf`, 2011.

[23] bobzilla, arkasha, uhtu. Wigle: Wireless network mapping. https://wigle.net/, 2001.

[24] Ciprian Borodescu. Web sites vs. web apps: What the experts think. https://www.visionmobile.com/blog/2013/07/web-sites-vs-web-apps-what-the-experts-think, 2013.

[25] Brave Software Inc. . Brave: A browser with your interests at heart. https://brave.com/, 2018.

[26] Brave Software Inc. What is brave ad replacement? `https://www.brave.com/about-ad-replacement/`, 2016.

[27] Ira Brodsky. Deathmatch: The mobile web vs. mobile apps. http://www.computerworld.com/article/3016736/mobile-wireless/the-mobile-web-vs-mobile-app-death-match.

[28] CEREN BUDAK, SHARAD GOEL, JUSTIN RAO, and GEORGIOS ZERVAS. Understanding emerging threats to online advertising. 2016.

[29] David Cancel, Felix Shnir, Alexei Miagkov, and Jose Maria Signanini. Ghostery makes the web cleaner, faster and safer! `https://www.ghostery.com/`, 2010.

[30] Sean Captain. This startup wants to end adblock plus "raping and pillaging" of online publishers. https://www.fastcompany.com/3055827/this-startup-wants-to-end-adblocks-raping-and-pillaging-of-online-publishers, 2016.

[31] Juan Pablo Carrascal, Christopher Riederer, Vijay Erramilli, Mauro Cherubini, and Rodrigo de Oliveira. Your browsing behavior for a big mac: Economics of personal information online. In *Proceedings of the 22nd international conference on World Wide Web*, WWW'13, 2013.

[32] Juan Miguel Carrascosa, Jakub Mikians, Ruben Cuevas, Vijay Erramilli, and Nikolaos Laoutaris. I always feel like somebody's watching me: measuring online behavioural advertising. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '15, page 13. ACM, 2015.

[33] Antonios A Chariton, Eirini Degkleri, Panagiotis Papadopoulos, Panagiotis Ilia, and Evangelos P Markatos. Dcsp: Performant certificate revocation a dns-based approach. In *Proceedings of the 9th European Workshop on System Security*, EuroSec '16, 2016.

[34] Antonios A Chariton, Eirini Degkleri, Panagiotis Papadopoulos, Panagiotis Ilia, and Evangelos P Markatos. Ccsp: A compressed certificate status protocol. In *INFOCOM 2017-IEEE Conference on Computer Communications*, 2017.

[35] Amir Chaudhry, Jon Crowcroft, Heidi Howard, Anil Madhavapeddy, Richard Mortier, Hamed Haddadi, and Derek McAuley. Personal data: thinking inside the box. In *Proceedings of The Fifth Decennial Aarhus Conference on Critical Alternatives*, pages 29–32. Aarhus University Press, 2015.

[36] Tom Chavez. Data: Deja vu all over again? https://adexchanger.com/considering-digital/tom-chavez/, 2010.

[37] Catalin Cimpanu. Tweak to chrome performance will indirectly stifle cryptojacking scripts. https://www.bleepingcomputer.com/news/security/ tweak-to-chrome-performance-will-indirectly-stifle-cryptojacking-scripts/, 2018.

[38] Cliqz GmbH. Ghostery makes the web cleaner, faster and safer! https://www.ghostery.com/blog/.

[39] Cliqz GmbH. Cliqz: The no-compromise browser. https://cliqz.com/en/, 2018.

[40] Coinhive. Monetize your business with your users' cpu power. https://coinhive.com/#javascript-api.

[41] CoinWarz. Monero network hashrate chart and graph. https://www.coinwarz.com/network-hashrate-charts/monero-network-hashrate-chart.

[42] Devin Coldewey. Thousands of major sites are taking silent anti-ad-blocking measures. https://techcrunch.com/2017/12/27/thousands-of-major-sites-are-taking-silent-anti-ad-blocking-measures/.

[43] Josh Constine. Facebook crushes q2 earnings, hits 1.71b users and record share price. https://techcrunch.com/2016/07/27/facebook-earnings-q2-2016/, 2016.

[44] Maxence Cornet. Coinhive review: Embeddable javascript crypto miner - 3 days in. https://medium.com/@MaxenceCornet/coinhive-review-embeddable-javascript-crypto-miner-806f7024cde8, 2017.

[45] Aldo Cortesi. An interactive console program that allows traffic flows to be intercepted, inspected, modified and replayed. https://mitmproxy.org/, 2015.

[46] Matthew Cortland. 2017 adblock report. https://pagefair.com/blog/2017/adblockreport/, 2017.

[47] CryptoMining24.net. Cpu for monero. https://cryptomining24.net/cpu-for-monero/, 2017.

[48] Angel Cuevas, Ruben Cuevas, Raquel Aparicio, and Jose Gonzalez. Fdvt: Data valuation tool for facebook users. In *Proceedings of the Conference on Human Factors in Computing Systems*, CHI '17, 2017.

[49] CYREN. Cyren – Cloud-based Internet Security Solytions. http://commtouch.com/.

[50] Soteris Demetriou, Whitney Merrill, Wei Yang, Aston Zhang, and Carl A. Gunter. Free for all! assessing user data exposure to advertising libraries on android. In *Proceedings of the 23rd Annual Network and Distributed System Security Symposium*, NDSS'16, 2016.

[51] Ernesto Van der Sar. The pirate bay website runs a cryptocurrency miner (updated). https://torrentfreak.com/the-pirate-bay-website-runs-a-cryptocurrency-miner-170916/.

