5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012)

Selected Papers*

Vigo, Spain

July 13, 2012

* HotPETs has no official proceedings: this document compiles selected papers and is published online only to facilitate discussions during the workshop. Selected papers are *not* included in PETS proceedings not to preclude later publication, of a full paper, in another venue.

Contents

Foreword 3

Program 4

Dr. Balachander Krishamurthy. Internet privacy: Towards more transparency (Invited Talk) 5

Greg Norcie, Kelly Caine and Jean Camp. Eliminating Stop-Points in the Installation and Use of Anonymity Systems: A Usability Evaluation of the Tor Browser Bundle 6

Luca Invernizzi, Christopher Kruegel and Giovanni Vigna. Message In A Bottle: Sailing Past Censorship 19

Indrajeet Singh, Michael Butkiewicz, Harsha Madhyastha, Srikanth Krishnamurthy and Sateesh Addepalli. Building a Wrapper for Fine-Grained Private Group Messaging on Twitter 34

Lukasz Olejnik, Claude Castelluccia and Artur Janc. Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns 48

Nevena Vratonjic, Vincent Bindschaedler, Kévin Huguenin and Jean-Pierre Hubaux. Location Privacy Threats at Public Hotspots 64

Mishari Almishari and Gene Tsudik. Exploring Linkability of User Reviews 79

Kurt Partridge, Manas A. Pathak, Ersin Uzun and Cong Wang. PiCoDa: Privacy-preserving Smart Coupon Delivery Architecture 94

Foteini Baldimtsi, Gesine Hinterwalder, Andy Rupp, Anna Lysyanskaya, Christof Paar and Wayne P. Burleson. Pay as you go 109

Michael Brennan. Perspectives on Academic Impact from Inside the Federal Trade Commission 118

Foreword

As the amount and sensitivity of personal information disseminated on the Web increase, so do related privacy concerns (or so we like to nag about). The ambition of HotPETs 2012 – the 5th Workshop on Hot Topics in Privacy Enhancing Technologies – is to foster new ideas, spirited debates, and controversial perspectives on privacy (and lack thereof). Now at its 5th edition, HotPETs celebrates this anniversary with some fun facts:

1. This year, a record number of 20 papers were submitted.[1] They were evaluated on the basis of research significance, novelty, and "hotness" of their topic. Each submission was reviewed by both co-chairs and external reviewers were asked to help in case of conflicts. Selection of papers was much harder than expected, due to the appreciably high quality of submissions. In the end, we selected 9 papers. [2]

2. HotPETs 2012 features an excellent invited speaker – Dr. Balachander Krishamurthy – and an outstanding program. Selected papers are presented in 3 sessions: (1) *Censoring Censorship: dream of reality?*, (2) *Privacy Erosion: New results and some bad news*, and (3) *Privacy Protection: Finally some good news!*

3. The 5th HotPETs edition keeps good traditions, such as the afternoon ice cream break, and introduces a new one: the best presentation award – a symbolic fun-seeking privacy-geek-oriented recognition that stimulates speakers to engage the audience (or the other way around, we are not completely sure).

We hope that HotPETs 2012 lives up to the expectations and brings up informal, yet inspiring, thought-provoking discussion within the world of Privacy Enhancing Technologies.

Finally, we wish to thank authors of submitted papers, speakers, attendees, keynote speaker Balachander Krishnamurthy, external reviewers, PETS Program Chairs Simone Fischer-Huebner and Matthew Wright, and, especially, webmaster Jeremy Clark and General Chair Carmela Troncoso (we really bugged the heck out of them!).

Have a great time!

Emiliano De Cristofaro
Julien Freudiger
Vigo, July 13, 2012

[1] Claiming the record number of submissions, although unsubstantiated and not supported by any scientific evidence, is utterly justified by the following pie chart: http://dilbert.com/strips/comic/2009-03-07/

[2] Are you really looking for the acceptance rate?

Program

5th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2012)
Vigo, Spain, July 13, 2012.

09:45	10:00	Initial Remarks

Session 1: Censoring Censorship – Dream or Reality?

10:00	10:25	Eliminating Stop-Points in the Installation and Use of Anonymity Systems: A Usability Evaluation of the Tor Browser Bundle. *Greg Norcie, Kelly Caine and Jean Camp*
10:25	10:50	Building a Wrapper for Fine-Grained Private Group Messaging on Twitter. *Indrajeet Singh, Michael Butkiewicz, Harsha Madhyastha, Srikanth Krishnamurthy and Sateesh Addepalli*
10:50	11:15	Message In A Bottle: Sailing Past Censorship. *Luca Invernizzi, Christopher Kruegel and Giovanni Vigna*
11:15	11:45	**Coffee Break**
11:45	1:00	**Keynote Talk**: Internet privacy: Towards more transparency *Balachander Krishamurthy, AT&T Labs–Research*
1:00	2:30	**Lunch**

Session 2: Privacy Erosion – New results and some bad news

2:30	2:55	Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns. Lukasz Olejnik, Claude Castelluccia and Artur Janc
2:55	3:20	Location Privacy Threats at Public Hotspots. *Nevena Vratonjic, Vincent Bindschaedler, Kévin Huguenin and Jean-Pierre Hubaux*
3:20	3:45	Exploring Linkablility of User Reviews. *Mishari Almishari and Gene Tsudik*
3:45	4:30	**Coffee Break**

Session 3: Privacy Protection – Finally some good news!

4:30	4:55	PiCoDa: Privacy-preserving Smart Coupon Delivery Architecture. *Kurt Partridge, Manas A. Pathak, Ersin Uzun and Cong Wang*
4:55	5:20	Pay as you go. *Foteini Baldimtsi, Gesine Hinterwalder, Andy Rupp, Anna Lysyanskaya, Christof Paar and Wayne P. Burleson*
5:20	5:45	Perspectives on Academic Impact from Inside the Federal Trade Commission. *Michael Brennan*
5:45	6:00	Closing Remarks

Internet privacy: Towards more transparency (Invited Talk)

Balachander Krishnamurthy

AT&T Labs–Research

Abstract

Internet privacy has become a hot topic recently with the radical growth of Online Social Networks (OSN) and attendant publicity about various leakages. For the last several years we have been examining aggregation of user's information by a steadily decreasing number of entities as unrelated Web sites are browsed. More recently we have found leakage of user's data to the same aggregating entities as a result of interaction with OSNs. I will present results from several studies on leakage of personally identifiable information (PII) via Online Social Networks (both traditional and mobile OSNs) and popular non-OSN sites. Linkage of information gleaned from different sources presents a challenging problem that has captured the attention of technologists, privacy advocates, government agencies, and the multi-billion dollar online advertising industry.

I will present the current status of both technical and non-technical attempts to ameliorate the problem. Economics might hold the key in increasing transparency of the largely hidden exchange of data in return for access of so-called free services.

Speaker's biography:

Balachander Krishnamurthy is a member of technical staff at AT&T Labs–Research. His focus of research of is in the areas of Internet privacy, Online Social Networks, and Internet measurements. He has authored and edited ten books, published over 80 technical papers, holds thirty five patents, and has given invited talks in thirty five countries.

He co-founded the successful Internet Measurement Conference and the Workshop on Online Social Networks. He has been on the thesis committee of several PhD students, collaborated with over seventy five researchers worldwide, and given tutorials at several industrial sites and conferences.

His most recent book "Internet Measurements: Infrastructure, Traffic and Applications" (525pp, Wiley, with Mark Crovella), was published in July 2006 and is the first book focusing on Internet Measurement. His previous book "Web Protocols and Practice: HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement" (672 pp, Addison-Wesley, with Jennifer Rexford) is the first in-depth book on the technology underlying the World Wide Web, and has been translated into Portuguese, Japanese, Russian, and Chinese.

Bala is homepageless and not on any OSN but many of his papers can be found **here**.

Eliminating Stop-Points in the Installation and Use of Anonymity Systems: a Usability Evaluation of the Tor Browser Bundle

Greg Norcie, Kelly Caine, L Jean Camp

Indiana University

Abstract

Broad-based participation in anonymizing technology can be hindered by a lack of usability. Since systems such as Tor give $\frac{1}{n}$ anonymity, the existence of stop-points is of particular concern because wider adoption of Tor equals better anonymity for all users of the system. Stop points are places in an interface at which an individual is at risk from being presented with a situation which may prevent them from moving forward with installation or use. Sixty four percent of users in our study encountered at least one stop-point while using the Tor Browser Bundle. While each stop-point may be minor in isolation, the cumulative effect is great and has the potential to harm the overall anonymity of the system. In this paper we enumerate these stop points and detail specific design changes that may address them. We also provide more generic but generalizable recommendations for the design of anonymity systems. Taken together, these suggestions may enhance usability and thus anonymity of individual users as well as system level anonymity.

1 Introduction

Tor is an anonymity network that uses onion routing [11] to prevent third parties from observing a user's web connection. The term onion routing refers to Tor's layered encryption (analogous to the layers in an onion), which prevents third parties from observing the content or linking the source and destination in the internet traffic of a participant in the Tor network.

As Dingledine and Mathewson[10] point out, there is a strong network effect present in anonymity networks like Tor. In any system where anonymity is $\frac{1}{n}$, additional users increase the anonymity of the system. Thus, barriers that reduce the number of people who adopt an anonymity system like Tor, reduce the anonymity all users of the system receive. Therefore, any increase in the usability of Tor that reduces barriers to adoption and increases the anonymity Tor provides. While many papers have examined the technical aspects of Tor's security [21][19][18], relatively few papers have examined Tor's usability (see [25] and [6] for notable exceptions). Our work is further distinguished by the dual focus on both flaws in the Tor interface, as well as their resolutions.

Tor can be run in a variety of configurations. Previous work [6] offered a cognitive walkthrough (but no subsequent lab study) of Tor, and suggested that Torpark, a Firefox variant with Tor incorporated into the browser was the most usable. Tor later released an official "Tor Browser Bundle" (TBB) which similarly contains a standalone Firefox variant with Tor built in, a design change backed up by the data collected by Clark et al.

Our work complements and extends existing work on improving the usability of Tor by identifying"stop-points" in the installation and use of the Tor Browser Bundle. Stop-points are places in an interface where a user may find themselves facing a requirement for user action such that the user is unable to proceed. [14]

However, a stop point does *not* necessarily prevent a user from going forward — any point at which the user is going forward while confused about the functionality or state of the system is also a stop point.

In this paper, we evaluate stop points during both the installation and the use of the Tor Browser Bundle, in a lab study with 25 participants. Based on this study, we utilize our results to specify immediate changes that can be made to the Tor Browser Bundle Interface. We also present a set of design recommendations which can be applied not only to the Tor Browser Bundle, but are also generalizable to other anonymity systems.

In the next section we describe related work. We place our study in the larger context of three main bodies of work: usable security, privacy enhancing technologies, and design heuristics. Given the breadth of these domains, these descriptions are necessarily brief. Following this overview of related work we describe our method in Section 3. In Section 4 we enumerate our findings. We then explicate these findings and their implications in Section 5, including both a general discussion that places the results in the context of related work and suggests design recommendations both specifically for Tor as well as for anonymity systems generally. Finally, we outline limitations of our study and then conclude in Section 7.

2 Related Work

There are three bodies of literature that inform this work. The three domains on which we draw for this work are laboratory evaluation of usability (particularly for security and privacy); privacy enhancing technologies; and design heuristics.

2.1 Usable Security in the Laboratory

In Whitten and Tygar's "Why Johnny Can't Encrypt" [25], one of the first usable security papers published, Whitten and Tygar performed a cognitive walkthrough and laboratory study examining the usability of PGP 5.0. Whitten and Tygar state that security software is usable if users are aware of the tasks they need to perform, are able to successfully perform said tasks without making dangerous errors, and are comfortable enough with a security interface to continue using it. Based on their evaluation of PGP 5.0, Whitten and Tygar note several problematic properties of computer security.

The major issues Whitten and Tygar noted are that users are unmotivated to protect their information, that it is difficult to produce abstractions for many security functions, that there is a lack of feedback when performing security tasks. Whiten and Tygar also describe what they call the "barn door property" - that data, once lost, cannot be reclaimed (evoking the futility of locking a barn once the animals have escaped). Whitten and Tygar also draw on economics of security literature (eg [24]) to apply the "weakest link property'. The weakest link property states that a security system is only as strong as it's weakest link. Often the weakness in a security system is the lack of usability in the interface, or the failure of an interface to match the mental model of the users. These findings have been reified in later work including [8] and [4], respectively.

Whitten and Tygar concluded that interfaces must be designed to guide and inform security decisions. Our work draws on this but targets on specific points of potential failure, and lessso on macro observations as described by Whitten et al.

Maxion and Reeder implemented a similar laboratory examination of usability of access control. As with our work, they determined that individuals who may believe that they have implemented the correct settings are not consistently correct. Indeed, as few as 25% were able to complete a basic ACL task using XPFP [23].

Other analyses of usable security or privacy include Inglesant and Sasse, who found that while individuals do in fact care about security, password policies are too inflexible and not designed for humans

— security policies are too inflexible to match their capabilities [16]. A follow-up study illustrated that graphical passwords had similar difficulties [3].

Engelman et al. [13] examined whether users would tolerate security delays in a Mechanical Turk task asking them to test a new web based PDF viewer, reading a document and reporting the frequency that a word appeared in a particular document, and then presenting them with a delay. Engleman et al. found that found that users were much more likely to cheat on the Mechanical Turk task when presented with either a non-security explanation for the delay, such as a simple loading bar, or a vague security explanation, such as changing the loading bar to simply read "Performing security scan." Conversely, users were less likely to cheat when given a concrete security explanation — that online documents often contain viruses and that the PDF reader was performing a virus scan.

2.2 Privacy Enhancing Technologies

The idea of passing an encrypted message between series of intermediate nodes was first discussed by Chaum [5]. The classic Chaumian mixnet is similar to the Tor network in that each packet is subject to a series of encrypting operations so that no node in the network can determine both the initiator and target of a communication. Similarly, each message has a theoretical requirement for three hops. However, in mix networks packets are delayed and reordered (thus preventing timing attacks but increasing latency) as opposed to there being a stable path for any session.

There have been several high anonymity, high latency systems such as MixMinion [9]. Zero Knowledge Systems' "Freedom" was released in beta in early 2000. Tarzan provided transport-layer anonymization using a decentralized, distributed Chaumian mix. [15] Freenet offered anonymous and resilient publishing using distributed content and shared keys; however, the initiator of a request could be identified. [7] Free Haven offered anonymous storage as well as protecting against traffic analysis. [7]

Yet none of these platforms ever gained popularity. Indeed, many of them never went beyond laboratory demonstration projects. Onion routing was first presented in 1998. [22] The second generation onion router work was published in 2004, when Dingledine was with Free Haven.[11] Before 2000 the majority of anonymizing systems that were used in practice were single hop proxies, e.g. [2]. For cryptographic PETs, Tor is unique in its acceptability and adoption, with the number of users in the hundreds of thousands. The latest instantiation of a more usable version of Tor combines the proxy with Tor along with a browser in one package called the Tor Browser Bundle (TBB). This simplified installation and interface has the potential to expand Tor to a broader user base.

2.3 Design Heuristics

Usability design heuristics predate the field of usable security by many years. Molich and Nielsen [20] wrote a widely cited set of design heuristics for human-computer interaction. Whitten and Tygar [25] expanded on this work, pointing out that secure systems have additional usability requirements. Users of privacy enhancing technologies need to be aware of the security tasks they need to perform. The user must then be able to perform these task(s) without making any dangerous errors, and then be comfortable enough with the interface to continue using it.

Clark, Oorschot, and Adams performed a cognitive walkthrough of several early Tor interfaces, but did not follow up with a lab study.The authors concluded that Torpark (a self contained browser bundle similar to the TBB our lab study evaluates) was the most usable option for novice Tor users. [6]

3 Methodology

3.1 Recruitment and Procedures

We recruited 25 undergraduates from a large midwestern university. Students were given lab credit for participating in the experiment. Students who were not comfortable participating were given the option to instead write a one page essay on Tor's basic functionality. Our study was approved by the Indiana University Institutional Review Board.

All participants were seated at VMware image of Windows 7, and given an instruction sheet, as well as a short questionnaire where they were instructed to record any usability issues they encountered throughout the study. Users also had their on-screen actions using screen capture software. The instruction sheet that was handed out provided users with the URL for the Tor Project (`http://torproject.org`). Users were then directed by the instruction sheet to download the Tor Browser Bundle (TBB), run the TBB, and use the TBB to download a new desktop background for their lab machine.

Before being given their instructions, users were informed that the experiment was a usability experiment, and thus the normal rules for a lab did not apply. Normally, students in a lab are expected to complete their tasks with minimal aid from the instructor, except for clarification of instructions, working through any complexities on their own. Before the experiment, users were briefed that their instructions were purposefully vague, and that if they were unclear how to proceed at any time, they should raise their hand so that the experimenter could assist them. Participants were also given the definition of stop points presented earlier in this paper. The participants were told that the lab was designed to find "stop-points", and that participants should raise their hands if they encountered a "stop point" that they could not proceed beyond. A post-task survey also asked users if they had encountered any issues. Users who raised their hands were instructed to note their issue on their post-task survey, and then told how to proceed past the stop-point. Finally, we recorded each user's screen, and were able to go back and examine their recordings if a user's textual response contained ambiguity.

3.2 Ecological Validity

Normally, in usable security experiments, experimenters strive to find non-expert users. However, it is the author's belief that typical users of Tor are moderately security savvy. Furthermore, since our users were security savvy, it actually strengthens the authors' argument that the interface is less usable than is desirable if even security experts make errors when using it.

Furthermore, since normal users of Tor are aware that they are engaging in security task, we did not attempt to hide the nature of the task from participants as one might do in say, a phishing study.

3.3 Sample Information

Demographic information was collected from all users during the exit survey. The sample was 88% male (22/25). Participant ages ranged from 20 to 37, with a median of 22.7 and a mode of 21. Participants were asked if they had heard of Tor. They were also asked to rank their familiarity with Tor, as well as their familiarity with computer security on a 1 to 7 scale (1 meaning "not at all familiar" and 7 meaning "very familiar"). While 84% (21/25) of users had heard of Tor, the users were by no means Tor experts. When asked "How familiar are you with Tor?", users responded with an average of 2.13 on a 7 point scale. Users were slightly more familiar with computer security. When asked "How familiar are you with computer security?", users reported an average of 4.5 on a 7 point scale.

3.4 Coding

Each post-task survey asked, in addition to demographic questions, two free response questions:

1. "Did you encounter any problems when installing the Tor Browser Bundle?"
2. "Did you encounter any problems when using the Tor Browser Bundle?"

The 25 study participants reported a total of 41 stop-points in these two free response questions. Two coders independently coded the results to these questions, assigning each complaint to one of of seven mutually exclusive categories. Categories were generated post-hoc after a holistic evaluation of the free response questions:

A.) Long launch time: The user noticed a lag between clicking the icon to start the Tor Browser Bundle, and the TBB window opening.

B.)Browsing Delay: Browsing through the TBB had a noticeable lag.

C.) Download Clarity: User wasn't sure where on website to download the TBB

D.) Window discriminability: User wasn't sure which window was TBB and which was a normal browser.

E.) Archive confusion: Problems unzipping the TBB package.

F.) Icon Salience: Problems finding the icon to start the file ("Start Tor Browser")

G.) Security Measure Confusion: Security measures taken by the TBB (such as redirecting from Google CAPTCHA, to DuckDuckGo) confused users.

Final intercoder agreement was calculated using Cohens Kappa, a method of calculating observer agreement of categorical data that accounts for agreements due to chance. Overall intercoder agreement between the two coders was Cohens Kappa = .72. Kappas of .61 - .80 are considered substantial. [17] After the first pass of coding, there was 100% coder agreement.

Now we will move onto the heart of our paper. Details on which of the issues that these categories describe were most prevalent are discussed in section 4 We then discuss the design implications of these findings, and present a set of design heuristics based on them in section 5.

4 Results

As reported in 3 our users were mostly moderately security savvy college students. Eighty percent of our users (20/25) were college aged (18-22) and all were currently enrolled an at least one computer science or informatics class. The study participants reported being familiar with computer security on a 7 point likert scale with 7 point likert scale with 1 being "Not at all familiar" and 7 being "Very familiar", the average participant scored a 4.5 when asked "How familiar are you with computer security? However,while 84% (21/25) of participants had heard of Tor, the average familiarity was only 2.13 on a 7 point likert scale. Table 1 summarizes the demographics of our participants.

Gender	
Male	22
Female	3
Age	
18–22	20
22+	5
Class Year	
Freshman (1)	0
Sophomore (2)	1
Junior (3)	9
Senior (4)	15
Other	0
Heard of tor?	
Yes	21
No	4
Familiar w/ Tor?	
1 (Not at all familiar)	11
2	5
3	7
4	0
5	2
6	0
7 (Extremely Familiar)	0
Familiar w/ computer security?	
1 (Not at all familiar)	0
2	0
3	1
4	9
5	10
6	3
7(Extremely Familiar)	0

Table 1: Results summary

We found that 36% of users (9/25) reported having no problems installing or using the Tor Browser Bundle. The remaining 16 users reported a total of 41 individual issues. As we can see from Table 2, the majority of the issues users encountered centered around launch time, browser delay, and window discriminability.

Category	Description	N	%
Long launch time	The user noticed a lag between clicking the icon to start the Tor Browser Bundle, and the TBB window opening.	13	40.6%
Browsing delay	Browsing through the TBB had a noticable lag.	6	18.8%
Window discriminability	User wasn't sure which window was TBB and which was a normal browser.	4	12.5%
Archive Confusion	Problems unzipping the TBB package.	4	12.5%
Icon salience	Problems finding the icon to start the file ("Start Tor Browser")	3	9.4%
Security Measure Confusion	Security meausures taken by the TBB (such as redirecting from Google CAPTCHA, to DuckDuckGo) confused users.	3	9.4%
Download Clarity	User wasn't sure where on website to download the TBB	3	9.4%

Table 2: Type of problems Tor problems encountered

Now that we have established what problems have been experienced by our participants, we will discuss the implications of these findings in section 5, and present a set of design principles based on them.

5 Discussion

Based on our results, we will present two sets of design implications. First, we will present a set of design recommendations specific to the Tor Browser Bundle. Then, we will generalize our results to provide a generalized set of design heuristics for creators of anonymity systems.

5.1 Tor Design Issues and Solutions

Issue: Long Launch Time (13/41)

Many users noticed a long delay between clicking "Start Tor Browser Bundle" and the Tor Browser Bundle opening. A typical scenario would be that the user would click on "Start Tor Browser Bundle". At this point, Vidalia (the graphical controller for Tor, whose interface confused users) appeared. Many users incorrectly assumed after 30 seconds or so that all their internet traffic was anonymized and proceeded to open Firefox or Internet Explorer.

Solution(s):

Users of Tor are likely willing to trade speed for privacy. Simply taking steps to inform users that the TBB may take a while to open and that such delay is normal could substantially alter a user's perception of the

Tor Browser Bundle. As Molich and Nielsen point out, systems need to provide feedback to users. [20] The typical user assumes that if a program fails to respond within a certain time frame, that either a process has run in the background, or an error has occurred. By providing an informative dialog instructing users to wait for the browser window to open, the confusion Tor users experience can be avoided.

Issue: Browsing Delay (6/41)

Many users noted that browsing with Tor was slower than browsing over a typical internet connection.

Solution(s):

Inform users (on the download page, during installation, and/or on the initial home page) that Tor may be slower than traditional connections, due to its traffic flowing through a series of intermediary nodes. Tor has traditionally been slower than a typical internet connection[12]. Building on Egelman's work with Mechanical Turk users,[13] the authors theorize that users are given *realistic* expectations about Tor's speed, they will not attribute this lack of speed to an error. In fact, this type of information may instead help users develop a more accurate mental model. Thus, insted of becoming frustrated, users may instead picture their packets traversing several nodes as they wait, thus gaining a sense of security from the delays sometimes introduced by Tor.

Issue: Window Discriminability (4/41)

Some users (many of whom also experienced a long launch time) had trouble discriminating which window was the Tor Browser Bundle and which was a normal browser window. This caused users to (erroneously) use a non-protected Firefox session to perform study tasks.

Solution(s):

The Tor Browser Bundle could consider using a more distinct program icon. The TBB could also change the browser's chrome to visually separate it from other browser windows with custom colors and icons matching those found on the TBB's default homepage (`https://check.torproject.org`), as shown in figure 1.

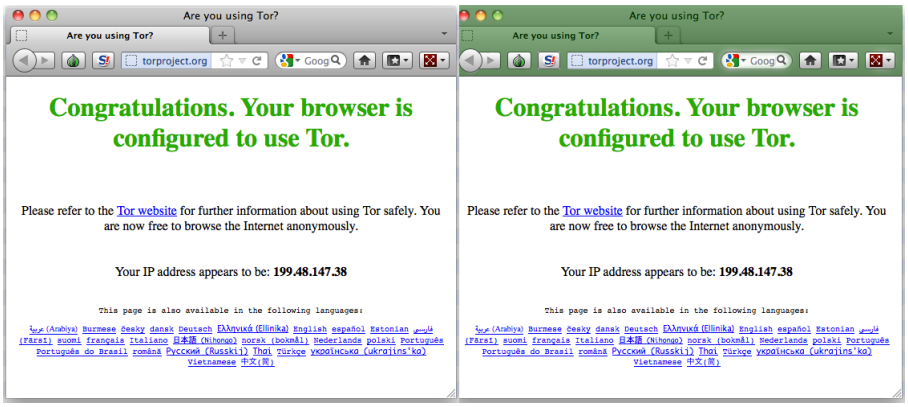


Figure 1: TBB interface before (left) and after (right) enhanced discriminability changes

Finally, the Tor Browser Bundle could hiding the Vidalia control panel, since having two applications with two separate browser windows and two separate icons often confuses users. Finally, if user accidentally closes the Tor Browser, Vidalia continues to inform the user that they are connected to the Tor network, as pictured in figure 2. This caused several users who accidentally closed the Tor Browser to erronously believe that *all* browser traffic was being anonymized.

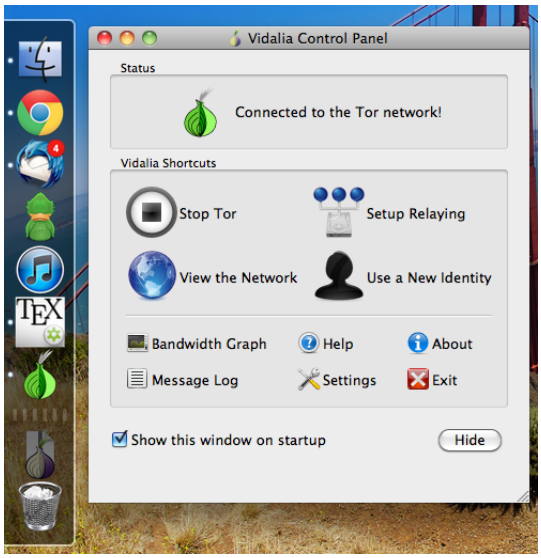


Figure 2: Vidalia reporting a connection to the Tor network, even though the Tor Browser window has been closed.

Issue: Icon Salience (4/41)

Some users were unclear how to start the Tor Browser Bundle, or thought that the Tor Browser Bundle would start automatically. The participant would not realize that the "Start Tor Browser Bundle" This could lead to serious errors, such as when one participant assumed after unzipping the TBB that all traffic was now anonymous, and proceeded to attempt to complete the study tasks using an unprotected system browser.

Solution(s):

the Tor Browser Bundle could place an icon on the desktop / dock. The Tor browser bundle could note at some point between downloading and installation that the user must click "Start Tor Browser Bundle" to begin. Alternatively, the Tor Browser Bundle could launch automatically after installation.

Issue: Download Clarity (4/41)

Some users were unsure which package to download and/or accidentally downloaded the wrong operating system's version of the Tor Browser Bundle.

Solution(s):

The download page could provide larger logos for each operating system, along with larger, bolded text describing which operating system a given package is for.

Issue: Security Measure Confusion (3/41)

Some security measures that the TBB takes, such as redirecting to Google searches to DuckDuckGo, and disabling certain types of active content confused non-technical users who did not understand why a given action had been redirected or a pop up box had been generated.

Solution(s):

Prior to performing any redirects, the TBB could provide a jargon free explanation of why a security measure is being taken. For example, before redirecting to DuckDuckGo, a pop up could appear and state: *"Google keeps a record of your history. Using DuckDuckGo will allow you to search anonymously."*

Issue: Archive Confusion (3/41)

Some users expected a guided "wizard" installer, and did not realize they had to click on "Start Tor Browser Bundle" once unzipping had occurred, leading to confusion.

Solution(s):

This issue is not necessarily a problem with Tor, but as we discuss later in our design heuristics, installation of the TBB is a prerequisite for *using* the Tor Browser Bundle. While the TBB developers cannot control the usability of the host operating system, a prominent note could be made on the download page that users will need to unzip the Tor Browser Bundle prior to using it.

5.2 Summary of Design Implications for Tor

In the previous section we described seven stop points and potential solutions based on our results in section 4. Each design recommendation was discussed in the context of the coding category that documented it. We found that vast majority of the issues were created by long launch times, browsing delay, and window discriminability.

5.3 Potential Design Heuristics For Anonymity Systems

Based on the above issues, we can arrive at a set of general design recommendations that may generalize to other anonymity systems. Any system that allows users to "hide in the crowd" can benefit from these heuristics, which aim to maximize adoption of a given anonymity system. Earlier we described a set of specific solutions — now we present more general heuristics applicable to a wide body of anonymity systems.

Heuristic 1: Installation precedes operation: Even the most well designed user interface is useless if the user never reaches it. The authors of anonymity software should strive to assist users who are installing the software. Download pages should try and make educated guesses as to what operating system a user is running, and provide users with simple heuristics for determining their operating system. For example, next to a link to download the Windows version of an anonymity software package, the page could state "If your computer is not a Mac, you probably want this version."

Heuristic 2: Ensure users are aware of trade-offs: Today's users have come of age in a time of widespread broadband adoption. Delays longer than a few seconds may cause users to question whether their connection is faulty. While ideally anonymity software should strive to deliver content with as little latency as possible, users of anonymity software are usually willing to trade speed for privacy. However, the software must provide feedback to the user to let them know that a given operation has

not failed. Just like a user is willing to accept a slower connection via a crowded internet cafe wifi network, a user is willing to accept a delay in exchange for anonymous communication.

Heuristic 3: Say why, not how: Sometimes an anonymity system must take a security measure, such as redirecting away from a site which may leak identity information, or disabling browser features such as cookies or javascript. Users desire to be told *why* a given security precaution is being taken. These explanations should avoid technical jargon and use real world metaphors whenever possible. Users who wish to understand at a deeper level should be given the option to drill down to a more detailed technical explanation.

6 Limitations

There are several limitations to this study. First, our sample was skewed heavily towards undergraduate males. Our sample size (N = 25), could have been larger. Both of these factors may affect the generalizability of our results. Also, while this paper provides several design heuristics, these principles have not been tested in the laboratory, and future work is needed to verify that indeed enhance the usability of any anonymity systems. For example, a future study should create a new version of the Tor Browser Bundle incorporating this paper's design recommendations, and test whether participants actually find the interface more usable when these changes have been implemented.

7 Conclusions

Based on our survey, we have discovered a number of usability issues in the Tor Browser Bundle. As Back et al. succinctly state "in anonymity systems usability, efficiency, reliability and cost become security objectives because they affect the size of the user base which in turn affects the degree of anonymity it is possible to achieve."[1].

We noted that the long launch time, browsing delay, and window discriminability were the issues most often cited by participants. Based on these issues, we presented a set of three design heuristics to help minimize usability issues in anonymity systems.

We said that "installation precedes operation" precedes operation, since if the installation of an anonymity system frustrates the user, they may never reach the UI, no matter how well designed it is. We also suggested that makers of anonymity systems ensure users are aware of the speed trade-offs in anonymity systems, and set appropriate expectations. Finally, with our "Say why, not how." heuristic, we encourage developers to explain why security measures which impact the user experience are taken, and that these explanations avoid technical jargon.

By applying these design heuristics, developers can help make the Tor Browser Bundle (or any similar anonymity system) more usable, and thus, more secure.

8 Future Work

While this work presented a set of heuristics for designing anonymity systems, as well as a few diagrams giving examples of possible changes to the Tor Browser Bundle, we neglected to present a unified design incorporating our suggestions.

Furthermore, while have presented a number of suggestions, but we have not experimentally verified them. Thus future work could continue along two lines. First, we could develop a new Tor Browser Bundle interface, taking into account the findings in this paper. After developing this new interface, we could verify via lab study that our interface changes were effective in increasing the usability of the Tor Browser Bundle.

References

[1] A. Back, U. Möller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In *Information Hiding*, pages 245–257. Springer, 2001.

[2] J. Boyan. The anonymizer. *Computer-Mediated Communication (CMC) Magazine*, 1997.

[3] S. Brostoff, P. Inglesant, and M. Sasse. Evaluating the usability and security of a graphical one-time pin system. In *Proceedings of the 24th BCS Interaction Specialist Group Conference*, pages 88–97. British Computer Society, 2010.

[4] L. J. Camp. Bringing mental models to privacy and security. *IEEE Technology And Society Magazine*, 28(3):37–46, 2009.

[5] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[6] J. Clark, P. Van Oorschot, and C. Adams. Usability of anonymous web browsing: An examination of tor interfaces and deployability. In *Proceedings of the 3rd symposium on Usable privacy and security*, pages 41–51. ACM, 2007.

[7] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.

[8] L. Cranor and S. Garfinkel. *Security and Usability*. O'Reilly Media, Inc., 2005.

[9] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 2–15. IEEE, 2003.

[10] R. Dingledine and N. Mathewson. Anonymity loves company: Usability and the network effect. In *Proceedings of the Fifth Workshop on the Economics of Information Security (WEIS 2006), Cambridge, UK, June*, 2006.

[11] R. Dingledine, N. Mathewson, and P. Syverson. Tor: the second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[12] R. Dingledine and S. Murdoch. Performance improvements on tor or, why tor is slow and what were going to do about it. 2009.

[13] S. Egelman, A. Acquisti, D. Molnar, C. Herley, N. Christin, and S. Krishnamurthi. Please continue to hold an empirical study on user tolerance of security delays. 2010.

[14] H. Elmore. Designing translucent security: Insights from a usability evaluation of pgp desktop. 2009.

[15] M. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 193–206. ACM, 2002.

[16] P. Inglesant and M. Sasse. The true cost of unusable password policies: password use in the wild. In *Proceedings of the 28th international conference on Human factors in computing systems*, pages 383–392. ACM, 2010.

[17] J. Landis and G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, pages 159–174, 1977.

[18] K. Loesing, S. J. Murdoch, and R. Dingledine. A case study on measuring statistical data in the Tor anonymity network. In *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010)*, LNCS. Springer, January 2010.

[19] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov. Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting. In *Proceedings of the 18th ACM conference on Computer and Communications Security (CCS 2011)*, October 2011.

[20] R. Molich and J. Nielsen. Improving a human-computer dialogue. *Communications of the ACM*, 33(3):338–348, 1990.

[21] S. J. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, SP '05, pages 183–195, Washington, DC, USA, 2005. IEEE Computer Society.

[22] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. *Selected Areas in Communications, IEEE Journal on*, 16(4):482–494, 1998.

[23] R. Reeder and R. Maxion. User interface dependability through goal-error prevention. In *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*, pages 60–69. Ieee, 2005.

[24] H. Varian. *Microeconomic analysis*, volume 2. Norton New York, 1992.

[25] A. Whitten and J. Tygar. Why johnny cant encrypt: A usability evaluation of pgp 5.0. In *Proceedings of the 8th USENIX Security Symposium*, volume 99, 1999.

Message In A Bottle: Sailing Past Censorship

Luca Invernizzi, Christopher Kruegel, Giovanni Vigna

UC Santa Barbara

Abstract

Exploiting recent advances in monitoring technology and the drop of its costs, authoritarian and oppressive regimes are tightening the grip around the virtual lives of their citizens. Meanwhile, the dissidents, oppressed by these regimes, are organizing online, cloaking their activity with anti-censorship systems that typically consist of a network of anonymizing proxies. The censors have become well aware of this, and they are systematically finding and blocking all the entry points to these networks. So far, they have been quite successful. We believe that, to achieve resilience to blocking, anti-censorship systems must abandon the idea of having a limited number of entry points. Instead, they should establish first contact in an online location arbitrarily chosen by each of their users. To explore this idea, we have developed Message In A Bottle, a protocol where any blog post becomes a potential "drop point" for hidden messages. We have developed and released a proof-of-concept application using our system, and demonstrated its feasibility. To block this system, censors are left with a needle-in-a-haystack problem: Unable to identify what bears hidden messages, they must block everything, effectively disconnecting their own network from a large part of the Internet. This, hopefully, is a cost too high to bear.

1 Introduction

The revolutionary wave of protests and demonstrations known as the *Arab Spring* rose in December 2010 to shake the foundations of a number of countries (e.g., Tunisia, Libya, and Egypt), and showed the Internet's immense power to catalyze social awareness through the free exchange of ideas. This power is so threatening to repressive regimes that censorship has become a central point in their agendas: Regimes have been investing in advanced censoring technologies [37], and even resorted to a complete isolation from the global network in critical moments [10]. For example, Pakistan recently blocked Wikipedia, YouTube, Flickr, and Facebook [23], and Syria blocked citizen-journalism media sites [44].

To sneak by the censorship, the dissident populations have resorted to technology. A report from Harvard's Center for Internet & Society [38] shows that the most popular censorship-avoidance vectors are web proxies, VPNs, and Tor [13]. These systems share a common characteristic: *They have a limited amount of entry points*. Blocking these entry points, and evading the blocking effort, has become an arms race: China is enumerating and banning the vast majority of Tor's bridges since 2009 [50], while in 2012, Iran took a more radical approach and started blocking encrypted traffic [47], which Tor countered the same day by deploying a new kind of traffic camouflaging [46].

In this paper, we take a step back and explore whether it is possible to design a system that is so pervasive that it is impossible to block without disconnecting from the global network.

Let's generalize the problem at hand with the help of Alice, a dissident who lives on the oppressed country of Tyria and wants to communicate with Bob, who lives outside the country. To establish a communication channel with Bob in any censorship-resistant protocol, Alice must know *something* about Bob. In the case of anonymizing proxies or mix-networks (e.g., Tor), this datum is the address of one of the entry points into the network. In protocols that employ steganography to hide messages in files uploaded to media-hosting sites (such as Collage [8]) or in network traffic (such as Telex [54]), Alice must know the location of the first rendezvous point.

The fact that Alice has to know something about Bob inevitably means that the censor can learn that too (as he might *be* Alice). Bob cannot avoid this without having some information to distinguish Alice from the

censor (but this becomes a chicken-and-egg-problem: How did Bob get to know that?). We believe that this initial *something* that Alice has to know is a fundamental weakness of existing censorship-resistant protocols, which forms a crack in their resilience to blocking. For example, this is the root cause of the issues that Tor is facing with distributing bridge addresses to its users without exposing them to the censor [45]. It is because of this crack that China has been blocking the majority of Tor traffic since 2009 [50]: the number of entry points is finite, and a determined attacker can enumerate them by claiming to be Alice.

In Message In A Bottle (MIAB), we have designed a protocol where Alice knows the *least possible* about Bob. In fact, we will show that Alice must know Bob's public key, and nothing else. Alice must know at least Bob's public key to authenticate him and be sure she is not talking to a disguised censor. However, contrary to systems like Collage and Telex, there is *no rendezvous point* between Alice and Bob.

This may now sound like a needle-in-a-haystack problem: If neither Alice nor Bob know how to contact the other one, how can they ever meet on the Internet? In order to make this possible and reasonably fast, MIAB exploits one of the mechanisms that search engines employ to generate real-time results from blog posts and news articles: *blog pings*. Through these pings, Bob is covertly notified that a new message from Alice is available, and where to fetch it from. We will show that, just like a search engine, Bob can monitor the majority of the blogs published on the entire Internet with limited resources, and in quasi real-time. In some sense, every blog becomes a possible meeting point for Alice and Bob. However, there are over 165 million blogs online [5], and since a blog can be opened trivially by anybody, for our practical purposes they constitute an infinite resource. We have estimated that, to blacklist all the possible MIAB's drop points, the Censor should block 40 million fully qualified domain names, and four million IP addresses (as we will show in Section 5.3, these are conservative estimates). For comparison, blacklisting a single IP address would block Collage's support for Flickr (the only one implemented), and supporting additional media-hosting sites requires manual intervention for each one.

In MIAB, Alice will prepare a message for Bob and encrypt it with his public key. This ciphertext will be steganographically embedded in some digital photos. Then, Alice will prepare a blog post about a topic that fits those photos, and publish it, along with the photos, on a blog of her choosing. Meanwhile, Bob is monitoring the stream of posts as they get published. He will recognize Alice's post (effectively, the bottle that contains the proverbial message), and recover the message.

We envision that MIAB might be used to bootstrap more efficient anonymity protocols that require Alice and Bob to know each other a little better (such as Collage, or Telex).

Our main contributions are:

- A new primitive for censorship-resistant protocols that requires Alice to have minimal knowledge about Bob;
- A study of the feasibility of this approach;
- An open-source implementation of a proof-of-concept application of MIAB that lets user tweet anonymously and deniably on Twitter.

2 Threat model

The adversary we are facing, the Censor, is a country-wide authority that monitors and interacts with online communications. His intent is to discover and suppress the spreading of dissident ideas.

Determining the current and potential capabilities of modern censors of this kind (e.g., Iran and China) is a difficult task, as the inner workings of the censoring infrastructure are often kept secret. However, we can create a reasonable model for our adversary by observing that, ultimately, the Censor will be constrained by economic factors. Therefore, we postulate that the censorship we are facing is influenced by these two factors:

- The censoring infrastructure will be constrained by its cost and technological effort
- Over-censoring will have a negative impact on the economy of the country

From these factors, we can now devise the capabilities of our Censor. We choose to do so in a conservative fashion, preferring to overstate the Censor's powers than to understate them. We make this choice also because we

understand that censorship is an arms race, in which the Censor is likely to become more powerful as technology advances and becomes more pervasive.

We assume that it is in the Censor's interest to let the general population benefit from Internet access, because of the social and economic advantages of connectivity. This is a fundamental assumption for any censorship-avoidance system: If the country runs an entirely separate network, there is no way out[1].

Because the Censor bears some cost from over-blocking, he will favor blacklisting over whitelisting, banning traffic only when it deems it suspicious. When possible, the Censor will prefer traffic disruption over modification, as the cost of deep-packet inspection and modification is higher than just blocking the stream. For example, if the Censor suspects that the dissidents are communicating through messages hidden in videos on YouTube, he is more likely to block access to the site rather than re–encoding every downloaded video, as this would impose a high toll on his computational capabilities. Also, we assume that if the Censor chooses to alter uncensored traffic, he will do so in a manner that the user would not easily notice.

The Censor might also issue false TLS certificates with a Certificate Authority under its control. With them, he might man-in-the-middle connections with at least one endpoint within his country.

As part of his infrastructure, the Censor might deploy multiple monitoring stations anywhere within his jurisdiction. Through these stations, he can capture, analyze, modify, disrupt, and store indefinitely network traffic. In the rest of the world, he will only be able to harness what is commercially

The Censor can analyze traffic aggregates to discover outliers in traffic patterns, and profile encrypted traffic with statistical attacks on the packets content or timing. Also, he might drill down to observe the traffic of individual users.

The Censor will have knowledge of any publicly available censorship-avoidance technology, including MIAB. In particular, he might join the system as a user, or run a MIAB instance to lure dissidents into communicating with him. Also, he might inject additional traffic into an existing MIAB instance to try to confuse or overwhelm it.

3 Design

In its essence, MIAB is a system devised to allow Alice, who lives in a country ruled by an oppressive regime, to communicate confidentially with Bob, who resides outside the country. Alice does not need to know any information about Bob except his public key.

In particular, MIAB should satisfy these properties:

- **Confidentiality:** The Censor should not be able to read the messages sent by Alice.
- **Availability:** The Censor should not be able to block MIAB without incurring unacceptable costs (by indiscriminately blocking large portions of the Internet).
- **Deniability:** When confidentiality holds, the Censor should not be able to distinguish whether Alice is using the system, or behaving normally.
- **Non-intrusive deployment:** Deploying and operating a MIAB instance should be easy and cheap.

To achieve these properties, the MIAB protocol imposes substantial overhead. We do not strive for MIAB's performance to be acceptable for low latency (interactive) communication over the network (such as web surfing). Instead, we want our users to communicate past the Censor by sending small messages (e.g., emails, articles, tweets). Also, MIAB can be employed to exchange the necessary information to bootstrap other anonymity protocols that require some pre-shared secret, as we will discuss in Section 4.

The only requirement that Alice must satisfy to use this protocol is to be able to make a blog post. She can create this post on any blog hosted (or self-hosted) outside the Censor's jurisdiction.