[52] Deepen Desai, Dhruval Gandhi, Mohd Sadique, and Manohar Ghule. Cryptomining is here to stay in the enterprise. https://www.zscaler.com/blogs/research/ cryptomining-here-stay-enterprise.

[53] Executive Digital. Programmatic & rtb. http://executive-digital.com/programmatic-rtb/.

[54] Digital Advertising Alliance. Youradchoices gives you control. http://youradchoices.com/, 2018.

[55] Disconnect. A faster, safer internet is one click away. https://disconnect.me/, 2011.

[56] Bruce Bujon Dominik Schurmann. Adaway open source ad blocker for android. `https://adaway.org/`, 2012.

[57] Brannon Dorsey. Browser as botnet, or the coming war on your web browser. Radical Networks., 2018.

[58] DoubleClick. Rtb decrypt price confirmations. https://developers.google.com/ad-exchange/rtb/response-guide/decrypt-price, 2016.

[59] Justin Driskill. Ad size guide. http://theonlineadvertisingguide.com/ad-size-guide/300x250/, 2016.

[60] Peter Eckersley. How unique is your web browser? In *Proceedings of the 10th International Symposium on Privacy Enhancing Technologies*, PETs' 10, 2010.

[61] Editors of Wired Magazine. How WIRED Is Going to Handle Ad Blocking. `https://www.wired.com/how-wired-is-going-to-handle-ad-blocking/`, 2016.

[62] Jo el van Bergen. Mixed content weakens https. https://developers.google.com/web/fundamentals/security/prevent-mixed-content/what-is-mixed-content, 2017.

[63] Hazem Elmeleegy, Yinan Li, Yan Qi, Peter Wilmot, Mingxi Wu, Santanu Kolay, Ali Dasdan, and Songting Chen. Overview of turn data management platform for digital advertising. *Proc. VLDB Endow.*, 2013.

[64] eMarketer Podcast. emarketer releases new us programmatic ad spending figures. https://www.emarketer.com/Article/eMarketer-Releases-New-US-Programmatic-Ad-Spending-Figures/1016698, 2017.

[65] Emily Cadman Emily Steel, Callum Locke and Ben Freese. How much is your personal data worth? http://www.ft.com/cms/s/2/927ca86e-d29b-11e2-88ed-00144feab7de.html, 2013.

[66] Let's Encrypt. Percentage of web pages loaded by firefox using https. https://letsencrypt.org/stats/, 2017.

[67] 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*, CCS '16, pages 1388–1401, New York, NY, USA, 2016. ACM.

[68] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. Cookies that give you away: The surveillance implications of web tracking. In *Proceedings of the 24th International Conference on World Wide Web*, WWW'15, 2015.

[69] Shayan Eskandari, Andreas Leoutsarakos, Troy Mursch, and Jeremy Clark. A first look at browser-based cryptojacking. In *Proceedings of IEEE Security & Privacy on the Blockchain*, S&B 2018, 2018.

[70] European Union Agency for Network and Information Security (ENISA). Cryptojacking - cryptomining in the browser. https://www.enisa.europa.eu/publications/info-notes/cryptojacking-cryptomining-in-the-browser, 2017.

[71] Marjan Falahrastegar, Hamed Haddadi, Steve Uhlig, and Richard Mortier. *Tracking Personal Identifiers Across the Web*, pages 30–41. Springer International Publishing, Cham, 2016.

[72] Famlam Fanboy, MonztA and Khrin. Easylist - overview. https://easylist.to/, 2018.

[73] FANDOM Lifestyle Community. Prepaid data sim card wiki - spain. http://prepaid-data-sim-card.wikia.com/wiki/Spain, 2017.

[74] Federal Trade Commission. Children's online privacy protection act (coppa). https://www.ftc.gov/enforcement/rules/rulemaking-regulatory-reform-proceedings/childrens-online-privacy-protection-rule, 2000.

[75] Klint Finley. Google's new ad blocker changed the web before it even switched on. https://www.wired.com/story/google-chrome-ad-blocker-change-web/.

[76] Rand Fishkin. Mobile web vs mobile apps: Where should you invest your marketing? https://moz.com/blog/mobile-web-mobile-apps-invest-marketing-whiteboard-friday.

[77] The Office for Creative Research. Floodwatch. https://floodwatch.o-c-r.org/.

[78] Electronic Frontier Foundation. Https everywhere. https://www.eff.org/https-everywhere, 2018.

[79] Brian Fung. Hackers have turned politifact's website into a trap for your pc. https://www.washingtonpost.com/news/the-switch/wp/2017/10/13/hackers-have-turned-politifacts-website-into-a-trap-for-your-pc/, 2017.

[80] Cuiyun Gao, Hui Xu, Yichuan Man, Yangfan Zhou, and Michael R. Lyu. Intelliad understanding in-app ad costs from users perspective. *CoRR*, abs/1607.03575, 2016.

[81] Arpita Ghosh, Mohammad Mahdian, R. Preston McAfee, and Sergei Vassilvitskii. To match or not to match: Economics of cookie matching in online advertising. *ACM Trans. Econ. Comput. 2015*, 2015.

[82] Arpita Ghosh and Aaron Roth. Selling privacy at auction. In *Proceedings of the 12th ACM Conference on Electronic Commerce*, pages 199–208, New York, USA, 2011. ACM.

[83] Phillipa Gill, Vijay Erramilli, Augustin Chaintreau, Balachander Krishnamurthy, Konstantina Papagiannaki, and Pablo Rodriguez. Follow the money: Understanding economics of online aggregation and advertising. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 141–148, New York, NY, USA, 2013. ACM.