3.1 Components

Before explaining the details of the MIAB protocol, we must introduce two concepts that will play a crucial role in our system. The first is the notion of a *blog ping*. The second is the metadata that is associated with digital photos.

[1] We are strictly speaking about traditional ways to leak messages to the Internet through network communication.

3.1.1 Blog pings.

A blog ping is a message sent from a blog to a centralized network service (a *ping server*) to notify the server of new or updated content. Blog pings were introduced in October 2001 by Dave Winer, the author of the popular ping server `weblogs.com`, and are now a well-established reality. Over the last ten years, the rising popularity of pings pushed for the development of a wealth of protocols that compete for performance and scalability (e.g., FriendFeed's SUP [26], Google's PubSubHubbub [27], and rssCloud [43]).

Search engines use blog pings to efficiently index new content in real time. Since search engines drive a good part of the traffic on the Internet, blog writers adopt pings to increase their exposure and for Search Engine Optimization. Consequently, the vast majority of blogging platforms support pings, and have them enabled by default (e.g., Wordpress, Blogger, Tumblr, Drupal).

3.1.2 Digital photo metadata.

Digital cameras store metadata embedded within each photo. This metadata keeps records of a variety of information, including the condition of the shot, GPS coordinates, user comments, copyright, post-processing, and thumbnails. Three metadata-encoding formats dominate in popularity, and are supported by the vast majority of cameras and photo editing software: IIM [31], XMP [55], and EXIF [16].

IIM (Information Interchange Model) is an older standard developed in the nineties by the International Press Telecommunications Council. It was initially conceived as a standard for exchanging text, photos and videos between news agencies. In 1995, Adobe's Photoshop adopted IPTC-IIM to store metadata in digital photos, and it is now ubiquitous in the digital camera world.

The second format, XMP (Extensible Metadata Platform) was developed by Adobe in 2001. XMP is XML/RDF-based, and its standard is public and open-source.

The last format is EXIF (Exchangeable Image File Format). It was developed by the Japanese Electronics Industry Development Association (JEIDA) in 1998, and evolved over the years. It is based upon the TIFF format.

A particular EXIF tag, `MakerNote`, stores manufacturer-specific binary content.

To observe the popularity of each of these formats, we downloaded 15,000 photos from Flickr. To reach a broad number of photographers, we selected the photos using Flickr popular tags [20]. The distribution of the metadata formats in our dataset is shown in Table 1. EXIF is by far the most popular format, as it was present in 96.63% of the photos with metadata (note that a photo might contain more than one format). We also observed that the `MakerNote` tag usually accounts for the majority of metadata space.

3.2 The MIAB Protocol

Our scene opens with Alice, who lives in a country controlled by the Censor. Alice wants to communicate with Bob, who is residing outside the country, without the Censor ever knowing that this communication took place. To do so, Alice sends a message with the MIAB protocol following these steps:

1. Alice authors a blog post of arbitrary content. The content should be selected to be as innocuous as possible.
2. Alice snaps one or more photos to include in the post.
3. Alice uses the MIAB client software to embed a message M into the photos. The message is hidden using shared-key steganography [42]. We will identify the shared key as K_s. Alice chooses the shared key arbitrarily.
4. K_s is encrypted with the public key of Bob K_P. This cipher-text is hidden in the photos' metadata, using tags that appear inconspicuous to the Censor (e.g., `Exif.Photo.ImageUniqueID`, which contains an arbitrary identifier assigned uniquely to each image). To better understand the location of the various data embedded in photos, please refer to Figure 1.
5. Alice publishes the blog post, containing the processed photos. Alice can choose the blog arbitrarily, provided it supports blog pings. We will discuss this in more detail in Section 5.2.
6. The blog emits a ping to some ping servers.
7. Meanwhile, Bob is monitoring some of the ping servers, looking for steganographic content encrypted with his public key. Within minutes, he discovers Alice's post, and decrypts the message.

Figure 1: Alice's message

	Number of photos	% of dataset	% of photos with metadata
No metadata	2093	13,95%	
EXIF present	12,472	83.15%	96.63%
XMP present	2972	19.81%	23.03%
IPTC-IIM present	666	4.44%	5.16%

Table 1: Metadata formats in photos on Flickr

8. Bob reads the message, and acts upon its content (more on this later).

Looking at Figure 1, the reader might wonder why we did not choose to store the message M, encrypted with K_P, directly into the metadata. The reason is that available space is limited (e.g., in our implementation, we found about 13.5 bytes of space). Instead, the image data is of the order of megabytes in size, and we can embed in it a message of a few kilobytes (this size depends on the size of the image, and the steganographic scheme used).

Also, notice that Steps 3 and 4 are essentially a simple public-key image steganography (PKIS) [51] scheme: A message is hidden in an image using a public key, and can only be retrieved with the corresponding private key. Note that any PKIS scheme could be used in place of Steps 3 and 4, and might improve the deniability of MIAB (for example, because it would not be necessary to alter the images' metadata). In our proof-of-concept implementation of MIAB, we have chosen the presented scheme because we can use shared-key image-steganography tools, which are widely available (e.g., Outguess [40], StegHide [29]). We are not aware of any open-source implementation of any other PKIS scheme. MIAB's deniability is a direct consequence of the PKIS scheme used, but our main contribution is, then, to create an highly-available protocol that is difficult and expensive for the Censor to block.

To be able to use MIAB, Alice needs a copy of the software, which will also contain Bob's public key. This bundle can be downloaded before the Censor becomes too controlling, or can be published around the web in different formats, so that the Censor cannot easily fingerprint and block it. Optionally, it can be distributed through spam, or offline via USB drives. We expect that the Censor will also have a copy of the software.

4 Applications

The MIAB protocol can be a useful component for a variety of applications. We will outline the ones we believe are most interesting, one of which we implemented.

4.1 Covert messaging

In this setting, Alice wants to send a message (e.g., email or tweet) out of the country in a deniable fashion. She does so simply by using the MIAB protocol to publish her message embedded in a blog post, leaving instructions to Bob on how to handle it. To showcase this application, we have implemented a Twitter proxy that relays the messages found through MIAB to Twitter, publishing them under the `@corked_bottle` handle. MIAB could also be used to communicate with a censorship-resistant microblogging service, like `#h00t` [2], or a privacy-preserving one, like Hummingbird [12].

To achieve two-way messaging, Alice composes a comment to the blog post she is about to publish. She sends this comment to Bob with MIAB, appended to her message. When Bob wants to reply to Alice's message, he steganographically encodes his answer inside the comment, and posts the comment on Alice's blog (or any blog Alice has chosen for posting her messages). Alice will either be notified of Bob's action by email or RSS feed, or she will check her blog every now and then. Alice's behavior should not be considered suspicious by the Censor, since blog writers have a habit of checking comments on their posts frequently.

To appear even more inconspicuous to the Censor, Alice might arbitrarily pick any other blog available on the Internet, and tell Bob to post the comment containing the reply there. A good meeting point for this is one of the blogs and online newspapers (provided they allow comments) that Alice reads on a regular basis.

How Bob hides his reply will depend on Alice's directives. An option is that Alice includes an image (or provides a private link to one) in her comment, and Bob embeds the message into this image. Since Alice and Bob have already established a shared secret, there is no need for metadata. Another option is that Bob encodes his reply in the comment's text. Note that our scheme authenticates only Bob, as not being identified is in Alice's interest. However, after the first message is sent, Bob and Alice share a session key, so that the Censor cannot pretend to be Alice and take over her role in the current message exchange.

This method of covert messaging suffers a noticeable overhead with respect to standard messaging. To achieve better efficiency, we envision that MIAB could be used to bootstrap some faster, shared-key protocol.

4.2 Bootstrapping Collage

Collage [8] is a protocol that uses user-generated content (e.g., photos, tweets) as drop sites for hidden messages. Differently from the MIAB approach, in Collage, the drop sites must be decided upon in advance: These rendezvous points are the secret that Alice and Bob share. To put and retrieve the messages from these rendezvous points, the users of Collage have to perform a series of *tasks*. For example, if the drop site is a photo posted on Flickr [19] under the keyword "flowers," the *sender task* will be the series of HTTP requests required to post a photo with that keyword, and the *receiver task* will be composed of requests designed to retrieve the latest photos with that tag.

To bootstrap a Collage installation, the database of tasks that Collage employs must be distributed offline. This database needs to be constantly updated as the Censor reacts and the drop sites change (both in location and structure). It is, therefore, crucial that this bootstrap database is up-to-date: Otherwise, the agreement on the drop points between sender and receiver will be lost, breaking the communication channel.

Once Collage has been bootstrapped, further updates can be received through the Collage network. To receive these updates, however, the Collage client must be connected to the Internet. When the connectivity is sporadic, the client's database might become obsolete, and a new bootstrap round will become necessary.

We believe that MIAB is a good fit for bootstrapping Collage, because the only information that Alice must know about Bob in order to request a current copy of the task database is Bob's public key. This key should rarely change, so it can be shipped with the software. If the Censor is already too controlling to allow this, the key and the software (which is just a few kilobytes in size) can be published online in a variety of formats, so that it will be difficult for the Censor to find and block them all.

5 Implementation

To demonstrate the feasibility of MIAB, we have implemented a proof-of-concept application that can be used to post anonymous messages on Twitter, circumventing the block on social networks imposed by Tyria's Censor.

To provide a more open evaluation, we have published the code online: The application can be downloaded at `http://www.message-in-a-bottle.us`, together with our public key.

We have set up a small cluster of machines that monitors one of the most popular blog ping servers (`webblogs.com`[2]), looking for MIAB messages (our cluster plays Bob's part in the algorithm). When a message is found, its content is posted on Twitter under the handle `@corked_bottle`.

Also, the code can be used with a new public/private key-pair to verify how messages are recovered.

5.0.1 Public-key cryptography

Our application can use either GnuPG or X.509 certificates. We ship a self-signed X.509 certificate, containing the RSA public key.

5.0.2 Steganography

We use `outguess` [41] as the steganographic primitive. This tool works on Linux, *BSD, and Windows, so it will most probably run on Alice's computer. We are aware that an attack on `outguess` is known [25]. We choose this particular software out of convenience: Any steganographic tool can be used, and the change required

[2]For example, blogs hosted on Google's `blogger.com` ping this server, unless the blog owner disables this.

in the code is trivial. See also Section 3.2 for a discussion on an alternative approach with public-key image steganography.

5.0.3 Blog-fetching cluster

We use four servers with Intel Xeon 2.40GHz processors, 4 Gigabytes of RAM, and high-speed Internet connection.

5.0.4 Photo metadata

We hide the output of $\text{RSA}_{K_P}(K_s)$ in EXIF and XMP metadata, and in the filename. The ciphertext is stored in base 10, 16, or 60, depending on the format stored in the tags. We employ two photos, hiding 13.5 bytes of key per photo. The list of tags employed, and some sample content, is provided in Table 2. The Table also reports the number of bits, calculated with Shannon's entropy, that are *undetectable*. That is, the bits whose values have a high entropy throughout our dataset, and for which it is difficult to tell if they have been altered. In the example data in the Table, the undetectable bits are highlighted in bold. Note that the sample content has been extracted from a photo downloaded from Flickr [19]. In the case of tags holding a `DateTime` object, we hide the ciphertext only in the minutes and seconds.

To show that this ciphertext could be hidden in a variety of locations, we store a part of it in the digits contained in the filename (e.g., `DSC01234.JPG`). All these decision are arbitrary, and every MIAB instance can make different choices.

Metadata tag	Example	Format	Description	Undetectable bits
`Exif.Photo.SubSecTimeOriginal`	**22**	Ascii	Centiseconds when the photo has been taken (the date and time up to the seconds precision are kept in the DateTimeOriginal tag)	6.64
`Exif.Photo.SubSecTimeDigitized`	**90**	Ascii	Centiseconds for the DateTimeDigitalized tag	6.64
`Exif.Photo.SubSecTime`	**75**	Ascii	Centiseconds for the DateTime tag	6.64
`Exif.Photo.ImageUniqueID`	**713E4D7FC20E42C1 A1617CE642C28017**	Ascii	Arbitrary, unique identifier assigned to the photo	16
`Xmp.xmpMM.OriginalDocument`	**3856450D6BAB4CE5 AFD64DAF89C3C198**	Ascii	Arbitrary, unique identifier assigned original document, when editing a photo	16
`Xmp.aux.ImageNumber`	**49,202**	Ascii	Photoshop's auxiliary image number	16
`Exif.Image.ImageNumber`	**233**	Long	Unique number assigned to a photo	16
`Exif.Image.DateTime`	2011-09-25 15:**52:04**	Ascii	The date and time of image creation (we only alter the minutes and seconds).	11.81
`Exif.Photo.DateTimeOriginal`	2011-08-23 11:**27:27**	Ascii	The date and time when the original image data was generated (we only alter the minutes and seconds)	11.81

Table 2: EXIF and XMP tags: example photo taken from Flickr

5.1 Evaluation

MIAB relies on Bob's ability to fetch and process all the blog posts created on the Internet in real time. To prove that this is feasible, we have done so with our cluster.

Over the period of three months (72 days), we have seen $814,667,299$ blog posts, as shown in Figure 2. The average number of posts seen per day is $11,314,823$, and the highest traffic we have experienced is $13,083,878$ posts in a day. Being a proof-of-concept, our implementation delivers suboptimal performance in terms of blog throughput (it is written in Python). So, we had to use four machines to keep up with the post rate. To estimate the performance limits of a single machine fetching and analyzing web pages, we tested Apache Nutch [22], a high-performance web crawler written in Java and built upon Apache Hadoop [21]. In our experiment, we were able to fetch more than 15 million pages per day. Since we have observed that our MIAB cluster is limited by its

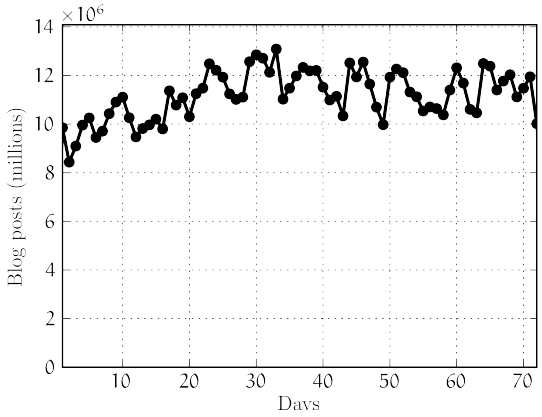


Figure 2: Blogs pings received over a period of three months

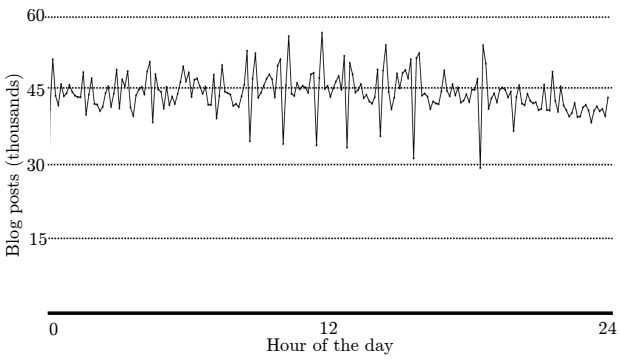


Figure 3: Blog ping received during day one (5 minutes slices)

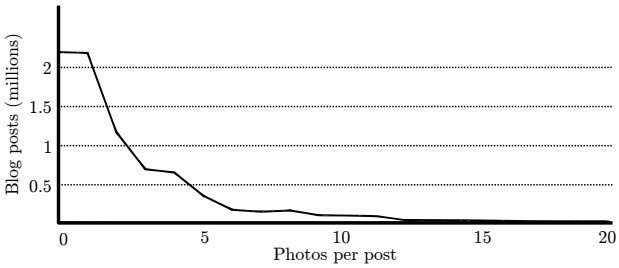


Figure 4: Number of photos per blog during day one

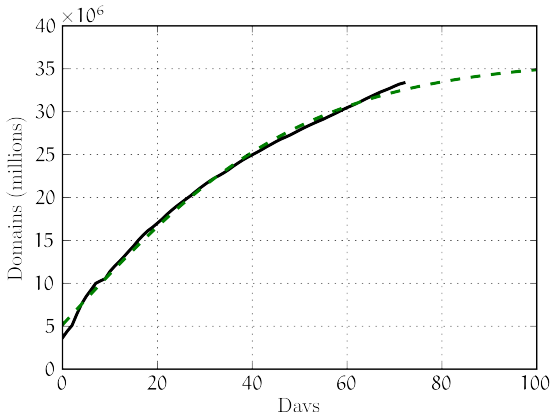


Figure 5: Blacklist size, when targeting fully qualified domain names

capacity of fetching and parsing the blog posts, and not by the image processing, we expect that an optimized implementation of MIAB should be able to achieve a similar performance.

The ping server we chose releases every five minutes the pings that it received during that period. To keep up with the ping rate, we process all the blog posts of one chunk of pings before the next one is released. Therefore, the maximum latency between the publishing of a post with MIAB content and our tweeting is ten minutes (five minutes in the ping server's queue, and another five before we process the post), with an average of five minutes.

During the course of a day, we have found that the rate of blog posts is quite stable, with an average of $44,146$ posts every five minutes, and a low standard deviation of $3,963$. See Figure 3 for an example.

We observed that approximately half of the blog posts contain photos. The distribution of the number of photos per post is shown in Figure 4.

5.2 Choosing the Right Blog

To send a message through MIAB, Alice has to publish a post on a blog that supports pings. Alice might already have posting privileges on such a blog: In this case, she can just use it. Otherwise, Alice needs to open one, or be added as an author to an existing one. It is therefore interesting to see how difficult it is for Alice to open a blog supporting pings.

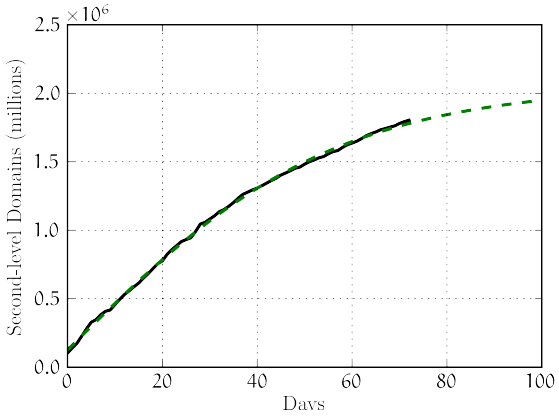


Figure 6: Blacklist size, when targeting second-level domains

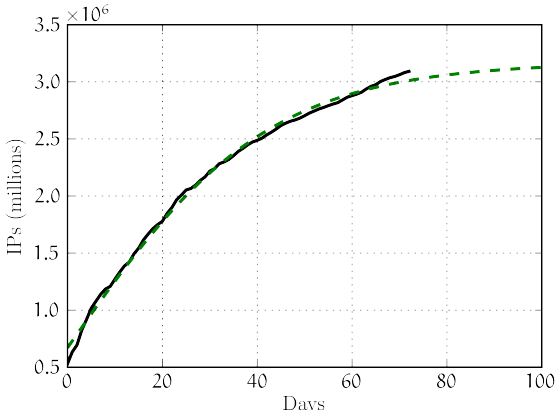


Figure 7: Blacklist size, when targeting IP addresses.

Platform	Number of sites (%)	Ping support
WordPress	63.49	yes
Joomla!	11.17	yes
Drupal	8.57	yes
Blogger	3.14	yes

Table 3: Blogging platform popularity: Alexa's top million (source: `builtwith.com` [7])

Platform	Number of sites	Ping support
WordPress	32	yes
TypePad	16	yes
Moveable Type	20	yes
BlogSmith	14	n/a, proprietary
custom made	8	n/a
Drupal	4	yes
Blogger	3	yes
Expression Engine	1	n/a, proprietary
Scoop	1	no
Bricolage	1	no, CMS

Table 4: Blogging platforms: 100 most popular blogs (source: `pingdom.com` [39])

In Table 3, we show the four most popular blogging platforms for the Alexa's top million websites in February 2012. These platforms account for more than 85% of all blogs, and they all support pings.

In Table 4, we show the platform that the 100 most popular blog chose to use in 2009. Of these, at least 75 support pings.

If the blog Alice has chosen does not support blog pings, Alice can perform the ping manually, as most ping servers offer this feature (e.g., `http://blogsearch.google.com/ping`).

5.3 Estimating the Blocking Set

The Censor might attempt to block all blogs by blacklisting their domain names, or IP addresses. It is essential to MIAB's availability that the blocking set (that is, the set of domains or IP addresses to block) is large, as this makes the maintenance of a blacklist impractical (this will be further discussed in Section 6.1). To estimate the size of the blocking set, we generated three blacklists from three months of recorded pings. It can be seen that these blacklists target Fully Qualified Domain Names (e.g., `myblog.blogspot.com.`, see Figure 5), second-level domains (e.g., `blogspot.com.`, see Figure 6), and IP addresses (see Figure 7).

These blacklists are cumbersome in size. For example, according to Netcraft, during the time of our experiment, there were $662,959,946$ FQDNs running web servers on the Internet [36]. In our experiment, we would have blacklisted $33,361,754$ FQDNs, which is 5% of the ones found by Netcraft. Blocking second level domains is also not practical. In our experiment, we blacklisted $1,803,345$ of them, which account for 1.4% of the global domain registrations (which are $140,035,323$, according to DomainTools [14]). Note that our blacklists only cover blogs that sent pings during our experiment. The total numbers will be higher, as the trend shown in

Figures 5, 6, and 7 suggests. To better quantify this, we checked the (hypothetical) blacklists's coverage on the pings that were emitted the day after we stopped updating the blacklists. We found that the blacklist targeting FQNDs had 12% miss ratio, the one targeting second level domains had 11% miss ratio, and the one targeting IP addresses had 11% miss ratio.

Even if the Censor chooses to pay the price of over-blocking and enforces these blacklists, Bob and Alice can still communicate though MIAB, thanks to a weakness in this blacklisting approach. That is, a blog might have multiple domain names (or IP addresses) assigned to it, but its pings are usually sent by only one of them. So, Alice can edit her blog at `blog.alice.com`, a domain name that never sends pings and, therefore, is not in the blacklist, and configure her blog to emit pings from the secondary address `alice.blogspot.com`. Bob will receive the ping, and visit the latter address to obtain the MIAB messages. In turn, the Censor also sees the ping, and will blacklist the secondary address in Alice's country. Alice will be still able to modify her blog through `blog.alice.com`, sneaking by the blacklist, and Bob will be able to visit `alice.blogspot.com`, because the Censor's block does not affect him.

6 Security Analysis

In this section, we discuss MIAB's resilience against the attacks that the Censor might mount to break the security properties that MIAB guarantees: availability, deniability, and confidentiality.

Since we lack a formal verification of the security of the steganographic primitives, and the powers of a real Censor are unknown, these goals are necessarily best effort.

6.1 Availability

The Censor might prevent Alice from sending messages through MIAB. Our main argument against this is that the Censor will be forced to ban a large part of the web, and he is likely unwilling to do so because of the pushback from its population and economic drawbacks. We will now describe in more detail the attacks that the Censor might devise to limit availability. For this discussion, we assume that deniability holds: In particular, the Censor cannot detect content generated with MIAB.

The Censor could *block blogging platforms*. In fact, this has already been attempted: Kazakhstan has been banning blogging platforms and social networks since 2008 [32]. In 2011, a court established they contributed to the spread of religious terrorism [24]. Despite the block, Kazakh bloggers have moved to self-hosted or less popular blogging platforms, and continue to grow in numbers [32]. There are over 165 million blogs on the Internet [5], and a good part of them are self-hosted, making the maintenance of a blacklist impractical (see Section 5.3). Even if the Censor is able to fingerprint blogs and successfully block each of them, there are other services that generate blog pings, for example, commenting services (e.g., Disqus, which is installed on 750,000 websites). Also, for her MIAB messaging, Alice might open a blog in favor of the ruling party: The Censor will be less likely to block that, because it shows that the population is supporting the regime. The Censor might also use pings to identify blogs, blocking any domain that emits a ping. This approach, which we already discussed in Section 5.3, is also vulnerable to a denial of service attack: Bob could forge bogus pings claiming to be any arbitrary domain (blog pings, just like emails, do not authenticate the sender), forcing the Censor to add it to the blacklist. Iterating this attack, Bob can de facto render the blacklist useless. To counter this, the Censor will have to maintain a whitelist of allowed domains. This is harmful to the regime, because of the social and economic costs of over-blocking, and can be easily circumvented (as we discussed in Section 5.3).

The Censor might want to *block pings*, or *shut down ping servers*. This attack is irrelevant when the blog (or ping server) is hosted outside the Censor's jurisdiction.

The Censor might try to *prevent Alice to post* on any blog. This requires both the ability to fingerprint a blog, and the technological power to deeply inspect the network traffic to detect and block posting. The cost of this attack increases as Alice will most likely be using a TLS connection (i.e., `https://`) to make the post, requiring the Censor to either man-in-the-middle the connection or perform statistical timing attacks. Also, Alice can create

a post in a variety of ways, for example using client applications or by email (as many blogging platforms support that). The Censor therefore has to block all these vectors.

The Censor might try to *coerce content hosts to drop* MIAB content. However, to do so, he needs to identify such content (when deniability should prevent him from doing so). Also, the Censor has to drop the content fast: Bob will fetch the blog after just a few minutes, since blog pings are sent in real time.

The Censor might try to *overwhelm* MIAB *by creating large quantities of bogus messages*. Bob should be able to provide a fair service to its users by rate limiting blogs, and ban them when they exhibit abusive behavior. Since Bob will first inspect the list of pings, he will avoid fetching blogs that are over their quota or blacklisted. Also, search engines will fetch the same blog pings and will analyze and index their content. Since the Censor, to create a large number of blog posts, will have to either generate the content automatically or replay content, search engines should detect his behavior and mark it as spam. Blog ping spam (or *sping*) is already a problem that search engines are facing, since pings are used in Search Engine Optimization campaigns. Because of this, much research has been invested into detecting this type of behavior. Bob could leverage that research by querying the search engine for the reputation of blog or blog posts. Using a combination of blacklists, rate limitation, and search engines reputation systems, Bob is likely to thwart this attack.

The Censor might *perform traffic analysis* to distinguish regular blog posts from MIAB ones. However, since the blog post is created by a human, this is unlikely to be successful.

The Censor might decide to *strip metadata* from the digital pictures uploaded to blogs, or *re-encode the images* to remove any steganographic content. First, this is computationally expensive for the Censor. Moreover, MIAB can easily evolve and hide the ciphertext in other parts of the post (e.g., in the text), or use a steganographic system that can withstand re-encoding [28].

6.2 Deniability

The Censor might try to identify who is using MIAB, thus compromising deniability.

The Censor might try to *detect steganography*. MIAB uses steganography as a primitive, so its deniability relies on a good steganographic algorithm, and will benefit from any advancement in steganography. We discussed the use of public-key image steganography in Section 3.2. Moreover, Bob might support several steganographic algorithms, and let Alice pick the one she feels safe to use. Her choice can be encoded anywhere in the post (e.g, the fist letter of the title).

The Censor might try to *detect anomalies in digital photo metadata*. This is not relevant if the public-key image steganographic algorithm chosen does not alter them. Also, the content stored in the metadata is ciphertext: Assuming that the cipher is sound, the attacker cannot distinguish ciphertext from a random permutation of the input (otherwise, the cipher would fail an *indistinguishability under chosen ciphertext attack*). This attack is irrelevant if we choose appropriately the metadata tags where the ciphertext is hidden. Several tags are well-suited to carry arbitrary data (e.g., `Exif.Image.OriginalRawImageDigest` `Exif.Photo.ImageUniqueID`). Since a blog post can contain several photos (in Section 5.1, we have shown that it is quite common that a post contains up to five photos), the size of the ciphertext is not a factor that forces us to use detectable tags. Also, in our implementation, we chose not to use the EXIF's `MakerNote` tag, which stores proprietary binary data. This tag, however, usually accounts for the majority of the size of the metadata, and has several vendor-specific tags where additional ciphertext could be stored. We decided against using it in our proof-of-concept implementation, because we found enough bits of available entropy within the standard tags (see Table 2).

The Censor might try to *detect anomalous blogging behavior*. Since Alice manually publishes the post, it will be difficult to detect anomalous behavior in a single post (provided that the message that Alice composes is innocuous). However, the Censor might detect an anomaly if the posting frequency is higher than the other blogs hosted on the same platform. Alice can mitigate this risk by keeping under control her post rate. Also, Alice can use a blogging platform that encourages frequent posts with photos (i.e., a photoblog, or a *moblog* - a mobile blogging platform). Alice might also have multiple online personas, each of which posts on a different blog.

The Censor might try to *run an instance of* MIAB *to trick Alice into communicating with him*. This is a limitation common to any system that relies on a public key to identify the receiving party. This is mitigated by publishing Bob's public key in a variety of formats on the Internet, so that the Censor cannot find them and block or alter them.

The Censor might try to *perform a timing attack, correlating the effect of the action specified in the* MIAB *message* (in our implementation, the posting of a new tweet) *with the publishing of a new post*. We can mitigate this by letting the MIAB user specify when he wants the aforementioned action to be taken, inserting random delays and generating decoys (in our implementation, tweeting a message that was not received).

6.3 Confidentiality

To break confidentiality, the Censor must get access to the plaintext of the message that Alice sent. The message is encrypted with a public key, of which only Bob knows the private counterpart. Assuming that the cryptographic primitive is secure, the Censor will not be able to read the message. Also, the Censor might run an instance of MIAB to trick Alice, as we discussed in the previous section.

7 Related Work

There has been an extensive and ongoing discussion on anonymous and censorship-resistant communication. In this section, we review the main directions of research, highlighting how they measure against the goals that we have set for MIAB.

7.1 Anonymization proxies

The most established way to achieve online anonymity is through proxies, possibly with encrypted traffic [1, 33]. In 2010, a report from Harvard's Center for Internet & Society [38] shows that 7 of the 11 tools with at least 250,000 unique monthly users are simple web proxies.

These systems focus on hiding the user identities from the websites they are surfing to. They deliver a high performance but, in doing so, they do not satisfy any of the goals of MIAB: They can be easily be blocked, since it is easy to discover the addresses of the proxies and block them (low availability). Also, even if the traffic is encrypted, the users cannot deny that they were using the system (no deniability), and it is possible to mount fingerprinting and timing attacks to peek into the users' activity [6, 30].

One of the first systems that address deniability is Infranet [17]. Uncensored websites deploying Infranet would discreetly offer censored content upon receiving HTTP traffic with steganographic content. Availability is still an issue, since the Censor might discover and block these websites, so the collaboration of a large number of uncensored websites becomes essential.

7.2 Mix networks

Mix networks (e.g., Tor [13], Mixminion [11]) focus on anonymity and unlinkability, offering a network of machines through which users can establish a multi-hop encrypted tunnel. In some cases, depending on the location of the proxies with respect to the Censor, it is possible to link sender and receiver [4, 35]. These systems do not typically provide deniability, although Tor tries to mask its traffic as an SSL connection [48].

The availability of these systems depends on the difficulty to enumerate their entry points, and their resistance to flooding [15]. Protecting the entry nodes has proven to be an arms race: For example, the original implementation of Tor provided a publicly-accessible list of addresses. These entry points were blocked in China in 2009 [49]. In response, Tor has implemented bridges [45], which is a variation on Feamster's key-space hopping [18]. However, these bridges have some limitations that can expose the address of a bridge operator when he is visiting websites through Tor [34]. Also, distributing the bridges' addresses without exposing them to the Censor is theoretically impossible (since the Censor could play Alice's role, and Tor cannot distinguish one from the other). However, it might be possible to increase the cost of discovering a bridge so much that the Censor finds it impractical to enumerate them. This problem remains very complex and, so far, none of the approaches attempted has succeeded:

Proof of this is the fact that China has been blocking the vast majority of bridges since 2010, as the metrics published by Tor show [50].

7.3 Anonymous publishing

Here, the focus is on publishing and browsing content anonymously. Freenet [9] and Publius [53] explore the use of peer-to-peer networks for anonymous publishing. These systems deliver anonymity and unlinkability, but do not provide deniability or availability, as the Censor can see where the user is connecting, and block him.

Another direction that is being explored is to make it difficult for the Censor to remove content without affecting legitimate content (Tangler [52]). This system suffers similar pitfalls as Freenet, although availability of the documents is improved by the *document entanglement*.

7.4 Deniable messaging

Systems for deniable messaging can be split into two categories: The ones that require no trusted peer outside the country (e.g., CoverFS [3], Collage [8]), and the ones that do (e.g., Telex [54]). CoverFS is a FUSE file system that facilitates deniable file sharing amongst a group of people. The file system synchronization happens though messages hidden in photos uploaded to a web media service (Flickr, in their implementation). As mentioned by the authors, while CoverFS focuses on practicality, the traffic patterns could be detected by a powerful Censor. This would affect its availability and deniability.

Collage, which we already described in Section 4.2, improves CoverFS' concept of using user-generated content hosted on media services to establish covert channels. Collage has higher availability, as the sites where the hidden messages are left change over time, and it provides a protocol to synchronize these updates. As the authors point out, Collage deniability and availability are challenged when the Censor performs statistical and timing attacks on traffic. Also, if the Censor records the traffic, he can join Collage, download the files where the messages are embedded, and find out from his logs who uploaded them. To join the Collage network, a user needs to have an up-to-date version of tasks database, which contains the rendez-vous points. Since its distribution has to be timely, this is challenging (and can be solved with MIAB). Also, without regular updates, the tasks database can go stale, which requires a new bootstrap.

In Collage, the tasks defining the rendez-vous points are generated manually. Although every media-sharing site can potentially be used with Collage, generating all these tasks requires substantial work. MIAB, instead, can use millions of domains out of the box without human intervention, which makes monitoring more difficult.

A Censor might also join Collage and disseminate large quantities of bogus messages. Since the burden of finding the right message is on Collage's users, they would be forced into fetching many of the bogus messages, thus exposing themselves to statistical attacks. This could be mitigated by rate limiting the message fetching, but this decision would result in unbounded delays between the posting of the message and its retrieval, challenging the availability of the system. An effective solution for this denial-of-service is to generate a separate task list for any pair of people that need to communicate with each other. This can be done using MIAB as a primitive to disseminate these databases to the parties, so that they can bootstrap a Collage session. Once the two parties are exchanging messages, they can arbitrate the Collage channel they want to employ, depending on the performance that their communication requires.

Telex is a system that should be deployed by Internet Service Providers (ISPs) that sympathize with the dissidents. A user using the Telex system opens a TLS connection to an unblocked site (that does not participate in the protocol), steganographically embedding a request in the TLS handshake. When the ISP discovers the request, it diverts the SSL connection to a blocked site. While Telex deniability is sound, it can be subject to a denial of service by the Censor. Also, the Censor can discover ISPs implement Telex by probing the paths to various points on the Internet, and prevent the traffic originating from the country from going through those ISPs. Overall, Telex comes close to satisfying the goals that we set for MIAB. However, it requires the ISP collaboration, which is hard to set up, as the authors mention. MIAB, on the other hand, requires minimal resources (a good

Internet connection and a single machine should suffice, when implemented efficiently), and hence, it can be setup today, as we demonstrated by hosting our public implementation of this system.

A simpler solution to our problem is to use an webmail service running outside the censored country (e.g., Gmail) to send a message to an address where a anti-censorship service is listening. In fact, this is one of the channels used to distribute Tor briges (by sending an email to `bridges@torproject.org`). Since the Censor can downgrade the dissident's connection to the webmail provider and view the traffic, this method is not deniable. Also, the Censor can block emails sent to that address, challenging availability.

8 Conclusions

We have introduced MIAB, a deniable protocol for censorship resistance that works with minimal requirements. MIAB can be used as a standalone communication system, or to bootstrap protocols that achieve higher performance, at the cost of more demanding requirements. We have demonstrated MIAB feasibility with the implementation of a proof-of-concept prototype, and released its code open-source. The deployment of a MIAB instance requires minimal configuration and resources, since it relies on well-established and popular technologies (blog pings and blogging platforms). Also, we have shown that MIAB is resilient to attacks to its availability, deniability and confidentiality. Although a powerful Censor might be able to disrupt MIAB, in doing so he will suffer a high cost, effectively cutting the population of his jurisdiction out of a major part of the Internet.

References

[1] Anonymizer. Home page. `http://plone.anonymizer.com/`.

[2] D. Bachrach, C. Nunu, D. Wallach, and M. Wright. #h00t: Censorship resistant microblogging. *arXiv:1109.6874*.

[3] A. Baliga, J. Kilian, and L. Iftode. A web based covert file system. In *Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, page 12. USENIX Association, 2007.

[4] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource routing attacks against tor. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 11–20. ACM, 2007.

[5] BlogPulse. Report on indexed blogs. `http://goo.gl/SEpDH`, 2011.

[6] D. Brumley and D. Boneh. Remote timing attacks are practical. *Computer Networks*, 2005.

[7] BuiltWith. Content management system distribution. `http://trends.builtwith.com/cms`, 2012.

[8] S. Burnett, N. Feamster, and S. Vempala. Chipping away at censorship firewalls with user-generated content. In *USENIX Security Symposium*, 2010.

[9] I. Clarke, O. Sandberg, B. Wiley, and T. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.

[10] CNN. Egyptians brace for friday protests as internet, messaging disrupted. `http://articles.cnn.com/2011-01-27/world/egypt.protests_1_egyptian-authorities-muslim-brotherhood-opposition-leader`.

[11] G. Danezis, R. Dingledine, and N. Mathewson. Mixminion: Design of a type iii anonymous remailer protocol. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 2–15. IEEE, 2003.

[12] E. De Cristofaro, C. Soriente, G. Tsudik, and A. Williams. Hummingbird: Privacy at the time of twitter. *IEEE Symposium on Security and Privacy*, 2012.

[13] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*, pages 21–21. USENIX Association, 2004.

[14] DomainTools. Internet statistic. `http://www.domaintools.com/internet-statistics/`.

[15] N. Evans, R. Dingledine, and C. Grothoff. A practical congestion attack on tor using long paths. In *Proceedings of the 18th conference on USENIX security symposium*, pages 33–50. USENIX Association, 2009.

[16] EXIF. Homepage. `http://www.exif.org`.

[17] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing web censorship and surveillance. In *Proceedings of the 11th USENIX Security Symposium, August*, 2002.

[18] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger. Thwarting web censorship with untrusted messenger discovery. In *Privacy Enhancing Technologies*, pages 125–140. Springer, 2003.

[19] Flickr. Homepage. `http://flickr.com`.

[20] Flickr. Popular tags. `http://www.flickr.com/photos/tags/`.

[21] T. A. Foundation. Apache hadoop. http://hadoop.apache.org/.
[22] T. A. Foundation. Apache nutch. http://nutch.apache.org/.
[23] A. France-Presse. Pakistan blocks facebook over mohammed cartoon. http://www.google.com/hostednews/afp/article/ALeqM5iqKZNUdJFQ6c8ctdkUW0C-vktIEA.
[24] A. France-Presse. Kazakhstan blocks popular blogging platforms, 2011.
[25] J. Fridrich, M. Goljan, and D. Hogea. Attacking the outguess. In *ACM Workshop on Multimedia and Security*, 2002.
[26] FriendFeed. Simple update protocol. http://code.google.com/p/simpleupdateprotocol/.
[27] Google. Pubsubhubhub. http://code.google.com/p/pubsubhubbub/.
[28] R. Greenstadt. Zebrafish: A steganographic system. *Massachusset Institute of Technology*, 2002.
[29] S. Hetzl. Steghide. http://steghide.sourceforge.net/.
[30] A. Hintz. Fingerprinting websites using traffic analysis. In *Privacy Enhancing Technologies*, 2002.
[31] IPTC-IIM. Homepage. http://www.iptc.org.
[32] S. Kelly and S. Cook. Freedom on the net. *Freedom House*, 2011.
[33] B. Labs. The lucent personalized web assistant. http://www.bell-labs.com/project/lpwa/, 1997.
[34] J. McLachlan and N. Hopper. On the risks of serving whenever you surf: vulnerabilities in tor's blocking resistance design. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 31–40. ACM, 2009.
[35] S. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *Security and Privacy, 2005 IEEE Symposium on*, pages 183–195. IEEE, 2005.
[36] Netcraft. May 2012 web server survey. http://news.netcraft.com/archives/2012/05/02/may-2012-web-server-survey.html.
[37] H. Noman and J. York. West censoring east: The use of western technologies by middle east censors, 2010-2011. 2011.
[38] J. Palfrey, H. Roberts, J. York, R. Faris, and E. Zuckerman. 2010 circumvention tool usage report. 2011.
[39] Pingdom. The blog platforms of choice among the top 100 blogs. http://royal.pingdom.com/2009/01/15/the-blog-platforms-of-choice-among-the-top-100-blogs/, 2009.
[40] N. Provos. Outguess - universal steganography. http://outguess.org, 1999.
[41] N. Provos. Defending against statistical steganalysis. In *10th USENIX security symposium*, 2001.
[42] N. Provos and P. Honeyman. Hide and seek: An introduction to steganography. *Security & Privacy, IEEE*, 2003.
[43] rssCloud. Homepage. http://rsscloud.org.
[44] TechCrunch. Syrian government blocks bambuserâĂŹs live video of crisis. http://eu.techcrunch.com/2012/02/17/syrian-government-blocks-bambusers-live-video-of-crisis/.
[45] Tor. Bridges. https://www.torproject.org/docs/bridges.
[46] Tor. Help users in iran reach the internet. https://lists.torproject.org/pipermail/tor-talk/2012-February/023070.html.
[47] Tor. Iran blocks encrypted traffic. https://blog.torproject.org/blog/iran-partially-blocks-encrypted-network-traffic.
[48] Tor. Iran blocks tor; tor releases same-day fix. https://blog.torproject.org/blog/iran-blocks-tor-tor-releases-same-day-fix.
[49] Tor. Tor partially blocked in china. https://blog.torproject.org/blog/tor-partially-blocked-china.
[50] Tor. Tor users via bridges from china. https://metrics.torproject.org/users.html?graph=bridge-users&start=2010-01-01&end=2012-02-20&country=cn&dpi=72#bridge-users.
[51] L. Von Ahn and N. Hopper. Public-key steganography. In *Advances in Cryptology-EUROCRYPT 2004*, 2004.
[52] M. Waldman and D. Mazieres. Tangler: a censorship-resistant publishing system based on document entanglements. In *ACM conference on Computer and Communications Security*. ACM, 2001.
[53] M. Waldman, A. Rubin, and L. Cranor. Publius: A robust, tamper-evident, censorship-resistant web publishing system. In *USENIX Security Symposium*, 2000.
[54] E. Wustrow, S. Wolchok, I. Goldberg, and J. A. Halderman. Telex: Anticensorship in the network infrastructure. In *USENIX Security Symposium*, 2011.
[55] XMP. Homepage. http://www.adobe.com/products/xmp/.

Building a Wrapper for Fine-Grained Private Group Messaging on Twitter

Indrajeet Singh[1], Michael Butkiewicz[1], Harsha V. Madhyastha[1],
Srikanth V. Krishnamurthy[1], Sateesh Addepalli[2]

[1] University of California, Riverside [2] Cisco Systems

Abstract

User privacy has been an increasingly growing concern in online social networks (OSNs). While most OSNs today provide some form of privacy controls so that their users can protect their shared content from other users, these controls are typically not sufficiently expressive and/or do not provide fine-grained protection of information. In this paper, we consider the introduction of a new privacy control—group messaging on Twitter, with users having fine-grained control over who can see their messages. Specifically, we demonstrate that such a privacy control can be offered to users of Twitter *today* without having to wait for Twitter to make changes to its system. We do so by designing and implementing `Twitsper`, a wrapper around Twitter that enables private group communication among existing Twitter users while preserving Twitter's commercial interests. Our evaluation shows that our implementation of `Twitsper` imposes minimal server-side bandwidth requirements and incurs low client-side energy consumption. Our `Twitsper` client for Android-based devices has been downloaded by over 1000 users and its utility has been noted by several media articles.

1 Introduction

OSNs have gained immense popularity in the last few years since they allow users to easily share information with their contacts and even discover others of similar interests based on information they share. However, not all shared content is meant to be public; users often need to ensure that the information they share is accessible to only a select group of people. Though legal frameworks can help limit with whom OSN providers can share user data, users are at the mercy of controls provided by the OSN to protect from other users the content they share. In the absence of effective controls, users concerned about the privacy of their information are likely to connect with fewer users, share less information, or even avoid joining OSNs altogether.

Previous proposals to address these privacy concerns on *existing OSNs* either (a) jeopardize the commercial interests of OSN providers [30, 24] if these solutions are widely adopted and thus, are likely to be disallowed, or (b) require users, who are currently accustomed to free access to OSNs, to pay for improved privacy [26, 37, 4]. On the other hand, though *new OSNs* have been developed with privacy explicitly in mind [15, 7], these OSNs have seen limited adoption because users are virtually "locked in" to OSNs on which they have already invested significant time and energy to build social relationships. Consequently, users have, in many cases today, raised privacy-related concerns in the media; though OSNs have introduced new privacy controls in response to these concerns (e.g., Facebook friend lists, Facebook groups, Google+ circles), such controls do not provide sufficiently fine-grained protection.

In light of this, we consider the privacy shortcomings on Twitter, one of the most popular OSNs today [16]. Twitter offers two kinds of privacy controls to users—a user can either share a message with all of her followers or with one of her followers; there is no way for a user on Twitter to post a *tweet* such

that it is visible to only a subset of her followers. In this paper, we fill this gap by providing fine-grained controls to Twitter users, enabling them to conduct private *group communication*. Importantly, we provide this fine-grained privacy control to Twitter users by implementing a wrapper that builds on Twitter's existing API, without having to wait for Twitter to make any changes to its service.

As our primary contribution, we design and implement `Twitsper`—a wrapper around Twitter that provides the option of private group communication for users, without requiring them to migrate to a new OSN. Unlike other solutions for group communication on Twitter [8, 18, 21], `Twitsper` ensures that Twitter's commercial interests are preserved and that users do not need to trust `Twitsper` with their private information. Further, in contrast to private group communication on other OSNs (e.g., Facebook, Google+), in which a reply/comment on information shared with a select group is typically visible to all recipients of the original posting, `Twitsper` strictly enforces privacy requirements as per a user's social connections (all messages posted by a user are visible only to the user's followers).

When designing `Twitsper`, we considered various choices for facilitating the controls that we desire; surprisingly, a simple approach seemed to be the best fit for fulfilling all our objectives. Thus, our implementation of `Twitsper` is based on this simple design which combines a Twitter client (that retains much of the control logic) with a server that maintains minimal state. Our evaluation demonstrates that this simple design does achieve the best tradeoffs between several factors such as backward compatibility, availability, client-side energy consumption, and server-side resource requirements.

Overall, our implementation of `Twitsper` is proof that users can be provided fine-grained privacy controls on existing OSNs, without waiting for OSN providers to make changes to their platform. Our client-side implementation of `Twitsper` for Android phones has been downloaded by over 1000 users and several articles in the media have acknowledged its utility in improving privacy and reducing information overload on Twitter [20, 14, 19].

2 Related work

Characterizing privacy leakage in OSNs: Krishnamurthy and Willis characterize the information that users reveal on OSNs [31] and how this information leaks [32] to other entities on the web (such as social application providers and advertising agencies). Our thesis is that ensuring OSN providers do not leak user information requires legal measures that mandate appropriate controls. It is not in the commercial interests of OSN providers to support systems that hide information from them. Therefore, we focus on enabling users to protect their information from other *undesired* users, rather than from OSN providers.

Privacy controls offered by OSNs: Google+ and Facebook permit any user to share content with a *circle* or *friend list* comprising a subset of the user's friends. However, anyone who comments on the shared content has no control; the comment will be visible to all those with whom the original content was shared. Even worse, on Facebook, if Alice comments on a friend Bob's post, Bob's post becomes visible to Alice's friend Charlie even if Bob had originally not shared the post with Charlie. Facebook also enables users to form groups; any information shared with a group is not visible to users outside the group. However, a member of the group has to necessarily share content with all other members of a group, even if some of them are not her friends. Twitter, on the other hand, enables users to restrict sharing of their messages either to only all of their followers (by setting their account to *private* mode) or to exactly one of their followers (by means of a *Direct Message*), but not to a proper subset. We extend Twitter's privacy model to permit private *group communication*, ensuring that the privacy of a user's reply to a message shared with a group is in keeping with the user's social connections.

Distributed social networks: Several proposals to improve user privacy on OSNs have focused on de-centralizing OSNs (e.g., Vis-a-Vis [37], DiSo [4], and PeerSoN [26]). These systems require a user to store her data on a personal device or in the cloud, thus removing the need for the user to trust a central

Category	%
Consider privacy a concern	77
Would like to control who sees information they post	70
Declined follower requests owing to privacy concerns	50

Table 1: Results of survey about privacy shortcomings on Twitter.

OSN provider. However, users have put in tremendous effort in building their social connections on today's OSNs [5, 16]; rebuilding these connections on a new OSN is not easy. Thus, unlike these prior efforts, we build a backward-compatible privacy wrapper on Twitter.

Improving privacy in existing OSNs: With Lockr [38], the OSN hosting a user's content is unaware of with whom a user is sharing content; Lockr instead manages content sharing. Other systems allow users to share encrypted content, either by posting the encrypted content directly on OSNs [30, 24, 25] or via out-of-band servers [12]. Users can share the decryption keys with a select subset of their connections (friends). Hummingbird [27] is a variant of Twitter in which the OSN supports the posting of encrypted content in such a manner that preserves user privacy. Narayanan et al. [34] ensure users can keep the location information that they divulge on OSNs private via private proximity testing. All of these techniques either prevent OSN providers from interpreting user content, or hide users' social connections from OSNs. Since neither is in the commercial interests of OSN providers, these solutions are not sustainable if widely adopted. In contrast, we respect the interests of OSN providers while exporting privacy controls to users.

Group communication: Like `Twitsper`, listserv [29] enables communication between groups of users. However, unlike with `Twitsper`, group communications on listserv lack a social structure and listserv was never designed with privacy in mind. Prior implementations of group messaging on Twitter, such as GroupTweet [8], Tweetworks [18], and Twitter Groups [21], have either not focused on privacy—they require users to trust them with their private information—or require users to join groups outside their existing social relationships on Twitter.

3 Motivating User Survey

While privacy concerns with OSNs have received significant coverage [31, 32], the media has mostly focused on leakage of user information on OSNs to third-parties such as application providers and advertising agencies. Our motivation is the need for a more basic version of privacy on OSNs—protecting content contributed by a user from other users on the OSN, which has began to receive some attention [10]. To gauge the perceived need amongst users for this form of privacy, we conducted an IRB approved user study across 78 users of Twitter [1]. Our survey questioned the participants about the need they see for privacy on Twitter, the measures they have taken to protect their privacy, and the controls they would like to see introduced to improve privacy. Table 1 summarizes the survey results. More than three-fourths of the survey participants are concerned about the privacy of the information they post on Twitter, and an almost equal fraction would like to have better control over who sees their content. Further, rather tellingly, half the survey takers have at least once rejected requests to connect on Twitter in order to protect their privacy. These numbers motivate the necessity of enabling users on Twitter to privately exchange messages with a subset of their followers, specifically allowing them to choose which subset to share a message with on a per-message basis.

[1] Participants consisted of staff and students at UCR.

Proposal	Backward Compatible	Preserves Commercial Interests	No Added Trust Required
Distributed OSNs	×	×	✓
Encryption	✓	×	✓
Separating content providers from social connections	✓	×	×
Existing systems for group messaging on Twitter	✓	✓	×
`Twitsper`	✓	✓	✓

Table 2: Comparison of `Twitsper` with previous proposals for improving user privacy on OSNs.

4 System design goals

Given the need for enabling private group messaging on Twitter, we next design `Twitsper` to provide fine-grained privacy controls to Twitter users. Our over-arching objective in developing `Twitsper` is to offer these controls to users without having to wait for Twitter to make any changes to their service. Our design for `Twitsper` is guided by three primary goals.

Backward compatible: Rather than developing a new OSN designed with better user controls in mind (e.g., proposals for distributed OSNs [37, 26, 4]) we want our solution to be compatible with Twitter. This goal stems from the fact that Twitter already has an extremely large user base—over 100 million active users [22]. Since the value of a network grows quadratically with the growth in the number of users on it (the network effect [33]), Twitter users have huge value locked in to the service. To extract equal value from an alternate social network, users will not only need to re-add all of their social connections, but will further require all of their social contacts to also shift to the new service. Therefore, we seek to provide better privacy controls to users by developing a wrapper around Twitter, eliminating the burden on users of migrating to a new OSN and thus maximizing the chances of widespread adoption of `Twitsper`.

Preserves commercial interests: A key requirement for `Twitsper` is that it should not be detrimental to the commercial interests of Twitter. For example, though a user can exchange encrypted messages on Twitter to ensure that she shares her content only with those with whom she shares the encryption keys, this prevents Twitter from interpreting the content hosted on its service. Since Twitter is a commercial for-profit entity and offers its service for free, it is essential that Twitter be able to interpret content shared by its users. Twitter needs this information for several purposes: to show users relevant advertisements, to recommend applications of interest to the user, and to suggest others of similar interest with whom the user can connect. Though revealing user-contributed content to Twitter opens the possibility of this data leaking to third-parties (either with or without the knowledge of the provider), user content can be insured against such leakage via legal frameworks (e.g., enforcement of privacy policies [23]) or via information flow control [39]. On the other hand, protecting a user's content from other users requires enabling the user with better controls—our focus in building `Twitsper`.

No added trust: In attempting to give users better controls without waiting for Twitter to change, we want to ensure that users do not have another entity to trust in `Twitsper`; users already have to trust Twitter with their information. Increasing the number of entities that users need to trust is likely to deter adoption since users would fear the potentially greater opportunity for their information to leak to third-parties. Therefore, we seek to ensure that users do not need to share with `Twitsper`'s servers any information they want to protect, such as their content or their login credentials. Tools such as TaintDroid [28] can be used to verify that `Twitsper`'s client application does not leak such information to `Twitsper`'s servers. We design `Twitsper` for the setting where `Twitsper`'s servers are not malicious by nature, but are inquisitive listeners; this attacker model is similar to that used in prior work (e.g., [35]).

Table 2 compares our proposal with previous solutions for improving user privacy on OSNs. Unlike

API call	Function
PrivSend(msg, group)	Send *msg* to all users specified in *group*
isPriv?(msg)	Determine if *msg* is a private message
PrivReply(msg, orig_msg)	Send *msg* to all of the user's followers who received *orig_msg*

Figure 1: `Twitsper`'s API beyond normal Twitter functionality.

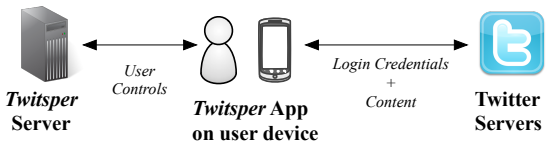


Figure 2: System architecture.

proposals for distributed OSNs, `Twitsper` enables users to reuse their social connections on Twitter, and unlike calls for exchange of encrypted content, we respect Twitter's commercial interests. Moreover, we introduce user controls via `Twitsper` without adding another entity for users to trust, unlike proposals such as Lockr [38], which call for the separation of social connections from content providers. Lastly, in contrast to prior implementations of group messaging on Twitter such as GroupTweet [8], Tweetworks [18] and Twitter Groups [21], we ensure that Twitter is privy to private conversations but `Twitsper` is not.

5 `Twitsper` design

Next, we present an overview of `Twitsper`'s design. We consider various architectural alternatives and discuss the pros and cons with each. Our design objectives guide the choice of the architecture that presents the best trade-off. As mentioned earlier, surprisingly, a fairly straightforward simple approach seems to yield the best trade-offs and is thus, used as the basic building block in `Twitsper`.

Basic definitions: First, we define a few terms related to Twitter and briefly explain the Twitter ecosphere.

- ***Tweet*:** A tweet is the basic mode of communication on Twitter. When a user posts a tweet, that message is posted on the user's Twitter page (i.e., http://twitter.com/username), and is seen on the timeline of everyone following the user.
- ***Direct Message*:** A direct message is a one-to-one private tweet from one user to a specific second user, and is possible only if the latter follows the former.
- @ ***Reply*:** A user can use a @*reply* message to reply to another user's tweet; this message will also appear on the timeline of anyone following both users.
- ***Twitter page*:** Every user's Twitter page (http://twitter.com/username) contains all tweets and @*reply* messages posted by the user. By default, this page is visible to anyone, even those not registered on Twitter. If a user sets her Twitter account to be private, all messages on her page are visible to any of the users following her account.
- ***Timeline*:** A user's timeline is the aggregation of all tweets, direct messages, and @*reply* messages (sorted in chronological order) visible to that user. In addition to her timeline, note that a user can view *any* tweet or @*reply* message posted by any user that she follows by visiting that user's Twitter page.

Interface: Our primary goal is to extend Twitter's privacy model. In addition to sharing messages with all followers or precisely one follower, we seek to enable users to privately share messages with a subset of their followers. To do so, we extend Twitter's API with the additional functionality shown in Table 1.

First, the *PrivSend* API call allows users to post *private* messages that can be seen by one or more members in the user's network, who are *specifically* chosen to be the recipients of such a message. However, simply enabling a message to be shared with a group of users is insufficient. To enable richer communica-

Design	Twitter's interests preserved	No added trust	Easily scales	Same text size	Always available	Linkable to orig message
Supporting server	✓	✓	✓	✓	✓	×
Embed lists	✓	✓	✓	×	✓	×
Encryption	×	✓	✓	✓	×	✓
Community pages	×	×	×	✓	×	×
Dual accounts (No longer possible)	×	✓	✓	✓	✓	✓

Table 3: Comparison of architectural choices.

tion, it is necessary that recipients of a message (shared with a group) be able to reply back to the group. In the case of discussions that need not be kept private, a user may choose to make her reply public so that others with similar interests can discover her. However, when Nina responds to a private message from Jack, it is unlikely that Nina will wish to share her reply with all the original target recipients of Jack's message since many of them may be "unconnected" to her. Nina will likely choose to instead restrict the visibility of her reply to those among the recipients of the original message whom she has approved as her followers. Therefore, the *PrivReply* API call enables replies to private messages, while preserving social connections currently established on Twitter via follower-followee relationships. Finally, the *isPriv?* API call is necessary to determine if a received message is one to which a user can reply with *PrivReply*. Hereafter, we refer to the messages exchanged with the *PrivSend* and *PrivReply* calls as whispers.

It is important to note that, since our goal is to build a wrapper around Twitter, rather than build a new OSN with these privacy controls, this extended API has to build upon Twitter's existing API for exchanging messages. Though Twitter's API may evolve over time, we rely here on simple API calls—to post a tweet to all followers and to post a Direct Message to a particular follower—that are unlikely to be pruned from Twitter's API. Also note that, in some cases, multiple rounds of replies to private messages can result in the lack of context for some messages for some recipients, since all recipients of the original whisper may not be connected with each other. In the trade-off between privacy and ensuring context, we choose the former in designing `Twitsper`.

Architectural choices: Next, we discuss various architectural possibilities that we considered for `Twitsper`'s design, to support the interface described above. While it may be easy for Twitter to extend their interface to support private group messaging, we note that Twitter has not yet done so in spite of the need for this amongst its users. Therefore, our focus is in designing `Twitsper` to offer this privacy control to users without having to wait for Twitter to make this change.

Using a supporting server: The simplest architecture that one can consider for `Twitsper` is to have clients send a whisper to a group of users by sending a Direct Message to each of those users. To enable replies, a supporting server can maintain the list of the original recipients of a whisper; when a client sends a whisper, it can send the identifiers of the Direct Messages and the list of recipients to the supporting server. Thus, a user can query this supporting server to check if a received Direct Message corresponds to a whisper. When the user chooses to reply to a whisper, the user's client can retrieve the list of recipients of the original whisper from the server, locally compute the intersection between those recipients and the user's followers, and then send Direct Messages to all those in the intersection.

If the supporting server is unavailable, users can continue to use Twitter as before, except that the metadata necessary to execute the *isPriv?* and *PrivReply* API calls cannot be retrieved from the server. However, the client software can be modified to allow a recipient to obtain relevant mappings (list of recipients of a whisper) from the original sender. Another option is to have the client embed the list of recipients of a whisper in every Direct Message sent out as part of a whisper. However, given Twitter's 140 character limit per Direct Message, this can be a significant imposition, reducing the permissible length of the message content.

This design places much of the onus on the client and may result in significant energy consumption for the typical use case of Twitter access from smart phones. On the flip side, in this architecture, the content posted by a user is *never* exposed to the supporting server i.e., privacy from `Twitsper`'s server is preserved. The server is simply a facilitator of group communications across a private group and only maintains meta data related to whispers. Further, Twitter is able to see users' postings and thus its commercial interests are protected. We note that the alternative of the client sending messages to the supporting server for retransmission to the recipients is not an option, since this would require users to trust the supporting server with the content of their messages.

This design however does have some shortcomings. Twitter lacks sufficient context to recognize that the set of Direct Messages shared to send a whisper constitute a single message rather than a local trending topic. Similarly, Twitter cannot link replies with the original message, since all of this state is now maintained at the supporting server.

Using encryption to hide content: To address the shortcoming in the previous architecture of being unable to link replies to the original whispers, in our next candidate architecture, clients post whispers just as they would a public message (tweet) but encrypt it with a group key which is only shared with a select group of users (who are the intended recipients of the message). This reduces the privacy problem to a key exchange problem for group communications. An out-of-band key exchange is possible.

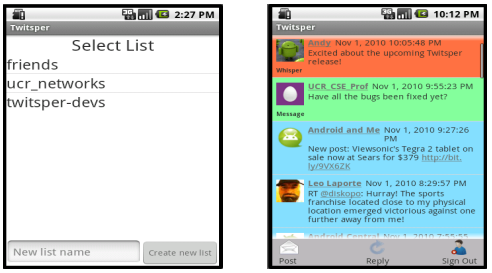
However, since only intended recipients can decrypt a tweet, Twitter's commercial interests are compromised. Furthermore, filtering of encrypted postings not intended for them is necessary at the recipient's side; if not, a user's Twitter client will display indecipherable noise from these postings. In other words, the approach is not backward compatible with Twitter. Note here that if these issues are resolved, e.g., by sharing encryption keys with Twitter, encryption can be used with any of the other architectural choices considered here to enhance privacy.

Using community pages to support anonymity: Alternatively, one may try to achieve anonymity and privacy by obfuscation. Clients post tweets to a obfuscation server, which in turn re-posts messages on behalf of users to a common "community" account on Twitter. Except for the server, no one else is aware of which message maps to which user. When a user queries the obfuscation server for her timeline, the server returns a timeline that consists of messages from her original timeline augmented with messages meant for that user from the "community" page. The obfuscation prevents the exposure of private messages to undesired users. Since the "community" page is hosted on Twitter, the shortcoming of the encryption-based architecture is readily addressed—Twitter has access to all information unlike in the case of encryption. An approach similar to this was explored in [36].

However, this architecture has several drawbacks. First, Twitter cannot associate messages with specific users; this precludes Twitter from profiling users for targeted advertisements and such. Second, all users need to trust the obfuscation server with the contents of their messages. Finally, since the architecture is likely to heavily load the server (due to the scale), the viability of the design in practice becomes questionable. When the server is unavailable, no private messages can be sent or received.

Using dual accounts: In our last candidate architecture, every user maintains two accounts. The first is the user's existing account, and a second private account is used for sending whispers. Since a Direct Message from one user to another is possible only if they are connected, links between these private accounts are dynamically created so as to allow a private message to propagate only to the desired users. However, Twitter has been quite clear that an application that creates significant fluctuation in links between users will be considered detrimental to the Twitter ecosphere [1], thus discouraging the dynamic creation and deletion of associations between private accounts. Further, a supporting server that has the relevant credentials of all users is necessary to setup and authorize follower-followee links between accounts dynamically; requiring users to trust their OSN login credentials with a supporting server is undesirable.

A different method for facilitating selective sharing of information is to use an @*reply* from these private accounts. However, as of mid-2009, Twitter discontinued the "capability" of @*reply* messages between



(a) List selection (b) A user's timeline

Figure 3: `Twitsper` on Android OS

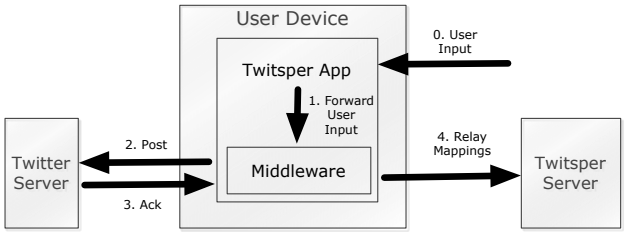


Figure 4: Steps for posting a whisper

disconnected users after concluding that less than 1% of the users found this feature useful and that it contributes to spam messages [13]. Thus, @*reply* messages posted from these disconnected private accounts will not be visible to intended recipients. Other problems with this architectural choice are that Twitter is unable to associate private messages with the normal accounts of users and responding to private messages is a challenge.

Figure 3 summarizes the comparison of the various architectural choices with respect to our design goals. While no solution satisfies all desirable properties, we see that the use of a supporting server presents the best trade-off in terms of simplicity and satisfying our goals. Therefore, we choose this to be the architectural choice for implementing `Twitsper`, whose architecture is as shown in Figure 2.

6 Implementation

In this section, we describe our implementation of the `Twitsper` client and server. Given the popularity of mobile Twitter clients, we implement our client on the Android OS [2, 3].

Generic implementation details. Normal tweets (public) and Direct Messages are sent with the `Twitsper` client as with any other Twitter client today. We implement whispers using Direct Messages as described before. Recall that direct messaging is a one-to-one messaging primitive provided by Twitter. Mappings from Direct Messages to whispers are maintained on our `Twitsper` server.

`Twitsper`'s whisper messages are always sent to a group of selected users. The client handles group creation by creating a list of users on Twitter. This list can either be public (its group members are viewable by any user of Twitter) or private for viewing only by its creator. The client sends a Direct Message via Twitter to each group member and gathers the message IDs returned by Twitter. These IDs and the Twitter list ID are then sent to the `Twitsper` server. The server creates and stores a mapping of the message IDs to the IDs of the group members.

When a `Twitsper` client receives a Direct Message, it queries the `Twitsper` server to check whether the message is a whisper or a standard Direct Message. If the server finds a mapping from the message ID to a list of users, this indicates that the message corresponds to a whisper. The server reports its finding to the client. A key feature of our system is that since whispers are sent as Direct Messages, whispers can still be received and viewed by legacy users of Twitter who have not adopted `Twitsper`; such users cannot however reply to whispers.

`Twitsper` allows a whisper recipient to reply not only to the sender, but also to a *subset* of the original group receiving the whisper. This subset is simply the intersection of the original group and the *followers* of

the responding user. Thus, it respects the social relations established by users. A whisper recipient's client is provided with the list ID corresponding to the group, and the client can then retrieve the user IDs on that list from Twitter if the original whisper sender made the list public. If the list is private, the recipient's response can only be received by the original sender. In the future, we plan to permit `Twitsper` users to modify the list associated with a particular whisper in order to enable inclusion of new users in the private group communication or removal of recipients of the original whisper from future replies.

Server implementation details: Our server is equipped with an Intel quad-core Nehalem processor, 24 GB of RAM, and one 7200 RPM 1 TB hard disk drive. The `Twitsper` server is implemented as a multi-threaded Java program. The main thread accepts incoming connections and assigns a worker thread, chosen from a thread pool, to service each valid API call. The server stores whisper mappings in a MySQL database. In order to ensure that writing to the database does not become a bottleneck we have multiple connections to the database; we observed that without this, the server performance was affected. These connections are used by worker threads in a round-robin schedule. Note that our server does not store any personal information or credentials of any user. The flow of information in case of a tweet (public) or a direct message remains unchanged. Only in the case of a whisper does the use of our system become necessary. The contents of a whisper are never sent to our server; only unique message IDs for the consequent Direct Messages (along with a list ID) are sent. This ensures that the server can never "overhear" conversations between users unless it has either a user's password,which with `Twitsper` is never transmitted to the server.

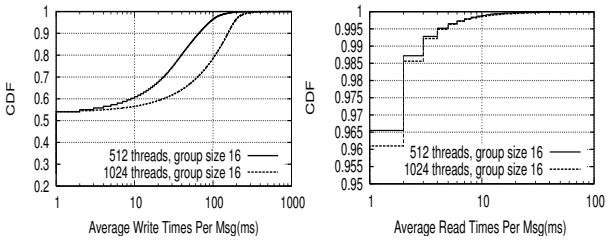
Client implementation details: Our client was written for Android OS v1.6 and was tested on the Android emulator as well as on three types of Android phones (Android G1 dev, Motorola Droid X, and HTC Hero). We use the freely available twitter4j package to access the Twitter API. The client is also multi-threaded and separates the UI (user-interface) thread from the processing, the network, and disk I/O threads. This ensures a seamless experience to the user without causing the screen to "freeze" when the client is performing disk or network I/O. We profiled the power consumption of our implementation to identify inefficiencies and iteratively improved the relevant code. These iterative refinements helped us decrease the dependence on the network by caching frequently retrieved user profile images, while maintaining a thread pool rather than the fork and forget model adopted by most open source implementations of other Twitter clients, so as to not over-commit resources.

When the `Twitsper` server is unavailable, we cache whisper mappings on the client and piggyback this data with future interactions with the server. On the other hand, recipients of whispers interpret them as Direct Messages and cannot reply back to the group until the server is again reachable. In future versions of `Twitsper`, we will enable recipients to directly query the client of the original sender if `Twitsper`'s server is unavailable.

We also color code tweets, Direct Messages and whispers, while maintaining a simple and interactive UI. Example screen shots from our `Twitsper` client are shown in Figure 3. Our client application is freely available on the Android market, and to date, our `Twitsper` Android application has been downloaded by over 1000 users.

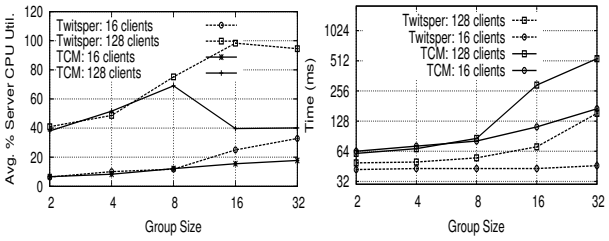
7 Evaluation

Next we present our evaluation of `Twitsper`. For the purposes of benchmarking, we also implement a version of `Twitsper` wherein a client posts a whisper by transmitting the message to the `Twitsper` server, which in turn posts Direct Messages to all the recipients on the client's behalf. Though, as previously acknowledged, this design clearly violates our design goal of users not having to trust `Twitsper`'s server, we use this *thin client* model (TCM) (we refer to our default implementation as the *fat client* model or `Twitsper` itself) as a benchmark to compare against. One primary motivation for using TCM as a point of



(a) DB write time (b) DB read time

Figure 5: Database performance



(a) CPU utilization (b) Service time per client

Figure 6: Server Metrics

comparison is that it can reduce the power consumption on phones (since battery drainage is a key issue on these devices). We also compare `Twitsper`'s energy consumption on a smart phone with that of a popular Twitter client to demonstrate its energy thriftiness.

Server-side results: First, we stress test our server by increasing the rate of connections it has to handle. In this experiment, we use one or more clients to establish connections and send dummy metadata to our server. All clients and the server were on the same local network and thus, network bandwidth was not the constraining factor. We monitored CPU utilization, disk I/O, and network bandwidth with Ganglia [6] and iostat to detect bottlenecks. We vary the target group size of whispers as well as the number of simultaneous connections to the server.

Disk. In Figure 5b, we plot the time taken by each thread to read information relevant to a message from the database (we preloaded the database with 10 million entries to emulate server state after widespread adoption); Figure 5a depicts the CDFs of the write times to the database. We see that as the number of clients increase, so do the database write times, but not the read times. Thus, as the system scales, the bottleneck is likely going to be the I/O for writing to the disk.

CPU. Next, we compare the server performance of TCM and `Twitsper`. We will refer to the version of the server which works in tandem with `Twitsper`, and handles only whisper metadata, as the `Twitsper` server. The TCM server must, in addition, handle the actual sending of whispers to their recipients. It is to be expected that the overhead of the TCM server would increase the computational power needed to service each client. Figures 6a and 6b show the average CPU utilization and user service time, respectively, for each server version. We see in Figure 6a that the `Twitsper` server has a higher CPU utilization than the TCM server. This is because the TCM server spends more idle time (Figure 6b) while servicing each client since it needs to wait on communications with Twitter. So even though more CPU resources are being spent per client with the TCM server, the average CPU utilization is lower.

Another interesting feature noted from these graphs is that certain increases in group size cause the server to more than double its service time. These sharp increases in service time in Figure 6b have corresponding drops in CPU utilization in Figure 6a. This is due to our server's disk writes being the throughput bottleneck. Since in each test we either double the number of client connections or the group size, we would expect a CPU bottleneck to manifest itself with drastic service time increases (of $\approx 200\%$). Instead, the data points to a disk write bottleneck where the client must wait for an acknowledgment of the server database's successful write. We verify with iostat that our hard drive is used at 100% utilization during these periods. We are currently investigating the effect of adding more disks.

Network. Figure 9a shows the number of bytes in and out with the TCM and `Twitsper` servers for a single client connection. Each line in Figure 9a represents a single client sending one whisper message to a group size which is varied (x-axis). We see that increasing the group size does not cause a large increase in the received bytes as compared to the case with only 2 group members. This illustrates that the overhead increase with recipient group size (which causes either the receipt of more message IDs with the `Twitsper` server or the receipt of more recipient user IDs with the TCM server) is very minor when compared to the

Interface	Twitsper	Other
LCD	13325	10127
CPU	755	1281
3G	4812	8232

Figure 7: Total power consumption (mJ)

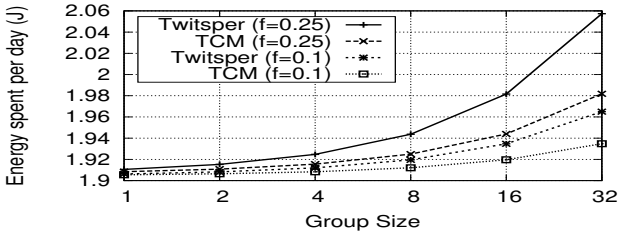


Figure 8: Client power consumption

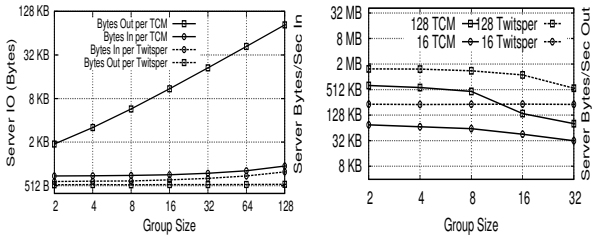
resources consumed by the SSL connection between the client and the server. The only additional overhead with the TCM server is the transfer of the actual whisper messages from the client; this manifests as the constant offset between these two curves. Since the Twitsper server has to only send a confirmation to the user that its whisper meta data was received correctly, the bytes out is independent of the recipient group size. In contrast, the burden of having to send whispers to each recipient (as a separate Direct Message) is on the TCM server. Increasing group size (x-axis) increases the number of Direct Messages sent to Twitter and this quickly results in an overshoot of the single client SSL connection overhead.

Figures 9b and 9c show the bandwidth consumed at the server as the number of bytes in and out per second. In Figure 9b, we see that the Twitsper server does not experience a reduction in transmission rate until it hits 128 clients and a group size of 16. At this point, we hit a disk bottleneck in writing client message metadata to our database. For the TCM server, we see a rate reduction even in the 16 clients case as we increase the group size; this is due to the latency incurred in the message exchange with Twitter. We hit a similar hard disk bottleneck at 128 concurrent client connections with the TCM server, as similar metadata needs to be stored with both server setups.

Comparing Twitsper and TCM clients: While Twitsper offers higher CPU utilization as well as lower bandwidth requirements, the power consumption at the client is a key factor in ensuring adoption of the service. To evaluate its client side power performance, we measure *the amount of energy* needed to make a single post with Twitsper to Twitter and to send a message to our server. We use the PowerTutor [11] application to measure the power consumed at the client. We made 100 posts back to back and measure the average energy consumed.

Figure 8 compares TCM and Twitsper based on the power consumed on a phone. The figure shows the energy consumption per day on an Android phone, for an average Twitter user who sends 10 messages per day and has 200 followers [9]. Our experiments suggest that the best implementation depends on the fraction of a user's messages that are private (denoted by f) and the typical size of a list to which private messages are posted. The energy consumption with Twitsper is significantly greater than that with the TCM client when f is large or the group sizes are big. However, since we expect private postings to constitute a small fraction of all information sharing and that such communication will typically be restricted to small groups, energy consumption overhead with Twitsper is minimal. Even in the scenarios where client-side energy consumption increases, the energy consumed is still within reason, e.g., the energy consumed per client across various scenarios is within the range of 1.9 J to 2.5 J, which is less than 0.005% of the energy capacity of typical batteries (10 KJ, as shown in [11]). Further, as we show next, the majority of the energy consumed in practice is by the user's interaction with the phone's display, whereas the energy we consider here is only that required to simply send messages, and does not include displaying and drawing graphics on the screen.

Comparison with another popular Twitter client: We next compare the power consumption of Twitsper with that of a popular Twitter client (TweetCaster[17]), which supports the default privacy options on Twitter. We begin the test after both clients had been initialized and had run for 15 seconds. We then send a message from each of the clients and refresh the home screen; there was at least one update to the



(a) Bytes in/out per client (b) Incoming bandwidth (c) Outgoing bandwidth

Figure 9: Network activity on server; same legend on (b) and (c)

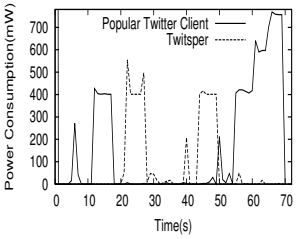


Figure 10: Comparison of power consumption

home screen. As seen from the traces of the power consumed in Figure 10, `Twitsper`'s power consumption is comparable. This shows that `Twitsper` only imposes energy requirements on the mobile device that are comparable to other Twitter clients. We observe that there is no noticeable loss in performance since both clients were made to carry out the same tasks functionally.

In the above test, even though the screen was kept on for as little a time as possible (less than 10% of the total time) the LCD accounted for close to 50% of the aggregate energy consumed, as seen from Figure 7. Referring the reader back to Figure 8, we see that as the group size increases there is only a marginal increase in the energy consumption associated with the sending of messages. Even if 25% of the messages are whispers and the average group size is 32 (which we believe is quite large), the energy consumed only increases from 1.92 J (for a single tweet) to 2.05 J—an increase of less than 15%; given that the LCD power consumption dominates, this is not a significant energy cost.

8 Conclusions

Today, for users locked in to hugely popular OSNs, the primary hope for improved privacy controls is to coerce OSN providers via the media or via organizations such as EFF and FTC. In this paper, to achieve privacy without explicit OSN support, we design and implement `Twitsper` to enable fine-grained private group messaging on Twitter, while ensuring that Twitter's commercial interests are preserved. By building `Twitsper` as a wrapper around Twitter, we show that it is possible to offer better privacy controls on existing OSNs without waiting for the OSN provider to do so.

Next, we plan to implement fine-grained privacy controls on other OSNs such as Facebook and Google+ as well, using a similar approach of building on the API exported by the OSN. Given the warm feedback received by `Twitsper`, we hope that the adoption of `Twitsper` and its follow-ons for other OSNs will persuade OSN providers themselves to offer fine-grained privacy controls to their users.

References

[1] Aggressive follower churn in spam subsection of "the twitter rules". `http://bit.ly/a62bx1`.

[2] Android operating system. `http://www.android.com/`.

[3] Comscore: Android is now highest-selling smartphone OS. `http://bit.ly/euR4Yb`.

[4] DiSo project. `http://diso-project.org/`.

[5] Facebook traffic reaches nearly 375 million monthly active users worldwide, led by us. `http://bit.ly/c0Z3UQ`.

[6] Ganglia. `http://ganglia.sourceforge.net/`.

[7] Google Plus numbers belie social struggles. http://bit.ly/pPIwDr.

[8] Grouptweet. http://www.grouptweet.com/.

[9] New data on Twitter's users and engagement. http://bit.ly/cu8P2s.

[10] Please rob me. http://www.pleaserobme.com/.

[11] Powertutor. http://bit.ly/hVaXh1.

[12] Priv(ate)ly. http://priv.ly/.

[13] Retweet this if you want non-followers replies fixed. http://bit.ly/YwLYw.

[14] Selectively tweeting via free Android application Twitsper. http://bit.ly/gas5bS.

[15] Social networks offer a way to narrow the field of friends. http://nyti.ms/j7dOsC.

[16] Tweet this milestone: Twitter passes MySpace. http://on.wsj.com/dc25gK.

[17] Tweetcaster. http://tweetcaster.com/.

[18] Tweetworks. http://www.tweetworks.com.

[19] Twitsper app for Android enhances Twitter security and privacy. http://bit.ly/avDV8l.

[20] Twitsper, group tweeting app, could change Twitter as we know it. http://huff.to/cYkGvH.

[21] Twitter Groups! http://jazzychad.net/twgroups/.

[22] Twitter reveals it has 100m active users. http://www.guardian.co.uk/technology/pda/2011/sep/08/twitter-active-users.

[23] Twitter suspends twidroyd & UberTwitter over privacy claims. http://bit.ly/hRcZ1w.

[24] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin. Persona: An online social network with user-defined privacy. In *SIGCOMM*, 2009.

[25] F. Beato, M. Kohlweiss, and K. Wouters. *Scramble! Your Social Network Data*, volume 6794 of *Lecture Notes in Computer Science*, chapter 12, pages 211–225. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[26] S. Buchegger and A. Datta. A case for P2P infrastructure for social networks - opportunities and challenges. In *6th International Conference on Wireless On-demand Network Systems and Services (WONS)*, 2009.

[27] E. De Cristofaro, C. Soriente, G. Tsudik, and A. Williams. Hummingbird: Privacy at the time of Twitter. In *IEEE Symposium on Security and Privacy (S&P)*, 2012.

[28] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *OSDI*, 2010.

[29] D. A. Grier and M. Campbell. A social history of bitnet and listserv, 1985-1991. *IEEE Annals of the History of Computing*, 2000.

[30] S. Guha, K. Tang, and P. Francis. NOYB: Privacy in online social networks. In *WOSN*, 2008.

[31] B. Krishnamurthy and C. Willis. Characterizing privacy in online social networks. In *WOSN*, 2008.

[32] B. Krishnamurthy and C. Willis. On the leakage of personally identifiable information via online social networks. In *WOSN*, 2009.

[33] S. J. Liebowitz and S. E. Margolis. Network externality: An uncommon tragedy. *The Journal of Economic Perspectives*, 1994.

[34] A. Narayanan, N. Thiagarajan, M. Lakhani, M. Hamburg, and D. Boneh. Location privacy via private proximity testing. In *NDSS*, 2011.

[35] R. A. Popa, H. Balakrishnan, and A. J. Blumberg. VPriv: Protecting privacy in location-based vehicular services. In *USENIX Security Symposium*, 2009.

[36] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web transactions. *ACM TISSEC*, 1998.

[37] A. Shakimov, H. Lim, R. Cáceres, L. P. Cox, K. Li, D. Liu, and A. Varshavsky. Vis-ã-Vis: Privacy-preserving online social networks via virtual individual servers. In *COMSNETS*, 2011.

[38] A. Tootoonchian, S. Saroiu, Y. Ganjali, and A. Wolman. Lockr: Better privacy for social networks. In *CoNEXT*, 2009.

[39] N. Zeldovich, S. Boyd-Wickizer, and D. Mazières. Securing distributed systems with information flow control. In *NSDI*, 2008.

Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns

Łukasz Olejnik[1], Claude Castelluccia[2], Artur Janc[3]

[1] INRIA, Grenoble, France, lukasz.olejnik@inria.fr
[2] INRIA, Grenoble, France, claude.castelluccia@inria.fr
[3] Google, Inc., Mountain View, USA, aaj@google.com

Abstract

We present the results of the first large-scale study of the uniqueness of Web browsing histories, gathered from a total of $368,284$ Internet users who visited a history detection demonstration website. Our results show that for a majority of users (69%), the browsing history is unique and that users for whom we could detect at least 4 visited websites were uniquely identified by their histories in 97% of cases. We observe a significant rate of stability in browser history fingerprints: for repeat visitors, 38% of fingerprints are identical over time, and differing ones were correlated with original history contents, indicating static browsing preferences (for history subvectors of size 50). We report a striking result that it is enough to test for a small number of pages in order to both enumerate users' interests and perform an efficient and unique behavioral fingerprint; we show that testing 50 web pages is enough to fingerprint 42% of users in our database, increasing to 70% with 500 web pages. Finally, we show that indirect history data, such as information about *categories* of visited websites can also be effective in fingerprinting users, and that similar fingerprinting can be performed by common script providers such as Google or Facebook.

1 Introduction

Motivations: A body of prior work has studied the leakage of private information when users browse the Web. In addition to data obtained by direct observation of Web traffic, known vectors for privacy loss range from explicit inclusion of third-party scripts [11] to long-standing browser mechanisms which allow Web authors to query the contents of a user's DNS or browser caches [6, 25], history store [9] or configuration information [5].

In this paper we analyze the consequences of the existing ability of Web authors to determine which websites a user has visited. Specifically, we investigate whether a user's browsing history, i.e. the list of visited websites, constitutes a fingerprint which can be used to uniquely identify and/or track the user. As users' Web browsing preferences are directly related to the content in which they are interested, it is suspected that such preferences may be individual in nature, akin to traditional biometric mechanisms such as friction ridges of a finger, or retinal patterns.

Since, as we discuss in Section 2.1, there exist several largely reliable methods to determine if a user has visited a particular website, we expect that such knowledge can be easily gathered by invasive webmasters [10]. Therefore, a careful study of the consequences of such history detection from a privacy perspective is necessary.

Contributions: Our investigation is based on the analysis of a large dataset of $382,269$ users' browsing histories obtained in a real-world scenario [8]—each browsing history is a subset of websites visited by a given user. We also convert these histories into interest profiles by labeling each history element with a

category obtained by querying a website categorization service [17]. The interest profile of a user is then defined as the categories of the sites he/she visited.

We then analyze these two datasets, focusing on the following questions:

- How are web histories and interest profiles distributed? Are they unique or similar? How large is the set of visited websites which must be queried to accurately distinguish between users if at all possible?
- Are web histories and interest profiles stable? In other words, do they constitute good behavioral fingerprints and can tracking websites rely on such data?
- Can web histories and interest profiles collected by service providers, such as Facebook or Google, be used to fingerprint users in the absence of any other history detection or tracking mechanisms?

Our findings indicate that, under the sample we studied, the vast number of users' Web histories are distinct and in fact unique; and that this is still the case when analyzing the general categories of visited website sets, rather than individual websites themselves. Strikingly, it is also possible to attribute a distinct history fingerprint to a user by testing just a small number of pre-defined popular pages and such fingerprints, to some extent, are stable over time; in certain cases such detected browsing history sets are recreated after a one-time clearing of the browsing history.

The potential risks and privacy implications result from a combination of sensitivity and persistence of this type of interest-based behavior data. In general, it is not simple to change one's browsing habits—if Web browsing patterns are unique for a given user, history analysis can potentially identify the same user across multiple Web browsers, devices, or if the user permanently changes her physical location.

Paper organization: This paper is organized as follows. First, we provide an overview of existing methods which allow the detection of users' browsing histories, and prior work on Web-based fingerprinting techniques. We then outline our methodology for gathering and processing browsing history data. In Section 4.1 we analyze the distinctiveness of browsing profiles. In Section 4.2 we perform an analogous analysis using only high-level website categories. In Section 4.3 we review the stability of users' browsing histories. We conclude by outlining some countermeasures to history detection and analyzing the history tracking potential of third-party script providers such as Google and Facebook (Section 5).

Ethical considerations: In this study, we utilize data gathered by a proof-of-concept Web application which was created to inform users about the risks of history detection and describe mitigations [8]. Users visiting the site automatically executed the default history test and the detected contents of their browsing histories were sent to a server so that it could display all gathered information back to the user. As such, the dataset contains real history information including records of user visits to potentially sensitive websites which might, in certain cases, be harmful to users if revealed to other individuals or organizations (e.g. employers).

Recognizing the significant problem which arises from gathering such information, we have taken the following precautions when analyzing and storing data:

- Apart from showing each user the contents of their browsing history as detected by our system, all data was analyzed and displayed in aggregate, without uncovering any user-identifiable information.
- The system does not allow any users to view any past history detection results (including their own) and does not use any mechanisms for tracking users (i.e. cookies)
- All log data was deleted from Internet-facing hosts and was used solely to prepare aggregate information presented in this work.

We also believe that an important consideration is the fact that such usage data is obtainable by any website visited by the user, and the detection techniques are widely known. In fact, we are aware of several

toolkits which gather user history data and send them covertly to their origin websites [10]; the desire to understand the implications of such privacy leaks is a major motivation for this work.

2 Background

2.1 History detection mechanisms

Web authors have in their arsenal a variety of techniques which enable them to query the contents of a visitor's browsing history. One of the most well-known approaches, and the one used to gather data for this work, is the querying of URLs in the browser's history using the CSS :visited mechanism [9]. While modern browsers introduced fixes [2] for high-speed history detection via this technique, certain interactive attacks are still possible [19]. In addition, about 25% desktop user agents and an even higher proportion of mobile browsers are still susceptible to this technique [18].

An alternative approach is the timing analysis allowing detection of items in a Web browser's cache, introduced by Felten et al. [6] and recently perfected by Zalewski [25]. Yet another history detection vector is the timing of DNS queries [12, 20]. Even in the absence of such client-side timing attacks, in many cases it is possible to reveal if a user is logged into a particular website by analyzing error messages or timing server responses ; however, this technique is likely more difficult to generalize in a real attack.