[84] Global Coin Report. Here's how monero (xmr) gets to $1,000. https://globalcoinreport.com/heres-monero-xmr-gets-1000/, 2018.

[85] Daniel G. Goldstein, R. Preston McAfee, and Siddharth Suri. The cost of annoying ads. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 459–470, New York, NY, USA, 2013. ACM.

[86] Dan Goodin. Ad network uses advanced malware technique to conceal cpu-draining mining ads. https://arstechnica.com/information-technology/2018/02/ad-network-uses-advanced-malware-technique-to-conceal-cpu-draining-mining-ads/.

[87] Google. Doubleclick for publishers ? small business. https://www.google.com/doubleclick/publishers/small-business/, 2016.

[88] Google AdSense. Guide to ad sizes. https://support.google.com/adsense/answer/6002621, 2016.

[89] Google Developers. Real-time bidding protocol: Cookie matching. https://developers.google.com/ad-exchange/rtb/cookie-guide, 2015.

[90] Google Developers. Rtb decrypt price confirmations. https://developers.google.com/ad-exchange/rtb/response-guide/decrypt-price, 2016.

[91] Google Developers. Mixed content weakens https. `https://developers.google.com/web/fundamentals/security/prevent-mixed-content/what-is-mixed-content#mixed_content_weakens_https`, 2017.

[92] Google Inc. Google AdWords. https://www.google.com/adwords/.

[93] Google Inc. "the arrival of real-time bidding and what it means for media buyers". https://static.googleusercontent.com/media/www.google.com/en//doubleclick/pdfs/Google-White-Paper-The-Arrival-of-Real-Time-Bidding-July-2011.pdf, 2011.

[94] Google Inc. Doubleclick manager. https://www.doubleclickbygoogle.com/solutions/digital-marketing/bid-manager/, 2016.

[95] Jay Graves. Ssl pinning for increased app security. https://possiblemobile.com/2013/03/ssl-pinning-for-increased-app-security/, 2013.

[96] Annabelle Green. Customer data collection increased to improve customer experience, research finds. http://business-reporter.co.uk/2016/07/20/customer-data-collection-increased-improve-customer-experience-research-finds/, 2016.

[97] Patrick Greenfield. Government websites hit by cryptocurrency mining malware. https://www.theguardian.com/technology/2018/ feb/11/government-websites-hit-by-cryptocurrency-mining-malware, 2018.

[98] WILSON ANDREWS GREGOR AISCH and JOSH KELLER. The cost of mobile ads on 50 news websites. http://www.nytimes.com/interactive/2015/10/01/business/cost-of-mobile-ads.html, 2015.

[99] Tren Griffin. Tren's advice for twitter. https://25iq.com/2017/01/06/trens-advice-for-twitter/, 2017.

[100] Saikat Guha, Bin Cheng, and Paul Francis. Challenges in measuring online advertising systems. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, pages 81–87, New York, NY, USA, 2010. ACM.

[101] Saikat Guha, Bin Cheng, and Paul Francis. Privad: Practical privacy in online advertising. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, 2011.

[102] Jiaping Gui, Ding Li, Mian Wan, and William G. J. Halfond. Lightweight measurement and estimation of mobile ad energy consumption. In *Proceedings of the 5th International Workshop on Green and Sustainable Software*, GREENS '16, pages 1–7, New York, NY, USA, 2016. ACM.

[103] Jiaping Gui, Stuart Mcilroy, Meiyappan Nagappan, and William G. J. Halfond. Truth in advertising: The hidden cost of mobile ads for software developers. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, pages 100–110, Piscataway, NJ, USA, 2015. IEEE Press.

[104] Beau Hamilton. Google has quietly dropped ban on personally identifiable web tracking. https://tech.slashdot.org/story/16/10/22/008216/google-has-quietly-dropped-ban-on-personally-identifiable-web-tracking, 2016.

[105] Seungyeop Han, Jaeyeon Jung, and David Wetherall. A study of third-party tracking by mobile apps in the wild, 2012.

[106] William T Harding, Anita J Reed, and Robert L Gray. Cookies and web bugs: What they are and how they work together. 2001.

[107] Patrick Holland. Verizon, t-mobile, at&t and sprint unlimited plans compared. https://www.cnet.com/news/how-does-verizon-unlimited-plan-stack-up-against-the-others/, 2017.

[108] Danny Yuxing Huang, Hitesh Dharmdasani, Sarah Meiklejohn, Vacha Dave, Chris Grier, Damon McCoy, Stefan Savage, Nicholas Weaver, Alex C Snoeren, and Kirill Levchenko. Botcoin: Monetizing stolen cycles. In *Proceedings of Annual Network and Distributed System Security Symposium*, NDSS'14, 2014.

[109] IAB. Openrtb api specification version 2.4. http://www.iab.com/wp-content/uploads/2016/01/OpenRTB-API-Specification-Version-2-4-DRAFT.pdf, 2015.

[110] IAB Technology Laboratory. Openrtb (real-time bidding). `https://www.iab.com/guidelines/real-time-bidding-rtb-project/`, 2017.

[111] IHS Technology. Paving the way: how online advertising enables the digital economy of the future. `https://www.iabeurope.eu/files/9614/4844/3542/IAB_IHS_Euro_Ad_Macro_FINALpdf.pdf`, 2015.