Such history detection attacks have been successfully demonstrated in various scenarios. Wondracek et al. showed the potential to deanonymize social network users [21]. Jang et al. demonstrated that such techniques are indeed in use in the wild as well as study different leakage channels [10] which only makes this threat more significant. The work done in [9] revealed the susceptibility of the majority of Internet users to high-speed history detection and showed that it can uncover large amounts of data about users. Timing analysis techniques have been successfully demonstrated [25] and were practically used to discover the contents of the users shopping carts [3].

In addition to potentially being leaked to third-party Web authors, such Web browsing data is revealed to legitimate service providers such as DNS server operators, third-party script providers [11] or ad networks, even if it is not explicitly gathered. Thus, we expect information about websites visited by Web users as part of day-to-day browsing is quite easily obtainable by a variety of parties.

2.2 Web fingerprinting

There are few results on behavioral profiling and fingerprinting analysis based on large samples of data or results of real-world surveys. One recent and prominent is an excellent study of browser fingerprints in [5], where the fingerprinting is based on plugins, fonts and other browser configuration.

Another recent and important example is [24], where the authors study a large data sample from users of Hotmail and Bing and focus on the potential of tracking relating only to the host information and other such as browser cookies and User-Agent string. Fingerprinting potential based on the detection of browsers' configuration using JavaScript is also analyzed by Mowery et al. [15]; similar techniques have been employed by Eckersley in his experiment [5].

Different aspects of timing, as well as DNS cache timing attack are explored by Jackson et al. [7]. If the attacker has access to the routing nodes, he can use the network flow to fingerprint the users, as shown in [23]. Behavioral biometry, where fingerprints are based on the behavioral aspects and traits such as typing dynamics or voice analysis are described in [14, 13, 22]

3 Methodology

Data analyzed in this paper was gathered in the What The Internet Knows About You project [8], aimed at educating users and browser vendors about the consequences of Web browser history detection. For the overall discussion of the system refer to [9] where the authors discuss the susceptibility of Web users to such techniques, their performance, mitigations, as well as give a detailed review of history detection mechanisms.

3.1 Experimental Setup

The experimental system utilized the CSS :visited history detection vector [4] to obtain bitwise answers about the existence of a particular URL in a Web browser's history store for a set of known URLs.

The system leveraged a two-tiered architecture to first detect the "primary links", i.e. top-level links such as `www.google.com`, and use that knowledge to query for secondary resources associated with the detected primary link, such as subpages within each site.

The history detection demonstration site gained popularity and between January 2009 and May 2011 we gathered $441,627$ profiles of unique users who executed a total of $988,128$ tests. We expect that our data sample comes from largely self-selected audience skewed towards more technical users, who likely browse the Web more often than casual Internet users.

In this paper we refer to $382,269$ users who executed the default "popular sites" test of over 6,000 most common Internet destinations.

The "popular sites" list was created out of 500 most popular links from Alexa [1], $4,000$ from the Quantcast popular websites list [16], lists of common government and military websites, and several custom-chosen URLs selected for their demonstration and education potential. In this work we analyze data about visited "primary links" only, without analyzing any detected subresources within a website.

This approach did not make it possible to obtain user's entire browsing history, and that was not the aim of the study. However, this was not necessary, as it is sufficient to show that the actual subsets are unique: a considerable unique number of profiles within a large dataset most likely indicates the overall uniqueness of the whole history superset of these Web users (if the subsets of some supersets are unique, those supersets must consequently be unique).

Other history detection techniques we describe will likely have different semantics and capabilities, but will, in the majority of cases, be able to obtain similar answers about whether a user had visited a particular website. Thus, we can analyze the gathered data without loss of generality.

3.2 Data Collection

For each user, the set of all visited links found by the detection algorithm was sent to the server so that it could display results to the user—in the majority of cases, the detection phase took less than 2 seconds for the set of 6,000 popular sites.

In addition to history test results, the system gathered each user's browser version information (User-Agent header), IP address as visible to our webserver and the test date. The system did not make use of any side-information such as cookies, Flash cookies or other persistent storage techniques to track users and did not provide any capability to review any history results except for the most recent test run.

An important aspect of the system was educating users about the privacy risks associated with history detection; along with the results page listing detected URLs we provided users with general information about the problem, references to related research, and mitigations against history detection. Thus, it was assumed that a considerable number of users will clear their histories after visiting the site, so in our data

analysis we refer to history data received during the first test execution for a given user (except for the stability analysis of repeat visitors which analyzes data from subsequent test runs).

3.3 Terminology

The data we are summarizing in this work may be perceived as a collection of Web history profiles H_s (for sites). If a profile p_i, which is a collection of visited sites, is present in H_s, then $Count(p_i)$ is the number of occurrences of p_i in H_s. If $Count(p_i) = 1$, this profile is unique in a dataset (it is present only once, and thus relates to a unique user). When the number of detected links for a particular user is very small (1-3), it is likely that $Count(p_i)$ may be greater than zero as an obvious consequence of the pigeonhole principle. If $Count(p_i) > 1$ then this profile is not unique but we may treat is as distinct in the dataset.

Sites on the Internet may be attributed to different categories, for example `www.google.com` is a search engine, and `cnn.com` is a news portal. Therefore the sites in our dataset can be attributed to different categories and it is straightforward to define a category profiles collection H_c, where the Web pages are mapped to categories.

Moreover, H_s may be converted into a bit vector collection V_s in the following way: create a list of sites ordering them by popularity in the discussed dataset and map every profile $p_i \in H_s$ to a vector $v_i \in V_s$; a bit in this vector is set if a page on a given position is visited (present in p_i). Such conversion allows us to operate on bit slices of these vectors in the rest of our analysis.

4 Web Behavioral Fingerprints

To establish the potential of Web history traces as a fingerprinting mechanism we perform the analysis at three different levels of granularity. First, we analyze the raw data which includes all websites we detected in a user's browsing history. Second, we convert the raw history profile to a category profile where each website is assigned to one of several buckets (e.g. shopping, news, or social networking). Third, we convert such a category vector to a tracking vector as visible to scripts downloaded from the `facebook.com` and `google.com` domains (Section 5).

At each point we examine the stability of the resulting fingerprints.

4.1 Web History Profile Uniqueness

4.1.1 Methodology

We analyzed history profiles by computing distributions of the profiles per size of the profile (number of visited sites) to inspect profile diversity, both for all users and specifically for mobile browser users. The similarity metric we utilize is the Jaccard Index. For two sets A, B the Jaccard Index is computed as $\frac{|A \cap B|}{|A \cup B|}$. Two sets are equal if Jaccard Index is 1, and they are certainly similar (correlated) if it is larger than 0.7.

4.1.2 Results

Figure 1 displays the distribution of Web history profiles according to their size (pink line). It also shows the distribution of unique (red line) and non-unique profiles (green line). A profile is unique if it is associated with a single user. The unique distribution was prepared by counting all unique history sets - if a Web history profile appeared more than once ($Count(p_i) > 1$ for $p_i \in H_s$, which is especially the case of history sizes 1-5) it is counted as a single fingerprint (it is distinct but has non-zero siblings, thus non-unique).

A profile is non-unique if it is common to several users; a percentage of unique profiles is also presented. This percentage is computed by dividing, for each profile size, the number of unique profiles by the total number of profiles.

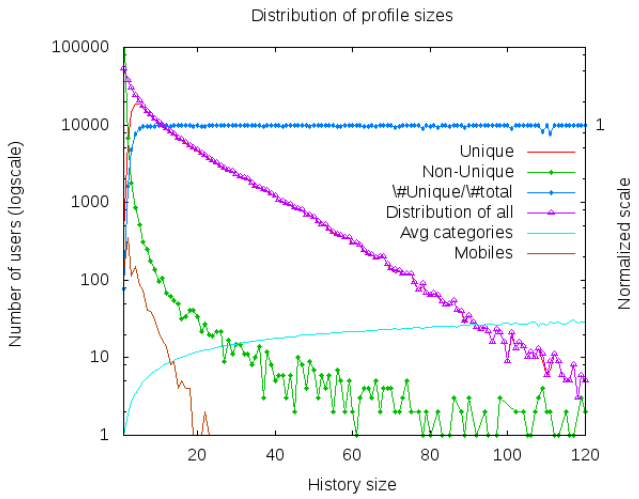


Figure 1: Distributions of: general, unique and non-unique Web history profiles per history size, normalized with respect to total number of histories in each history set. Average number of categories per history size is also shown.

It is interesting to note the clearly visible peak of detected sites - there are far more profiles with detected history size greater than 3, compared to smaller history sizes. The general distribution is also shown; it is easy to observe the prevailing unique profiles starting close to the point $X = 4$, indicating that detecting as few as four visited sites provides a useful fingerpriting signal.

In our dataset, the average number of visited sites per profile is 15, and the median is 10. However, analyzing just the history sizes larger than 4 ($223,197$ of such profiles) results in the average number of links 18 (median 13) with 98% profiles being unique.

For history sizes $1-4$ the ratio of unique profiles to total profiles ranges from 0.008 to 0.76 but in average, for all the profiles, 94% of users had unique browsing histories. This result suggests that browsing patterns are, in most circumstances, individual to a given user.

Figure 1 also displays the distribution of the Web history of mobile users (orange lines). The patterns of mobile Web usage is created by looking for a specific User-Agent HTTP header, detecting Web browsers on iPhone, iPad, Android and BlackBerry devices. The graph presents data from 1256 users. Even though this was not a large sample, with respect to overall number of profiles, different usage patterns are observed—specifically, the detected history sizes are smaller, which might suggest that the Web use on mobile devices is not as frequent or large as it is with non-mobile Web browsing.

To understand relative differences in observed history profiles, we analyzed the similarity between the fingerprints from different users. Within all of the history sizes, we measured the similarity using Jaccard index for all distinct (rather than unique) Web history profile pairs, i.e. if a profile was seen several times it is being treated as a single fingerprint. The averaged (by total number within history size set) result as a function of the history size is presented in Figure 2. The fingerprints were largely dissimilar, which confirms that enumerating actual browsing interests is feasible.

4.1.3 Selection

For a given website, what is its importance when studying history fingerprint uniqueness? If a given Web page is not very popular, or not visited very often, what is the "share" it provides to the overall uniqueness of a profile? Due to the sparseness of the set of visited sites for most users, we hypothesize that the most

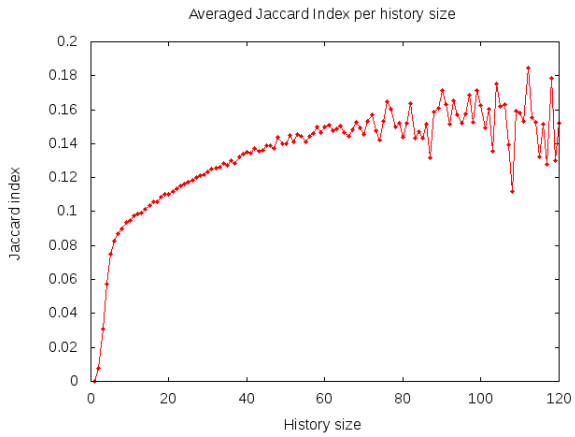


Figure 2: Jaccard Index as a function of history size. It is always smaller than 0.2 hence the profiles are dissimilar

commonly visited sites will be most useful in creating fingerprints. Of course, if a site is visited by only one person it contributes significantly to the uniqueness of this particular profile. However, it is of no value when establishing fingerprints of other users; it would likely negatively affect the performance of any real data-collecting implementation.

To analyze this, we sorted all of the websites with respect to their popularity metric in our data. After that, each Web history profile was converted to a vector representation. A bit in a vector was set if an associated page has been visited. Therefore, the most popular pages were the left-most bits. We studied slices of these vectors focusing on the the first K left-most bits, for different values of K. We created a frequency distribution of such a uniqueness set, as shown on Figure 3. The X axis represents the number of distinct profiles, as counted from the dataset, which correspond to a specific anonymity set (Y axis), ordered from largest to smallest. For example, the point (X=10;Y=1000) indicates that the 10th most popular profile is shared by 1000 users.

In order to improve readability the scale is logarithmic. As is seen on axis X, over $250,000$ of profiles belong to the set 1 (from the axis Y) and thus they are unique. The previous analysis based on subvectors is also presented here. Vectors of sizes 10 (only 10 sites corresponding to maximum of 1024 possible unique profiles) are not enough to create a large set of unique attributes. However, increasing the vector size to 50 provides a very accurate approximation of the data from the full website set. Thus, testing for as few as 50 well-chosen websites in a user's browsing history can be enough to establish a fingerprint which is almost as accurate as when $6,000$ sites are used.

The information-theoretic surprisal is defined as $I = -log_2(F(x))$, where F is a probability density function related to an observation of a particular fingerprint in our dataset. Figure 4 shows a cumulative distribution function of surprisal computed for both Web history and category (vector) profiles. Unique profiles contribute over $18b$ of surprisal and this is the case for the majority of profiles. For general Web history profiles, about 70% of them are close to the maximum surprisal, and for only 50 links in the history this is still the case for about 50% of profiles. Although the maximum surprisal is (unsurprisingly) reached when all sites are considered, it can be approximated by a vector of size 500, as shown in Table 2.

We conclude that the most important sites for fingerprinting are the most popular ones because a considerable number of history profiles are still distinct, even in a small slice of 50 bits. This perhaps counter-intuitive observation may be a result of applying the binomial coefficient: if we assume that in average a user has visited 20 sites of the 50 most popular sites, there are still $\binom{50}{20} = 4.71 \times 10^{13}$ combinations possible. It is worth noting that although users' histories have been tested against more than $6,000$ Web pages, this

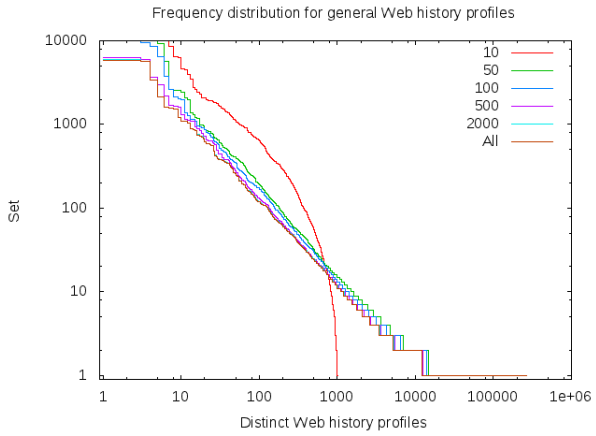


Figure 3: Frequency distributions computed for different bit slices. Even conservatively, only the first $50b$ are sufficient to obtain a large number of unique fingerprints.

does not correspond to a space size of 2^{6000} because of factors such as website popularity, and visitedness correlations based on user interests. Moreover, as is shown in the table 2, it is clear that the dynamics of changes are different than for a uniformly random distribution.

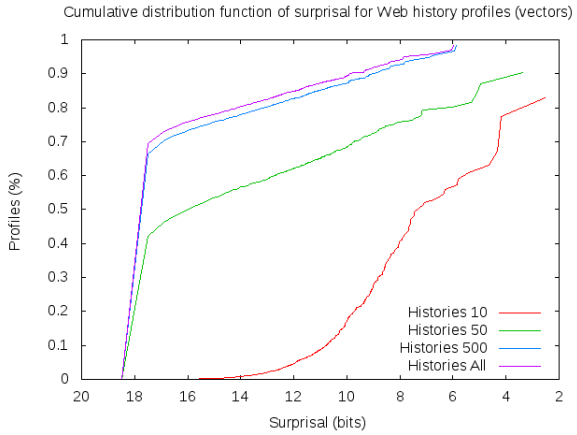


Figure 4: Cumulative distribution function of surprisal for Web history profile vectors of different sizes

The importance of this finding should not be underestimated. If a small number of links is sufficient to prepare an efficient fingerprint for a user, then existing techniques such as DNS or browser cache timing analysis can be used to perform similar analysis.

4.2 Category Profile Uniqueness

4.2.1 Methodology

To extend our analysis, we converted each history profile into a category profile. This was performed by replacing each website of a profile by the general category it belongs to by using the Trend Micro Site Safety Center categorization service [17]. This service takes as input a website URL and outputs a category. Trend Micro uses 72 different interest categories (including news, games, social networking and others).

Category	Count	Category	Count
Search Engines / Portals	1.0000	Social Networking	0.9145
Shopping	0.7891	Computers / Internet	0.7855
News / Media	0.7489	Streaming Media/MP3	0.7229
Entertainment	0.6734	Reference	0.5098
Games	0.2958	Pornography	0.2679
Auctions	0.2436	Government / Legal	0.2385
Software Downloads	0.2201	Blogs / Web Comm.	0.2062
Photo Searches	0.1970	Peer-to-peer	0.1954
Email	0.1666	Business / Economy	0.1552
Sports	0.1250	Financial Services	0.1052
Pers.Net.Stor./File Downl.Srv	0.0950	Travel	0.0920
Adult / Mature Content	0.0588	Education	0.0559
Internet Telephony	0.0521	Chat/Instant Messaging	0.0519
Personals / Dating	0.0516	Internet Radio and TV	0.0493
Vehicles	0.0452	Restaurants / Food	0.0378

Table 1: 30 most popular categories normalized with respect to Search Engines/Portals $(417,750)$. Remaining categories not listed here sum up to 0.35.

4.2.2 Results

We computed a unique set of interests for every Web history profile by discarding repeated occurrences of the same category in profiles. This resulted in $164,043$ distinct category profiles, out of which 88% are unique (i.e. only attributed to a unique user).

Consequently, we can observe that a large number of users have unique personal browsing interests even when analyzed using the more coarse-grained category metric. Figure 1 shows the average number of unique categories per each profile according to history sizes. In a real scenario of an advertising provider, multiple repetitions of each category in the profiles are likely used to enumerate the strength of interest in the category which provides additional information; however, in our analysis, we did not utilize this signal.

Table 1 shows the categories popularity in the profiles, with respect to the most popular category - *Search Engines / Portals*.

Additionally, we converted such category profiles into vectors, similar to the earlier analysis of Web history profiles. The first element of the vector corresponds to the most popular category, i.e. *Search Engines*, the second element to the second most popular category, i.e. *Social Networking*, and so on.

We then computed the frequency distributions for different sizes of the category vector, as in Figure 5.

The associated cumulative distributions are presented on Figure 6. Results show that subvectors of size 30-50 are seen to be enough to prepare a meaningful profile and still maintain a large number of unique profiles. In terms of uniqueness potential, it is sufficient to analyze 30 categories as the data quickly follow the same long tail pattern as in "raw" Web history profiles. However, categories of size 10 are clearly not sufficient because they do not carry enough entropy to distinguish between the large number of profiles in our dataset.

4.2.3 Summary

The conversion from Web history profiles to only use each website's category decreased the overall number of unique profiles. However, we observe that even with the coarser-grained metric there is still a large number of distinct profiles. Assuming conservative profiling and discarding the interest rates for given

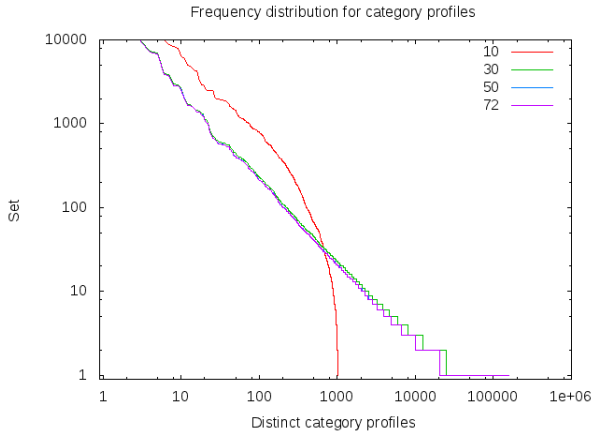


Figure 5: Frequency distributions computed for different bit slices (categories).

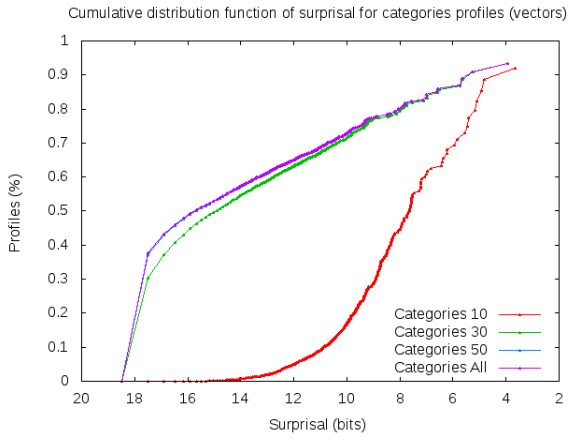


Figure 6: Cumulative distribution (category profiles) function of surprisal for history profiles vectors of different sizes

categories, we were still able to attribute a unique category profile to 39% users.

4.3 Stability of history profiles

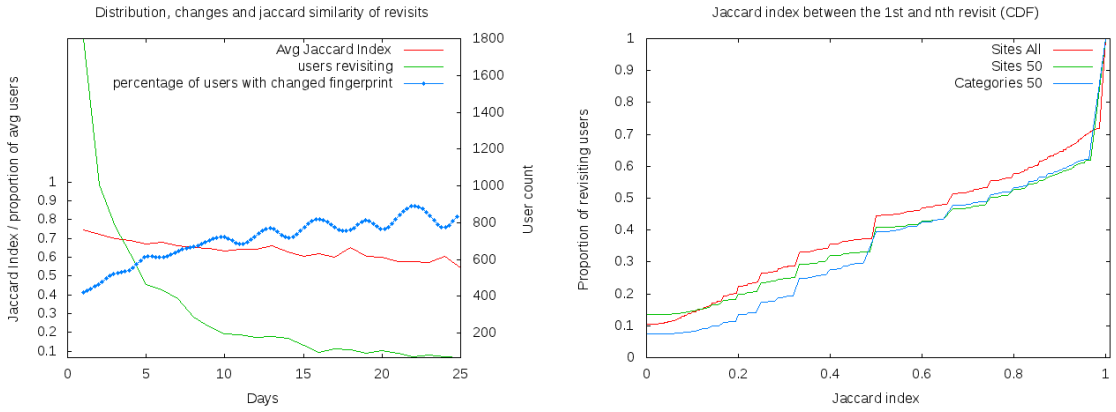
In order to analyze the potential for user tracking using history fingerprints we must understand the stability of Web preferences and browsing history contents. Many Internet users browse the Web on a daily basis, and their browsing history is constantly populated with new websites and subresources. However, all that is not necessarily crucial to address the stability problems since it is sufficient to limit the testing to a small list of sites (e.g. if a user's fingerprint is unique for a subset of n sites, visiting a website outside of the tested set will not affect it).

To quantify this, we analyze history contents of repeat visitors to our test site. Since the dataset is timestamped, it is possible to verify how time affects the history or even if the users cleared their history after learning about websites' ability to detect sites in their history. If a user, identified by the tuple (IP, User-Agent), visited the site on a day_1 and day_n (for $n > 1$), we computed the differences $\{x|day_n - day_1, \text{for all } n > 1\}$ and the similarity between these potentially different Web histories for these two days.

In the analysis only the first 25 days are shown because after this period the number of revisits were small (although some of the users revisited the site even after a year and a half and sometimes the fingerprints for these revisits were also identical). For this analysis we considered profiles constructed from the most popular 50 sites (using subvectors as described before). It can be seen on Figure 7a and it suggests that in considerable number of cases the history remains similar with time, which is especially the case for the first few days after the initial visit. When considering up to 500 most popular sites, the figure do not change significantly.

The cumulative distribution function of the Jaccard Index between the users who re-entered the site is depicted on Figure 7b and as the peak shows, as many as 38% of users had a strongly correlated history.

For 50 subvectors, the fraction of users with a correlated history was 57% and category profile subvectors of size 50 show a similar trend; the number of observed changes is significant. A comparison of this figure to the similar analysis from [5] indicates that Web behavioral fingerprints are slightly less stable than those based on browser configuration for the first few days; the situation becomes comparable for the following days.



(a) **(b)**

Figure 7: (a) The horizontal axis represents the time difference between the first and consecutive visits to the site. The average Jaccard similarity is high for the first days. (b) CDF of Jaccard Index between the fingerprints for the 1st and nth revisit.

Vector Slice	All	Google	Facebook
50	155117	30398	195
100	196899	85682	4130
500	244990	160190	73606
whole	255210	177459	91915

Table 2: Number of unique profiles for different bit slices and vectors: All, Google and Facebook (368284 in total for "popular sites" test)

5 Privacy Risks Analysis

Previously we have shown that browsing preferences are often unique to users and fingerprints derived from them are stable over time in some cases. It is also clear that Web authors can employ known techniques to mount attacks against Web users. Here we focus on a different aspect: what are the possibilities to perform the above analysis by Web service providers? It is important to note that Web service providers have almost constant access to the users' browsing content and are potentially in an ideal position to track the user. In some of the cases users already have accounts on their services which immediately provides information about their identity to the service provider.

5.1 Tracking

Choosing two of the most prominent Web service providers: Google and Facebook and by using real-world scenario data we have verified the extent to which the users may be tracked.

5.1.1 Methodology

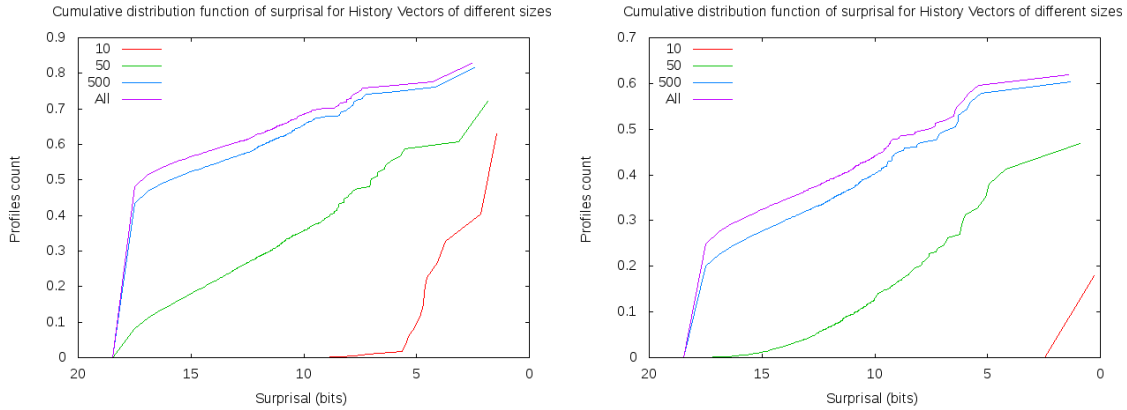
In order to verify to what extent large Web service providers could recreate our profiles, we converted the Web history profiles to the more specific *tracking profiles* which contained only the sites on which we detected scripts from Google and Facebook. In this subsection we also compare them with the data obtained by analyzing "raw" Web histories, as well as category profiles.

5.1.2 Results

After constructing such tracking profiles we computed the distribution of profile sizes and compared them with the one shown on Figure 1. As can be seen, the conversion to tracking profiles decreased the overall sizes of such profiles; however, many of them are still quite large. Figure 8a depicts the distribution of these profiles for "raw" Web history profile (vectors) and these tailored against Facebook and Google services.

We performed the same analysis as in the case of Web history profiles. The numbers of unique fingerprints observed according to the subvector sizes are presented in Table 2. Smaller vector slices for Google and Facebook mean smaller number of unique tracking profiles; for the whole vector ("All") this is sufficient. Smaller vector slices in the former case may result from the nature of the "popular sites" list, which was not prepared to specifically measure this setting. The discrepancy between Google and Facebook is likely due to the fact that Google is providing more Web services than Facebook (AdSense, Analytics, etc.).

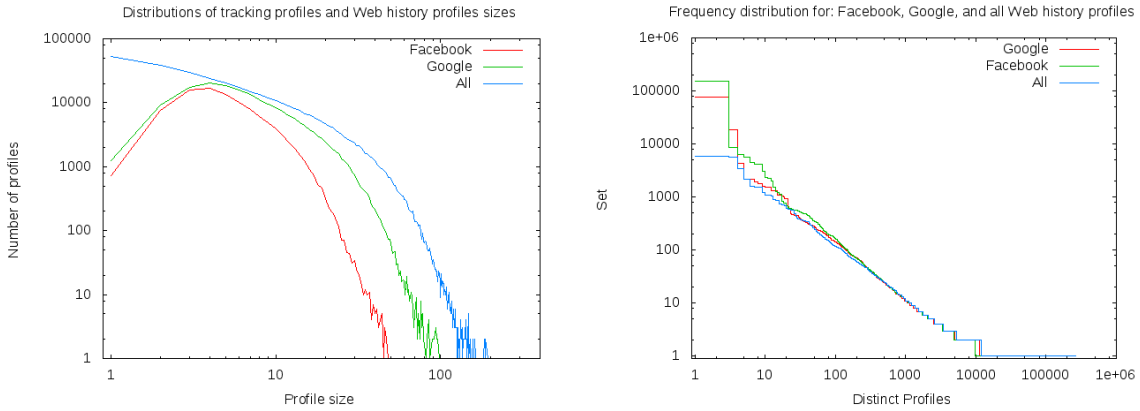
We also computed profile frequency distributions; results are depicted on Figure 8b. The number of unique fingerprints for specific cases of Google and Facebook are smaller, but still considerably large and make analysis possible. They also both occupy larger sets (compared to Web history profiles), and especially in the case of Facebook we observed far less unique fingerprints. The "popular sites" list of pages created



(a) Google **(b)** Facebook

Figure 9: Cumulative distribution (Google and Facebook tracking profiles) function of surprisal for history profiles vectors of different sizes.

for the experiment was not tailored in order to test this particular issue, but as can be seen, there are still many unique profiles which makes this analysis relevant.



(a) Distributions **(b)** Frequency

Figure 8: Distributions and frequency distributions of profiles (All, Google, Facebook).

We then computed the surprisal of the tracking profiles for different bit vector sizes. Figures 9a and 9b show cumulative distribution functions of surprisal for Google and Facebook (vectors). They should be compared to Figure 4. A similar behavior is clearly seen, especially for Google profiles with about 50% of profiles still being unique. The results for Facebook are not as accurate (about 25%).

6 Countermeasures

The problem of information leakage in the modern browsers is not entirely surveyed and researched. Therefore it is not possible to give a perfect and general solution. Particular techniques which allow for history detection can also be merely partially addressed.

The data analyzed in this paper was gathered by performing CSS :visited history detection, which is now generally fixed in the modern browsers, although it will continue to work for older browser installations which constitute to a considerable fraction of Web users. Script blocking may be helpful against certain attacks, but other techniques may still be possible (as was the case with CSS detection which could be conducted without JavaScript). Scripts can, in general, block the 3rd party content present on the sites but this approach often limits the user experience and cannot always be recommended for the average Web user. Plugins (such as AdBlock) blacklisting certain Web resources exist and can defend against this in some cases.

Even after deploying all of available defenses, a user is still prone to other techniques involving timing analysis which can currently only be solved by clearing/removing browser caches, which introduces a usability/privacy trade-off. If a user clears her browsing history and disables browser caching, she will still be prone to other attacks such as DNS cache timing which cannot be cleared as easily.

In the end, the user cannot defend against unknown privacy leaks. If a user wishes to defend against fingerprinting by testing a known pre-defined list of pages, she can ensure that she has visited all of the tested pages—this way, it will be impossible to create a conclusive fingerprint or to enumerate the user's browsing interests. However, we don't expect this to be a realistic mitigation for most Web users.

7 Conclusion

Our work presents the first large-scale analysis of Web browsing histories and their potential for fingerprinting and tracking Web users. In our dataset composed of $368,284$ web histories, more than 69% of users have a unique fingerprint, with a surprisal larger than 18 bits.

The results indicate that Web browsing histories, which can be obtained by a variety of known techniques, may be used to divulge personal preferences and interests to Web authors; as such, browsing history data can be considered similar to a biometric fingerprint. Since Web histories are largely unique to person and, as shown, stable over time in some of the cases, they can be understood as an identifier and potentially be used to strengthening of tracking, in addition to revealing information about a particular user's browsing interests.

We also observed a striking result: it is sufficient to test just a small number of pre-defined sites to obtain vast numbers of different Web history profiles. This indicates that even inefficient techniques for history detection can allow history-based user fingerprinting. Currently there are no known and fully-reliable mitigations to such tracking. By converting visited website data to category profiles, we were able to map the personal interests in much the same way as it is being done by advertising companies. Such profiles, when studying only unique types of categories, were still unique. An analysis of tracking potential (on two examples of Google and Facebook, shown in in Section 5) brings us to a conclusion that Web service providers are also in a position to re-create users' browsing interests.

An interesting question is whether it's possible to extrapolate our data to the entire Internet population. In other words, out of the 2 billion Internet users, what is the percentage of users that have an unique web-history or category profile? As discussed in [5], this number is very difficult, if not impossible, to compute; however, we argue that, as opposed to browser fingerprints, web-histories contain more semantic information that can be exploited to reduce the population size. For example, by considering the language of visited pages, the population size can be reduced from few billions to few millions users. Furthermore, some location-specific sites, such as weather forecast sites, can be used to reduce the population size to few hundred thousands users. Finally, as opposed to the browser fingerprints that rely on a fixed set of browser characteristics, in our case the number of tested or tracked sites can always be increased in order to increase the fingerprint size, and therefore the percentage of unique web-histories.

In the end we believe that Web browsing preferences can be used as an efficient behavioral fingerprint

which is in many cases stable over time. Attacks employing existing techniques such as timing analysis make it simple to conduct large-scale fingerprinting of Web users' interests; this is aided by our observation that only a small set of sites needs to be tested in order to obtain a vast number of unique profiles.

References

[1] Alexa. Alexa 500. `http://alexa.com`.

[2] L. D. Baron. Preventing attacks on a user's history through css :visited selectors. `http://dbaron.org/mozilla/visited-privacy`, 2010.

[3] A. Bortz and D. Boneh. Exposing private information by timing web applications. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 621–628, New York, NY, USA, 2007. ACM.

[4] Bugzilla. Bug 147777 - :visited support allows queries into global history. `https://bugzilla.mozilla.org/show_bug.cgi?id=147777`, 2002.

[5] P. Eckersley. How unique is your web browser? In *Privacy Enhancing Technologies*, pages 1–18, 2010.

[6] E. W. Felten and M. A. Schneider. Timing attacks on web privacy. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 25–32, New York, NY, USA, 2000. ACM.

[7] C. Jackson, A. Bortz, D. Boneh, and J. C. Mitchell. Protecting browser state from web privacy attacks. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 737–744, New York, NY, USA, 2006. ACM.

[8] A. Janc and L. Olejnik. What the internet knows about you. `http://www.wtikay.com/`.

[9] A. Janc and L. Olejnik. Web browser history detection as a real-world privacy threat. In *ESORICS*, pages 215–231, 2010.

[10] D. Jang, R. Jhala, S. Lerner, and H. Shacham. An empirical study of privacy-violating information flows in JavaScript Web applications. In A. Keromytis and V. Shmatikov, editors, *Proceedings of CCS 2010*, pages 270–83. ACM Press, Oct. 2010.

[11] B. Krishnamurthy and C. 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.

[12] S. Krishnan and F. Monrose. Dns prefetching and its privacy implications: when good things go bad. In *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, LEET'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.

[13] B. Miller. Vital signs of identity. *IEEE Spectr.*, 31:22–30, February 1994.

[14] R. Moskovitch, C. Feher, A. Messerman, N. Kirschnick, T. Mustafic, A. Camtepe, B. Löhlein, U. Heister, S. Möller, L. Rokach, and Y. Elovici. Identity theft, computers and behavioral biometrics. In *Proceedings of the 2009 IEEE international conference on Intelligence and security informatics*, ISI'09, pages 155–160, Piscataway, NJ, USA, 2009. IEEE Press.

[15] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham. Fingerprinting information in javascript implementations. In *Proceedings of W2SP 2011*, IEEE Computer Society, 2011.

[16] Quantcast. Quantcast. `http://www.quantcast.com/`.

[17] Trendmicro. Trend micro site safety center. `http://global.sitesafety.trendmicro.com`.

[18] W3Schools. W3schools online web tutorials. `www.w3schools.com`.

[19] Z. Weinberg, E. Y. Chen, P. R. Jayaraman, and C. Jackson. I still know what you visited last summer: Leaking browsing history via user interaction and side channel attacks. In *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, SP '11, pages 147–161, Washington, DC, USA, 2011. IEEE Computer Society.

[20] C. E. Wills, M. Mikhailov, and H. Shang. Inferring relative popularity of internet applications by actively querying dns caches. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, IMC '03, pages 78–90, New York, NY, USA, 2003. ACM.

[21] G. Wondracek, T. Holz, E. Kirda, and C. Kruegel. A practical attack to de-anonymize social network users, ieee security and privacy. In *IEEE Security and Privacy*, Oakland, CA, USA, 2010.

[22] R. V. Yampolskiy and V. Govindaraju. Behavioural biometrics: a survey and classification. *Int. J. Biometrics*, 1:81–113, June 2008.

[23] T.-F. Yen, X. Huang, F. Monrose, and M. K. Reiter. Browser fingerprinting from coarse traffic summaries: Techniques and implications. In *Proceedings of the 6th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA '09, pages 157–175, Berlin, Heidelberg, 2009. Springer-Verlag.

[24] T.-F. Yen, Y. Xie, F. Yu, R. P. Yu, and M. Abadi. Host fingerprinting and tracking on the web:privacy and security implications. In *19th Annual Network and Distributed System Security Symposium (NDSS) 2012, Internet Society*, 2012.

[25] M. Zalewski. Browser security handbook, part 2. `http://code.google.com/p/browsersec/wiki/Part2`, 2009.

Location Privacy Threats at Public Hotspots

Nevena Vratonjic, Vincent Bindschaedler, Kévin Huguenin, and Jean-Pierre Hubaux

School of Computer and Communication Sciences, EPFL, Switzerland

Abstract

Location privacy has been extensively studied over the last few years, especially in the context of location-based services where users purposely disclose their location in order to benefit from convenient context-aware services. However, to date little work has been devoted to the case of users' location privacy being compromised *by others*.

This paper studies a concrete and widespread example of such situations, specifically the location-privacy threat created by access points (public hotspots, FON, home routers, *etc*.) making use of Network Address Translation. Indeed, because users connected to the same hotspot share a unique public IP, a single user making a location-based request is enough to allow the service provider to map the IP address of the hotspot to its geographic coordinates, thus compromising the location privacy of all the other connected users. Even in the case where IP addresses change periodically (e.g., by using DHCP), an attacker is still able to update a previous (IP, Location) mapping by inferring IP changes from authenticated communications (e.g., cookies, usernames).

The contribution of this paper is four-fold: (i) We identify a threat to users' location privacy caused by the use of shared public IP addresses; (ii) we formalize and analyze theoretically the aforementioned problem; (iii) we quantify the privacy threat theoretically and we assess the current state experimentally by using real traces from existing hotspots; and (iv) we discuss various countermeasures that can be used to thwart the threat.

1 Introduction

With the ubiquity of mobile devices with advanced capabilities, it is becoming the norm for users to be constantly connected to the Internet. In particular, users can benefit from many online services, while on-the-go. Among others, location-based services (LBSs) are increasingly gaining popularity. With an LBS, users share their location information with a service provider in return for context-aware services, such as finding nearby restaurants. Users also enjoy sharing location information with their friends on social networks (e.g., Facebook and Twitter). For example, they can then find friends in the vicinity or recommend places they visit. Although very convenient, the usage of LBSs raises serious privacy issues.

Location privacy is a particularly acute problem as location information is valuable to many potential adversaries. In particular, because many other pieces of information can be inferred from users' locations (e.g., participation in a political gathering), authoritarian regimes have incentives to collect such information. Location information is essential for many online service providers whose business models revolve around personalized services. A prominent example is (mobile) online advertising—an ever-increasing business whose annual revenue is tens of billions of US Dollars (e.g., $\$22.4$ billion in the US in 2011 [17])— as so-called location-specific ads based on the location information are significantly more appealing to users [13].

Usually, users do not disclose their location information to non-LBS service providers. These service providers can still obtain it through *IP-location*: determining the location of a (mobile) device from its IP address. However, available solutions do not provide sufficient precision for the commercial needs of service providers [13, 16], especially in the case of dynamic IPs. Another way for the service providers to obtain

a user's location is via transitivity, relying on other users to disclose their location and that of other users in their vicinity: if a provider knows the location of user B and that user A is close to user B, the provider knows roughly the location of A. Examples of such situations are when users report neighboring users (e.g., Bluetooth), or when they *check-in* (i.e., report their own location) on online social networks and tag friends who are with them. In some cases, even if the proximity information is not directly revealed by users, the adversary is still able to infer it, as we shall show.

In this paper, we study a location-privacy threat users are exposed to on a daily basis: when a user connects to the Internet through the same access point (AP) as other users (e.g., a public hotspot, home router) who make LBS queries, the service provider learns the user's location. Indeed, because all of the devices connected to a public hotspot, implementing network address translation, share the AP's public IP address, allocated (typically via DHCP) by an Internet Service Provider (ISP), when users generate LBS queries, the service provider learns the fine-grained geographic location of the AP and maps it to the AP's public IP. IP addresses remain the same for a certain amount of time (at least during the DCHP lease time), therefore for any connection for which the source IP is the same as the AP's IP, the service provider can conclude that the device is located nearby the location of the AP. Unfortunately, the user is usually not aware of this threat and more importantly, protecting her location privacy is no longer in the user's control. With such a system, the accuracy of the estimated location depends on the range of the AP (typically under a hundred meters) and on the accuracy of the locations reported by users in LBS queries.

The (IP, Location) mapping the adversary obtained for the AP stays valid until the IP address changes. Static IPs, long DHCP leases and re-assignment of the same IPs to the same clients therefore have a negative effect on location privacy. However, even when the IP address is renewed and changes, service providers have means to learn about the IP change, for example, due to the widespread use of authenticated services (e.g., e-mails, online social networks). Consider a user connected to the AP who checks her e-mail shortly before and after an IP address change. As a unique authentication cookie is appended to both requests, the service provider can conclude that the same user has connected with a new IP and can therefore update the (IP, Location) mapping with the new IP address.

The contribution of our work is four-fold: (i) We identify the threat to users' location privacy, which arises from the use of shared public IP addresses. (ii) We formalize and analyze the problem. (iii) We quantify theoretically the location privacy threat and evaluate experimentally its scale based on traces from existing hotspots deployed across the EPFL campus. Even at a moderately visited hotspot, we observe the large scale of the treat: the adversary learns the location of the AP only a few hours after users start connecting and over 24 hours he can locate up to 79% of the users. And (iv) we discuss various countermeasures that could thwart the threat. To the best of our knowledge, this is the first paper that addresses this problem.

Our paper is organized as follows. In Section 2 we provide the relevant background. We describe the system architecture, the adversary and the threat model in Section 3. We formalize the problem in Section 4 and we analytically quantify the location privacy threat. We further evaluate the threat based on traces from deployed hotspots and present the results in Section 5. In Section 6 we discuss possible countermeasures and new business opportunities created by the threat. Finally, we conclude the paper in Section 7.

2 Background

In this section, we provide relevant background on the technical aspects underlying the considered problem.

IPv4 (public) address allocation To communicate on the Internet, hosts need public IP addresses. An IP address can be either *static*, i.e., permanently fixed, or *dynamic*, i.e., periodically obtained from a pool of available addresses, typically through the Dynamic Host Configuration Protocol (DHCP) [7]. The host can use the IP for a limited amount of time specified with the *DHCP lease*. For convenience, upon DHCP lease

expiration, hosts are often re-assigned the same IP. More than 62% of dynamic IPs on average remain the same over a period of at least 24h [28].

Network Address Translation (NAT) In order to cope with IP address depletion, Network Address Translation (NAT) was introduced [23]. NAT hides an entire IP address space, usually consisting of private IP addresses, behind one or several public IP addresses. It is typically used in Local Area Networks (LAN), where each device has a private IP, including the gateway router that runs NAT. The router is also connected to the Internet with a public IP assigned by an ISP. As traffic is routed from the LAN to the Internet, the private source IP in each packet is translated on-the-fly into the public IP of the router: traffic from all of the hosts in LAN appears with the same public IP – the public IP of the NAT router.

Geolocation Mobile devices determine their positions by using their embedded GPS or an online geolocation service. With a GPS, the computation takes place locally by using satellites positions and a time reference. Commercial GPS provides highly accurate location results ($\sim$30 meters) [14], especially in "open sky" environments. With online geolocation services (e.g., Google or Skyhook) a device shares some information about its surroundings, typically a list of nearby cell towers and Wi-Fi APs (i.e., their MAC addresses) together with their signal strengths, based on which the geolocation server estimates the location of the device by using a reference database. This database is built typically by deploying GPS-equipped mobile units that scan for cell towers and Wi-Fi AP and plot their precise geographic locations. In addition, they take into account inputs reported by users with GPS-equipped devices who provide both their coordinates and the surrounding parameters. The accuracy of such systems is in the range of 10 meters [22].

3 System Model

In this section, we elaborate on the considered setting, notably NAT access points, the location-privacy threat, and the adversary.

3.1 System Architecture

We consider a *NAT Access Point* setting, a prevalent network configuration, where users connect to the Internet through an access point (AP), such as a public hotspot, a home (wireless) router or an open-community Wi-Fi AP (e.g., FON), as depicted in Fig. 1. An AP, located at (x_1,y_1), is connected to the Internet by a given Internet Service Provider (ISP) and provides connectivity to the authorized users. The AP has a single *dynamic public* IP address that is allocated with DHCP by the ISP: The AP's public IP address is selected from a given DHCP pool of available IP addresses and is valid during the DHCP lease time. When connecting to the AP, each device is allocated a *private* IP address and the AP performs a Network Address Translation (NAT). Consequently, in the public network, all of the connections originating from the devices connecting through the AP will have the same source IP, which is the public IP address of the AP.

While connected to the Internet through an AP, users make use of various online services including news, search engines, e-mail, social networks, location-based and online geolocation services. Services can be used either in an authenticated (e.g., e-mail) or unauthenticated way (e.g., Web search). We consider that the requests a server receives from the devices connected to the AP are of the following types:

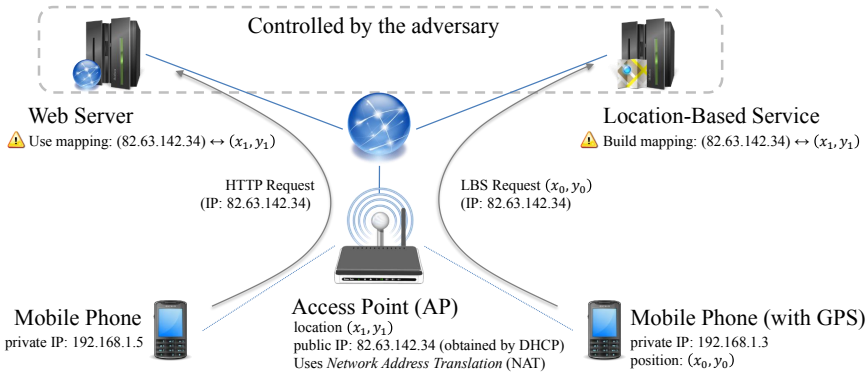


Figure 1: System and threat model. Devices connect through a NAT Access Point and share a single public IP address (the AP's IP). A user making an LBS request reveals her location (thus the approximate location of the AP)) to the adversary who can then build the (IP, Location) mapping. When another user connects to a different server (controlled by the adversary) the adversary can use the (IP, Location) mapping to locate the user because she connects with the same IP.

1. Geolocation requests: `Geo-Req(`MACs`)`, where MACs refer to the MAC addresses of the APs and cell towers in the range of the device;
2. LBS requests: `LBS-Req(`(x_0, y_0)`)`, where (x_0, y_0) denotes the estimated coordinates of the device[1] (assumed to be close to the AP's location (x_1, y_1)) obtained by the user's device using one of the techniques described in Section 2;
3. Authenticated standard (i.e., that are neither LBS nor Geolocation) requests: `Auth-Req(`tok`)`, where tok represents any information that allows for user authentication (e.g., cookie or username);
4. Unauthenticated standard requests: `Req()`.

Requests of the first two types contain an estimate of the AP's coordinates, thus they both allow the server to build the (IP, (x_1, y_1)) mapping. Consequently, there is no need to distinguish between these two types of requests, and we simply refer to both as LBS requests. For all four types of connections, the server knows the source IP addresses, specifically the AP's public IP address.

3.2 Adversary and Threat Models

We consider an adversary whose goal is to learn users' locations, for instance, to make a profit by providing geo-targeted mobile ads and recommendations (e.g., a private company) or to track users (e.g., an authoritarian regime). The adversary has access to the information collected by a number of servers that provide online services described above. Governments can eavesdrop on individuals' communications to obtain such information. As for private companies, Google, for instance, provides Web searches (Google), e-mail (GMail), social networking (Google+), and geolocation and location-based services (Google Maps). As such, it receives requests of the four types and consolidates all the information obtained [11]. The extent to which these services are used is exacerbated by their deep integration in the widely spread Android operating system. In addition, Google has an advertising network and thus has a strong incentive to obtain and monetize information about users' locations. Microsoft (with Bing, Hotmail, Bing Maps, and Windows

[1]We assume that all LBS requests concern users' actual locations, or that the server has means to distinguish between such LBS requests and other LBS requests.

Phones) and Apple (with iCloud and iPhone) are other relevant potential candidates for the considered adversary. In this paper, we focus on the case where the adversary has access to all the four types of requests. The adversary is assumed to be *honest-but-curious*, meaning that he passively collects information but does not deviate from the specified protocol (e.g., purposely returning inaccurate answers to LBS queries).

Given such an adversarial model, we consider the threat of the adversary who knows the location of a user without it being explicitly disclosed: The threat comes from the fact that the adversary can build mappings between the AP's IP addresses of the APs and their geographic coordinates based on LBS requests he receives from other users connected to the APs. And, because all requests (coming from devices connected through the AP) share the same public IP, the adversary can subsequently infer the location of the other users. More specifically, considering the example depicted in Fig. 1, when the LBS provider's server (assumed to be controlled by the adversary) receives an LBS request for position (x_1, y_1), which is the actual position of the user (herself located close to the AP) determined by her GPS-equipped mobile phone, it can map the AP's public IP address (i.e., `82.63.142.34`) to the approximated AP's location (i.e., (x_1, y_1)). Later, when another user, connected through the AP, makes a request to a server (also controlled by the adversary), then the server can exploit the obtained mapping and infer from the source IP (i.e., the AP's public IP again: `82.63.142.34`) that the second user is also at the same location (i.e., (x_1, y_1)). The adversary can subsequently provide geo-targeted mobile ads. If the adversary is interested in tracking users, he can locate any user who makes an authenticated request before the IP changes.

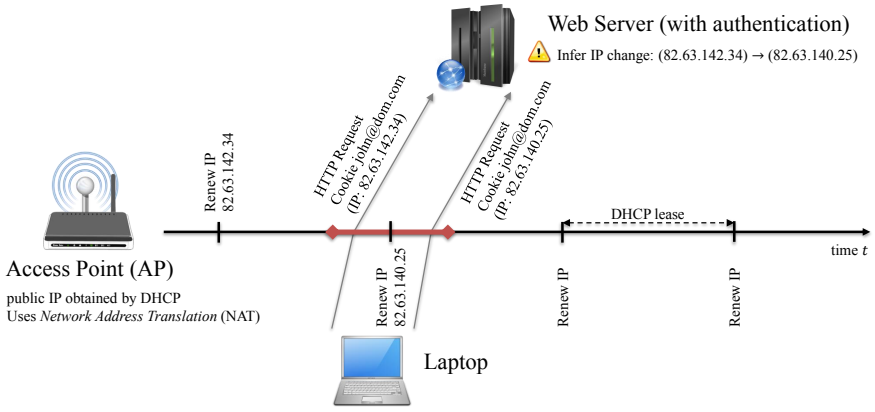


Figure 2: AP's IP address renewal and updating of the (IP, Location) mapping. A user generates an authenticated request (with a unique cookie) during a DHCP lease interval in which the adversary has obtained the (IP, Location) mapping, shortly before the DHCP lease expires and the AP is assigned a new IP. Shortly after the IP change, the same user generates another authenticated request (with the same cookie) from the new IP. As both requests occurred in a short time interval, the adversary can infer that the AP's IP changed from `82.63.142.34` to `82.63.140.25` and update the mapping.

We assume that the IP addresses in the DHCP pool can be assigned to clients at very distant locations (e.g., a nation-wide ISP that assigns IP addresses among the whole set of its clients scattered all over the country). Consequently, the fact that the AP's public address is dynamic limits in time the extent of the threat: If the AP is assigned a new address by the ISP (with DHCP), the mapping built by the adversary becomes invalid, unless the adversary is able to infer the IP address change. The inference can be based on authenticated requests as depicted in Fig. 2: A request, authenticated by cookie `john@dom.com` and originating from IP address `82.63.142.34`, is shortly followed by another request authenticated by the same cookie `john@dom.com` but originating from a different source IP address (i.e., `82.63.140.25`). There are two options: either the AP's IP is changed or the user has moved and is now connected from a different AP. If the inference time interval (delimited with diamonds in Fig. 2) around the IP renewal time is

short enough, then the adversary can infer, with high confidence, that the IP has changed and he knows its new value.

In summary, the problem we study is as follows. Considering a single AP, time is divided into intervals corresponding to DHCP leases, during which the AP's public IP address remains the same. At a certain point in time, the adversary knows the location of the AP associated to the IP because (i) a user made an LBS request earlier in the time interval or (ii) the adversary knew the location corresponding to the public IP address from the previous interval **and** a user made an authenticated request shortly before and after the public IP address was renewed. The location-privacy threat is to be evaluated with respect to the number of users whose location is known by the adversary. In the first case (geo-targeted mobile ads), the adversary needs to know the location of the user *when* the user makes a requests: the victims are therefore the users who make a standard request *after* the adversary learns the (IP, Location) mapping (during the same DHCP lease). In the second case (tracking), the adversary can maintain a log of the users who connected during a DHCP lease and locate them *a posteriori* if he learns the (IP, Location) mapping at some point during the same DHCP lease: the victims are the users who make an authenticated request *during* a DHCP lease in which the adversary learns the (IP, Location) mapping. Due to space limitations, we will evaluate the threat only with respect to an adversary who aims to exploit *current* location information through geo-targeted ads. The case of tracking can be studied by following the same line of reasoning.

4 Formalization and Analysis

In this section, we model the aforementioned setting and quantify theoretically the location-privacy threat. The notations are summarized in Table 1 (p. 15).

4.1 Model

We consider an access point *AP*, a passive adversary $\mathcal{A}$, and a set of users who connect to *AP* and make requests to servers controlled by $\mathcal{A}$. We study the system over the continuous time interval $[0, +\infty)$. At each time instant t, *AP* has a single public IP obtained through DHCP. Every T time units, starting at time 0, the DHCP lease expires and *AP* is either re-assigned the same IP or allocated a new one. We model this with independent random variables drawn from a Bernoulli distribution: with probability p_{New} *AP* is assigned a new IP, and with probability $1-p_{\text{New}}$ it is re-assigned the same IP. We divide time into successive sub-intervals I_k, $k \geq 0$, of duration T, corresponding to the DHCP leases: $I_k = [kT, (k+1)T]$. Each sub-interval is aligned with a DHCP lease. Therefore, within each sub-interval *AP*'s public IP address remains unchanged. For any time instant t, we denote by $\bar{t}$, the relative time within the corresponding sub-interval, that is $\bar{t} = t \mod T$.

Users connect to *AP*, remain connected for a certain time and then disconnect. While connected, users make requests, each of which is of one of the following types: LBS, authenticated, or standard. We model users who arrive and connect to *AP* with a homogeneous Poisson process[2] with intensity λ_{Arr}, which implies that the number $N_{\text{Arr}}(t)$ of users who arrive and connect to *AP* during any time interval of length t follows a Poisson distribution with parameter $\lambda_{\text{Arr}}t$:

$$\mathbf{P}\left[N_{\text{Arr}}(t) = n\right] = \frac{(\lambda_{\text{Arr}}t)^n}{n!}e^{-\lambda_{\text{Arr}}t}\ ,\ n \geq 0.$$

We denote the time users stay connected to the *AP* by T_{Dur}, which follows an exponential distribution with average $\frac{1}{\lambda_{\text{Dur}}}$. This means that the associated cumulative distribution function (cdf) and probability density

[2] All modeling choices in this section follow well-established conventions [19] and are backed up by several public Wi-Fi hotspot workload analysis (e.g., [9]).

function (pdf) are

$$f_{\mathrm{Dur}}(t) = \lambda_{\mathrm{Dur}} e^{-\lambda_{\mathrm{Dur}} t} \quad \text{and} \quad F_{\mathrm{Dur}}(t) = \mathbf{P}\left[T_{\mathrm{Dur}} < t\right] = 1 - e^{-\lambda_{\mathrm{Dur}} t}\ .$$

A noteworthy property of exponential distributions is *memory-lessness*: the probability distribution of the time spent *since* a given time instant t, provided that the user is still connected at time t, is the same for all t. In other words, $\mathbf{P}\left[T_{\mathrm{Dur}} > \delta t\right] = \mathbf{P}\left[T_{\mathrm{Dur}} > t + \delta t \mid T_{\mathrm{Dur}} > t\right]$.

We assume the system to be stationary with respect to user connections and disconnections. Based on Little's law [19], the average number of connected users at any time instant t is therefore constant and given by: $N_{\mathrm{Con}} = \lambda_{\mathrm{Arr}}/\lambda_{\mathrm{Dur}}$.

Users generate requests independently of each other. For each user, the three types of requests she makes are also independent: Standard and authenticated requests are modeled by independent homogeneous Poisson processes with intensity λ_{Std} and λ_{Auth}, respectively. In particular, the probability that at least one request of a type is made during an interval of length t is

$$P_{\mathrm{Std}}(t) = 1 - e^{\lambda_{\mathrm{Std}} t} \quad \text{and} \quad P_{\mathrm{Auth}}(t) = 1 - e^{\lambda_{\mathrm{Auth}} t}\ .$$

Another noteworthy property of Poisson processes is that the number of requests in two disjoint intervals are independent. We assume that each user makes a standard request when she connects to *AP*. For an LBS, we assume that only a proportion α_{LBS} of the users make LBS requests, and we model such requests by independent homogeneous Poisson processes with intensity λ_{LBS} for each user.

Fig. 3 depicts the user arrivals, departures, standard and LBS request processes and illustrates the key notations and concepts introduced in this section.

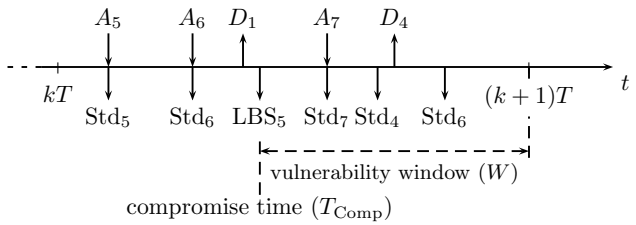


Figure 3: Threat caused by a user making an LBS request. A_i and D_i represent User i's arrival and departure, respectively. The time at which the first LBS request is made (LBS_5) is called the *compromise time* (T_{Comp}). From time T_{Comp} on, any user who makes a standard request is a victim. Users already connected at T_{Comp} are victims if they make a standard request after T_{Comp} and before leaving and before the end of the sub-interval, e.g., User 4. Users who connect after T_{Comp} are, *de facto*, victims as users make a standard request when they connect, e.g., User 7.

4.2 Threat

We first focus on a single sub-interval and quantify the location-privacy threat, with respect to the number of users whose location is known by the adversary because of others. Specifically, we call a *victim* a user who makes a standard request at a time at which the adversary knows the mapping (IP, Location).

Quantifying the threat in a sub-interval If at least one user connected to *AP* uses an LBS at some time instant (thus revealing her current location), $\mathcal{A}$ obtains the (IP, Location) mapping based on which it can locate other users.

We define the *compromise time* T_{Comp} as the first time within the sub-interval, when a user connected to *AP* uses an LBS, if such an event occurs, and T otherwise. At any time, there are on average N_{Con} users

connected to *AP*, out of which $\alpha_{\mathrm{LBS}} N_{\mathrm{Con}}$ potentially make LBS queries. The aggregated process of LBS requests is a Poisson process with intensity $\Lambda_{\mathrm{LBS}} = \alpha_{\mathrm{LBS}} N_{\mathrm{Con}} \lambda_{\mathrm{LBS}}$. Therefore, the probability that at least one LBS request (from the aggregated process) is made before time $\bar{t}$ in the sub-interval is

$$F_{\mathrm{Comp}}(\bar{t}) = \mathbf{P}\left[T_{\mathrm{Comp}} < \bar{t}\right] = 1 - e^{-\Lambda_{\mathrm{LBS}}\bar{t}} \quad \text{for } \bar{t} < T\ ,$$

and, in this case, the expected compromise time is $\frac{1}{\Lambda_{\mathrm{LBS}}}(1 - e^{-\Lambda_{\mathrm{LBS}}T})$. We call f_{Comp} the corresponding probability density function. The time interval that spans from the compromise time to the end of the sub-interval is called the *vulnerability window* (see Fig. 3) and the expected value W of its length is

$$\mathbf{E}\left[W\right] = T - \frac{1 - e^{-\Lambda_{\mathrm{LBS}}T}}{\Lambda_{\mathrm{LBS}}}. \tag{1}$$

Fig. 4 depicts the cumulative distribution function of the compromise time and its average value in an example setting. We observe that even with moderate AP popularity and LBS usage, the adversary obtains the mapping before the DHCP lease expires in 83% of the cases and he does so after 11 hours on average.

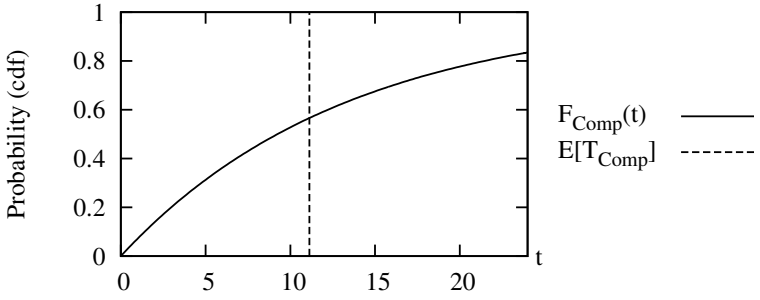


Figure 4: Cumulative distribution function of the compromise time T_{Comp} (expressed in hours). The parameters were set to $\lambda_{\mathrm{Arr}} = 5$ users/h, $\lambda_{\mathrm{Dur}} = 1/1.5$ (i.e., average connection time of one hour and a half), $\lambda_{\mathrm{LBS}} = 0.05$ req./h, and $\alpha_{\mathrm{LBS}} = 0.2$.

In order to compute the number of victims we distinguish between two groups of users: those who were connected when the first LBS request was made, e.g., User 6 in Fig. 3, and those who subsequently connected to *AP* during the vulnerability window (and are, *de facto*, victims as they make a standard request when they connect), e.g., User 7. We call V_1 and V_2 the number of victims in each group.

There are N_{Con} users connected at the compromise time (recall that there are on average N_{Con} users in the system at any time). Whenever we compute an expected value involving the number of users connected, Wald's equation [19] allows us to consider that the system is composed of exactly N_{Con} users. Provided that an LBS request is made within the sub-interval, the number of victims at that time is the number V_1 of connected users who make a subsequent standard request before leaving and before the end of the sub-interval. We compute the expected value of V_1 by applying the law of total probability, conditioning over both the compromise time and the time spent in the system:

$$\begin{aligned}
\mathbf{E}\left[V_1\right] &= N_{\mathrm{Con}} \int_{t=0}^{T} \int_{u=0}^{\infty} f_{\mathrm{Comp}}(t) f_{\mathrm{Dur}}(u) P_{\mathrm{Std}}(\min\left(u, T-t\right)) du\, dt \\
&= N_{\mathrm{Con}} \frac{\Lambda_{\mathrm{LBS}} \lambda_{\mathrm{Std}}}{(\lambda_{\mathrm{Std}} + \lambda_{\mathrm{Dur}}) - \Lambda_{\mathrm{LBS}}} \left[\frac{1 - e^{-T\Lambda_{\mathrm{LBS}}}}{\Lambda_{\mathrm{LBS}}} - \frac{1 - e^{-T(\lambda_{\mathrm{Std}} + \lambda_{\mathrm{Dur}})}}{(\lambda_{\mathrm{Std}} + \lambda_{\mathrm{Dur}})} \right]
\end{aligned} \tag{2}$$

We compute, on average, the number V_2 of users who connect to *AP* between the compromise time and the end of the sub-interval by applying the law of total probability, conditioning over the length of the vulnerability windows:

$$\mathbf{E}\left[V_2\right] = \mathbf{E}\left[N_{\mathrm{Arr}}(W)\right] = \lambda_{\mathrm{Arr}} \cdot \mathbf{E}\left[W\right] = \lambda_{\mathrm{Arr}} \left(T - \frac{1 - e^{-\Lambda_{\mathrm{LBS}} T}}{\Lambda_{\mathrm{LBS}}} \right) . \tag{3}$$

The average number of victims in a sub-interval is the expectation of the sum of the number of victims connected at the compromised time (V_1) and victims arriving within the vulnerability window (V_2). Naturally, this number has to be compared to the average number of users who have been connected at some point within the sub-interval: $V_{\mathrm{total}} = N_{\mathrm{Con}} + \lambda_{\mathrm{Arr}} T$. It can be observed in Fig. 5 that the proportion of victims $(\mathbf{E}\left[V_1\right] + \mathbf{E}\left[V_2\right])/V_{\mathrm{total}}$ increases with T. This is because all users who connect during the vulnerability window are victims. As the probability of the adversary obtaining the mapping before time T increases with T, V_1/V_{total} first increases. However, because V_1 is upper-bounded by N_{Con} and V_{total} increases with T, V_1/V_{total} eventually tends to 0. In the end, when the DHCP lease expires the location of more than half of the users is compromised.

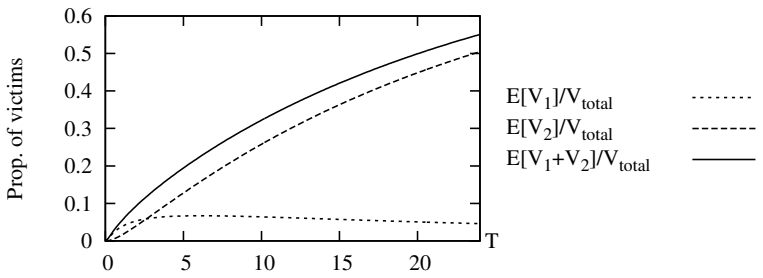
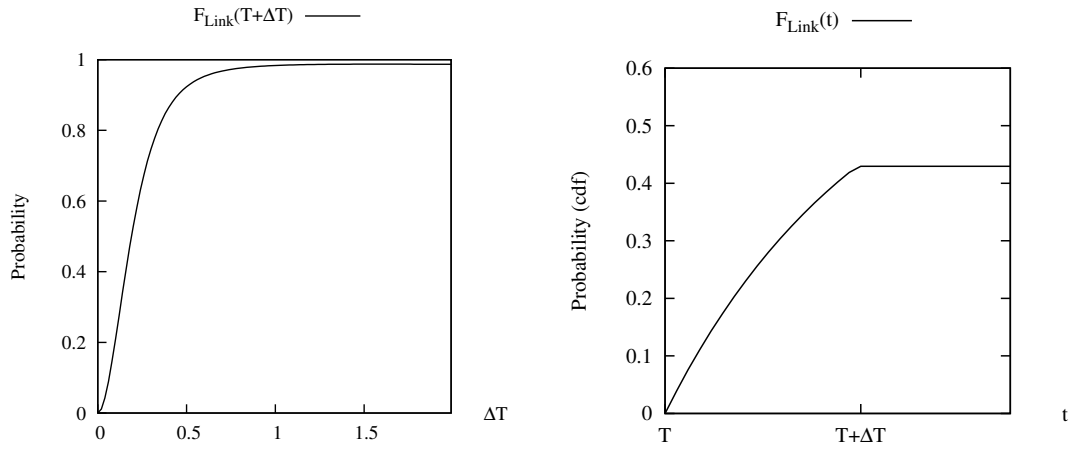


Figure 5: Proportion of victims within a sub-interval of length T, corresponding to a DHCP lease. The parameters were set to: $\lambda_{\mathrm{Arr}} = 5$ users/h, $\lambda_{\mathrm{Dur}} = 1$ (i.e., average connection time of one hour), $\lambda_{\mathrm{Std}} = 10$ req./h, $\lambda_{\mathrm{LBS}} = 0.05$ req./h, and $\alpha_{\mathrm{LBS}} = 0.2$. The dotted curve (resp. dashed) corresponds to the victims connected at (resp. arriving after) the compromise time. The solid curve represents the total proportion of victims.

Inferring IP change We consider two successive sub-intervals, without loss of generality I_0 and I_1, and we look at the probability F_{Link} that the adversary infers the IP change from authenticated requests. This situation occurs if at least one user makes both an authenticated request at most ΔT time units ($\Delta T < T/2$) before the IP change and another authenticated request at most ΔT time units after the IP change. Proceeding similarly as above, we can compute the probability of inferring the IP change by distinguishing between two groups of users: those who were connected at time $T - \Delta T$ and those who connected within $[T - \Delta T, T]$.

The linking probability can be thought of as depending both on t and ΔT. Fig. 6a depicts the linking probability at time $T + \Delta T$ as a function of ΔT. It can be observed that this probability rapidly converges to 1. Note that the fact that linking probability increases with ΔT is balanced by the decreased confidence of the adversary. This is because the probability that a user makes two authenticated requests from two distinct access points in the time interval $[T - \Delta T, T + \Delta T]$ (moving from one to the other) increases with ΔT. Fig. 6b depicts the linking probability as a function of t. It remains constant for $t \geq T + \Delta T$ because only authenticated requests made in the time interval $[T - \Delta T, T + \Delta T]$ are taken into account to infer the IP change. Note that with a value of ΔT as small as 5 minutes, which provides high confidence, the adversary can still infer the IP change with a probability of 43%.



(a) As a function of ΔT. **(b)** As a function of t.

Figure 6: Probability of inferring the IP change, i.e., F_{Link}. The parameters were set to: $\lambda_{\text{Arr}} = 5$ users/h, $\lambda_{\text{Dur}} = 1/1.5$, $\lambda_{\text{Std}} = 10$ req./h, $\lambda_{\text{LBS}} = 0.05$ req./h, $\lambda_{\text{Auth}} = 2$ req./h, $\Delta T = 5$ minutes, and $\alpha_{\text{LBS}} = 0.2$.

Quantifying the threat over multiple sub-intervals When the adversary infers the IP changes, the probability $F^{(k)}_{\text{Map}}(t)$ that the adversary knows the (IP, Location) mapping at time $t \in I_k$, $k \geq 1$ is

$$F^{(k)}_{\text{Map}}(\bar{t}) = F_{\text{Comp}}(\bar{t}) + (1 - F_{\text{Comp}}(\bar{t}))F^{(k-1)}_{\text{Map}}(T)\left((1 - p_{\text{New}}) + p_{\text{New}}F_{\text{Link}}(\bar{t})\right) \tag{4}$$

with initial condition $F^{(0)}_{\text{Map}}(\bar{t}) = F_{\text{Comp}}(\bar{t})$. Note that the assumption $\Delta T < T/2$ is required here. Indeed, this technical restriction ensures that the time interval $[kT - \Delta T, kT + \Delta T]$ (used by the adversary for the linking), does not overlap with the time interval $[(k-1)T - \Delta T, (k-1)T + \Delta T]$. Essentially, this makes the two intervals disjoint and therefore also independent with respect to authenticated requests, which allows us to multiply the corresponding probabilities. From Eq. (4), it can be seen that $F^{(k)}_{\text{Map}}(T)$ obeys the following recursive equation:

$$F^{(k)}_{\text{Map}}(T) = a + bF^{(k-1)}_{\text{Map}}(T)$$

where $a = F_{\text{Comp}}(T)$ and $b = (1 - F_{\text{Comp}}(T))\left((1 - p_{\text{New}}) + p_{\text{New}}F_{\text{Link}}(T)\right)$. This recursive equation can easily be seen to have as a solution $a(1 - b^{k+1})/(1 - b)$. As $b < 1$, $F^{(k)}_{\text{Map}}(T)$ converges to a finite value, i.e., $a/(1 - b)$.

The number of victims in the sub-interval I_k can be computed by replacing the density f_{Comp} in Eqs. (2) and (3) with the density of $F^{(k)}_{\text{Map}}$. The probability that the adversary has the mapping (IP, Location) at time t in sub-interval I_k, i.e., $F^{(k)}_{\text{Map}}$ is illustrated in Fig. 7. It can be observed in the figure that the mapping probability increases over time and, after the convergence, the adversary successfully obtains the mapping before the DHCP lease expires in 79% of the cases and before the half-lease in 60% of the cases.

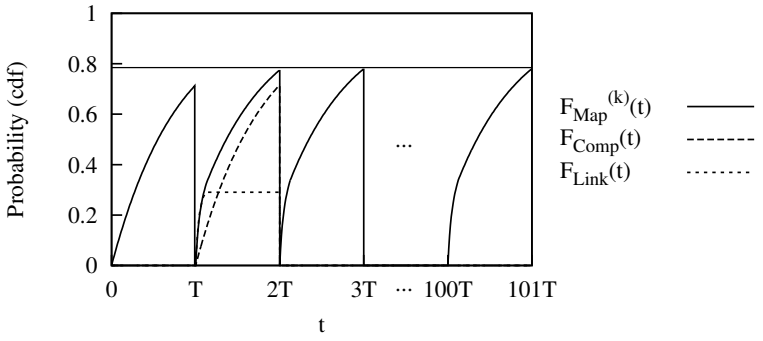


Figure 7: Probability of obtaining the (IP, Location) mapping over several sub-intervals. The solid curve represents the probability of obtaining the mapping before time t. The dashed curve represents the probability of obtaining the mapping from an LBS request. The dotted curve represents the probability of inferring the IP change. The parameters were set to $\lambda_{\mathrm{Arr}} = 5$ users/h, $\lambda_{\mathrm{Dur}} = 1/1.5$, $\lambda_{\mathrm{Std}} = 10$ req./h, $\lambda_{\mathrm{LBS}} = 0.035$ req./h, $\lambda_{\mathrm{Auth}} = 0.2$ req./h, $T = 24$ hours, $\Delta T = 3$ hours, $\alpha_{\mathrm{LBS}} = 0.1$, and $p_{\mathrm{New}} = 1$. Note: to highlight the respective contributions of the linking and compromise probabilities, some values differ from our previous setting (e.g., ΔT). In the first sub-interval, the linking probability is zero and the probability of having the mapping is the compromise probability. In subsequent sub-intervals, this probability $F^{(k)}_{\mathrm{Map}}(t)$ increases due to the potential inference of IP changes: it becomes a combination of $F_{\mathrm{Link}}(\bar{t})$ and $F_{\mathrm{Comp}}(\bar{t})$ (and the probability of having the mapping by the end of the preceding sub-interval).

5 Experimental Results

In this section, we complement our theoretical analysis with simulations based on traces from a network of Wi-Fi access points deployed on the EPFL campus.

Simulator We use a discrete event simulator which processes events according to a schedule. Simulation events are defined by their occurrence time and type, which is one of the following: arrival, departure, LBS request, standard request, authenticated request, and DHCP renewal. Arrivals and departures are taken from real traces, whereas other events are simulated according to the aforementioned model. We average the results of multiple simulations.

Dataset Our dataset consists of daily user session traces for the period of Oct. 2011 – Feb. 2012 from about 3,000 Wi-Fi APs deployed across the EPFL campus. Each log-entry contains the AP's unique ID, the anoymized MAC address of the device and the connection time. We select an AP used by on average 20 users simultaneously, a setting that corresponds to a bookstore or a hotel [9].

We compare the simulation results (averaged over a period of 50 days) against the results from our theoretical analysis and show the results in Fig. 8. Because our theoretical model assumes a homogeneous user arrival rates, we compute the expected proportion of victims and compromise time as if the arrival process started at 6:AM, the time at which a significant number of users start connecting to the AP in our traces. It can be observed that although the model does not capture the time-of-the-day effects of the user arrival process, both the theoretical and simulated expected proportions of victims matches when considering the entire period.

We observe that after 11:AM, only 5 hours after users start connecting to the AP, users' location privacy is compromised. By the end of the day, 80% or the users who connected through the AP were compromised. This moderate example shows the immense threat users are exposed to, as it is expected that for more popular hotspots, e.g., airports, the compromise time will be even shorter.

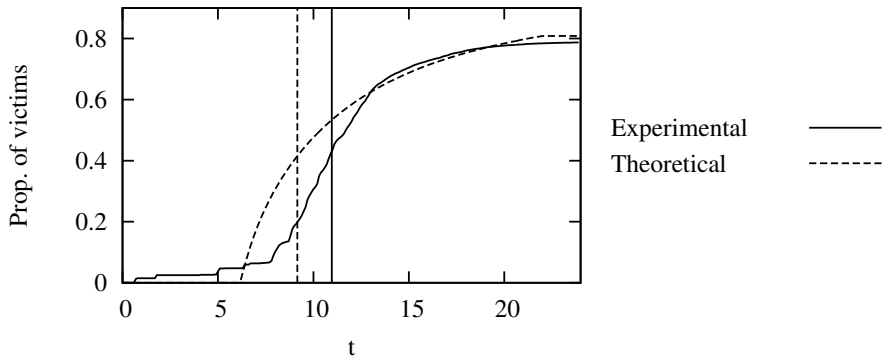


Figure 8: Experimental results. The parameters were set to: $\lambda_{\mathrm{Arr}} = 23.6$ users/h, $\lambda_{\mathrm{Dur}} = 1/0.9$ (users stay connected for 54 minutes on average), $\lambda_{\mathrm{Std}} = 10$ req./h, $\lambda_{\mathrm{LBS}} = 0.05$ req./h, and $\alpha_{\mathrm{LBS}} = 0.2$. The solid curve represents the experimental proportion of victims, whereas the dashed curve represents its theoretical counterpart. The vertical lines represent the respective average compromise times.

6 Countermeasures and Discussion

In this section, we discuss possible countermeasures and business opportunities.

Countermeasures Cryptographic primitives are efficient at protecting users' privacy, but because of the way networking protocols operate, they may not be sufficient, in particular, when the private information is the source IP address.

Hiding users' actual source IP from the destination (i.e., the adversary) naturally comes to mind as a straightforward countermeasure against the considered threat. This can be done in several ways. In relay-based anonymous communications, a user's traffic is forwarded from the source to the destination by several relay nodes, in such a way that the destination cannot know the user's source IP. Examples of such networks include Tor [6], mix networks [2, 5], or simple HTTP proxies [21]. With Virtual Private Networks (VPNs) [10], the user is assigned an IP address that belongs to a remote network (e.g., a corporate network [3] or commercial/public VPN [24, 20]). To the adversary, the user's requests appear to originate from within the remote network, whose location is different from that of the user. Unfortunately, most users are not aware of such techniques [26].

Alternatively, these countermeasures can be implemented by ISPs, for instance, by deploying a country-wide NAT that aggregates traffic from all hosts connected to the ISP at several gateways (e.g., Telefonica [25]) or by IP Mixing [18]. This also applies to operators of AP networks (e.g., Starbucks, AT&T Wi-Fi). However, they may not have incentives to implement such solutions.

A second approach to thwart the threat consists in decreasing the knowledge of the adversary, by reducing the accuracy of the reported location and by increasing the uncertainty about the AP's location. Examples of location PETs reducing the adversary's accuracy include spatial cloaking [12] and adding noise to reported locations [1]. To increase the adversary's uncertainty, [15] proposes to inject "dummy" requests, i.e., not related to the user's location. It is not easy for users to implement these PETs, because some geolocation requests are implemented in the operating system, which is often controlled by the adversary (e.g., Google Android). Moreover, when these techniques are implemented in a non-coordinated fashion, the adversary might still be able to infer the actual location by filtering out requests that stand out from the bulk (increasing its certainty) and averaging the remaining requests (increasing its accuracy). Better results can be achieved by having the AP operators implement the location-privacy preserving mechanisms, but they might lack incentives to do so.

Finally, as highlighted by our analysis, various other countermeasures can be implemented by the ISP

or the AP's owner: reduce the DHCP lease, always allocate a new IPs, trigger the IP change when the traffic is low or purposely impose silent periods around the renewal time (reducing the chances that the adversary infers the IP change from authenticated requests). Unfortunately, all these have a negative effect on the quality of service and impose a significant overhead in network management, and are thus unlikely to be deployed in practice. Besides technical countermeasures, one may also envision a "Do-not-geolocalize" initiative, similar to "Do-not-track" [8], allowing users to opt-out of being localized.

The evolution of the threat with IPv6 With IPv6, each host has a public IP, composed of a *prefix* (leftmost 64 bits), shared with other hosts in the same network, and a unique *host part* (rightmost 64 bits). Sharing a prefix is analogous to sharing a public IPv4 address behind a NAT: (IPv4, Location) mappings correspond to (Prefix, Location) mappings. Because IPv6 prefixes are intended to be less dynamic than IPv4 addresses, the threat is expected to be amplified.

Business opportunities Beyond threatening the location-privacy of users, the (IP, Location) mapping technique presented in this paper can be used as a novel IP-location solution potentially improving on existing solutions [16, 27]. Online service providers, such as Google and Microsoft, are in a position to build and monetize this service. Because ISPs hold the mappings and can prevent the service providers to build the mapping (using the aforementioned countermeasure) they can make a profit either by selling IP-location to service providers (e.g., Verizon in the US [4]) or by selling the privacy-protection to users.

7 Conclusion

In this paper we present a practical threat, effectively demonstrating that the location privacy of users connecting to hotspots can be compromised by others. The scale of the threat is immense because it simply leverages on the way most networks are designed. Our theoretical and experimental analysis allows us to quantify the threat and identify the key parameters and their impact on the adversary's success. We survey possible countermeasures and we find that adequate ones can be used to protect individual users' location privacy. However, to completely thwart the threat, these countermeasures need to be widely deployed.

We intend to further study this threat by focusing on the following aspects: (i) accuracy of a novel IP-location service, based on (IP, Location) mappings; (ii) refinement of the model, for instance by modeling users arrivals with inhomogeneous Poisson process to capture time-of-the-day effects; (iii) adversary's inference about IP changes, taking into account e.g., persistent connections, fingerprinting users' connections, and the trade-off between the probability of inferring the IP change and the adversary's confidence; and (iv) adversary's ability to track users' locations, as users move and connect to different APs over time.

References

[1] R. Agrawal and R. Srikant. Privacy-Preserving Data Mining. In *SIGMOD*, 2000.

[2] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[3] Cisco VPN Client. `http://www.cisco.com/en/US/products/sw/secursw/ps2308/index.html`, 2012.

[4] CNN. Your phone company is selling your personal data. `http://money.cnn.com/2011/11/01/technology/verizon_att_sprint_tmobile_privacy`, 2011.

[5] G. Danezis, R. Dingledine, D. Hopwood, and N. Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *IEEE S&P*, pages 2–15, 2003.

[6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security*, pages 303–320, 2004.

[7] R. Droms. Dynamic Host Configuration Protocol. RFC 2131, Mar. 1997. Updated by RFCs 3396, 4361, 5494.

[8] Federal Trade Commission. Protecting Consumer Privacy in an Era of Rapid Change: A proposed framework for businesses and policymakers. Preliminary FTC Staff Report., 2010.

[9] A. Ghosh, R. Jana, V. Ramaswami, J. Rowland, and N. Shankaranarayanan. Modeling and Characterization of Large-Scale Wi-Fi Traffic in Public Hot-Spots. In *INFOCOM*, pages 2921–2929, 2011.

[10] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A Framework for IP-Based Virtual Private Networks. RFC 2764, Feb. 2000.

[11] Google Privacy Policy. `http://www.google.com/intl/en/policies/privacy/preview/`, 2012.

[12] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *MobiSys*, 2003.

[13] Targeting Local Markets: An IAB Interactive Advertising Guide. Interactive Advertising Bureau, 2010.

[14] E. Kaplan. *Understanding GPS - Principles and applications*. Artech House, 2nd edition edition, December 2005.

[15] H. Kido, Y. Yanagisawa, and T. Satoh. An Anonymous Communication Technique using Dummies for Location-Based Services. In *ICPS*, pages 88–97, 2005.

[16] I. Poese, S. Uhlig, M. A. Kaafar, B. Donnet, and B. Gueye. IP Geolocation Databases: Unreliable? *ACM SIGCOMM Computer Communication Review*, 41(2):53–56, 2011.

[17] PricewaterhouseCoopers. Internet Advertising Revenue Report, 2011.

[18] B. Raghavan, T. Kohno, A. C. Snoeren, and D. Wetherall. Enlisting ISPs to Improve Online Privacy: IP Address Mixing by Default. In *PETs*, pages 143–163, 2009.

[19] S. M. Ross. *Stochastic Processes*. Wiley, 1995.

[20] Security Kiss. `http://www.securitykiss.com/index.php?lang=en`, 2012.

[21] M. Shapiro. Structure and Encapsulation in Distributed Systems: The Proxy Principle. In *ICDCS*, pages 198–204, 1986.

[22] Skyhook Location Performance. `http://www.skyhookwireless.com/location-technology/performance.php`, 2012.

[23] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022, Jan. 2001.

[24] Strong VPN. `http://www.strongvpn.com/`, 2012.

[25] Telefonica Implements NAT for ADSL Users. `http://bandaancha.eu/articulo/7844/usuarios-adsl-movistar-compartiran-misma-ip-mediante-nat-escasear\-ipv4`, 2012.

[26] Tor Metrics Portal. `https://metrics.torproject.org`.

[27] Y. Wang, D. Burgener, M. Flores, A. Kuzmanovic, and C. Huang. Towards Street-Level Client-Independent IP Geolocation. In *NSDI*, 2011.

[28] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How Dynamic are IP Addresses? In *ACM SIGCOMM*, 2007.

Appendix

Symbol	Definition
T	DHCP lease time
p_{New}	Probability of being assigned a new IP
I_k	k-th sub-interval
$\bar{t}$	Relative time within a sub-interval
λ_{Arr}	Rate of user arrivals at AP
$N_{\mathrm{Arr}}(t)$	Number of arrivals in an interval of length t
T_{Dur}	Time users stay connected to the AP
$1/\lambda_{\mathrm{Dur}}$	Average time users stay connected to the AP
$F_{\mathrm{Dur}}, f_{\mathrm{Dur}}$	Pdf/cdf of T_{Dur}
N_{Con}	Average number of users connected to the AP
$\lambda_{\mathrm{Std}}, \lambda_{\mathrm{Auth}}$	Rates of user standard/authenticated requests
$P_{\mathrm{Std}}(t), P_{\mathrm{Auth}}(t)$	Probability that a user makes at least one request during an interval of length t
α_{LBS}	Proportion of users who make LBS requests
λ_{LBS}	Rate of user LBS requests
Λ_{LBS}	Aggregated rate of users' LBS requests
T_{Comp}	First time an LBS request occurs in a sub-interval
$F_{\mathrm{Comp}}, f_{\mathrm{Comp}}$	Pdf/cdf of T_{Comp}
W	Length of the vulnerability window
ΔT	Time interval used to infer IP changes
$F_{\mathrm{Link}}(t)$	Probability of inferring IP change before time t
$F^{(k)}_{\mathrm{Map}}(t)$	Probability of having the mapping before time $t \in I_k$

Table 1: Table of symbols.

Exploring Linkability of User Reviews

Mishari Almishari and Gene Tsudik

Computer Science Department, University of California, Irvine
{malmisha,gts}@ics.uci.edu

Abstract

Large numbers of people all over the world read and contribute to various review sites. Many contributors are understandably concerned about privacy in general and, specifically, about linkability of their reviews (and accounts) across multiple review sites. In this paper, we study linkability of community-based reviewing and try to answer the question: *to what extent are "anonymous" reviews linkable, i.e., highly likely authored by the same contributor?* Based on a very large set of reviews from one very popular site (Yelp), we show that a high percentage of ostensibly anonymous reviews can be accurately linked to their authors. This is despite the fact that we use very simple models and equally simple features set. Our study suggests that contributors reliably expose their identities in reviews. This has important implications for cross-referencing accounts between different review sites. Also, techniques used in our study could be adopted by review sites to give contributors feedback about linkability of their reviews.

1 Introduction

In recent years, popularity of various types of review and community-knowledge sites has substantially increased. Prominent examples include Yelp, Tripadvisor, Epinions, Wikipedia, Expedia and Netflix. They attract multitudes of readers and contributors. While the former usually greatly outnumber the latter, contributors can still number in hundreds of thousands for large sites, such as Yelp or Wikipedia. For example, Yelp had more than 39 million visitors and reached 15 million reviews in late 2010 [1]. To motivate contributors to provide more (and more useful/informative) reviews, certain sites even offer rewards [2].

Some review sites are generic (e.g., Epinions) while others are domain-oriented, e.g., Tripadvisor. Large-scale reviewing is not limited to review-oriented sites; in fact, many retail sites encourage customers to review their products. e.g., Amazon and Netflix.

With the surge in popularity of community-based reviewing, more and more people contribute to review sites. At the same time, there has been an increased awareness with regard to personal privacy. Internet and Web privacy is a broad notion with numerous aspects, many of which have been explored by the research community. However, privacy in the context of review sites has not been adequately studied. Although there has been a lot of recent research related to reviewing, its focus has been mainly on extracting and summarizing opinions from reviews [6, 8, 18] as well as determining authenticity of reviews [10, 11, 13].

In the context of community-based reviewing, contributor privacy has several aspects: (1) some review sites do not require accounts (i.e., allow ad hoc reviews) and contributors might be concerned about linkability of their reviews, and (2) many active contributors have accounts on multiple review sites and prefer these accounts not be linkable. The flip side of the privacy problem is faced by review sites themselves: how to address spam-reviews and sybil-accounts?