[112] Interactive Advertising Bureau (IAB). Rothenberg says ad blocking is a war against diversity and freedom of expression. https://www.iab.com/news/rothenberg-says-ad-blocking-is-a-war-against-diversity-and-freedom-of-expression/, 2016.

[113] InvestingAnswers. Cost Per Thousand (CPM). http://www.investinganswers.com/financial-dictionary/businesses-corporations/cost-thousand-cpm-2917.

[114] investing.com. All cryptocurrencies. https://www.investing.com/crypto/currencies, 2018.

[115] Umar Iqbal, Zubair Shafiq, and Zhiyun Qian. The ad wars: retrospective measurement and analysis of anti-adblock filter lists. In *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, pages 171–183. ACM, 2017.

[116] iSixSigma. How to determine sample size, determining sample size. https://www.isixsigma.com/tools-templates/sampling-data/how-determine-sample-size-determining-sample-size/, 2017.

[117] Steve C. Lee John Hoffman, Jeffrey S. Jacobson. New jersey division of consumer affairs obtains settlement with developer of bitcoin-mining software found to have accessed new jersey computers without users' knowledge or consent. http://nj.gov/oag/newsreleases15/ pr20150526b.html, 2015.

[118] Surya Mattu Julia Angwin, Terry Parris Jr. Facebook is quietly buying information from data brokers about its users' offline lives. http://www.businessinsider.com/facebook-data-brokers-2016-12, 2016.

[119] Peter Kafka and Rani Molla. Recode - 2017 was the year digital ad spending finally beat TV. https://www.recode.net/2017/12/4/16733460/ 2017-digital-ad-spend-advertising-beat-tv, 2017.

[120] Kate Kaye. Nielsen in pact to use offline data for online ad targeting. https://www.clickz.com/nielsen-in-pact-to-use-offline-data-for-online-ad-targeting/77948/, 2009.

[121] Azeem J. Khan, V. Subbaraju, Archan Misra, and Srinivasan Seshan. Mitigating the true cost of advertisement-supported "free" mobile applications. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile '12, pages 1:1–1:6, New York, NY, USA, 2012. ACM.

[122] Kicelo and Dominik Schuermann. Adaway default blocklist. https://adaway.org/hosts.txt, 2016.

[123] Martin Kihn. Top 10 amazing secrets of dmps. http://blogs.gartner.com/martin-kihn/top-10-amazing-secrets-of-dmps/, 2016.

[124] Jacob Kleinman. Stop using whatsapp if you care about your privacy. https://lifehacker.com/stop-using-whatsapp-if-you-care-about-your-privacy-1825719172, 2018.

[125] Ben Kneen. Ssp to dsp cookie syncing explained. http://www.adopsinsider.com/ad-exchanges/cookie-syncing/, 2011.

[126] Knime. Seven techniques for dimensionality reduction. https://www.knime.org/blog/seven-techniques-for-data-dimensionality-reduction, 2015.

[127] Know Online Advertising Inc. Data Management Platform - DMP. http://www.knowonlineadvertising.com/programmatic-buying/data-management-platform-dmp/, 2013.

[128] Know Online Advertising Inc. Definition of backfill. `http://www.knowonlineadvertising.com/advertisingdictionary/backfill/`, 2013.

[129] Georgios Kontaxis, Michalis Polychronakis, Angelos D. Keromytis, and Evangelos P. Markatos. Privacy-preserving social plugins. In *Proceedings of the 21st USENIX Conference on Security Symposium*, SEC'12, 2012.

[130] Nicolas Kourtellis, Jeremy Blackburn, Cristian Borcea, and Adriana Iamnitchi. Special issue on foundations of social computing: Enabling social applications via decentralized social data management. *ACM Transactions on Internet Technology*, 15(1), 2015.

[131] Nicolas Kourtellis, Joshua Finnis, Paul Anderson, Jeremy Blackburn, Cristian Borcea, and Adriana Iamnitchi. Prometheus: User-controlled p2p social data management for socially-aware applications. In *Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware*, pages 212–231, Berlin, Heidelberg, 2010. Springer-Verlag.

[132] Balachander Krishnamurthy and Craig Wills. Privacy diffusion on the web: A longitudinal perspective. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 541–550, New York, NY, USA, 2009. ACM.

[133] Steve Kroft. The data brokers: Selling your personal information. http://www.cbsnews.com/news/the-data-brokers-selling-your-personal-information/, 2014.

[134] Mohit Kumar. Using VPN in the UAE? You'll Be Fined Up To $545,000 If Get Caught! http://thehackernews.com/2016/07/vpn-is-illegal-in-uae.html, 2016.

[135] Rainey Reitman Kurt Opsahl. The disconcerting details: How facebook teams up with data brokers to show you targeted ads. https://www.eff.org/deeplinks/2013/04/disconcerting-details-how-facebook-teams-data-brokers-show-you-targeted-ads, 2013.

[136] Hon Lau. Browser-based cryptocurrency mining makes unexpected return from the dead. https://www.symantec.com/blogs/threat-intelligence/browser-mining-cryptocurrency.

[137] Paul J Leach, Tim Berners-Lee, Jeffrey C Mogul, Larry Masinter, Roy T Fielding, and James Gettys. Encoding sensitive information in uri's. `https://tools.ietf.org/html/rfc2616#section-15.1.3`, 1999.

[138] Leading Edge Provider. Internet trends, stats & facts in the u.s. and worldwide 2016. http://www.leadingedgeprovider.com/2016/12/internet-trends-stats-facts-in-the-u-s-and-worldwide-2016/, 2016.