The goal of this paper is to explore and measure linkability of reviews by investigating how close and related are a person's reviews. That is, how accurately we can link a set of anonymous reviews to their original author. Our study is based on over $1,000,000$ reviews and $\simeq$ $2,000$ contributors from Yelp. This paper makes the following contributions:

1. We provide a privacy measurement study where we extensively assess and measure reviews' linkability and show that anonymous reviews are accurately de-anonymized in the presence of very simple features. For example, using only alphabetical letter distributions, we can link up to 83% (and 96% with few additional features) of the anonymous reviews to their real authors. We believe that the findings in this study are very important and alarming for reviewers who are concerned about their privacy.

2. We propose several models and improvements that quite accurately link "anonymous" reviews.

Our results have several implications. One of them is the ability to cross-reference contributor accounts between multiple (and similar) review sites. If a person regularly contributes to two similar review sites under different accounts, anyone can easily link them, since many people tend to consistently maintain their traits in writing reviews. This is possibly quite detrimental to personal privacy. Another implication is the ability to correlate reviews ostensibly emanating from different accounts that are produced by the same author. Our approach can thus be very useful in detecting self-reviewing and, more generally, review spam [10] whereby one person contributes from multiple accounts to artificially promote or criticize products or services.

One envisaged application of our technique is to have it integrated into review site software. This way, review authors could obtain feedback indicating the degree of linkability of their reviews. It would then be up to each author to adjust (or not) the writing style and other characteristics.

Organization: Section 2 provides background information about techniques used in our experiments. The sample dataset and study settings are addressed in Section 3. Next, our analysis methodology is presented in Section 4. Section 5 discusses issues stemming from this work. Then, Section 6 overviews related work and Section7 concludes the paper.

2 Background

This section provides some background about statistical tools used in our study. We use two well-known approaches based on: (1) Naïve Bayes Model [12], (2) Kullback-Leibler Divergence Metric [5]. We briefly describe them below.

2.1 Naïve Bayes Model

Naïve Bayes Model (NB) is a probabilistic model based on the eponymous assumption stating that all features/tokens are conditionally independent given the class. Given tokens: $T_1, T_2, ..., T_n$ in document D, we compute conditional probability of a document class C as follows:

$$P(C|D) = P(C|T_1, T_2, ..., T_n) = \frac{P(T_1, T_2, ..., T_n|C)P(C)}{P(T_1, T_2, ..., T_n)}$$
$$= \frac{P(T_1|C)P(T_2|C).....P(T_n|C)P(C)}{P(T_1, T_2, ..., T_n)}$$

Using the Naïve Bayes assumption,

$$P(T_1, T_2, ..., T_n|C) = P(T_1|C)P(T_2|C).....P(T_n|C)$$

To use NB for classification, we return the class value with maximum probability:

$$Class = argmax_C P(C|D) = argmax_C P(C|T_1, T_2, ..., T_n) \quad (1)$$

Since $P(T_1, T_2, ..., T_n)$ is the same for all C values, and assuming $P(C)$ is the same for all class values, the above equation is reduced to:

$$Class = argmax_C P(T_1|C)P(T_2|C).....P(T_n|C)$$

Probabilities are estimated using the Maximum-Likelihood estimator [5] along with Laplace smoothing [14] as follows:

$$P(T_i|C) = \frac{Num\ of\ T_i\ in\ D\ +\ 1}{Num\ of\ Tokens\ in\ D\ +\ Num\ Possible\ Token\ Values}$$

2.2 Kullback-Leibler Divergence Metric

Kullback-Leibler Divergence (KLD) metric measures the distance between two distributions. For any two distributions P and Q, it is defined as:

$$D_{kl}(P\|Q) = \sum_i P(i) log(\frac{P(i)}{Q(i)})$$

KLD is always positive: the closer to zero, the closer Q is to P. It is an asymmetrical metric, i.e., $D_{kl}(P\|Q) \neq D_{kl}(Q\|P)$. To transform it into a symmetrical metric, we use the following formula (that has been used in [20]):

$$SymD_{kl}(P,Q) = 0.5 \times (D_{kl}(P\|Q) + D_{kl}(Q\|P)) \quad (2)$$

Basically, $SymD_{kl}$ is a symmetrical version of D_{kl} that measures the distance between two distributions. As discussed below, it is used heavily in our study. In the rest of the paper, the term "KLD" stands for $SymD_{kl}$[1].

3 Data Set and Study Settings

Data Set. Clearly, a very large set of reviews authored by a large number of contributors is necessary in order to perform a meaningful study. To this end, we collected $1,076,850$ reviews for $1,997$ contributors from `yelp.com`, a very popular site with many prolific contributors. The minimum number of reviews per contributor is 330, the maximum – $3,387$ and the average – 539 reviews, with a standard deviation of 354. For the purpose of this study, we limited authorship to prolific contributors, since this provides more useful information for the purpose of review linkage. Note that 50% of the contributors authored fewer than 500 reviews and 76% authored fewer than 600. Only 6% of the contributors exceed $1,000$ reviews. Additionally, 50% of the contributors write reviews shorter than 140 words (on average) and 75% – have average review size smaller than 185. Also, 97% of contributors write reviews shorter than 300 words. The overall average review size is relatively small – 149 words.

Study Settings. Our central goal is to study linkability of relatively prolific reviewers. Specifically, we want to understand – for a given prolific author – to what extent some of his/her reviews relate to, or resemble, others. To achieve that, we first randomly order the reviews of each contributor. Then, for each contributor U with N_U reviews, we split the randomly ordered reviews into two sets:

1. First $N_U - X$ reviews: We refer to this as the **identified record** (IR) of U.
2. Last X reviews: These reviews represent the full set of anonymous reviews of U from which we derive several subsets of various sizes. We refer to each of these subset as an **anonymous record** (AR) of U. An AR of size i consists of the first i reviews of the full set of anonymous reviews of U. We vary the AR size for the purpose of studying the user reviews linkability under different numbers of anonymous reviews.

[1]Note that, under certain conditions, NB and asymmetrical KLD models could be equivalent. That is, $argmax_{Class}P(Class|T_1,T_2,...,T_n)$ is equivalent to $argmin_{Class}D_{kl}(Token\_distribution\|Class\_distribution)$, where $T_1,T_2,...T_n$ are the tokens of a document D and $Token\_distribution$ is their derived distribution. The proof for this equivalency is in [20]. However, this equivalence does not hold when we use the symmetrical version $SymD_{kl}$.

Since we want to restrict the AR size to a small portion of the complete user reviews set, we restrict X to 60 as this represents less than 20% of the minimum number of reviews for authors in our set (330 total). We use the **identified records** (IRs) of all contributors as the training set upon which we build models for linking anonymous reviews. (Note that the IR size is not the same for all contributors, while the AR size is uniform.) Thus, our problem is reduced to matching an anonymous record to its corresponding IR. Specifically, one anonymous record serves as an input to a matching/linking model and the output is a sorted list of all possible account-ids (i.e., IR sets) listed in a descending order of probability, i.e., the top-ranked account-id corresponds to the contributor whose IR represents the most probable match for the input anonymous record. Then, if the correct account-id of the actual author is among top T entries, the matching/linking model has a hit; otherwise, it is a miss. Consequently, our study boils down to exploring matching/linking models that maximize the hit ratio of the anonymous records for varying values of both T and AR sizes. We consider two values of T: 1 (perfect hit) and 10 (near-hit). Whereas, for the AR size, we experiment with a wider range of values which includes: 1, 5, 10, 20, 30, 40, 50 and 60. Note that in this paper, linkability ratio and hit ratio are used exchangeably.

Even though our focus is on the linkability of prolific users, we also attempt to assess performance of our models for non-prolific users. For that, we slightly change the problem setting by making the IR size smaller; this is discussed in Section 4.4.

4 Analysis

As mentioned in Section 2, we use Naïve Bayes (NB) and Kullback-Leibler Divergence (KLD) models. Before analyzing the collected data, we tokenize all reviews and extract four types of tokens:

1. **Unigrams:** set of all single letters. We discard all non-alphabetical characters.
2. **Digrams:** set of all consecutive letter-pairs. We discard all non-alphabetical characters.
3. **Rating:** rating associated with the review. (In Yelp, this ranges between 1 and 5).
4. **Category:** category associated with the place/service being reviewed. There are 28 categories in our dataset,

Note that we experimented our models on larger token sets, namely trigram and stemmed-word sets. Surprisingly, they mostly perform worse(in terms of linkability) than unigrams or digrams. Before proceeding, we re-cap abbreviations and notation in Table 1.

4.1 Methodology

We begin with the brief description of the methodology for the two models.

4.1.1 Naïve Bayes (NB) Model

For each account IR, we built an NB model, $P(token_i|IR)$, from its identified record. Probabilities are estimated using the Maximum-Likelihood estimator [5] and Laplace smoothing [14] as shown in 2. We then construct four models corresponding to the four aforementioned token types. That is, for each IR, we have $P_{unigram}$, P_{digram}, $P_{category}$ and P_{rating}.

To link an anonymous record AR to an account IR with respect to token type R, we first extract all R-type tokens from AR, $T_{R_1}, T_{R_2}, ...T_{R_n}$ (Where T_{R_i} is the i-th R token in AR). Then, for each IR, we compute the probability $P_R(IR|T_{R_1}, T_{R_2}, ...T_{R_n})$. Finally, we return a list of accounts sorted in decreasing order of probabilities. The top entry represents the most probable match.

NB	Naïve Bayes Model
KLD	Symmetrical Kullback-Leibler Model
R	Token Type: rating, unigram or digram
LR	Linkability Ratio
AR	Anonymous Record
IR	Identified Record
$SymD_{KLD}(IR, AR)$	symmetric KLD between IR and AR
$SymD_{KLD\_r}$	symmetric KLD of rating tokens
$SymD_{KLD\_c}$	symmetric KLD of category tokens
$SymD_{KLD\_l}$	symmetric KLD of lexical tokens
$SymD_{KLD\_r\_c}$	symmetric KLD of rating and category
$SymD_{KLD\_l\_r\_c}$	symmetric KLD of all tokens

Table 1: Notation and abbreviations.

4.1.2 Kullback-Leibler Divergence (KLD) Model

We use symmetric KLD (see Section 2) to compute the distance between anonymous and identified records. To do so, we first compute distributions of all records and then we smooth the distributions via Laplace smoothing [14](same as the probability estimation in explained in Naive Bayesian in Section 2). As before, we compute four distributions. To link AR with respect to token type R, we compute $SymD_{kl}$ between the distribution of R for AR and the distribution of R for each IR. Then, we return a list sorted in ascending order of $SymD_{KLD}(IR, AR)$ values. The first entry represents the account with the most likely match.

4.2 Study Results

We now present the results corresponding to the lexical tokens. Then, in the next section, we experiment with some combinations of lexical and non-lexical ones.

4.2.1 Lexical – Results

Figures 1(a) and 1(b) depict LRs (Top-1 and Top-10) for NB and KLD with the unigram token. As expected, with the increase in the anonymous record size, the LR grows: it is high in both Top-1 and Top-10 plots. For example, in Top-1 of both figures, the LRs are around: 19%, 59% and 83% for anonymous record sizes of 10, 30 and 60, respectively. Whereas, in Top-10 of both figures, the LRs are around: 45.5%, 83% and 96% for same record sizes. This suggests that reviews are highly linkable based on trivial single-letter distributions. Note that the two models exhibit similar performance.

Figures 1(c) and 1(d) consider the digram token. In both models, the LR is impressively high: it gets as high as 99.6%/99.2% in Top-1 for NB/KLD for an AR size of 60. For example, the Top-1 LRs in NB are: 11.7%, 62.9%, 87.5% and 97.1%, for respective AR sizes of 1, 5, 10 and 20. Whereas, in KLD, the Top-1 LRs for record sizes of 10, 30 and 60 are: 1.9%,74.9% and 99.2%, respectively.

Unlike unigrams – where LRs in both models are comparable – KLD in digram starts with LRs considerably lower than those of NB. However, the situation changes when the record size reaches 50, with KLD performing comparable to NB. One reason for that could be that KLD improves when the distribution of ARs is more similar to that of corresponding identified records; this usually occurs for large record sizes, as there are more tokens.

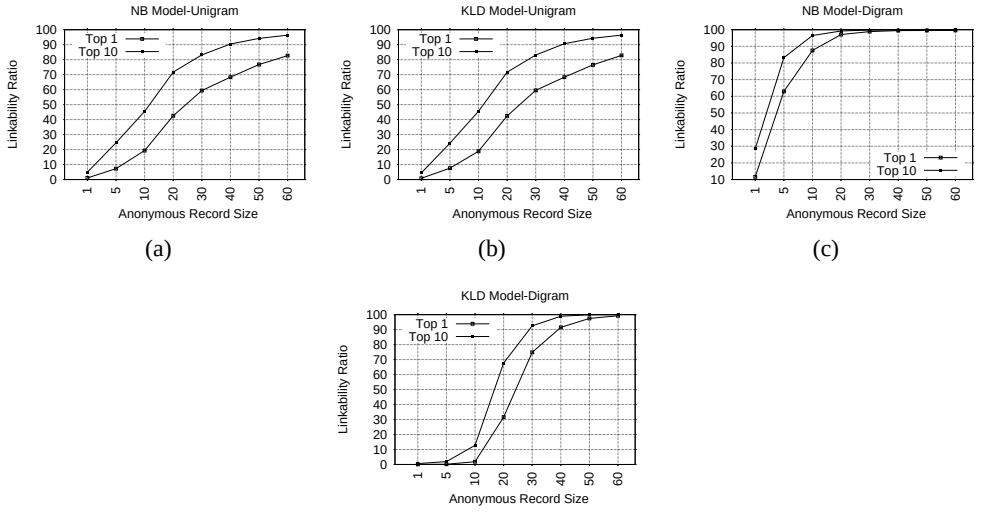


Figure 1: LRs of NB and KLD models for unigrams and digrams

Not surprisingly larger AR sizes entail higher LRs. With NB, a larger record size implies that, a given AR has more tokens in common with the corresponding IR. Thus, an increase in the prediction probability $P(IR|T_1, T_2, ...T_n)$. For KLD, a larger record size causes the distribution derived from the AR to be more similar to the one derived from the corresponding IR.

4.3 Improvement I: Combining Lexical with non-lexical Tokens

In an attempt to improve the LR, we now combine the lexical tokens with the non-lexical ones.

4.3.1 Combining Tokens Methodology

This is straightforward in the NB. We simply increase the list of tokens in the unigram- or digram-based NB by adding the non-lexical tokens. Thus, for every AR, we have $P(lexical\_token_i|IR)$, $P(category\_token_i|IR)$ and $P(rate\_token_i|IR)$.

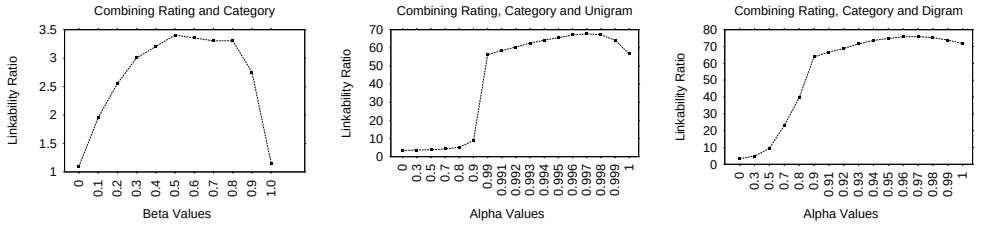
Combining non-lexical with lexical tokens in KLD is less clear. One way is to simply average $SymD_{KLD}$ values for both token types. However, this might degrade the performance, since lexical distributions may convey much more information than their non-lexical counterparts. Thus, giving them the same weight would not yield better results. Instead, we combine them using a weighted average. First, we compute the weighted average of rating and category $SymD_{KLD}$:

$$SymD_{KLD\_r\_c}(P, Q) =$$
$$\beta \times SymD_{KLD\_r}(P, Q) + (1 - \beta) \times SymD_{KLD\_c}(P, Q)$$

Then, we combine the above with $SymD_{KLD}$ of the lexical tokens to compute the final weighted average:

$$SymD_{KLD\_l\_r\_c}(P, Q) =$$
$$\alpha \times SymD_{KLD\_l}(P, Q) + (1 - \alpha) \times SymD_{KLD\_r\_c}(P, Q)$$

Thus, our goal is to get the right β and α values. Intuitively, lexical $SymD_{KLD}$ should have more weight as it carries more information. Since there is no clear way of assigning weight values, we experiment with several choices and pick the one with the best performance; we discuss the selection process below. We experiment only within the IR set and then verify the results generalize to the AR. This is done as follows:



(a) (b) (c)

Figure 2: Results of combining different tokens using different β and α values

First, for every IR, we allocate the last 30 reviews as a testing record and the remainder – as a training record. Then, we experiment with $SymD_{KLD\_r\_c}$ using several β values and set β to the value that yields the highest LR based on the testing records. Then, we experiment with $SymD_{KLDJ\_r\_c}$ using several α values and, similarly, pick the one with the highest LR.

Since β or α could assume any values, we need to restrict their choices. For β, we experiment with a range of values, from 0 to 1.00 in 0.1 increments. For α, we expect the optimal value to exceed 0.9, since the LR for lexical tokens is probably higher than non-lexical ones. Therefore, we experiment with the weighted average by varying α between 0.9 and 0.99 in 0.01 increments.

If the values exhibit an increasing trend (i.e., $SymD_{KLDJ\_r\_c}$ at α of 0.99 is the largest in this range) we continue experimenting in the $0.99 - -1.00$ range in 0.001 increments. Otherwise, we stop. For further verification, we also experiment with smaller α values: $0.0, 0.3, 0.5, 0.7,$ and 0.8, all of which yield LRs significantly lower than 0.9 for both the unigram and digram. We acknowledge that we may be missing α or β values that could further optimize $SymD_{KLDJ\_r\_c}$. However, results in the following section show that our selection yields good results.

Figure 2(a) shows LRs (Top-1) for β values. The LR gradually increases until it tops off at 3.4% with $\beta = 0.5$ and then it gradually decreases. Figure 2(b) shows LRs (Top-1) for α values in the unigram case. The LR has an increasing trend until it reaches 67.8% with $\alpha = 0.997$ and then it decreases. Figure 2(c) shows LRs (Top-1) for α values in the diagram case where it tops off at 75.9% with $\alpha = 0.97$. Thus, the final values are 0.5 for β and $0.997/0.97$ for $alpha$ in the unigram/digram case. Even though we extract α and β values by testing on a record size of 30, the results in following sections show that the derived weights are effective when tested on ARs of other sizes.

4.3.2 Combining Lexical with Non-Lexical Tokens – Results

Figures 3(a) and 3(b) show Top-1 and Top-10 plots in NB and KLD models of unigram tokens before and after combining them with rating and category tokens. Adding non-lexical tokens to unigrams substantially increases LRs in several record sizes. In NB, the gain in Top-1 LRs ranges from 0.25-18.9% (1.4 - 15.7% for Top-10 LRs). In KLD, the gain in Top-1 LRs ranges from 2.5-11.9 (2-7.8% in Top-10 LRs) for most record sizes. These findings shows how effective is combining the non-lexical tokens with the unigrams. In fact, we can accurately identify almost all ARs.

Figures 3(c) and 3(d) show the effect of adding ratings and categories to digrams. The overall effect is less: in NB (KLD) model, the increase in Top-1 LRs ranges from 0.3-1.8% (0.2-2.7%) for most record sizes. The increase is very similar in Top-10 plots.

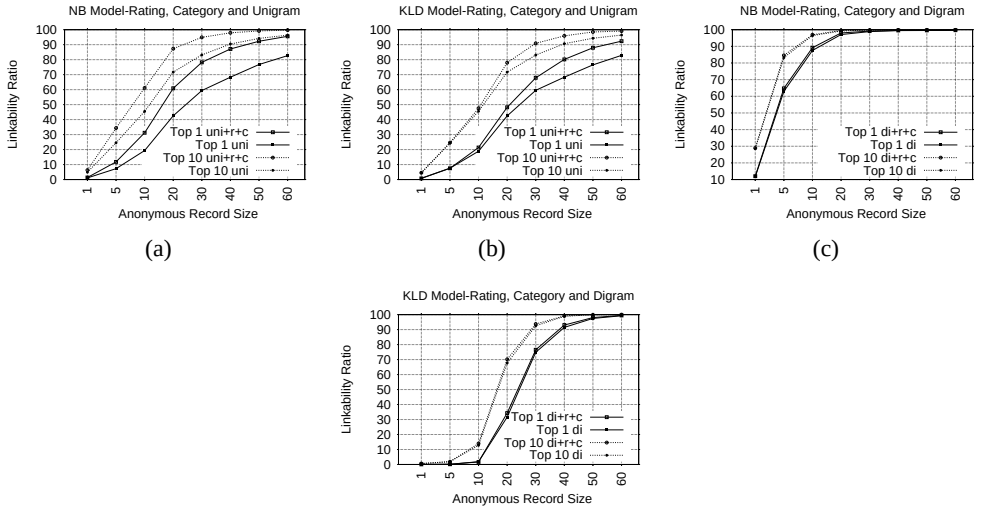


Figure 3: LRs for NB and KLD for combining ratings and categories with unigrams or digrams

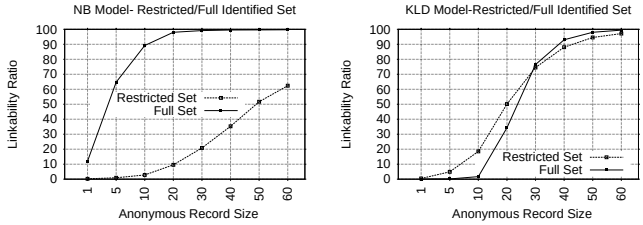


Figure 4: LRs for NB and KLD in full and restricted identified set

4.4 Restricting Identified Record Size

In previous sections, our analysis was based on using the full data set. That is, except for the anonymous part of the data set, we use all of the user reviews as part of our identified set. Although the LR is high in many cases, it is not clear how the models will perform when we restrict the IR size. To this end, we re-evaluate the models with the same problem settings, however, with a restricted IR size. We restrict the IR size to the AR size; both randomly selected without replacement.

Figures 4(a) and 4(b) show two Top-1 plots in NB and KLD models: one plot corresponds to the restricted identified set and the other – to the full set. Tokens used in the models consist of digrams, ratings and categories (since this combination gives the highest LR). Unlike the previous sections, where NB and KLD behaved similarly, the two models now behave differently when restricting the identified set. While NB performs better than KLD on the full set, the latter performs much better than NB when the identified set is restricted. In fact, in some cases, KLD performs better when the set is restricted.

The reason for this improved KLD performance might be the following: in the symmetric KLD distance function, the distributions of both the IR and AR have to be very close in order to match regardless of the size of the IR; unlike the NB, where larger training sets would lead to better estimates of the token probabilities and thus more accurate predictions.

In KLD, we achieve high LRs for many record sizes. For example, Top-1 LRs in the restricted set are 74.5%, 88% and 97.1% when the anonymous (and identified) record sizes are 30, 40 and 60, respectively.

Algorithm $Match\_All$**:** Pseudo Code

Input: (1) Set of ARs: $S_{AR} = \{AR_1, AR_2, ..., AR_n\}$
(2) Set of reviewer-ids / identified records:
$S_{IR} = \{IR_1, IR_2, ..., IR_n\}$
(3) Set of matching lists for each AR:
$S_L = \{List_{AR_1}, .., List_{AR_n}\}$

Output: Matching list: $S_M = \{(IR_{i_1}, AR_{j_1}), ..., (IR_{i_n}, AR_{j_n})\}$

set $S_M = \emptyset$
While $|S_{AR}| \neq 0$:
    Find AR_i with smallest $SymD_{KLD}$ in all lists in S_L
    Get corresponding reviewer-id IR_j
    Add (IR_j, AR_i) to S_M
    Delete AR_i from S_{AR}
    Delete $List_{AR_i}$ from S_L
    For each $List_t$ in S_L,
        Delete tuple containing IR_j from $List_t$
    End For
End While

NOTE 1: $List_{AR_i}$ in S_L is a list of pairs (IR_j, V_{ij}) where $V_{ij} = SymD_{KLD}(IR_j, AR_i)$, for all j
NOTE 2: $List_{AR_i}$ is sorted in increasing order of V_{ij}, i.e., IR_j with lowest $SymD_{KLD}(IR_j, AR_i)$ at the top.

Figure 5: Pseudo-Code for matching all ARs at once.

Whereas, the LRs in the full set for the same AR sizes are: 76.5% , 93% and 99.4%. When the record size is less than 30, KLD performs better in the restricted set than the full one. For example, when the AR size is 20, the LR in the restricted set is 50.1% and 34.3% in the full set. In NB, Top-1 LR in the restricted set is lower than the full set. For instance, it is 20.8%, 35.3% and 62.4% for AR sizes of: 30, 40 and 60, respectively. Whereas, for the same sizes, the LR is more than 99% in the full set.

This result has one very important implication: even with very small IR sizes, many anonymous users can be identified. For example, with only IR and AR sizes of only 30, most users can be accurately linked (75% in Top-1 and 90% in Top-10). This situation is very common since many real-world users generate 30 or more reviews over multiple sites. Therefore, even reviews from less prolific accounts can be accurately linked.

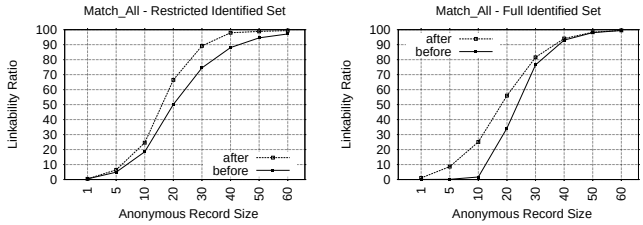
4.5 Improvement II: Matching all ARs at Once

We now experiment with another natural strategy of attempting to match all ARs at once.

4.5.1 Methodology

In previous sections, we focus on independently linking one AR at a time. That is, the input to our matching/linking model is one AR and the output is the user of the closest IR. If we change the problem settings and make the input a set of ARs(instead of one) where each AR belongs to a different user, we may be able to improve the linkabilty knowing that an AR cannot be mapped to more than one user. To this end, we construct algorithm $Match\_All()$ in Figure 5 as an add-on to the KLD models suggested in previous sections where the input is a set of ARs, each of which belongs to a different user. The number of ARs in the input is equal to the number of users in our dataset.

$SymD_{KLD}(IR_j, AR_i)$ symmetrically measures the distance between their (IR_j's and AR_i's) distributions. Since every AR maps to a distinct IR (AR_i maps to IR_i), it would seem that lower $SymD_{KLD}$ would lead to a better match. We use this intuition to design $Match\_All()$. As shown in the figure, $Match\_All()$



(a) (b)

Figure 6: Effects of $Match\_All()$ on LRs in full and restricted identified set: before and after plots

picks the smallest $SymD_{KLD}(IR_j, AR_i)$ as the map between IR_j and AR_i and then deletes the pair (IR_j, V_{kj}) from all remaining lists in S_L. The process continues until we compute all matches. Note that, for any $List_{AR_k}$, (IR_j, V_{kj}) is deleted from the list only when there is another pair (IR_j, V_{lj}) in $List_{AR_l}$, such that $SymD_{KLD}(IR_j, AR_l) \leq SymD_{KLD}(IR_j, AR_k)$, and IR_j has been selected as the match for AR_l.The output of the algorithm is a match-list: $S_M = \{(IR_{i_1}, AR_{j_1}), ..., (IR_{i_n}, AR_{j_n})\}$.

We now consider how $Match\_All()$ could improve the LR. Suppose that we have two ARs: AR_i and AR_j along with corresponding sorted lists L_i and L_j and assume that IR_i is at the top of each list. Using only KLD (as in previous sections), we would return IR_i for both ARs and thus miss one of the two. Whereas, $Match\_All$, would assign IR_i to **only** one AR – the one with the smaller $SymD_{KLD}(IR_i, ...)$ value. We would intuitively suspect that $SymD_{KLD}(IR_i, AR_i) < SymD_{KLD}(IR_i, AR_j)$ since IR_i is the right match for AR_i and thus their distributions would probably be very close. If this is the case, $Match\_All$ would delete IR_i (erroneous match) from the top of L_j which could help clearing up the way for IR_j (correct match) to the top of L_j.

We note that there is no guarantee that $Match\_All()$ will always work: one mistake in early rounds would lead to others in later rounds. We believe that $Match\_All()$ works better if $SymD_{KLD}(IR_i, AR_i) < SymD_{KLD}(IR_j, AR_i)$ $(j \neq i)$ holds most of the time.

In the next section, we show the results of $Match\_All()$ when we experiment with the KLD model with digram, rating and category tokens.[2].

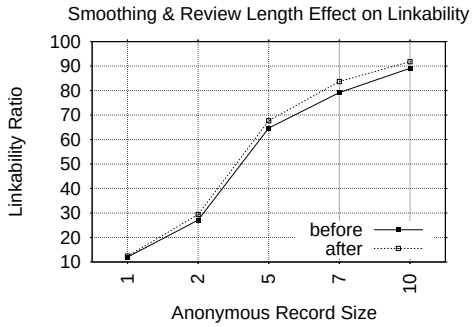
4.5.2 Results

Figures 6(a) and 6(b) show the effect of $Match\_All()$ on Top-1 LRs in both the restricted identified set and the full identified set, respectively. The combination of diagram, rating and category tokens are used. Each figure shows two Top-1 plots: one for the LR after using $Match\_All$ and the other – for the LR before using it. Clearly, $Match\_All$ is effective in improving the LR for almost all record sizes. For the restricted set, the gain in the LR ranges from 1.6-16.4% for nearly all AR sizes. A Similar increase is observed in the full set that ranges from 1-23.4% for most record sizes. This shows that the $Match\_All$ is very effective when used with diagram, rating and category tokens. The privacy implication of $Match\_All$ is important as it significantly increases the LR for small ARs in the restricted set. This shows that privacy of less prolific users is exposed even more with $Match\_All$.

4.6 Improvement III: Improving Linkability for Small Anonymous Records

Although most of the proposed exhibits high LR's when the AR size is large, the linkability is not as high for small record sizes. For improving the LR for small AR (in the full identified set), we consider the NB model that uses diagrams, ratings and categories as its tokens (see Section 4.3.2) as a base for our

[2]We also tried $Match\_All()$ with the NB model and it did not improve the LR.



(a)

Figure 7: Effects of smoothing and review length on LRs: before and after plots

improvement. We use this model as it performs the best for small ARs comparable to other models. To that end, we first change the way we smooth the probabilities as follows:

$$P(token_i|IR) = \frac{Num\ Token_i\ in\ D\ +\ \eta}{Num\ Tokens\ in\ D\ +\ \eta\ \times\ Num\ Possible\ Tokens}$$

Unlike the models in the previous sections (see Section 2)[3], η could take values other than 1. In fact, we experiment with several different values and we find that η value of 0.5 gives the best performance[4]. The intuition is that setting η to a value less than 1 may help downscale the effect of noisy digrams that the user rarely use. Additionally, we leverage the length of the reviews, the number of the alphabetical letters, as an additional feature to the model. We consider the length of the reviews as we intuitively believe that different users tend to write longer/shorter reviews than others. We model the length as a normal distribution and we use the maximum likelihood estimate to set the distribution parameters [14, 5].

Figure 7 shows the effect of this improvement. For clarity, we only show the improvement resulting from combining the two aforementioned steps. As shown, the Top-1 LRs gain roughly ranges from 0.5%-5%. For example, for AR size of 5, 7 and 10, the Top-1 LR approximately increases from 65%, 79% and 89% to 68%, 84% and 92%, respectively. Similar increases are observed in the Top-10 LR which reach 88%/98% for AR size of 5/10 (and up to 30%/54% for AR size of 1/2).

4.7 Study Summary

We now summarize the main findings and conclusions of our study.

1. The LR becomes very high – reaching up to $\sim$ 99.5% in both KLD and NB when using only digram tokens. (See Section 4.2.1).

2. Surprisingly, using only unigrams, we can link up to 83% in both NB and KLD models, with 96% in Top-10. (See Section 4.2.1). This suggests that reviewers expose a great deal merely from their single letter distributions.

3. Non-lexical tokens are very useful in tandem with lexical tokens, especially, the unigram: we observe a $\sim$19%/12% Top-1 LR increase in NB/KLD for some cases. (See Section 4.3.2).

[3]Here, document D refers to the Identified Record IR

[4]Note that we experiment η on only the training set and pick the best value

4. Relying only on unigram, rating and category tokens, we can accurately link 96%/92% of the ARs (size 60) in NB/KLD. (See Section 4.3.2).

5. Restricting the IR size does not always degrade linkability. In KLD, we can link as many as 97% ARs when the IR size is small. (See Section 4.4).

6. Linking all ARs at once (instead of each independently) helps improve accuracy. The gain is up to 16/23% in restricted/full set. (See Section 4.5.2).

7. Generally, NB performs better than KLD when we use the full identified set and KLD performs better when we use the restricted identified set.

8. Combining review length with different smoothing techniques is helpful in increasing the linkability for small AR and the Top-1/Top-10 LR reach 92%/98% for AR size of 10(See Section 4.6).

5 Discussion

Implications. We believe that the results of, and techniques used in, this study have several implications. One implication is the possibility to cross-reference accounts (and reviews) among multiple (similar)review sites. If a person contributes to two similar review sites under two identities, it is likely that sets of reviews from these sites can be linked. This could be quite detrimental to contributors' privacy. Another implication is the ability to correlate – on the same review site – multiple accounts that are in fact manipulated by the same person. This could make our techniques very useful in detecting review spam [10], whereby a contributor authors reviews under different accounts to tout (also self-promote) or criticize a product or a service.

Prolific Users. While there are clearly many more occasional (non-prolific) reviewers than prolific ones, we believe that our study of prolific reviewers is important, for two reasons. First, the number of prolific contributors is still quite large. For example, from only one review site – Yelp – we identified $\sim 2,000$ such reviewers. Second, given the spike of popularity of review sites [1], we believe that, in the near future, the number of such prolific contributors will grow substantially. Also, even many occasional reviewers, with the passage of time, will enter the ranks of "prolific" ones, i.e., by slowly accumulating a sufficient corpus of reviews over the years. Nevertheless, our study suggests that privacy is not high even for non-prolific users, as discussed in Section 4.5. For example, when both IR and AR sizes are only 20 (i.e., total per user contribution is 40 reviews), we can accurately link around 70% of anonymous records to their reviewers.

Anonymous Record Size. Our models perform best when the AR size is 60. However, for every reviewer in our dataset, 60 represents less than 20% of that person's total number of reviews. Also, using NB coupled with digram, rating, category and length features, we can accurately link most anonymous records when AR size is small (see Section 4.6).

Unigram Tokens. While our best-performing models are based on digram tokens, we also obtain high linkability results from unigram tokens that reach up to 83% (96% in the Top 10) in NB or KLD. The results improve to 96/92% when we combine unigrams with rating and category tokens. Note that the number of tokens in unigram-based models is 59 (26) tokens with (without) combining them with rating and category tokens. Whereas, the number of tokens in diagram-based models is 676 (709 when combined with rating and category tokens). This makes linkability accuracy based on unigram models very comparable to its diagram counterpart, while the number of tokens is significantly fewer. This implies a substantial reduction in resources and processing power in unigram-based models which would make them scale better. For example, if we assume that the attacker wants to link a set of anonymous reviews to *many* large review datasets, unigram-based models would scale better, while maintaining similar level of accuracy.

Potential Countermeasures. One concrete application of our techniques is via integration with the review site's front-end software in order to provide feedback to authors indicating the degree of linkability of their reviews. For example, when the reviewer logs in, a linkability nominal/categorical value (e.g. high, medium, and low) could be shown indicating how some of his/her reviews (selected randomly) are linkable to the rest. It would then be up to to the individual to maintain or modify their reviewing patterns to be less linkable. Another way of countering linkability, as suggested in [15], is for the front-end software to automatically suggest a different choice of words that are less revealing (less personal) and more common among many users. We suspect that, with the use of such words, reviews would be less linkable and lexical distributions for different users would be more similar.

6 Related Work

Many authorship analysis studies are in the literature. Among the most related recent studies are [16, 15, 3]. In [16], a large scale author identification techniques (based on linguistic stylometry) are evaluated on blog de-anonymization. While the problem formulation is similar to ours, there are notable differences. First, we study the linkability in a different context; i.e., user reviews. User reviews have ratings and categories, which prove useful in some scenarios, while blogs(used in [16]) do not. Additionally, user reviews are shorter while blogs could be as long as an article. Moreover, user reviews are mainly about user evaluations of a specific service/product while blogs could be very random, such as news reporting or literature-related work. Second, our study points to high linkability ratios in user reviews, nearly 100% Top-1 linkability ratio, where as in [16], the Top-1 linkability ratio is around 20% [5]. Third, our study shows high linkability ratios in the presence of very simple features. A related problem is explored in [15]. It focuses on identifying authors based on reviews in both single- and double-blinded academic peer-reviewing processes of scientific journals and conferences. Naïve Bayes classifier is used – along with word-based tokens – to identify authors and the best result is around 90%. This work is different from ours in several aspects. First, it explores the author identification in a very restricted domain; i.e., academic paper reviews. Second, the number of candidate authors is around 20 which is less than ours(~ 2000). Third, the number of features used in [15] is large where unigram, bigram, and trigrams based on words(a sequence of one, two and three words) are used. In ours, we only use unigrams and bigrams that are based on letters (in addition to the ratings and categories). The work in [3] also considered author identification and similarity detection by incorporating a rich set of stylistic features along with a novel technique(based on Karhunen-Loeve-transforms) to extract write-prints. An identification performance of 91% is achieved. The same approach is tested on a large set of Buyer/Seller Ebay feedback comments collected from Ebay. Such comments typically reflect one's experience when dealing with a buyer or a seller. Unlike our general-purpose reviews, these comments do not review products, services or places of different categories. Additionally, the scale of the problem is different and the analysis is performed for only 100 authors. An author identification technique based on frequent pattern write prints is shown in [9] and author identification techniques based on extracting lexical, syntactic, structural and content-specific features and then feeding them to some classifiers are shown in [21]. For a comprehensive overview of authorship analysis studies, we refer to [19].

While many of the author identification studies are somewhat similar to our present work, there are some notable differences. First, we perform authorship identification analysis in a context that has not been extensively explored – generic user reviews. User reviews are generally are less formal and less restricting in the choice of words. In a review, the author generally assesses something and thus the text conveys some evaluation and personal opinions. In addition, reviews contain other non-textual information, such as the ratings and categories of things being reviewed. These types of extra information provide added

[5] Note in [16], the identification accuracy is increased to 80% by not making a guess when there is not enough confidence; however, this does not increase the linkability ratio (recall is low).

leverage(shown in 4.3). Second, our problem formulation is different. We study linkability of reviews in the presence of a large number of prolific contributors where the number of anonymous reviews could be more than one (up to 60 reviews). Whereas, most prior work attempts to identify authors from a small set of authors, each with small sets of texts. Third, we show high linkability ratios in the presence of very simple features. For example, reviewers can be accurately identified from their letter distributions. These measurement results are very alarming for users concerned about their privacy.

Some work is done in recovering authors based on their ratings, using external knowledge such as [7] and [17]. Another related research effort assesses authenticity of reviews [10]. A related study in [4] proposes linguistic-based techniques to detect user attempts to hide their writing styles.

7 Conclusion

Large numbers of Internet users are becoming frequent visitors and contributors to various review sites. At the same time, they are concerned about their privacy. In this paper, we study linkability of reviews. Based on a large set of reviews, we show that a high percentage (99% in some cases) are linkable, even though we use very simple models and very simple features set. Our study suggests that users reliably expose their identities in reviews, which could be partly due to the way the users write their reviews and the places they select to review. This has certain important implications for cross-referencing accounts among different review sites and detecting people who write reviews under different identities. Additionally, techniques used in this study could be adopted by review sites to give contributors feedback about linkability of their reviews.

References

[1] Yelp By The Numbers. `http://officialblog.yelp.com/2010/12/2010-yelp-by-the-numbers.html`.

[2] Yelp Elite Squad. `http://www.yelp.com/faq#what_is_elite_squad`.

[3] A. Abbasi and H. Chen. Writeprints: A Stylometric Approach to Identity-Level Identification and Similarity Detection in Cyberspace. In *ACM Transactions on Information Systems*, 2008.

[4] S. Afroz, M. Brennan, and R. Greenstadt. Detecting Hoaxes, Frauds, and Deception in Writing Style Online. In *IEEE Symposium on Security and Privacy*, 2012.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[6] K. Dave, S. Lawrence, and D. M. Pennock. Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. In *international conference on World Wide Web*, 2003.

[7] D. Frankowski, D. Cosley, S. Sen, L. Terveen, and J. Riedl. You Are What You Say: Privacy Risks of Public Mentions. In *International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006.

[8] M. Hu and B. Liu. Mining and Summarizing Customer Reviews. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[9] F. Iqbal, H. Binsalleeh, B. Fung, and M. Debbabi. A unified data mining solution for authorship analysis in anonymous textual communications. In *Information Sciences (INS): Special Issue on Data Mining for Information Security*, 2011.

[10] N. Jindal and B. Liu. Opinion Spam and Analysis. In *ACM International Conference on Web Search and Data Mining*, 2008.

[11] N. Jindal, B. Liu, and E.-P. Lim. Finding Unusual Review Patterns Using Unexpected Rules. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, 2010.

[12] D. Lewis. Naive(bayes) at forty:the independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning*, 1998.

[13] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. Lauw. Detecting Product Review Spammers using Rating Behaviors. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, 2010.

[14] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[15] M. Nanavati, N. Taylor, W. Aiello, and A. Warfield. Herbert West – Deanonymizer. In *6th USENIX Workshop on Hot Topics in Security*, 2011.

[16] A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, and D. Song. On the Feasibility of Internet-Scale Author Identification. In *IEEE Symposium on Security and Privacy*, 2012.

[17] A. Narayanan and V. Shmatikov. Robust De-anonymization of Large Sparse Datasets. In *IEEE Symposium on Security and Privacy*, 2009.

[18] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Empirical Methods on Natural Language Processing Conference*, 2002.

[19] E. Stamatatos. A Survey of Modern Authorship Attribution Methods. In *Journal of the American Society for Information Science and Technology*, 2009.

[20] S. Yadav, A. K. Reddy, A. N. Reddy, and S. Ranjan. Detecting Algorithmically Generated Malicious Domain Names. In *Internet Measurement Conference*, 2010.

[21] R. Zheng, J. Li, H. Chen, and Z. Huang. A Framework for Authorship Identification of Online Messages: Writing Style Features and Classification Techniques. In *Journal of the American Society for Information Science and Technology*, 2006.

PiCoDa: Privacy-preserving Smart Coupon Delivery Architecture

Kurt Partridge[1*], Manas A. Pathak[2*], Ersin Uzun[3], Cong Wang[4*]

[1]Google Inc.
[2]Adchemy Inc
[3]Palo Alto Research Center
[4]Illinois Institute of Technology

Abstract

In this paper, we propose a new privacy-preserving smart coupon delivery system called PiCoDa. Like prior work on private behavioral targeted advertising, PiCoDa protects user data by performing targeting on the end-user's device. However, PiCoDa makes two more guarantees: it verifies a user's eligibility for a coupon, and it protects the vendor's privacy by not revealing the targeting strategy.

To accommodate the constraints of different targeting strategies, PiCoDa provides two targeting protocols that tradeoff user privacy and vendor privacy in different ways. We show how both designs meet requirements for user privacy, vendor protection, and robustness. In addition, we present simulation results of the protocols using realistic parameters to further validate the efficiency and effectiveness of PiCoDa.

1 Introduction

In recent years, online advertising has come to rely more heavily on behavioral targeting.Behavioral targeting allows vendors to provide more relevant messages by using indicators of interest from historical data about a user. From the user's perspective, better targeting is beneficial as it leads to more personalized service and less exposure to information that is not of interest to them. Particularly interesting is how behavioral targeting incorporate not just web data, but physical contextual data like location, time of day, and proximity to other individuals [9].

However, despite the incentives to both vendors and users, the current practice of behavioral targeting raises great privacy concerns among users [15]. In order to target accurately, vendors need to know adequate information about users, such as their demographics, geographic locations, purchase behaviors, browser and internet search histories. However, collecting such information is usually in conflict with user privacy. Enabling accurate behavioral targeting without compromising user privacy is a challenging problem.

In this paper, we propose a new privacy-preserving smart coupon delivery architecture called PiCoDa. Instead of doing the behavioral matching for coupon requirements on the vendor side, we propose to perform it on the user device. This allows a vendor to deliver targeted coupons while users keep full control of their behavioral data, which never leaves the user's device. Shifting the behavioral targeting computations to the client device is not a new idea and has been recently applied to the context of personalized search [16] and online advertising [12, 7, 6, 3]. However, targeted coupon delivery poses additional challenges beyond those demanded by targeted advertising. Shifting the computations for behavioral matching to the user's device alone is insufficient to protect both users and vendors. First, coupons must be delivered only to the

*Work was done when all authors were affiliated with PARC

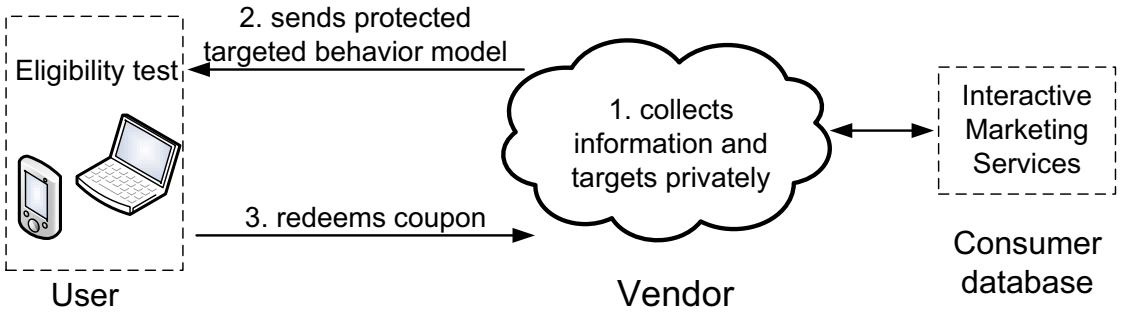


Figure 1: An Exemplary Architecture of Activity-based Targeting

eligible users to prevent coupon exploits. The simpler strategy of pushing down all the coupons in clear and asking the user's device to select ones the user is eligible for does not work well. It exposes the coupons to malicious users who are trying to get discounts they may not have qualified for. Second, a targeting system should ensure that non-eligible users learn nothing about the vendor's targeting strategy (i.e., the coupon's eligibility requirements) beyond their non-eligibility.

Given the complexity of the vendor's targeting strategy, we present two different operating modes for PiCoDa. The first is a non-interactive design that guarantees that behavioral data never leaves a user's device during the coupon targeting process. However, as discussed later in the paper, the non-interactive protocol is only suitable when the vendor's targeting strategy for a coupon is difficult to guess, i.e., its entropy is at least 80 bits. When the vendor's targeting strategy is not hard to guess, we propose a three-round interactive protocol between a PiCoDa server and a user device. In this case, some information about the user's data does leave the client device, but in an indecipherable form that is useless without later cooperation, which the user only provides when they redeem the coupon. We show later in the paper that both designs provide user privacy, vendor protection, and system robustness. Our simulation results with realistic parameter selections further validate the efficiency and effectiveness of our designs.

The rest of the paper is organized as follows. Section 2 introduces the system model, threat model, and our design goals. Then we provide the detailed description of PiCoDa in Section 3. Section 4 gives the security analysis, followed by Section 5, which reports simulation results. Section 6 overviews the related work, followed by our conclusions in Section 7.

2 Problem Statement

2.1 System Model

We consider PiCoDa, the privacy-preserving smart coupon delivery architecture, involves two different entities, as illustrated in Fig. 1: the *user*, who wants to enjoy personalized coupon delivery service while releasing as little as possible private personal data, and the *vendor*, who wants to provide accurate user targeting via behavior analysis while protecting itself from coupon exploitation attacks. In addition, we also assume a *consumer database* in our architecture, which provides proprietary background information (possibly coarse-grained) of consumers to help vendors conduct better targeting services.

The user has a mobile device, which maintains the personal information locally on the device. For simplicity, we assume the user's behaviorial data can be represented as a vector where each entry can be either an integer or real number, denoting representative statistics of different kinds of user behavior over a certain amount of time. For example, these elements in the vector could represent statistics of URL streams in the browsing history, or the number of times specific websites have been visited. The integers could also represent location traces, such as the number of times of different geo-locations that have been visited. The user local data **changes over time**, and provides the most recent targeting information for vendors.

The vendor, who maintains a PiCoDa server for coupon targeting, usually learns some information (e.g., name, mailing address, etc.) about users when they enroll in a loyalty or a coupon delivery program. They may also have a user disclosed profile or can contact commercial consumer databases to learn more (e.g., gender, ethnicity, marital status etc.) about their users. We assume such coarse-grained information is **static** or changes slowly in practice. By combining the static background information and its proprietary behavioral targeting models, a vendor can effectively create the eligibility requirements expressed as targeting strategy for a coupon and only those users that are eligible for the requirements will receive the coupons.

2.2 Behavior Encoding

A vendor may use many different criteria for deciding whether to deliver a particular user a coupon. For example, a vendor may want to reach loyal customers. This is partially reflected by how frequently the user has visited the vendor's store over the past month. Loyalty may also be measured by the user's past purchasing behavior. Or, the vendor may want to reach new potential customers. This might be inferred from visitors to a competitor's store. Vendors may wish to filter prospective coupon recipients according to the probability of the coupon increasing the recipient's future loyalty. This might be informed by a record of coupon deliveries in the past. Finally, physical constraints or inconvenience factors may also be considered. It does not make sense to send a user in New York a coupon redeemable only at California.

In order to accurately answer above questions, vendors must start from the user's behavior raw data and generate/estimate an eligibility strategy $\mathbf{w}$. In order to be effective, $\mathbf{w}$ must be expressive enough to reflect vendor's strategies. In mobile domain, we consider features from four different types of user behavior: browsing history, geographic traces, purchasing information, and message/contact information. For each coupon, frequency and feature counts might be aggregated over several weeks, while the targeting window could cover a shorter period of a few days.

Fig. 2 gives a more detailed list of features that might be used in a typical targeting scenario. A vendor chooses the features and values of the features they wish to target, and assembles them into a vector. For example, to target users that visited the vendor's website at least twice in the past five days, had visited the vendor's retail store three times in the past 10 days, and had made four purchases in the past month, the vector $\mathbf{w}$ might be encoded as $(2, 3, 4)$ for features (w_1, w_2, w_3) where w_1 is number website visits in the past five days, w_2 is the number of retail store visits in the past 10 days, and w_3 indicates the number of purchases in the past month. In our current system, we restrict criteria to approximate or exact equality matching.

2.3 Security Threats and Design Goals

We consider the protection in the PiCoDa architecture design from both user and vendor sides. Both the user's local behavioral information and vendor's targeting strategy (i.e., the proprietary algorithms utilizing the user's up-to-date behavioral data, purchased or collected static data about users and eligibility requirements of a particular coupon) should be protected. In other words, the system should satisfy the following properties:

- **User data privacy**. Our design aims to guarantee that no user behavioral data is revealed to the vendor during the targeted coupon delivery unless the coupon is redeemed. In case the vendor's targeting strategy is complicated and hard to guess, our design further aims to achieve an ideal case in which the communication between user's device and a PiCoDa server is one way: from server to user device.

- **Vendor protection**. The vendor's coupons and its delivery strategy should be protected from non-eligible users during the coupon delivery process. That is, from the information pushed down to user devices, a user either learns he is eligible for a coupon or learns nothing beyond his non-eligibility for that particular coupon.

TARGETING CRITERIA	EXAMPLE METRIC
Browsing History	
overall interest	Fraction of queries in vendor's campaigned product line over all queries in past month
recent interest	Total number of relevant search queries in the last 5 days
recent interest	Total number of visits to product line-relevant webpages in the last 5 days
loyalty	Fraction of webpage views of vendor's product line relative to general web activity
Geolocation Trace Features	
overall interest	Fraction of visits at vendor's retail store versus all shopping stores.
overall interest	Total time in hours the user has spent at vendor's retail stores.
in-market	Time in days since last visit to vendor's retail store
recent interest	Total number of visits to vendor's retail store in the past 5 days
loyalty	Fraction of visits to vendor's retail store versus competitor retail stores
Purchasing Information	
recent purchase volume	Total number of purchases made in the last 30 days
in-market	Total purchases of similar products falling into vendor's product lines.
in-market	Time since the last purchase within vendor's product line
in-market	Number of purchases of complementary products in past 2 weeks
coupon use	Fraction of purchases with coupons over all purchases.
Messaging/Contact Information	
recent interest	Fraction of messages containing keywords related to vendor's product line category in past 5 days
in-market	The number of the user's recent contacts that have purchased within the vendor's product line

Figure 2: Example features for coupon targeting. This list is by no means exclusive or complete. Additional features may be used for improved targeting.

- **Robustness**. The system design should guarantee robust delivery of coupons to eligible users. Moreover, it should prevent users from faking the behavioral data to collect coupons in any illegitimate manner. However, this is an ambitious design goal in the presence of collusion attacks where eligible users share information with non-eligible users. Against such attacks, we propose a series of alternatives to decrease the marginal gain of a coupon, thus discouraging users from arbitrarily trying different behavioral data to maliciously collect and redistribute coupons.

In short, our goal is to enable vendors to deliver behaviorally targeted coupons without accessing any user sensitive information and protect themselves from disclosing any valuable data and against coupon exploitation attacks in the process. However, it is important to note that we do not attempt to protect user's privacy when he actually redeems a coupon due to two reasons: (1) Users willing to redeem the coupon inevitably reveal their eligibility for the coupon at the point of redemption. (2) Vendors need to utilize users' feedback, in terms of the coupon redemption results, to evaluate and improve their targeting strategies.

3 PiCoDa Protocols

In this section,we first discuss expressing coupon eligibility requirements based on the user's behavioral model and the vendor's strategy. Then, we present the details of two PiCoDa protocols for non-interactive and interactive operation.

3.1 Coupon Eligibility Requirement

Following existing literature on behavior targeting (e.g., [11, 3]), we use vectors to represent both vendor side targeting strategy and user side behavioral model. As defined above, the vendor's targeting strategy is represented by an n-dimensional vector $\mathbf{w} = (w_1, w_2, \ldots, w_n)$. Each user's behavioral model, which contains a series of features from the daily behavior events collected by the mobile device, is also denoted by an n-dimensional vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$. The targeting process thus depends on the eligibility test between the vendor's strategy $\mathbf{w}$ and the user's model $\mathbf{x}$, and we assume $\mathbf{w}$ and $\mathbf{x}$ are from the same n-dimensional space. Based on different encodings of the user's behavior and the vendor's strategy, the eligibility of a coupon is determined by the following three cases:

1. Entries in $\mathbf{w}$ approximately match entries in $\mathbf{x}$ — in this case, vendors rely on a series of predictive features to distribute coupons. Those features are numerical values computed from user's daily behavior stream. How close $\mathbf{w}$ and $\mathbf{x}$ are is measured by certain distance metrics, such as Euclidean distance and/or cosine distance [8], depending on the different application scenarios.

2. Entries in $\mathbf{w}$ exactly match entries in $\mathbf{x}$ — in this case, vendors rely on a series of deterministic rules to distribute coupons. For example, the vendor may ignore other entries in $\mathbf{w}$ but only care about whether the user has been to a certain local retail store or has been the vendor's loyalty program member to offer him a coupon. Since these rules are usually encoded as discrete binary or categorical values, distance based similarity measurement might no longer be meaningful.

3. Hybrid of case (1) and (2).

For distance based eligibility testing, various data preprocessing techniques, such as min-max normalization [8], can be applied to the derived model vector $\mathbf{w}$ and $\mathbf{x}$. For presentation simplicity, we assume these preprocessing steps are appropriately coordinated between a PiCoDa server and a user device before any coupon delivery takes place.

As discussed in the following sections, our eligibility test only guarantees that the users getting a coupon are eligible for it. However, by design, it is possible for a subset of eligible users not to get a coupon. We argue that small number of false negatives are acceptable to vendors. As long as the number of false positives, i.e., non-eligible users getting a coupon, is negligible, we consider vendor's interest not to be violated. The rest of this section shows how we leverage these assumptions in the design of PiCoDa.

3.2 Protocol 1: Privacy-preserving Non-interactive Coupon Targeting

In non-interactive protocol, the PiCoDa server pushes down the targeting strategy in a protected form to the user, who then performs a blind matching with local behavioral model to determine his eligibility status for a coupon. Such a design has the benefit that nothing leaves the user's device before the user actually gets a coupon. An assumption in the non-interactive targeting design is that the vendor's targeting strategy must be hard to guess, i.e., its entropy should be at least 80bits, to be secure against brute-force guessing attacks. Otherwise, a malicious user can find a behavioral model that matches vendor's strategy by trial and error.

Compared to having rules that dictates exact match on few variables, high entropy requirement for targeting strategy can be more easily met if $\mathbf{w}$ contains many predictive features with numerical values. Thus, we focus on the first case of eligibility test, where a user's eligibility for a coupon is determined by measuring the similarity between the vendor's targeting strategy $\mathbf{w}$ and the user's behavioral model $\mathbf{x}$. As noted previously, the eligibility test has to happen at the user's device for user data privacy and better scalability.However, the targeting strategy $\mathbf{w}$ cannot be pushed down to the user device in clear as it would violate two of previously mentioned design goals: vendor protection and robustness. To achieve the challenging goal of privacy-preserving non-interactive coupon targeting, we use locality sensitive hash functions to implement private similarity tests.

Locality Sensitive Hashing. Locality-sensitive hashing (LSH) denotes the method to perform probabilistic dimension reduction of high-dimensional data [1]. Its key idea is to hash the input data points (using specially-designed locality-sensitive hash functions), such that for similar data points (close to each other), the collision probability is much higher than for those that are far away. For different distance metrics, the family of LSH is defined differently [1]. For presentation simplicity, we take the LSH defined over cosine distance as an example in our targeting scheme description, though LSH for other distance metrics, such as Euclidean distance [5] or Hamming distance [1] etc., can also be used.

The cosine distance metric can be represented by the angle between two vectors $\mathbf{w}$ and $\mathbf{x}$, $\Theta(\mathbf{w}, \mathbf{x}) = \cos^{-1}(\frac{\mathbf{w}\cdot\mathbf{x}}{||\mathbf{w}||\cdot||\mathbf{x}||})$.

For this distance measure, Charikar [4] gives the following LSH family $\mathcal{F}$. By drawing each component of an n-dimensional random vector $\mathbf{r}$ from the Gaussian distribution $\mathcal{N}(0, 1)$ independently, the hash function $f_{\mathbf{r}}(\cdot)$ computed over an n-dimensional vector $\mathbf{q}$ is given by:

$$f_{\mathbf{r}}(\mathbf{q}) = \begin{cases} 1 & \text{if } \mathbf{r}\cdot\mathbf{q} \geq 0 \\ 0 & \text{if } \mathbf{r}\cdot\mathbf{q} \leq 0 \end{cases}$$

This construction divides the entire input space of the dataset by the hyperplane represented by the vector $\mathbf{r}$; two vectors lying on the same side of the hyperplane defined by $\mathbf{r}$ hash to the same value. The likelihood of two vectors $\mathbf{w}$ and $\mathbf{x}$ hashing to the same LSH value depends on their cosine similarity, i.e.,

$$p = \Pr[f_{\mathbf{r}}(\mathbf{w}) = f_{\mathbf{r}}(\mathbf{x})] = 1 - \frac{\Theta(\mathbf{w}, \mathbf{x})}{\pi}. \quad (1)$$

As using one hash function $f_{\mathbf{r}}$ from the family $\mathcal{F}$ does not give accurate enough results for the locality sensitive hash, in practice, it is suggested to use a set of K hash functions $f_{\mathbf{r}_1}, \ldots, f_{\mathbf{r}_K}$, and the final hash value is obtained by concatenating their output. This K-bit LSH function, denoted as $F(\cdot)$, maps an n-dimensional vector $\mathbf{x}$ into a K-bit string.

Non-interactive Protocol for Approximate Matching. However, due to the locality sensitivity, where similar vectors will be hashed together, LSH no longer has the one-way property of a cryptographic function. In other words, it is possible for adversaries to infer information on the pre-image of LSH from the locality sensitive hash values. To enhance the security strength, we propose to apply a cryptographic hash function, like SHA1, to the locality sensitive hash values, before the targeting.

Specifically, instead of pushing down the value of $F(\mathbf{w})$ and asking the user to compare $F(\mathbf{w}) \stackrel{?}{=} F(\mathbf{x})$, the PiCoDa server can push down $h(F(\mathbf{w}))$ and $F(\cdot)$ to the user. The user computes $F(\mathbf{x})$ and tests if $h(F(\mathbf{w})) \stackrel{?}{=} h(F(\mathbf{w}))$. If the test matches, the user is potentially eligible for a coupon to redeem. If not, then due to the one way property of $h(\cdot)$, the user will learn nothing about the $F(\mathbf{w})$ from the received hash values. The non-interactive protocol between the PiCoDa server and the user is as follows. We let $\phi(\cdot)$ denote a pseudorandom function, $\mathsf{Enc}(\cdot)$ denote a semantically secure encryption function, and $\mathsf{Sig}(\cdot)$ denote some secure digital signature scheme.

1. The PiCoDa server sends down to the user:
   $h(F(\mathbf{w}))$, $\mathsf{Enc}_{\mathsf{key}}(coupon||UID||nonce||\ Sign)$, and $F(\cdot)$. Here $coupon$ denotes the actual content of coupon, UID specifies the user, $nonce$ is a fresh random number per hash value pushed down. Also $\mathsf{key} = \phi(F(\mathbf{w}))$, $Sign = \mathsf{Sig}_{vendor}(coupon||UID||nonce)$.

2. The user tests his behavioral data and checks if $h(F(\mathbf{x})) \stackrel{?}{=} h(F(\mathbf{w}))$. If yes, the user continues to plug $F(\mathbf{x})$ into $\phi(\cdot)$ to get the trapdoor key, and further open the encrypted coupon. If no match, the user learns nothing beyond the fact of his non-eligibility, due to the assumption that $\mathbf{w}$ is hard to guess.

3. During redemption, the validity of a coupon can be checked by verifying the signature, $Sign$, and the UID of the redeeming user.

Parameter Selection. Because $F(\mathbf{w})$ outputs K-bit string, we have to ensure K is sufficiently large, e.g. K = 80, such that it is not feasible for malicious users to enumerate. However, larger K value would also reduce the probability of two similar points hashing together (see Eq. (1)), due to the fact that $p > p^K$ for any $0 < p < 1$ and $K > 1$. Since the success of the eligibility test depends on the similarity of the two vectors $\mathbf{w}$ and $\mathbf{x}$, setting large K might result in less or even no successful matches.

Following the methodology in LSH community [1], to maintain the correctness of the high probability matching, one approach is to push down a set of L independent concatenated LSH functions $F_1(\cdot), \ldots, F_L(\cdot)$, and ask the user to find if any of the L hash values matches his own result. Note that the probability for the user to find any match among the L hash values is at least $1 - (1 - p^K)^L$, where p is determined by the similarity of $\mathbf{w}$ and $\mathbf{x}$ via Eq. (1). Clearly, by increasing L, we increase the value of $1-(1-p^K)^L$, and thus maintain the high probability matching for true positive of the eligibility test. By increasing K, we decrease the value of $1-(1-p^K)^L$, and thus suppress the low probability matching for the false positive. As a result, choosing large K and L amplifies the gap between the true positive and false positive of the eligibility test. However, the side-effect is the extra computation burden at the user side.

In practice, because only the seed used to randomly sample the vectors for function $F(\cdot)$ needs to be sent, the bandwidth for transmitting multiple LSH functions is not a concern. To avoid intensive computation cost for the eligibility test, the PiCoDa server can distribute the L hash values over a certain targeting time window instead of in one batch. For example, it can push down 10 different hash values on the same targeting vector $\mathbf{w}$ with 10 randomly different LSH functions every day to one user in a 2-week time window to reach the requirement of $L = 140$. If the user behavioral model matches any of those hash values, an eligible coupon is to be delivered. Otherwise, after this targeting time-window, the PiCoDa server can start a new cycle and push down hash values based on some different targeting strategy.

Remark. When the PiCoDa server pushes down $h(F_i(\mathbf{w}))$ to each user each time, where $i = 1, \ldots, L$, it must ensure that every $F_i(\mathbf{w})$ are at least have 1 bit difference. As a result, it can ensure $\mathsf{key} = \phi(F_i(\mathbf{x}))$ is only usable for that specific user with UID and the specific coupon with $nonce_i$. Since $F_i(\cdot)$ are defined by random vectors, even for the same $\mathbf{w}$, making each $F_i(\mathbf{w})$ different should not be difficult to achieve in practice. For ease of presentation, we defer the security analysis to Section 4.

3.3 Protocol 2: Privacy-preserving Interactive Coupon Targeting

As discussed in Section 3.1, there are cases in which the vendor's strategy is deterministic instead of approximate. For example, the vendor may only care about whether the user has been to a certain local retail store to distribute him a coupon. Further, such deterministic rules are usually not complicated, i.e., do not have high enough entropy. This may be the result of that vendors don't care about certain entries in strategy $\mathbf{w}$ or because in certain scenarios only a few entries matter for the targeting purposes. Thus, directly pushing down the hash values $h(\mathbf{w})$ (due to deterministic match, we don't need LSH any more) to users for local matching no longer works, as simple guessing attacks via enumeration on value of $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ become feasible.

To cover this case, we propose a design by making certain relaxations of our stringent constraints. That is, our protocol now requires users and the PiCoDa server to interact during the coupon delivery session. But we still ensure that the vendor's strategy is protected against non-eligible users, and users behavioral data is not revealed to the vendor unless they choose to redeem the coupon (if they are eligible).

Assuming the vendor only cares about m entries in $\mathbf{w}$ with index $\mathcal{I} = (i_1, i_2, \ldots, i_m)$. In the following, we adopt techniques from "Password-authenticated key agreement" [2] as a base for our protocol design. Let $\mathbb{G}$ denotes a finite cyclic group with generator g. This group could be $\mathbb{Z}_P^*$ where P is a large prime

with 1024 bits. Both g, P and hash function $h(\cdot)$ are public. The interactive protocol of PiCoDa operates as follows:

1. The PiCoDa server picks random values $r, a \in \mathbb{Z}_P^*$, computes $H_v = h(w_{i_1}||w_{i_2} \dots w_{i_m}||r)$ and $g^a \mod P$, and sends $\{\mathsf{Enc}_{H_v}(g^a), r, \mathcal{I}\}$ to the user.

2. The user picks $\{x_i\}$ vis $\mathcal{I}$ and computes $H_x = h(x_{i_1}||x_{i_2} \dots x_{i_m}||r)$. He picks a random $b \in \mathbb{Z}_P^*$, computes $g^b \mod P$, and sends $\mathsf{Enc}_{H_x}(g^b)$ to the PiCoDa server.

3. The PiCoDa server uses H_v to decrypt $\mathsf{Enc}_{H_x}(g^b)$ and gets decrypted value V. It then sends $\mathsf{Enc}_{\mathsf{V}^a}(coupon||UID||nonce_i||Sign_i)$ to the user.

4. The user uses H_x to decrypt $\mathsf{Enc}_{H_v}(g^a)$ and gets decrypted value X. He then uses X^b to decrypt $\mathsf{Enc}_{\mathsf{V}^a}(coupon||UID||nonce_i||Sign_i)$.

Note that in step (3) and (4), after decryption, we have $\mathsf{V} = g^b$ and $\mathsf{X} = g^a$ if and only if $H_v = H_x$. Otherwise, both V and X are just some indistinguishable random values. In step (4), when $H_x = H_v$, it's easy to know $\mathsf{X}^b = \mathsf{V}^a = g^{ab}$. And the eligible user gets the coupon in the final step. In the meanwhile, this eligible user knows his x_i for $i \in \mathcal{I}$ equals vendor's corresponding w_i. However, when $H_x \neq H_v$, non-eligible users still know nothing about vendor's strategy. The detailed security analysis in Section 4.

Remark. To prevent users faking their behavioral data –by colliding with eligible users who received the coupon before them–, PiCoDa server may collect commitments from user devices to their behavior vectors (e.g., $h(x_i||i||UID||nonce)$) and require them to open the commitments while redeeming a coupon (e.g., by revealing the $nonce$). Alternatively, PiCoDa server may run the first 2 steps of the above protocol with all users before executing step 3 with any of them, or hide the coupon encryption key in step 3 (e.g., by using $h(\mathsf{V}^a||nonce)$ as the encryption key) and open it (e.g., by revealing the $nonce$) to all the users at the same time. Finally, we remark that in practice, the PiCoDa server initiates this protocol with each user only once per coupon, which practically excludes the threat of coupon exploit from malicious users by limiting guessing and exhaustive search opportunities.

3.4 Further Discussion: Dealing with the Hybrid Case

We have discussed the vender's approximate strategy and deterministic strategy. What if the vendor needs both? One way to achieve that is to concatenate the two protocols. In particular, the vendor first uses the non-interactive protocol to do the LSH based targeting. The eligible users passing the test have the choice of whether to proceed to do the interactive protocol or not. This concatenation can be applied to the case of tiered coupon distribution systems, where approximate matching corresponds to loose eligibility requirements and the vendor delivers broadly targeted coupons, like $1-off for a sports retailer. When the user chooses to proceed for interactive protocol, it comes to a more personalized targeting. For example, such coupons can be for only a few users that are very important and highly loyal to the vendor. Of course, with the coupon becomes more personalized or higher-tiered, the users are willing to reveal more of themselves (when redeeming the coupon).

Instead of doing tiered coupon delivery, the vendor might be only interested in offering coupons if and only if both the approximate and the deterministic strategies have positive matches simultaneously. In this case, we can put both LSH values and the deterministic rules in the cryptographic hash $h(\cdot)$ in the step (1) of either the non-interactive protocol[1] or the interactive protocol, the remaining of the protocols follows directly. Take the interactive design for example. Let $H_v = h(w_{i_1}||w_{i_2} \dots w_{i_m}||F(\bar{\mathbf{w}})||r)$, where $\bar{\mathbf{w}}$ contains the $n - m$ remaining entries of the original $\mathbf{w}$ seeking for approximate match. Because the

[1]The combined strategy of the non-interactive protocol must have an entropy larger than 80bits.

interactive protocol does not allow users/PiCoDa server to do offline guessing/enumeration attack, we no longer need the 80-bit requirement for the LSH outputs. In other words, we only choose an appropriately small number of LSH output bits K such that we can use $L = 1$ to simplify the eligibility test. The protocol goes exactly the same as in Section 3.3 and thus is omitted.

Remark. From the ease of management point of view, the non-interactive mode of PiCoDa is easier to operate since all the targeting hash values could be pre-generated. The PiCoDa server does not even have to be always online as there are no interactions. For the interactive operation mode, the PiCoDa server needs to interact with every user per coupon delivery, which can be less scalable than the non-interactive case. However, it does give the vendor more flexibility when choosing targeting strategies. In both cases, users maintain the full control of their behavioral data until they redeem the coupons. Non-eligible users know nothing about the vendor's targeting strategy.

4 Security Analysis

4.1 User Data Privacy Protection

Non-interactive Coupon Targeting. User privacy is protected in the sense that all the eligibility matching happens at user's mobile device and no data leaves the phone before the user redeems the coupon. However, when a user decides to redeem the coupon, he or she must disclose to the PiCoDa server his or her eligibility status. In this case, the vendor can learn that $F(\mathbf{w}) = F(\mathbf{x})$, where $F(\cdot)$ is the public locality sensitive hash function.

Though we limit the privacy-preservation to the targeting process, it is worth further understanding on how much the fact $F(\mathbf{w}) = F(\mathbf{x})$ reveals about $\mathbf{x}$. Since we choose $K = 80$, which divides the whole n-dimensional space into 2^{80} subspaces. Thus, if there are enough reasonable points in the same subspace, then user's $\mathbf{x}$ can further be protected in a k-anonymity manner. If in the worst case there is only one point in the subspace, then the vendor can exactly pinpoint $\mathbf{x}$ via $\mathbf{x} = \mathbf{w}$. However, since the user knows the subspace as well, the user can also exactly pinpoint vendor's $\mathbf{w}$ from the eligibility test, which violates vendor's own protection requirement. So we argue that the vendor has enough incentives not to select small subspace so as to protect the targeting strategy $\mathbf{w}$. As a result, whenever a user finds a match, his behavioral data $\mathbf{x}$ sharing the same subspace with $\mathbf{w}$ will also be protected from that same large subspace.

Interactive Coupon Targeting. In this case, users exchange information with the PiCoDa server. But based on the security strength of "password-authenticated key agreement" [2], we still ensure that users have full control of their behavioral data. First, information uploaded to the PiCoDa server in the protocol is just some random encryption value; Second, even after the decryption, the PiCoDa server cannot tell whether the user's $\mathbf{x}$ matches the strategy $\mathbf{w}$. In other words, before the coupon redemption, the PiCoDa server or vendor learns nothing about the coupon targeting result. Thus, user data privacy is well-protected.

4.2 Vendor Protection

Non-interactive Coupon Targeting. By protecting vendor, we aim to ensure the eligibility test on user's device either reveals the fact to user that $F(\mathbf{w}) = F(\mathbf{x})$ or nothing about vendor's targeting strategy $\mathbf{w}$ except that $F(\mathbf{w}) \neq F(\mathbf{x})$. For the latter, due to our two-layered hash construction with large $K = 80$, the user knows nothing from his unmatched eligibility test. This is because reverse-engineering $h(F(\mathbf{w}))$ is computationally infeasible, assuming $\mathbf{w}$ itself is hard to guess, i.e., with 80-bit entropy.

But if there is a match, then the user knows $F(\mathbf{w}) = F(\mathbf{x})$. Using the aforementioned argument where the vendor selects a large enough subspace defined by random vectors for $F(\cdot)$, the vendor's $\mathbf{w}$ cannot be exactly pinpointed by a single user.

Interactive Coupon Targeting. In the interactive case, there are only a small number of rules or entries in the targeting strategy $\mathbf{w}$, or the vendor selectively cares a portion of entries in $\mathbf{w}$. Thus, protecting both the interesting entry index and the values can become important to vendor. Currently, our design does not

protect which entries are important in $\mathbf{w}$. Knowing this information might give the user some advantage to infer the actual values in $\mathbf{w}$, based on other context information. However, the vendor can instruct the PiCoDa server to initiate the protocol with each user per coupon only once, and thus each user only has one chance to guess the correct value in vendor's targeting strategy $\mathbf{w}$. From a practical point of view, the threat of correct guessing and other coupon exploits can be negligible. Further, following the same reasoning for user privacy protection, we can ensure that non-eligible users know nothing about the actual values in $\mathbf{w}$ from the eligibility test.

Remark. Note that neither design maintains the vendor protection against users having a match from the eligibility test. A positive matching result inevitably reveals some information about the vendor's strategy to the users. Ideally the vendor would prefer not to expose the strategy at all. To mitigate the negative effect of exposing targeting strategies to eligible users, we propose a series of alternative approaches in the next section.

4.3 Robustness

Previous discussions show that users who are originally not eligible for a coupon learn nothing beyond the failure of the eligibility test, and thus are not able to provide useful information to harm the system. However, eligible users who already get the coupons might be willing to share information of their behaviors, e.g., via blogs, or social networks to their friends. These users might give good pointers for other users to mimic the behavior and narrow down the brute-force guessing space directly on $\mathbf{x}$ for $\mathbf{w}$ or $F(\mathbf{w})$. Unfortunately, there is no perfect solution for the vendor to defeat a user that is faking his behavior. In the following, we provide a series of alternatives to address the problem. Our goal is to prevent or discourage users from arbitrarily trying different behavior $\mathbf{x}$ to maliciously collect and redistribute coupons.

Using Trusted Computing Technology. Our first approach is to rely on trusted computing technology to mitigate the concern of user's faking behavior. It can be achieved via Trusted Platform Module (TPM) [14], which offers hardware based root of trust and has already been adopted by many major laptop vendors in the market. Physically attached to a computer, the TPM chip is accessed by software from upper layers using a well-defined command set, through which, the TPM can facilitate cryptographic functionalities like hardware pseudo-random number generation, key generation, signing and encryption/decryption etc. Thus, we can use TPM's capabilities to do code attestation and verification for the device and the application software.

According to the latest work-in-progress specification version 2.0 of Mobile Trusted Module by the TCG [13], TPM is expected to be soon in place on smart phones from major phone manufacturers. Assuming users cannot temper the process running on device collecting user's behavioral data, then users have to actually conduct the behavior accordingly to get the coupon, like visiting the stores, or accumulating enough purchase records. The marginal gain of coupon can thus be easily diminished by the cost of non-eligible user's actually mimicing/conducting those possibly non-trivia behaviors.

Commitment Based Approach. We can also ask users to commit to their behavioral data $\mathbf{x}$'s periodically or before receiving coupons. In this case, they cannot arbitrarily change their behavior to maliciously collect coupons, even if they learn information by colluding with each others. In Section 3.3, we have outlined few commitment based approaches for interactive targeting of PiCoDa. In the following, we demonstrate another example via using Pedersen's commitment [10] scheme for non-interactive targeting mode of PiCoDa.

Assume both the vendor and the user agree on some group $\mathbb{G}_q$ of prime order q and two generators g, g_0 for which the discrete logarithm problem is hard. Whenever the user conducts certain behaviors, represented by some element x_i in the behavioral model $\mathbf{x}$, the user picks a random $\tau_i \in \mathbb{Z}_q$ and sends a commitment $C_{\tau_i}(x_i) = g_0^{\tau_i} g^{x_i}$ to the PiCoDa server. Here due to the randomness of τ_i, x_i is protected. Given commitments $C_{\tau_i}(x_i)$ for $i = 1, \ldots, n$, the PiCoDa server could later verify the result of vector product $\mathbf{r} \cdot \mathbf{x}$ from the LSH computation (See Section 3.2), based on the homomorphic property of the commitment

construction. Specifically, the user redeeming the coupon sends $\mathbf{r} \cdot \mathbf{x}$ together with the randomness $\{\tau_i\}$ embedded in the commitment. The PiCoDa server verifies $\prod_{i=1}^{n} C_{\tau_i}(x_i)^{r_i} = g^{\mathbf{r}\cdot\mathbf{x}} g_0^{\sum r_i \cdot \tau_i}$ and thus check if the corresponding bit of LSH output is correct.

Compared to TPM based approach, commitment based approach requires the user to send commitments to the PiCoDa server, which could be against the original motivation of a non-interactive targeting design. However, commitments do not have to be done very frequently, because $\mathbf{x}$ values are usually aggregate information over a certain amount of time. If the coupon targeting time-window is set to be 2 weeks, i.e., $\mathbf{x}$ measures the user's behavior over the past 2 weeks, then asking users to send non-revealing information to the PiCoDa server once per 2 weeks can be reasonably acceptable .

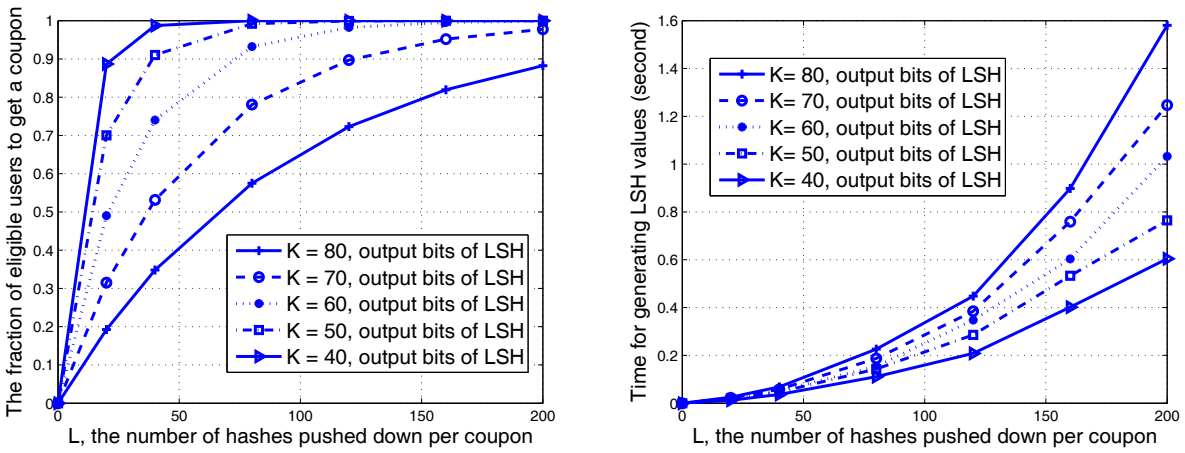
Relying on External Third Parties. In practice, we can also rely on external third parties to help prevent users from faking behaviors. The immediately available third party for the role could be the wireless carriers. The carriers keep a track of their mobile users's geographic locations all the time. One viable approach is to have vendors and wireless carriers setup some service agreement such that periodically vendors can rely on wireless carriers to verify users' location data. Thus, all the geographic related behavior can be verified by vendor. Specifically, when a user has received coupons from the proposed PiCoDa protocols and decides to redeem one, he can give the permission (e.g., request with his signature) to the vendor for verifying their geographic traces at the carrier's. Note that because the user's eligibility for the coupon is inevitably revealed to the vendor at the time of redemption (see Section 2.3), the fact that the vendor verifies the user's behavior authenticity through external third parties is not a violation of PiCoDa's design goals on user data privacy protection. Considering a large potion of elements in behavioral model $\mathbf{x}$ might be location related, users only need to prove or commit on other non-location related behaviors and save the computation and bandwidth cost. Following the same intuition, other similar third parties might include: central ad-network dealers, like Google, Yahoo! for helping verifying user's browsing behaviors. Mobile apps platform holders like Apple and Google could help verify user's app-related behaviors.

Relying on Probabilistic Matching Property. Another factor we should take into consideration for discouraging users from faking their behaviors is the probabilistic matching. By selecting appropriate parameters of K and L, the vendor can fine-tune the probability of the successful matching between $\mathbf{w}$ and $\mathbf{x}$ via LSH. For example, even if $\mathbf{w}$ and $\mathbf{x}$ are quite close with each other such that the angle $\theta(\mathbf{w}, \mathbf{x})$ normalized by π is just 0.1, choosing $K = 80$ and $L = 200$ could still leads to a successful match with probability as low as 0.043. This means for 1000 users who are potentially eligible for a coupon represented by a targeting strategy $\mathbf{w}$, the eligibility test only gives coupons to at most 43 users on average.

As mentioned in Section 3.1, having a relatively small fraction of eligible users get the coupons is not violating vendor's business interest. Therefore, for those users who originally do not have the correct behavioral data and want to learn information shared by others to try their luck, the slim rate of successful matching can be really discouraging for their motivation of collusion.

Relying on Coupon Redemption History. Each time a user redeems a coupon, the vendor knows the user's then behavioral model $\mathbf{x}$ is within a certain distance of the eligibility model $\mathbf{w}$ where $\theta(\mathbf{x}, \mathbf{w}) < d$. Over time, such relationship $\theta(\mathbf{x}, \mathbf{w}) < d$ might reveal a certain pattern. Exploring the common patterns from those relationships can help the vendor identify the inconsistencies of user's behavioral models in consecutive coupon matching/redeeming sessions over a enough long time period. Another approach is to combine the coupon redemption pattern with aforementioned commitment schemes. For example, if a user has redeemed a 5% coupon over the past three coupon delivery cycles, and then suddenly wants to redeem a coupon for "buy 1 with 1 free" that can be suspicious. The vendor could honor the coupon this time but start to request the user's behavior commitments for future eligibility verification. The more coupons a user has redeemed, the more difficult for the user to fake things.

Remark. Due to space limitation, we do not try to enumerate a comprehensive list and believe there could



(a) The eligibility fraction of targeting.

(b) The user side computation cost.

Figure 3: User eligibility and LSH generation times in the non-interactive design, for different choices of K and L.

be other options available as more research effort is put on the topic. We argue that putting together all the listed alternatives or operating them in parallel, where some of them can be overlapping, could significantly raise the bar for unfaithful users. Also, the overall effect for the proposed PiCoDa system for private coupon targeting is much better than the current simple coupon code based ecosystems, in terms of ensuring user data privacy, vendor protection as well as system robustness.

5 Performance Evaluation

We evaluate PiCoDa through simulation to validate the running times for realistic parameter values. Both mechanisms of PiCoDa are implemented in C++ on a workstation with Intel Core 2 CPU running at 3.0GHz. The OpenSSL library is used to implement cryptographic functions like SHA1 and AES etc.

In a typical targeting scenario, the vendor's targeting strategy may cover the user's behavior events from domains like page views, query search results, GPS traces, purchase history, messages, and contacts. And for the behavior events from each domain, there can be a series of measurements to be reported as features in $\mathbf{w}$ and $\mathbf{x}$. In our simulation, without loss of generality we set the dimensionality of user's behavioral model and vendor's targeting strategy $|\mathbf{w}| = |\mathbf{x}| = 30$. The cosine distance-based similarity measurement is used, and the hash based commitment scheme is included in the design to ensure that no user could fake their behavioral data. Note that we only report timing performance for the protocol data. A practical implementation will require time to transmit the coupon contents.

Non-interactive Design: In this case, we fix the cosine distance threshold between $\mathbf{x}$ and $\mathbf{w}$ at 0.985, which means the largest tolerated angle between $\mathbf{w}$ and $\mathbf{x}$ is $\theta(\mathbf{x}, \mathbf{w}) = 18.2°$. The results for different choices of K and L are shown in Fig. 3. Fig. 3-(a) shows the fraction of eligible users that receive a coupon after the eligibility test. Depending on the application, this can be fine-tuned by the vendor by setting K and L appropriately. In particular, given any fixed K ranging from 40 to 80,[2] the vendor can always find some L less than 200 such that the fraction of eligible users that do receive a coupon is more than 90%.

On the other hand, K and L cannot be set arbitrarily, as high values might increase the computational burden on the user side for each coupon targeting as shown in Fig. 3. In our simulation, to avoid the transmission of LSH function, which is defined by a set of random vectors and can be large for large K and

[2]For K less than 80, we implicitly assume a hybrid case such that deterministic rules are combined with LSH output to satisfy the 80-bit entropy requirement of the targeting strategy.

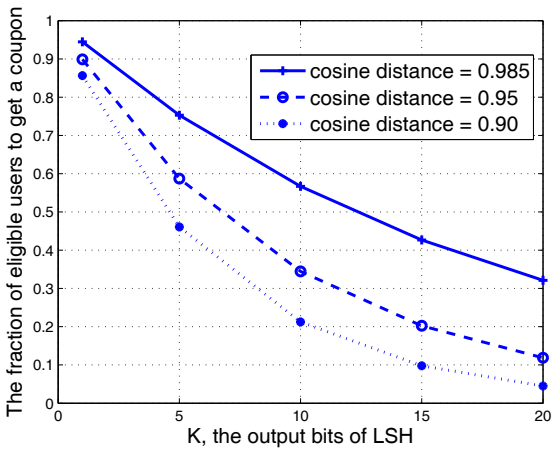


Figure 4: The eligibility fraction of interactive targeting for different choices of K.

L, only a random seed must be transmitted from the PiCoDa server to user, who then generates the LSH function on the fly. The timing result reported in Fig. 3-(b) thus involves both LSH function generation and LSH value computation. It can be seen that timing cost increases when either K or L is large. But even for the largest K and L values on the graph, the computation still requires less than 1.6 seconds. As coupon delivery does not need to happen in real time, this computational cost is likely to be acceptable in practice.

Interactive Design We also simulate the hybrid case in Section 3.4 using the interactive protocol of PiCoDa. As we no longer have the constraint on the large number output bits K of LSH, we can set $L = 1$ and choose an appropriate K value to fine-tune the acceptable accuracy with regard to different datasets. The fraction of eligible users that get a coupon for three different thresholds is shown in Fig. 4. Under those settings, i.e., $K < 20$, the computation cost for generating and evaluating LSH values is always less than 1 millisecond. Also, the three-round interaction between the PiCoDa server and the user is very efficient. Each step only involves one modular exponentiation together with AES encryption and hash operations, which takes less than 1.5 milliseconds.

Note that although our timing results are derived from simulation on a desktop machine, it is reasonable to expect that mobile devices will match this performance in the next few years, given the current trend of increasing mobile device processing power. We leave the empirical study of PiCoDa with a real dataset on mobile devices as future work.

6 Related Work

Privacy-preserving targeted behavior analysis has been explored by researchers in various forms [12, 7, 6]. Toubiana *et al.* proposed Adnostic [12], a browser extension that runs the behavioral profiling and targeting algorithm on the user browser's history database. Because the results are kept within the browser, users see ads relevant to their interests from a group of candidate ads (they suggest 20) without leaking information outside the browser. Adnostic uses homomorphic encryption and zero-knowledge proofs to allow the ad-network to correctly charge the corresponding advertisers, without seeing which ads are viewed by users (i.e., the so-called "charge per impression" model). Adnostic does not consider it a privacy breach when users reveal their ad click history. This is similar to our PiCoDa system, as we don't aim to protect user privacy when the user chooses to redeem the coupon. What differentiates PiCoDa and Adnostic is the security requirements. Adnostic only considers the user's privacy, while our PiCoDa system further ensures vendor protection, and enforces the eligibility test and coupon result validation for system robustness.