[139] Michael Learmonth. Online ad industry: Advertising is 'creepy'. http://adage.com/article/digital/online-ad-industry-advertising-creepy/140840/, 2009.

[140] Ilias Leontiadis, Christos Efstratiou, Marco Picone, and Cecilia Mascolo. Don't kill my ads!: Balancing privacy in an ad-supported mobile application market. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile'12, 2012.

[141] Christophe Leung, Jingjing Ren, David Choffnes, and Christo Wilson. App vs web. https://recon.meddle.mobi/appvsweb/, 2016.

[142] Christophe Leung, Jingjing Ren, David Choffnes, and Christo Wilson. Should you use the app for that?: Comparing the privacy implications of app- and web-based online services. In *Proceedings of the 2016 ACM on Internet Measurement Conference*, IMC '16, 2016.

[143] John Leyden. More and more websites are mining crypto-coins in your browser to pay their bills, line pockets. https://www.theregister.co.uk/2017/10/13/ crypto_mining/.

[144] John Leyden. Real mad-quid: Murky cryptojacking menace that smacked ronaldo site grows. http://www.theregister.co.uk/2017/10/10/ cryptojacking/.

[145] Natasha Lomas. Cryptojacking attack hits 4,000 websites, including uk's data watchdog. https://techcrunch.com/2018/02/12/ico-snafu/, 2018.

[146] Lukasz Olejnik and Claude Castelluccia. To bid or not to bid? measuring the value of privacy in rtb. http://lukaszolejnik.com/rtb2.pdf, 2015.

[147] Ziang Ma, Haoyu Wang, Yao Guo, and Xiangqun Chen. Libradar: Fast and accurate detection of third-party libraries in android apps. In *Proceedings of the 38th IEEE/ACM 38th International Conference on Software Engineering Companion*, ICSE'16, 2016.

[148] Bernard Marr. Where can you buy big data? here are the biggest consumer data brokers. https://www.forbes.com/sites/bernardmarr/2017/ 09/07/where-can-you-buy-big-data-here-are-the-biggest-consumer-data-brokers/, 2017.

[149] David Martin, Hailin Wu, and Adil Alsaid. Hidden surveillance by web sites: Web bugs in contemporary use. *Commun. ACM*, 46(12):258–264, December 2003.

[150] Jim Martin. What's the best cpu temperature? https://www.techadvisor.co.uk/how-to/desktop-pc/cpu-temp-3498564/, 2018.

[151] MaxMind Inc. Geoip databases & services: Industry leading ip intelligence. https://www.maxmind.com/en/geoip2-services-and-databases.

[152] Jonathan Mayer. Tracking the trackers: Microsoft advertising. *The Center for Internet and Society*, 2011.

[153] Kieren McCarthy. Cbs's showtime caught mining crypto-coins in viewers' web browsers. http://www.theregister.co.uk/2017/09/25/ showtime_hit_with_coinmining_script/.

[154] Claire Cain Miller and Somini Sengupta. Advertisers find new ways to track smartphone users. `http://www.bostonglobe.com/news/nation/2013/10/05/selling-secrets-phone-users-advertisers/ZSNNChJQvFuEcHJFsUJGUM/story.html`, 2013.

[155] Dan Mitchell. Online ads vs. privacy. `http://www.nytimes.com/2007/05/12/technology/12online.html`, 2007.

[156] Prashanth Mohan, Suman Nath, and Oriana Riva. Prefetching mobile ads: Can advertising systems afford it? In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 267–280, New York, NY, USA, 2013. ACM.

[157] MoPub. Mopub openrtb 2.3 integration guide. https://dev.twitter.com/mopub-demand/marketplace-integration/openrtb.

[158] MoPub Inc. Mopub platform. http://www.mopub.com/platform/.

[159] Brian Morrissey. Forbes starts blocking ad-block users. http://digiday.com/publishers/forbes-ad-blocking/, 2015.

[160] Richard Mortier, Jianxin Zhao, Jon Crowcroft, Liang Wang, Qi Li, Hamed Haddadi, Yousef Amar, Andy Crabtree, James Colley, Tom Lodge, Tosh Brown, Derek McAuley, and Chris Greenhalgh. Personal data management with the databox: What's inside the box? In *Proceedings of the ACM Workshop on Cloud-Assisted Networking*, pages 49–54, 2016.

[161] Muhammad Haris Mughees, Zhiyun Qian, and Zubair Shafiq. Detecting anti ad-blockers in the wild. *Proceedings on Privacy Enhancing Technologies*, 2017(3):130–146, 2017.

[162] Muhammad Haris Mughees, Zhiyun Qian, Zubair Shafiq, Karishma Dash, and Pan Hui. A first look at ad-block detection: A new arms race on the web. *CoRR*, abs/1605.05841, 2016.

[163] Anthony Muller. Ad-mageddon! ad blocking, its impact, and what comes next. https://marketingland.com/ad-mageddon-perspectives-ad-blocking-impacts-comes-next-227090, 2017.

[164] Troy Mursch. Cryptojacking: 2017 year-end review. https://badpackets.net/cryptojacking-2017-year-end-review/, 2017.

[165] Donny Nadolny. Bitcoin plus miner. https://wordpress.org/plugins/bitcoin-plus-miner/.

[166] Meiyappan Nagappan. Go ahead and add that extra ad library, but be careful about which one you add. https://www.developereconomics.com/add-extra-ad-library-but-be-careful-which-one, 2015.

[167] Natalie Lynn. Mobile web vs. mobile in-app advertising: Which is best for your campaign? https://gimbal.com/mobile-web-vs-mobile-in-app-ads/, 2016.

[168] David Naylor, Alessandro Finamore, Ilias Leontiadis, Yan Grunenberger, Marco Mellia, Maurizio Munafò, Konstantina Papagiannaki, and Peter Steenkiste. The cost of the "s" in https. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, pages 133–140, New York, NY, USA, 2014. ACM.

[169] Netimperative. Ad-blocking soars 10 http://www.netimperative.com/2016/01/ad-blocking-soars-10-in-just-3-months/.

[170] JAKOB NIELSEN. How long do users stay on web pages? https://www.nngroup.com/articles/how-long-do-users-stay-on-web-pages/.

[171] Nick Nikiforakis, Wouter Joosen, and Benjamin Livshits. Privaricator: Deceiving fingerprinters with little white lies. In *Proceedings of the 24th International Conference on World Wide Web*, CCS'15, 2015.

[172] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, S&P'13, 2013.

[173] Rishab Nithyanand, Sheharbano Khattak, Mobin Javed, Narseo Vallina-Rodriguez, Marjan Falahrastegar, Julia E. Powles, Emiliano De Cristofaro, Hamed Haddadi, and Steven J. Murdoch. Adblocking and counter blocking: A slice of the arms race. In *6th USENIX Workshop on Free and Open Communications on the Internet (FOCI 16)*, Austin, TX, 2016. USENIX Association.

[174] Official Journal of the European Union. Directive 95/46/ec (general data protection regulation). http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679.

[175] Lukasz Olejnik and Claude Castelluccia. To bid or not to bid? measuring the value of privacy in rtb. https://lukaszolejnik.com/rtb2.pdf.

[176] Lukasz Olejnik, Minh-Dung Tran, and Claude Castelluccia. Selling off user privacy at auction. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium*, NDSS'14, 2014.

[177] OpenX. Rtb macros. `http://docs.openx.com/Content/demandpartners/rtb_macros.html`.

[178] Pierluigi Paganini. Thousands of websites worldwide hijacked by cryptocurrency mining code due browsealoud plugin hack. https://securityaffairs.co/wordpress/68966/hacking/browsealoud-plugin-hack.html, 2018.

[179] Elias P. Papadopoulos, Michalis Diamantaris, Panagiotis Papadopoulos, Thanasis Petsas, Sotiris Ioannidis, and Evangelos P. Markatos. The long-standing privacy debate: Mobile websites vs mobile apps. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 153–162, Republic and Canton of Geneva, Switzerland, 2017. International World Wide Web Conferences Steering Committee.

[180] Panagiotis Papadopoulos, Antonios A. Chariton, Elias Athanasopoulos, and Evangelos P. Markatos. Where's wally?: How to privately discover your friends on the internet. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ASIACCS '18, pages 425–430, New York, NY, USA, 2018. ACM.

[181] Panagiotis Papadopoulos, Panagiotis Ilia, and Evangelos P Markatos. Truth in web mining: Measuring the profitability and cost of cryptominers as a web monetization model. *arXiv preprint arXiv:1806.01994*, 2018.

[182] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. *arXiv preprint arXiv:1805.10505*, 2018.

[183] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. The cost of digital advertisement: Comparing user and advertiser views. In *Proceedings of the 27th International Conference on World Wide Web*, WWW'18, 2018.

[184] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. Exclusive: How the (synced) cookie monster breached my encrypted vpn session. In *Proceedings of the 11th European Workshop on Systems Security*, EuroSec'18, pages 6:1–6:6, New York, NY, USA, 2018. ACM.

[185] Panagiotis Papadopoulos, Nicolas Kourtellis, Pablo Rodriguez Rodriguez, and Nikolaos Laoutaris. If you are not paying for it, you are the product: How much do advertisers pay to reach you? In *Proceedings of the 2017 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '17, pages 142–156, New York, NY, USA, 2017. ACM.

[186] Panagiotis Papadopoulos, Antonis Papadogiannakis, Michalis Polychronakis, Apostolis Zarras, Thorsten Holz, and Evangelos P. Markatos. K-subscription: Privacy-preserving microblogging browsing through obfuscation. In *Proceedings of the 29th Annual Computer Security Applications Conference*, ACSAC'13, 2013.

[187] Fotios Papaodyssefs, Costas Iordanou, Jeremy Blackburn, Nikolaos Laoutaris, and Konstantina Papagiannaki. Web identity translator: Behavioral advertising and identity privacy with wit. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, 2015.

[188] Patricia Gamer. Average revenue per user is an important growth driver. http://marketrealist.com/2015/02/average-revenue-per-user-is-an-important-growth-driver/, 2015.

[189] Andrea Peterson. Bankrupt radioshack wants to sell off user data. but the bigger risk is if a facebook or google goes bust. https://www.washingtonpost.com/news/the-

switch/wp/2015/03/26/bankrupt-radioshack-wants-to-sell-off-user-data-but-the-bigger-risk-is-if-a-facebook-or-google-goes-bust/, 2015.

[190] Phidgets Inc. What is a phidget? https://www.phidgets.com/docs21/What_is_a_Phidget, 2017.

[191] Melanie Pinola. How to test your computer's cpu temperature. https://www.lifewire.com/how-can-i-test-laptop-temperature-2377618, 2018.

[192] PricewaterhouseCoopers LLP,. Iab programmatic revenue report 2014 results. `http://www.iab.net/media/file/PwC_IAB_Programmatic_Study.pdf`, 2015.

[193] PulsePoint. Rtb implementation notes. http://docs.pulsepoint.com/display/BUYER/RTB+Implemen

[194] Raspberry Pi Foundation. Raspberry Pi 2 Model B. https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[195] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, and Phillipa Gill. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. 2018.

[196] redphx. Apk Downloader. https://chrome.google.com/webstore/detail/apk-downloader/cgihflhdpokeobcfimliamffejfnmfii.

[197] Rainey Reitman. How to opt out of receiving facebook ads based on your real-life shopping activity. https://www.eff.org/deeplinks/2013/02/howto-opt-out-databrokers-showing-your-targeted-advertisements-facebook, 2013.

[198] Jingjing Ren, Ashwin Rao, Martina Lindorfer, Arnaud Legout, and David Choffnes. Recon: Revealing and controlling pii leaks in mobile network traffic. In *Proceedings of the 14th Annual International Conference MobiSys*, 2016.

[199] Christopher Riederer, Vijay Erramilli, Augustin Chaintreau, Balachander Krishnamurthy, and Pablo Rodriguez. For sale : Your data: By : You. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, HotNets-X, pages 13:1–13:6, New York, NY, USA, 2011. ACM.

[200] Thomas Robitaille. psrecord: Record the cpu and memory activity of a process. https://github.com/astrofrog/psrecord, 2017.

[201] Guenter Roeck. Overview of the lm-sensors package. https://github.com/groeck/lm-sensors, 2015.

[202] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 12–12. USENIX Association, 2012.

[203] Ameer Rosic. What is monero? the ultimate beginners guide. https://blockgeeks.com/guides/monero/.

[204] rovo89. Xposed Module Repository. http://repo.xposed.info/.

[205] Khalid Saleh. Effectiveness of online advertising - statistics and trends. `https://www.invespcro.com/blog/effectiveness-online-advertising/`, 2016.

[206] samy.pl. evercookie - virtually irrevocable persistent cookies, 2014.

[207] Sandvine Incorporated. Sandvine: 70% of global internet traffic will be encrypted in 2016. https://www.sandvine.com/pr/2016/2/11/sandvine-70-of-global-internet-traffic-will-be-encrypted-in-2016.html, 2016.

[208] Kai Sedgwick. Mining Crypto In a Browser Is a Complete Waste of Time. https://news.bitcoin.com/mining-crypto-in-a-browser-is-a-complete-waste-of-time/, 2018.

[209] Sascha Segan. Verizon, at&t may be choking unlimited data users. https://www.pcmag.com/news/355963/verizon-at-t-may-be-choking-unlimited-data-users, 2017.

[210] Jerome Segura. Malicious cryptomining and the blacklist conundrum. https://blog.malwarebytes.com/threat-analysis/2018/03/malicious-cryptomining-and-the-blacklist-conundrum/, 2018.

[211] Judy Selby. The impact of big data decisions on business valuations. https://datafloq.com/read/impact-big-data-decisions-business-valuation, 2016.

[212] Selenium. Selenium – Web Browser Automation. http://www.seleniumhq.org/.

[213] Suranga Seneviratne, Harini Kolamunna, and Aruna Seneviratne. A measurement study of tracking in paid mobile applications. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, WiSec '15, 2015.

[214] Shashi Shekhar, Michael Dietz, and Dan S. Wallach. Adsplit: Separating smartphone advertising from applications. In *Proceedings of the 21st USENIX Security*, 2012.

[215] Denis Sinegubko. Malicious website cryptominers from github. https://blog.sucuri.net/2018/01/malicious-cryptominers-from-github-part-2.html, 2018.

[216] Nicola Smith. How publishers are turning up the heat in the ad-blocking war. https://www.theguardian.com/media-network/2016/sep/02/publishers-ad-block-users-hide-content, 2016.

[217] Jacopo Staiano, Nuria Oliver, Bruno Lepri, Rodrigo de Oliveira, Michele Caraviello, and Nicu Sebe. Money walks: A human-centric study on the economics of personal mobile data. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014.

[218] Statista Inc. Premium digital advertising spending worldwide from 2015 to 2020 (in billion u.s. dollars). https://www.statista.com/statistics/237974/online-advertising-spending-worldwide/, 2016.

[219] Statista Inc. Percentage of all global web pages served to mobile phones from 2009 to 2018. https://www.statista.com/statistics/241462/global-mobile-phone-website-traffic-share, 2018.

[220] Greg Sterling. Morgan stanley: No, apps aren't winning. the mobile browser is. http://marketingland.com/morgan-stanley-no-apps-arent-winning-the-mobile-browser-is-144303.

[221] Jason Summerfield. Mobile website vs. mobile app: Which is best for your organization? https://www.hswsolutions.com/services/mobile-web-development/mobile-website-vs-apps/.

[222] Andrew Tar. Proof-of-work, explained. https://cointelegraph.com/explained/proof-of-work-explained, 2018.

[223] The Editors of Wired. How wired is going to handle ad blocking. https://www.wired.com/how-wired-is-going-to-handle-ad-blocking/, 2016.

[224] The European Union Agency for Network and Information Security (ENISA). Malvertising. https://www.enisa.europa.eu/publications/info-notes/malvertising, 2016.

[225] The Linux Information Project. The ps command. http://www.linfo.org/ps.html, 2005.

[226] The Tor Project, Inc. Tor project: Anonymity online. https://www.torproject.org/, 2002.

[227] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy preserving targeted advertising. In *Proceedings of the 17th Annual Network and Distributed System Security Symposium*, NDSS '10, 2010.

[228] TRUSTe Technology Blog. Mobile tracking: How it works and why it's different. http://www.truste.com/developer/?p=86, 2016.

[229] Kane Tse. Understanding programmatic advertising: A brief look at its history. https://medium.com/wired-mesh/understanding-programmatic-advertising-a-brief-look-at-its-history-411dd5842304.

[230] Liam Tung. Windows: This sneaky cryptominer hides behind taskbar even after you exit browser. https://www.zdnet.com/article/windows-this-sneaky-cryptominer-hides-behind-taskbar-even-after-you-exit-browser/, 2017.

[231] Twitter Developers. Iab categorization with examples. https://dev.twitter.com/mopub/marketplace/iab-categorization.

[232] Zacharias Tzermias, Vassilis Prevelakis, and Sotiris Ioannidis. Privacy risks from public data sources. In Nora Cuppens-Boulahia, Frédéric Cuppens, Sushil Jajodia, Anas Abou El Kalam, and Thierry Sans, editors, *ICT Systems Security and Privacy Protection*, pages 156–168, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[233] UnhappyGhost Goldenstein. Fingerprinting defenses in the tor browser. http://www.unhappyghost.com/2015/02/forensics-fingerprinting-defenses-in-tor-browser.html.

[234] Narseo Vallina-Rodriguez, Jay Shah, Alessandro Finamore, Yan Grunenberger, Konstantina Papagiannaki, Hamed Haddadi, and Jon Crowcroft. Breaking for commercials: Characterizing mobile advertising. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, 2012.

[235] Narseo Vallina-Rodriguez, Srikanth Sundaresan, Abbas Razaghpanah, Rishab Nithyanand, Mark Allman, Christian Kreibich, and Phillipa Gill. Tracking the trackers: Towards understanding the mobile advertising and tracking ecosystem. *arXiv preprint arXiv:1609.07190*, 2016.

[236] Adarsh Verma. 6 easy ways to block cryptocurrency mining in your web browser. https://fossbytes.com/block-cryptocurrency-mining-in-browser/, 2018.

[237] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 1961.

[238] James Wagner. Protecting users from extension cryptojacking. https://blog.chromium.org/2018/04/protecting-users-from-extension-cryptojacking.html, 2018.

[239] Jeffrey Walton, JohnSteven, Jim Manico, Kevin Wall, and Ricardo Iramar. Certificate and Public Key Pinning. `https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning`.

[240] Xuetao Wei, Lorenzo Gomez, Iulian Neamtiu, and Michalis Faloutsos. Profiledroid: Multi-layer profiling of android applications. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 137–148, New York, NY, USA, 2012. ACM.

[241] Ryan Welton. Android SSL certificate pinning bypass. https://github.com/Fuzion24/JustTrustMe.

[242] WhistleOut Inc. Compare the best cell phone plans. https://www.whistleout.com/CellPhones, 2018.

[243] Sean Whitbeck. Rtb is growing like mad. is your mobile marketing keeping up? http://liftoff.io/rtb-growing-like-mad-mobile-marketing-keeping/, 2015.

[244] Michael Whitener. Cookies are so yesterday; cross-device tracking is insome tips. https://iapp.org/news/a/cookies-are-so-yesterday-cross-device-tracking-is-insome-tips/.

[245] Ben Williams. Adblock plus and (a little) more. https://adblockplus.org/blog/100-million-users-100-million-thank-yous.

[246] Courtney Williams. Mobile vs desktop ad space: challenges and opportunities. https://www.clearpivot.com/blog/mobile-vs-desktop-ad-space-challenges-and-opportunities, 2016.

[247] World Wide Web Consortium (W3C). Same origin policy. https://www.w3.org/Security/wiki/Same_Origin_Policy, 2010.

[248] Luke Wroblewski. Mobile web vs. native apps or why you want both. http://www.lukew.com/ff/entry.asp?1954, 2016.

[249] Wush Chi-Hsuan Wu, Mi-Yen Yeh, and Ming-Syan Chen. Predicting winning price in real time bidding with censored data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

[250] James Wyke. The zeroaccess botnet: Mining and fraud for massive financial gain. Sophos Technical Paper, 2012.

[251] Jun Yan, Ning Liu, Gang Wang, Wen Zhang, Yun Jiang, and Zheng Chen. How much can behavioral targeting help online advertising? In *Proceedings of the 18th International Conference on World Wide Web*, WWW'09, 2009.

[252] Chanmin Yoon, Dongwon Kim, Wonwoo Jung, Chulkoo Kang, and Hojung Cha. Appscope: Application energy metering framework for android smartphone using kernel activity monitoring. In *Presented as part of the USENIX Annual Technical Conference*, ATC'12, pages 387–400, 2012.

[253] Zhonghao Yu, Sam Macbeth, Konark Modi, and Josep M. Pujol. Tracking the trackers. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, 2016.

[254] Shuai Yuan, Jun Wang, and Xiaoxue Zhao. Real-time bidding for online advertising: Measurement and analysis. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, ADKDD '13, pages 3:1–3:8, New York, NY, USA, 2013. ACM.

[255] Jinyan Zang, Krysta Dummit, James Graves, Paul Lisker, and Latanya Sweeney. Who knows what about me? a survey of behind the scenes personal data sharing to third parties by mobile apps. http://techscience.org/a/2015103001, 2015.

[256] John Zorabedian. Wired to adblocker users: pay up for ad-free site or you get nothing. https://nakedsecurity.sophos.com/2016/02/10/wired-to-ad-blocker-users-pay-up-for-ad-free-site-or-you-get-nothing/, 2016.