Guha *et al.* present an architecture called Privad [7], which has similar goals of Adnostic but aims to

provide better privacy guarantees of user's local data. Specifically, Privad introduces a semi-trusted *dealer* between the ad-network and user in order to anonymize the user click behavior to prevent the ad-network from identifying the user. The report of a view/click still allows the ad-network to bill the advertisers and pay the publishers accordingly. Though Privad provides better privacy protection than Adnostic, the utilized anonymization mechanism also increases the cost for both performance and the click-fraud detection. As with Adnostic, the difference between Privad and PiCoDa is that Privad does not consider vendor side protection and does not perform the eligibility check during the ad targeting.

Fredrikson *et al.*'s RePriv [6] presents another in-browser approach to perform personalization without sacrificing user privacy. Unlike Adnostic and Privad, RePriv does not hide all the user's personal information. Rather, RePriv shifts the privacy control to the user, i.e., it explicitly asks the user's consent in any transfer of sensitive local information to different service providers for personalized content. While RePriv allows a wide range of personalized web applications to exist, shifting the control of personal information transfer also raises usability concerns over frequent interruptions and the difficulty of specifying preferences about personal information dissemination. PiCoDa instead adopts a different disclosure model. In particular, PiCoDa protects user data during or after the targeting process, unless the user chooses to redeem the coupon. Other differences include the enforced eligibility test and the vendor protection in our system.

7 Concluding Remarks

In this paper, we have studied the problem of privacy-preserving coupon targeting. Our goal is to enable vendors to deliver targeted coupons to eligible mobile users without compromising user privacy and without revealing their targeting strategies. The design of PiCoDa shifts the targeting from vendor side to user side. Specifically, for different vendor targeting strategies, we have provided two targeting protocols: a non-interactive one and a three-round interactive one. Our security analysis shows how both meet the system requirements of user privacy, vendor protection, and robustness. The timing performance from our simulation with realistic parameter selections further validates the efficiency and effectiveness of PiCoDa. Given the results, we conclude that PiCoDa extends existing work on privacy-preserving targeted advertising, which only considers user privacy but ignores vendor protection. Furthermore, we hope PiCoDa will inspire other privacy-preserving targeting services in which both the vendor protection and user privacy protection are demanded.

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51:117–122, 2008.

[2] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *Proc. of EUROCRYPT*, pages 139–155, 2000.

[3] M. Bilenko and M. Richardson. Predictive client-side profiles for personalized advertising. In *Proc. of ACM SIGKDD*, 2011.

[4] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. of the 34th Annual ACM Symposium on Theory of Computing*, 2002.

[5] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. of STOC*, pages 253–262, 2004.

[6] M. Fredrikson and B. Livshits. Repriv: Re-envisioning in-browser privacy. In *Proc. of IEEE Symposium on Security and Privacy*, 2011.

[7] S. Guha, B. Cheng, and P. Francis. Privad: practical privacy in online advertising. In *Proc. of NSDI*, 2011.

[8] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Morgan Kaufmann Pub, third edition, 2011.

[9] K. Partridge and J. Begole. Activity-based advertising. In J. Müller, F. Alt, and D. Michelis, editors, *Pervasive Advertising*. Springer-Verlag, London, UK, 2011. to appear.

[10] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proc. of CRYPTO, volume 576 of LNCS*, pages 129–140, 1991.

[11] K. Sugiyama, K. Hatano, and M. Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proc. of WWW*, 2004.

[12] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy preserving targeted advertising. In *Proc. of NDSS*, 2010.

[13] Trusted Computing Group. MTM 2.0 - Trusted Computing Group. Online at `http://www.trustedcomputinggroup.org/`.

[14] Trusted Computing Group. TPM Main Specification. Online at `http://www.trustedcomputinggroup.org/`.

[15] J. Turow, J. King, C. Hoofnatle, A. Bleakley, and M. Hennessy. Americans reject tailored advertising and three activities that enable it. *Departmental Papers (ASC)*, page 137, 2009.

[16] Y. Xu, B. Zhang, Z. Chen, and K. Wang. Privacy-enhancing personalized web search. In *Proc. of the 16th International World Wide Web Conference*, 2007.

Pay As You Go

Foteini Baldimtsi[1], Gesine Hinterwälder[2], Andy Rupp[3], Anna Lysyanskaya[1], Christof Paar[2,4], Wayne P. Burleson[2]

[1] Computer Science Department, Brown University
{foteini,anna}@cs.brown.edu
[2] Department of Electrical and Computer Engineering, University of Massachusetts
{hinterwalder,burleson}@ecs.umass.edu
[3] AGT Group (R&D) GmbH, Germany
arupp@agtgermany.com
[4] Horst Görtz Institute for IT Security, Ruhr-University Bochum, Germany
christof.paar@rub.de

Abstract

Until recently, the desire for user privacy in intelligent transportation systems seemed to be at odds with the cost constraints of payment tokens. There is need for low-cost payment devices that can be produced in massive quantities while sophisticated cryptographic techniques seem to be too computationally intensive to be used in such devices. Our Pay-as-you-Go approach will demonstrate that it is possible to obtain privacy on very low-cost tokens, while retaining the benefits of useful data collection. We explore privacy-preserving payment protocols suitable for use in public transportation and show ways to efficiently implement those protocols on potential payment devices. Pay-as-you-Go is a multi-disciplinary research project, funded by NSF, that includes a diverse team of computer scientists, electrical and computer engineers and transportation engineers from Brown University and the University of Massachusetts.

Keywords: Electronic-cash, transportation payment systems, RFID security and privacy, mobile payments, refunds, anonymous-credentials

1 Introduction

Consider a large metropolitan area such as New York City. Its public transportation system is what makes the city work. But it does not work for free: the millions of passengers it serves every day must pay for the ride. Let us take a look at the underlying payment systems.

The simplest, and oldest, payment system is with actual cash, tokens, or tickets. One of its advantages is that the passengers do not leave behind an electronic trail of their comings and goings. However, it also has severe limitations: physical payments require cashiers or customized payment booths or turnstiles; it is hard to adapt the system to variable pricing or collect statistics that lead to better service, etc.

As a result, pre-paid or monthly cards (those that need to be swiped, or sometimes contactless cards) such as MetroCards in NYC and Charlie Cards in Boston have replaced the systems based on physical tokens. Contactless devices have also made paying tolls easier: systems such as E-ZPass give drivers a device that automatically pays for their highway tolls as they drive by the toll booth.

These convenient systems introduce concerns as to the privacy of their customers. Essentially, one's MetroCard or Charlie Card is a persistent identifier, and the MTA in New York, or MBTA in Boston, has the ability to locate an individual in a large metropolis, which prompts concerns among privacy advocates [31].

Additionally, these devices do not necessarily offer security for the transportation authorities either — for example, the Charlie Card was shown vulnerable to forgery by 3 MIT students doing a class project [29].

Privacy is an especially challenging problem in this context since it not only spans cryptographic theory and many engineering fields but extends into public policy areas such as environmental justice policy and sociology issues such as "fair access to all". However, in order to enable a large-scale deployment and broad acceptance of such a payment system, adequate security and privacy mechanisms are an essential requirement. Indeed, current users of FasTrak, the electronic toll collection system of California, rank "more secure technology to prevent security and privacy issues" in the top three recommendations of a recent study [26].

One may argue that giving up one's privacy is a small price to pay for such important benefits as ease and convenience, not to mention the fact that the information collected can facilitate advanced traveler information dissemination, traffic management, travel time estimation, emergency management, congestion pricing and carbon emissions control, and environmental justice assessments. However, perhaps one could get the best of both worlds: all the benefits of MetroCards without sacrificing privacy?

In theory, cryptographic techniques that make this possible exist. Electronic cash schemes (e-cash) have all the privacy benefits of actual physical cash. How can we implement them on constrained devices such as a MetroCard? How do we make them work with the same speed and convenience as non-privacy-preserving MetroCards? How do we still allow to collect the same useful information about traffic patterns, even while preserving the privacy of individuals?

This paper is about the Pay-As-You-Go (PAYG) project, which seeks to answer these questions. Our goal is to bridge the gap between theoretical constructions and practical implementation on RFID devices. Our starting point as far as crypto is concerned is state-of-the-art electronic cash schemes; we want to make them as efficient as possible, in part by customizing them to the scenario at hand. On the other end, our starting point is state-of-the-art RFID devices; we want to achieve highly efficient implementations of e-cash that would be appropriate for this scenario. Working from both ends of the problem, we will obtain a solution that offers speed and convenience on the one hand, and cryptographic guarantees of security and privacy on the other. By incorporating additional cryptographic techniques, we can derive additional benefits, such as variable pricing and privacy-preserving data collection.

Our results so far are promising. On the crypto end, we want (1) provable security and (2) high efficiency. The most efficient electronic cash scheme proposed in the literature is due to Brands [11]; unfortunately we have shown that known techniques do not allow to prove its security [4] making it inadequate for our purposes. Luckily, a scheme due to Abe [1] offers comparable efficiency and provable security at the same time. However, Abe's scheme doesn't allow the encoding of user's attributes (such as age, address, etc.) which is essential for the transportation setting. Encoding user's attributes in the coins allows us to implement additional features in our system such as variable pricing (e.g. reduced fare for senior customers) and privacy-preserving data collection. We are able to create a new scheme based on Abe's that allows the encoding of user's attributes in the coins withdrawn while being efficient and provably secure [3]. We further show how to make variable-priced schemes as efficient as fixed-priced schemes [28] by using the idea of pre-payment with refunds. We give more details in Section 2.

On the implementation end, one challenge is imposed by the time constraints of transportation payment systems. To avoid congestion in front of turnstiles, a payment transaction should be completed in 200 to 300 ms. As we will see, this is a considerable challenge given the computational complexity of advanced payment protocols. The efficiency of withdrawing money, i.e., for charging a payment token, is less critical, but should also not take longer than a few seconds. Another set of challenges are related to the payment token itself. First, it should be based on inexpensive hardware due to the potentially very high volume and the need to replace payment cards frequently. Secondly, it should be able to communicate and work contactlessly and without battery. These two conditions are in conflict with the need to run very complex cryptographic operations. We demonstrate that through optimized implementation techniques it is

possible to realize advanced payment protocols even on RFID tokens with extremely limited computational resources. Such hardware platforms are close approximations of low-cost platforms that will be used in future payment tokens. Section 3 gives more details on the implementation techniques.

Thus, we are on the cusp of having RFID devices implementing cryptographically secure electronic cash. This opens up other application areas for which, up until now, cryptography was considered prohibitively inefficient.

1.1 Related Work

Heydt-Benjamin et al. [18] were the first to mention integrated transportation payment systems as an area of interest for cryptographic research and proposed the use of recent advances in anonymous credentials and e-cash systems for electronic payments in public transportation systems. Sadeghi, Visconti, and Wachsmann presented an efficient RFID-based e-ticket scheme for transit applications [30]. However, they assumed the existence of external trusted devices and their system did not protect privacy against a prying transportation authority but only against prying outsiders.

In 2009, Blass et al. [8] proposed an offline, pre-paid, "privacy-preserving" payment system for transit applications. The most remarkable feature of this approach is that the whole system solely relies on a 128-bit hash function (and lots of precomputed data: 18 TB stored at the backend and 1 GB per reader). However, again, a user's privacy in their system was not protected from the entity responsible for issuing transportation tokens.

Popa, Balakrishnan, and Blumberg [24] proposed a privacy-preserving payment system for location-based services such as mileage-based toll collection, insurance pricing based on driver behavior, etc. In 2010, Balasch et al. [2] presented PrETP which is a location-private toll-collecting system where the on-board payment units can prove that they use genuine data and perform correct operations while disclosing the minimum amount of location data. In 2011 Meiklejohn et al. [19] presented another location-private system for toll collection that was based on PrETP but fixed a serious problem that PrETP was vulnerable to colluding free-loading users. Their system, called Milo, doesn't reveal any information, even when drivers misbehave and collude.

All these schemes were developed for a scenario where users subscribe for a service and pay by the end of a billing period. In the transit scenario, we cannot assume that each user has access to a trusted PC to settle accounts: an untrusted vending machine is more realistic. PAYG therefore choses the e-cash route achieving real-time, secure payments that also guarantee location privacy.

2 Electronic Cash

E-cash is an electronic version of physical cash, which provides at least as much anonymity and security as physical cash transactions. A typical e-cash scheme has three types of players: the Bank $\mathcal{B}$, Users $\mathcal{U}$ and Merchants $\mathcal{M}$ who are involved in the following procedures: (a) *setup* where the public and private keys of $\mathcal{B},\mathcal{U}$ and $\mathcal{M}$ are established; (b) *account opening* during which $\mathcal{U}$s and $\mathcal{M}$s register at the bank; (c) *withdrawal* where users obtain e-coins from the bank; (d) *spending* where users submit coins to merchants in exchange for goods/services; (e) *deposit* where merchants deposit coins back to their bank accounts; (f) other protocols to identify malicious behavior.

In the electronic world, coins are simply digital data with unique serial numbers. So, in order to prevent users from creating coins by themselves we require the bank to digitally sign the coins. Regular digital signatures are not good enough for the e-cash setting. The problem is that anonymity is violated since the bank is able to recognize the signatures that were issued during withdrawal when the coins are deposited and thus trace how the users spend their coins. To circumvent this problem, we use a special type of digital

signatures called *blind signatures*. The idea is that the user can obtain the bank's signature on some message without the bank getting any information about the message being signed. This seems to solve the problem of users creating coins themselves, but what if a user simply tries to spend the same coin more than once? This could be easily solved, if we requested the bank to be on-line all the time (so that it could simply check if a coin being spend was used before). However, this doesn't seem very practical especially for the transportation setting. It turns out that security can be provided even in the off-line scenario. In that case we need to somehow encode the user's identity in the coins he withdraws so that if he tries to double spend a coin, the bank will be able to identify him.

In 1982, Chaum [14] came up with the idea of an e-cash system that allows anonymous, unlikable payments, secure against double spending in the off-line setting. Following Chaum's paradigm many schemes were proposed [11, 21, 17, 15, 6, 12, 13]. The scheme due to Brands [11], which is currently implemented by Microsoft in their U-Prove project [22], has the most efficient protocol when it comes to spending an e-coin and initially seemed like a good candidate scheme for the PAYG project. However, no proof of security has been given for his scheme. First, the best double-spending guarantee known, suggested by Cramer et al. [16], holds under the knowledge of exponent assumption (KEA) which is a very strong assumption (we do not want to base the security of cryptographic schemes on such strong assumptions). Second and more importantly, we showed that Brands blind signature (at the heart of his e-cash scheme) cannot be proven *unforgeable* using any of the currently known techniques [4]. Our result is even more general: we essentially ruled out all known approaches to proving security of a broader class of blind signatures ("generalized blind schnorr signatures") in the random oracle model.

We were able to resolve the second issue by providing a modification of Brands blind signature, which we prove to be unforgeable. The idea was to use Pointcheval and Stern's [23] suggestion and associate more than one secret keys to the same public key. So, in our modification of Brands, the signer (Bank) has a public key of the form $H = G_1^{w_1} G_2^{w_2}$ where w_1, w_2 the secret keys (refer to our paper [4] for the complete protocol and the security proof). However, similar to the original Brands [11], it is still an open problem whether provable guarantees against double-spending can be given for this modification.

As mentioned in the introduction we need schemes that provide (1) provable security and (2) high efficiency. Security in transactions plays an important role for the PAYG project and we are going to rule out any construction that doesn't provide a formal cryptographic proof of security (such as Brands). The scheme due to Abe [1] is provably secure and efficient enough for the PAYG requirements. It's drawback however, is that it doesn't allow the encoding of user's attributes in the coins/tokens withdrawn. This means that features like variable pricing and privacy-preserving data collection cannot be implemented with it. We gave a new e-cash scheme, based on the one due to Abe, which is provably secure, efficient and at the same time allows the encoding of user's attributes [3]. In our new e-cash the user commits to a set of attributes during the account opening phase, those attributes are later encoded in the coins he withdraws from the bank and then, during the spending phase, the user can choose to reveal a subset of those attributes when needed.

In Table 1 we provide a comparison of the above mentioned e-cash schemes. Compact e-cash [12] is another famous scheme that satisfies all the security requirements, but is significantly more expensive during the withdrawal and more importantly the spending phase which makes it difficult to be used in the PAYG without some modifications.

Even very efficient e-cash schemes require fairly large storage space for coins, and the protocols for withdrawing or spending a coin are computationally expensive. Hence, it is beneficial to limit the amount of coins that a user has to spend to execute a payment, having to spend only a single coin to make a payment would be ideal. In transportation payment systems, this conflicts with the necessity of allowing flexible prices. Fares should not be flat but arbitrary (and adjustable) monetary amounts. Setting the denomination of a coin to be one cent certainly allows for flexible pricing but users would need plenty of them to pay for a trip. Setting the face value to two dollars reduces the number of required coins per trip but severely restricts the system of fares. If we wanted to do a tradeoff and have e-coins for different monetary values we

	Brands [11]		Brands modification [4]		Abe [1]		New E-cash [3]		Compact E-cash [12]			
									RSA		Pairings	
Efficiency	B/M[1]	U[2]	B/M	U	B/M	U	B/M	U	B/M	U	B/M	U
Withdrawal[3]	2	13	4	19	6	12	7	13	10	8	15	14p
Spend	7	0	9	0	11	1	11	1	NC[5]		5+6p	9+6p
E-coin size	6 elem.		9 elem.		9 elem.		9 elem.		~ 0		~ 0	
Blindness	✓		✓		✓		✓		✓		✓	
Unforgeability	✗[?]		✓		✓		✓		✓		✓	
DS[4] **security**	✗[?]		✗[?]		✓		✓		✓		✓	
Attributes	✓		✓		✗		✓		✓		✓	

[1] Bank/Merchant, [2] User
[3] In number of exponentiations, [4] Double Spending, [5] Non comparable

Table 1: Comparison of e-cash schemes

would need to deal with overpayments and change in a privacy-preserving way. This is especially difficult in transportation payment systems where usually bank and merchants are the same entity.

To circumvent this problem we designed a payment system, called P4R [28] (**P**rivacy-**P**reserving **P**re-**P**ayments with **R**efunds), that is based on the concept of pre-payments with refunds: users deposit money to obtain a bundle of coins. When the payment is less than the value of the coin, the user obtains a refund. The system allows to aggregate these refunds in a single token, thereby saving memory and increasing privacy. P4R can be constructed from any anonymous e-cash/credential scheme which can be modified in a way that coins/credentials can be shown twice without revealing the ID of a user (e.g., Brands' scheme, or our new e-cash scheme [3]). In the next section we give more details on how P4R works.

2.1 P4R Payment System

As mentioned above P4R is not a typical e-cash scheme but a pre-payment scheme with refunds. It is composed of three subsystems the Trip Authorization Token (TAT), the Refund Calculation Token (RCT), and the Refund Token (RT) system. The TAT system is an offline system. Here ticket vending machines play the role of the "bank" issuing TATs and (offline) readers at the entry turnstiles play the role of a "merchant" where tokens can be spent. The RT system is an online system. Here roles are reversed compared to the TAT system, i.e., readers at the exit turnstiles issue refunds and the (online) vending machines receive the tokens and disburse the users. Some details of the different subsystems are given below.

A TAT (aka ticket) is a credential that authorizes a user to make exactly one trip in the transportation system. A user initially makes a deposit at a vending machine to obtain a number of TATs where the cost of a TAT equals the cost of the most expensive trip. Of course, to reduce the deposit for a TAT it is also possible to have different types of TATs for different sections or zones of the transportation system. The withdrawal of a TAT is done in a blind fashion such that a later use cannot be linked. The ID of a user (e.g., a credit card number) is encoded in each TAT to prevent a repeated use for entering the transportation system (double-spending detection). At the beginning of a ride a user presents an unused TAT to the reader at the turnstile at which he wants to enter the system. If it is valid and the user can show (using a zero-knowledge proof) that he knows the ID encoded in the TAT, access to the transportation system is granted.

At the end of the trip when the user leaves the system the actual fare is determined at the exit turnstile. This is done as follows: When entering the system a user also receives an RCT (aka stamped ticket), which

contains a MAC on the TAT, the date and time, as well as on the ID of the reader. When he leaves the system he presents this RCT to the exit turnstile reader, which calculates the trip cost based on this information. To obtain a refund the user also provides a blinded version of his RT (blank RT tokens are available from the vending machines) to the reader. To prevent a user from re-using an RCT and thus claiming the same refund several times in a row, the idea is to bind an RCT to the TAT, which has just been used to enter the system, and force him to again prove the knowledge of the ID encoded into this TAT when he leaves the system. An RT is re-used to add up several refunds instead of having a separate RT per refund. This saves memory and increases the privacy of a user. At some point the user may decide to cash the collected refund or to buy new TATs using this money. To do this, the user presents his RT to the vending machine, which redeems the RT, if this token is not already marked as cashed in the central database. This refund subsystem can be implemented based on a new variant of Boneh-Lynn-Shacham signatures [9] or a variant of RSA signatures [27].

As for security, we can show that in P4R it is infeasible for malicious users to receive reimbursements, which exceed the overall deposit for TATs minus the overall fare of their trips. Note that this also covers fare evasion. In other words, the transportation authority does not lose money. With respect to privacy, we can show that any two users obtaining the same total refund amount cannot be distinguished. Furthermore, we argue that usually many sequences of trips should result in the same refund sum: The number of sequences actually equals the number of *integer partitions* of the refund sum. Hence, we are interested in the number $p_S(n)$ of partitions of integers n, where parts are restricted to come from a certain set S.

However, one should also be aware of the limits of this approach: In practice, we cannot guarantee that during a certain period of time many such sequences actually appear in the records of the transportation authority. For instance, it could happen that since the issue date of the refund token nobody but the owner of the token did trips that resulted in a particular refund amount (though in theory many sequences lead to this amount). Moreover, there might be a lot of different sequences of single trips leading to the same refund sum but one trip in all these sequences is the same. In this case, we know that a user redeeming this refund amount must have taken exactly this trip. Hence, the exact level of location privacy provided by our system depends on the frequency of individual trips and the set of single refund values, i.e., characteristics of the transportation system and user behavior. Nevertheless, we believe that for real-world transportation systems the limits described above are no real issues. In fact, a transportation authority could publish the set of possible trips and the corresponding refund values, which would allow to (partly) check the existence of such problems.

In summary, we ruled out Brands' e-cash, because of insufficient provable security guarantees [4]. We gave the first provably secure construction of e-cash with attributes, whose efficiency is comparable to that of Brands' e-cash [3], and showed how to construct P4R from e-cash with attributes [28].

3 Implementation

For the implementation, the PAYG project focuses on the user side, i.e. the payment devices. The turnstiles can be equipped with powerful hardware or connected to a back-end system, and hence do not represent an obstacle for the efficient execution of complex algorithms.

To keep the overall payment execution time low, payment devices that do not have to be physically connected to a machine are preferable, lending itself to the use of RF-communicating devices. As such, modern smartphones support NFC-technology, a standard that supports short range communication via RF-signaling. Additionally NFC-phones are equipped with powerful CPUs. However, it cannot be assumed that every customer possesses an NFC-capable smartphone. Furthermore, relying solely on battery powered devices is not desirable for a host of reasons, as for example cost and lifetime. Ideally the transportation authority would provide user devices for free or at least at a very low price. Passive RFID devices are the

emerging standard for low-cost tokens and are highly attractive for this application domain. However, they are also severely constrained in terms of computational capabilities and memory. Contactless smartcards allow short range communication, conform to NFC-technology, which enables the easy design of a hybrid system. Nevertheless for some setups, medium range communication, say over a distance of at least 1 m, is highly attractive. For example it can enable multimodal payment systems, including payments for toll roads. Pay-as-you-Go does not limit its investigations to one of those devices, but investigates advantages and disadvantages of the different device types.

Privacy-preserving payment protocols are based on public key cryptography, which requires the execution of computationally expensive algorithms. This conflicts with the need for very inexpensive payment devices that use only limited power. Modern contact-less smartcards provide hardware accelerators to support certain standard cryptographic protocols. But the usage of those hardware accelerators is stiff ([5], [7]), and an implementation of e-cash schemes on such devices leads to very long execution times.

A medium range passively powered RFID device that can be freely programmed is the Moo, a computational RFID chip designed at the University of Massachusetts Amherst [33]. Even though it is not a commercial product, it is a good representative for an extremely low-cost, medium-range payment token, which could be used in future transportation payment systems. This RFID tag is equipped with an MSP430F2618 microcontroller, a low-power microcontroller developed by Texas Instruments. The microcontroller uses a 16-bit RISC architecture. It has 8 KB RAM and 116 KB flash memory. Based on market prices of some contactless smartcards, it can be assumed that if mass-produced, the price for a Moo will be in the range of a few dollars. We implemented **Brands** and **Brands modif** from Table 1 for this device.

We base the implementation of both schemes on elliptic curve cryptography (ECC). ECC is especially suitable in very constrained environments. In comparison to other public-key schemes, as for example discrete logarithm (DL) based schemes, ECC requires much shorter operand lengths to achieve the same level of security. As such, a 160-bit curve offers the same level of security as a 1024-bit DL-based scheme. Nevertheless, even though ECC has a lower bit complexity than DL, sophisticated implementation techniques are required in order to achieve acceptable execution times of the payment schemes on devices as the Moo. We conclude from [10] that a 160-bit elliptic curve presents sufficient security for a micro-payment system. We chose to implement the schemes based on the curve that is specified as secp160r1 in [25]. This curve is based on a special prime that allows for a very efficient implementation of the reduction step in the underlying prime field.

The most time critical operation, and hence the function that dominates the execution time of the different protocols, is the point multiplication, which is the equivalent to an exponentiation in DL-based schemes. We used the Montgomery algorithm for point multiplication [20] to implement this function. To give an idea of the computational complexity: The Montgomery ladder [20] requires 160 point additions and 160 point doublings, which when using Jacobian coordinates need $12M + 4S$ (where M stands for modular multiplication and S stands for modular squaring) and $4M + 6S$, respectively. On a 16-bit microcontroller one element in the underlying 160-bit finite field is represented as an array of ten integers. Thus using operand-scanning form, a modular multiplication requires 100 and a modular squaring 55 integer multiplications. Hence $344\,000$ integer multiplications are necessary to execute a single point multiplication. Especially useful for this implementation is the multiply-and-accumulate instruction of the MSP430, which supports a 16x16-bit unsigned integer multiplication and accumulation of the 32-bit results [32]. In our implementation the execution of one point multiplication takes 6.8 million cycles.

Table 2 summarizes the token-side execution times of the payment scheme for the withdrawal and spending phase. The user authentication step during withdrawal, where the user proves ownership of his account, is not included. This authentication step has to be executed once, before one or several coins can be withdrawn. The execution times are presented for two frequencies of the token CPU, 4 Mhz and 16 Mhz. The microcontroller that is integrated on the Moo operates at a frequency of 4 Mhz. However, it can be operated at a maximum frequency of 16 Mhz.

We note that spending is considered the time-sensitive operation as this happens during the actual transportation, e.g., at a turnstile during rush hour. The achieved spending timings in the range of 10 ms are fully sufficient even for high throughput transportation situations. The withdrawal of coins could be done at home, where the user connects the payment token to his personal computer, and leaves it there for the charging period. In that case the withdrawal would not be very time-critical. Yet, there are several advantages of the withdrawal taking place at charging stations located near the entrance points of the public transportation system. In that case, the withdrawal (even of several coins) needs to be executable in a couple of seconds. There are approaches to improving the performance of the payment token. An interesting one is that during charging the payment device could be connected to the charging station, which supplies the passive tag with sufficient energy to clock it at higher frequencies, or to power additional hardware accelerators.

	Brands	**Brands Modif.**
Cycle count Withdrawal of a coin[1]	83	125
Cycle count Spending of a coin[1]	0.052	0.053
Execution time Withdrawal of a coin @16 MHz[3]	5	8
Execution time Spending of a coin @16 MHz[3]	0.0033	0.0033
Execution time Withdrawal of a coin @4 MHz[3]	21	31
Execution time Spending of a coin @4 MHz[3]	0.013	0.013
Code Size[2]	16570	17030

[1] in million cycles, [2] in bytes
[3] in seconds

Table 2: Timing results of user side protocol implementation

4 Conclusions and Future Work

The Pay-as-you-Go project has three main goals: (1) develop novel cryptographic algorithms for the transportation setting whose efficiency is suitable, and that allow for collecting useful data and yet guarantee privacy at the same time, (2) explore emerging hardware architectures and software for performing modern cryptographic operations on low-cost devices, and (3) test human factors and performance of privacy-preserving payment systems under realistic conditions of emerging transportation payment systems that must balance security and privacy with cost with usability. Our results so far include ruling out Brands e-cash scheme due to insufficient security guarantees, constructing new efficient and provable secure algorithms for the transportation setting that allow the encoding of user's attributes and can be used to construct a pre-payment with refunds (P4R) and implementing part of them. The next steps of the project include, in the crypto-end, the proposal of some security guarantees for the modified Brands protocol against double spending and exploring possible constructions that also allow private data collection. On the implementation end, the next steps will be the implementation of our new e-cash scheme with P4R as well as comparing different potential contactless payment devices.

5 Aknowledgements

This work was supported by NSF grant 0964379.

References

[1] M. Abe. A secure three-move blind signature scheme for polynomially many signatures. In *EUROCRYPT'01*.

[2] J. Balasch, A. Rial, C. Troncoso, B. Preneel, I. Verbauwhede, and C. Geuens. Pretp: privacy-preserving electronic toll pricing. USENIX Security'10.

[3] F. Baldimtsi and A. Lysyanskaya. Anonymous credentials light. Cryptology ePrint Archive, Report 2012/352, 2012. http://eprint.iacr.org/.

[4] F. Baldimtsi and A. Lysyanskaya. On the security of one-witness blind signature schemes. Cryptology ePrint Archive, Report 2012/197, 2012. http://eprint.iacr.org/.

[5] L. Batina, J.-H. Hoepman, B. Jacobs, W. Mostowski, and P. Vullers. Developing efficient blinded attribute certificates on smart cards via pairings. In *CARDIS'10*.

[6] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. Compact e-cash and simulatable vrfs revisited.

[7] P. Bichsel, J. Camenisch, T. Groß, and V. Shoup. Anonymous credentials on a standard java card. In *ACM CCS'09*.

[8] E.-O. Blass, A. Kurmus, R. Molva, and T. Strufe. Psp: private and secure payment with rfid. In *WPES'09*.

[9] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4):297–319, 2004.

[10] J. W. Bos, M. E. Kaihara, T. Kleinjung, A. K. Lenstra, and P. L. Montgomery. On the security of 1024-bit rsa and 160-bit elliptic curve cryptography. 2009.

[11] S. Brands. Untraceable off-line cash in wallets with observers. In *CRYPTO'93*.

[12] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *EUROCRYPT'05*.

[13] A. Chan, Y. Frankel, and Y. Tsiounis. Easy come - easy go divisible cash. In *EUROCRYPT'98*.

[14] D. Chaum. Blind signatures for untraceable payment. In *Crypto'82*.

[15] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO '88*.

[16] R. Cramer, I. Damgard, and J. B. Nielsen. On electronic payment systems. In *http://www.daimi.au.dk/ ivan/ecash.pdf*, 2010.

[17] N. Ferguson. Single term off-line coins. In *EUROCRYPT '93*.

[18] T. S. Heydt-benjamin, H. jin Chae, B. Defend, and K. Fu. Privacy for public transportation. In *PETs 2006*.

[19] S. Meiklejohn, K. Mowery, S. Checkoway, and H. Shacham. The phantom tollbooth: privacy-preserving electronic toll collection in the presence of driver collusion. In *USENIX 2011*.

[20] P. L. Montgomery. Speeding the pollard and elliptic curve methods of factorization. In *Mathematics of Computation*, 1987.

[21] T. Okamoto and K. Ohta. Universal electronic cash. In *CRYPTO'91*.

[22] C. Paquin. U-prove cryptographic specification v1.1. In *Microsoft Technical Report*, 2011.

[23] D. Pointcheval and J. Stern. Provably secure blind signature schemes. In *Asiacrypt'96*.

[24] R. A. Popa, H. Balakrishnan, and A. J. Blumberg. Vpriv: protecting privacy in location-based vehicular services. In *USENIX security symposium*, 2009.

[25] C. Research. Sec 2: Recommended elliptic curve domain parameters. 2000.

[26] P. F. Riley. The tolls of privacy: An underestimated roadblock for electronic toll collection usage. *Computer Law & Security Review*, 24(6):521 – 528, 2008.

[27] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[28] A. Rupp, M. Fischlin, and C. Paar. Pre-payments with refunds: A new cryptographic approach to privacy-preserving payments for transportation systems. In *Manuscript*, 2012.

[29] R. Russel, Z. Anderson, and A. Chiesa. Anatomy of a subway hack. In *DefCon*, 2008.

[30] A.-R. Sadeghi, I. Visconti, and C. Wachsmann. User privacy in transport systems based on rfid e-tickets. In *PiLBA*, 2008.

[31] B. Saderson. E-z pass could take toll on right to privacy. In *New York Post*, 1996.

[32] Texas Instruments Incorporated. MSP430x2xx Family User's Guide (Rev. H). 2011.

[33] H. Zhang, J. Gummeson, B. Ransford, and K. Fu. Moo: A batteryless computational rfid and sensing platform. 2011.

Perspectives on Academic Impact from Inside the Federal Trade Commission

Michael Brennan

Drexel University, Computer Science Department,
3141 Chestnut Street, Philadelphia, Pennsylvania

Abstract

The privacy field of academic computer science produces a large body of work with potential for impact on the lives of end users. An important avenue for this impact, and one of the most overlooked, is the Federal Trade Commission (FTC). This paper summarizes the impact that academic research has had in recent years on the policies, actions and opinions of the FTC and proposes approaches for increasing the impact of academic work. It also analyzes contributions of the non-academic research communities, argues for greater acceptance of action by the FTC and similar agencies as an impact measurement in academia, and details roadblocks that limit bridge-building between government and academia. The author has been is a staff technologist for the FTC's Division of Privacy and Identity Protection.

1 Disclaimer

The author has worked for the Federal Trade Commission since August of 2010. This paper reflects the personal opinions of the author and not the official opinions or positions of the Commission or the United States.

2 Introduction

How does academic privacy research result in real-world privacy protection or enhancement? This question of impact is not just an issue of broader research impact but a fundamental scientific research question in the field of privacy. A major means to accomplish impact is through existing regulation and oversight institutions. The call for papers for the Workshop on Privacy in Electronic Society points this out explicitly: "The need for privacy-aware policies, regulations, and techniques has been widely recognized" [7].

One approach towards achieving privacy is through action at the Federal Trade Commission (FTC or the Commission). The FTC is the only federal regulatory agency in the United States with jurisdiction over general privacy concerns. It has the authority to take action against companies with unfair or deceptive privacy practices through a variety of means including investigations resulting in orders that seek to correct the issue and set precedent for best practices. The FTC also has impact beyond the United States through action concerning multinational companies and cooperation with international data protection agencies[1].

The FTC has taken a number of influential actions in recent years to address allegedly deceptive and unfair privacy practices. These include consent orders against search and advertising giant Google [20], social media service Twitter [1], and the ad network Chitika [5]. Results of these orders include mandating independent privacy audits over the course of 20 years for Google and 10 years for Twitter, and requiring

[1] FTC Office of International Affairs `www.ftc.gov/oia/`

a link to opt out of targeted advertising in ads served by Chitika. The Commission has also issued an influential staff report proposing a new framework for protecting consumer privacy [9], representatives of the Commission have spoken on privacy issues in a number of major forums [23, 24] including the US Congress, and the Commission has regularly issued consumer education reports concerning technical privacy and security threats [10].

Identifying violations, anticipating harmful business practices, and protecting consumer privacy in technical domains takes a large amount of resources. The FTC does not have a dedicated technical research arm for privacy matters. A number of the actions and investigations carried out by the FTC are initiated by work in the academic and independent privacy and security research communities. Unfortunately I do not believe these communities realize the extent to which the FTC listens and relies upon their work. I base this assertion on my personal experience as a technologist at the Division of Privacy and Identity Protection. I have interacted with many researchers on current technical privacy issues and more often than not I am met with surprise that the FTC is knowledgeable and interested in these issues.

The primary impediment to research communities understanding and recognizing the impact of their work at the FTC is the fact that all ongoing investigations are nonpublic. The only point at which an investigation becomes public is in the case where a matter is settled or publicly closed. Many more issues never see the light of public scrutiny because they are closed or resolved privately. While unfortunate, this is a necessary aspect of the work of the FTC. An investigation into privacy-related events or actions of a company is not an implicit assertion of that company's guilt. It is my opinion, based on my experience, that the FTC recognizes the market impact of an investigation and does not wish to unjustly punish business based on an accusation.

Another issue is the lack of a consistent bridge between technical academia, independent researchers and the FTC. The FTC is a legal organization and, as such, builds most of its bridges with legal academia. The FTC has not maintained a significant presence at technical academic conferences. It makes sense that there is a lack of recognition among academics that their work can – and does – significantly affect the policy of the FTC. Only recently with the introduction of Ed Felten as Chief Technologist has the Commission began to build a bigger bridge to the technical sectors of academia.

3 The Federal Trade Commission

The Commission was established by the Federal Trade Commission Act of 1914. While the original goal was to regulate unfair competition, Congress has expanded the Commission 's powers to include a number of consumer protection issues. The FTC is divided into the Bureaus of Competition, Economics and Consumer Protection. The FTC is presently the only federal agency general jurisdiction over unfair and deceptive privacy practices in the United States. Much of this is done through the Bureau of Consumer Protection and its Division of Privacy and Identity Protection.

3.1 Division of Privacy & Identity Protection

The Division of Privacy and Identity Protection (DPIP) is the most recent division to be added to the Bureau of Consumer Protection. In addition to Section 5 of the FTC Act, DPIP also enforces the Fair Credit Report Act (FCRA) and the Gramm-Leach-Bliley Act. The FCRA gives consumers the right to know what information credit bureaus and consumer reporting agencies have on them. The Gramm-Leach-Bliley Act requires financial institutions to maintain the security and confidentiality of customer information.

Action at DPIP does not only take the form of investigations into illegal business practices. Consumer and business education, legislative analysis, white papers, policy outreach, and public speeches are just some examples of the tools at the disposal of DPIP and the FTC in general.

3.2 Mandate and Authority

A significant number of DPIP investigative actions regarding privacy are based on section 5 of the FTC Act. Specifically the statute that states "unfair methods of competition in or affecting commerce, and unfair or deceptive acts or practices in or affecting commerce, are hereby declared unlawful" [12].

Deception can take many forms but it is generally considered to be a misleading representation or an omission that is counter to a reasonable consumer's expectations. In the privacy or data security context, this might be an undisclosed practice that is counter to the representations made to consumers. One example is the case of Chitika where a mechanism was presented to consumers for opting out of behaviorally targeted advertising but the opt-out cookie expired 10 days after receipt [5]. Poor data security practices despite a representation that the privacy and security of a consumer will be protected may also be deceptive, as in the case of Twitter's failure to protect the personal information of consumers despite stating that they took a number of measures to protect this it from unauthorized access [4].

Unfairness generally means a demonstration of actual harm to consumers, not speculative harm. This may come in the form of financial harm, such as tricking consumers into purchasing fraudulent or unwanted goods or services, or, in some cases, emotional harm such as that which might occur due to exposure of sensitive health information.

4 Academic Impact at the FTC

The work performed by the FTC is greatly facilitated by input from academic research communities, journalists, and independent researchers. The role of academia is particularly important. Computer science publications lend facts and credibility to FTC action. This is often overlooked in academic circles despite the significant real world impact that takes place as a result of some publications. This section highlights some of the important work by academics and independent researchers that have informed FTC actions.

4.1 Demonstrated Impact in Investigations

Investigations within the FTC can become public in a few ways but the most common are through complaints that lead to settlement or litigation and through closing letters. In March of 2010 the FTC made public its investigation into Netflix by way of a closing letter [15]. The letter states the concerns of the FTC about the risk of re-identifying anonymized data in the second Netflix Prize data set. In addition to a citation of Narayanan and Schmatikov's work on the subject, the first such citation of an academic computer science paper in any closing letter by the commission, the FTC also cited work by Paul Ohm titled "Broken Promises of Privacy: Responding to the Surprising Failure of Anonymization" [19]. While Ohm's work was published in a legal venue, the UCLA Law Review, it is reliant on a bed of research by computer scientists that demonstrates the widespread use of ineffective anonymization techniques.

Direct references to academic research in documentation from the Commission is not the only way to identify impact. Oftentimes investigations are spurred by well documented complaints by organizations such as the Electronic Fronteir Foundation (EFF) and Electronic Privacy Information Center (EPIC). The recent investigation and subsequent settlement between the FTC and Facebook [6] followed a complaint by a coalition of consumer groups led by EPIC [2, 21]. The complaint cites a number of academic work, such as Arvind Narayan and Vitaly Schmatikov's work, "De-anonymizing Social Networks" [17] and writing by Ed Felten of Princeton University [11].

4.2 Impact in the 2010 Privacy Report

In December of 2010, the FTC issued a report entitled, "Protecting Consumer Privacy in an Era of Rapid Change: A Proposed Framework for Businesses and Policy Makers" [9]. It proposes a framework for balancing privacy with innovation based on consumer information. It reflects the FTC staff's position regarding current consumer privacy concerns.

The privacy report has had substantial impact in the area of consumer privacy. Over 450 public comments were submitted by multinational corporations, special interest groups and individual consumers. Comments by Microsoft, Google, AT&T, and Facebook illustrate the influence of the report and the FTC in general. Many of these comments note the impact that the FTC has had in recent years, including Google statement that "The Commission has been instrumental in discouraging practices that are harmful or deceptive and undermine user trust... Google supports the Commission's promotion of a framework to guide the privacy efforts of all commercial entities, coupled with continued consumer education and enforcement against bad practices." [3].

Every citation and footnote in the report indicates serious discussion over a specific issue and, in the case of research papers, reflects substantial impact on the policy being outlined in the report. A citation in the report indicates strong influence within the Commission. This is unlike many academic papers where a citation may simply be a passing acknowledgement of previous work. The end goal of research, especially with the field of privacy, is to effect change in the world. In those terms a citation in an FTC order or major report is near the highest form of real world impact and should thus be considered a high form of academic impact.

The report also received widespread media coverage including a space at the top of the front page of the New York Times [18]. It is not often that academic computer science research directly supports high profile media coverage.

The Commission's report cites several specific academic works. McDonald and Cranor's 2008 work, "The Cost of Reading Privacy Policies" [14], is used to support the Commission staff's opinion that lengthy privacy policies do not enable a consumer's ability to make informed privacy decisions. Egelman et. al. demonstrated that consumers were willing to pay more in exchange for better privacy protections in their 2009 CHI paper, "Timing is Everything? The Effects of Timing and Placement of Online Privacy Indicators" [8]. This work is used to support the FTC staff's argument that customers are uncomfortable being tracked and that they are willing to sacrifice potential benefits in order to maintain a greater level of privacy. Narayanan and Schmatikov's 2008 paper at the IEEE Symposium on Security and Privacy, "Robust De-anonymization of Large Sparse Datasets" was the foundation of DPIP's technical understanding of the privacy implications of a large data set released by Netflix [16].

The privacy report uses academic papers in exactly the way they are supposed to be used: as scientific evidence to inform and support a factually sound understanding of technical policy issues. These papers do not present an agenda, they present facts. And, ideally, these facts will serve as a foundation for effective policymaking.

4.3 Other Demonstrations of Impact

David Vladeck, the director of the Bureau of Consumer Protection, has spoken multiples times on the surreptitious collection of private information through a method known as "CSS history sniffing" [23, 24]. In speeches to both the International Association of Privacy Professionals and the Consumer Watchdog Conference, Vladeck revealed the influence of Jang et. al.'s "An Empirical Study of Privacy-Violating Information Flows in JavaScript Web Applications" published at CCS 2010 [13].

The FTC played a major role in closing down the ability for malicious websites to exploit CSS in order to mine the history of unsuspecting visitors. Browser vendors were urged to tackle the exploit which had

been known for nearly a decade but not seen in widespread use until recently. Jang's paper illustrated the popular use of this tactic and that persuaded the FTC to take action.

The FTC also regularly brings in speakers from the academic research community. The Commission's technologists are responsible for arranging technical speakers on issues relating to both specific investigations and generally relevant privacy and security issues. While these talks are not officially nonpublic, they are not formally announced to avoid indicating potential investigative agendas. The privacy roundtables of 2009 and 2010 indicate the level of interest the Commission has in bringing in knowledgeable academics with speakers such as Alessandro Acquisti, Fred Cate, Lorrie Cranor, Peter Eckersley, Arvind Narayanan, and many others[2]. Academic researchers have also testified in front of members of the Commission, such as Aleecia McDonald [22].

4.4 Influence by Non-Academic Research

While the focus of this paper is on academic publications that directly impact policy, research that falls outside of the published academic record is also vital. Security researchers, hackers, and independent journalists have a long history of ties with the academic privacy and security community as can be seen through a number of conferences such as Chaos Communication Congress, Hacking at Random, DEFCON and organizations like the Electronic Frontier Foundation, Chaos Computer Club and the Tor Project.

The FTC staff privacy report cites Samy Kamkar's *evercookie*[3] to demonstrate the potential for pervasive online tracking beyond the commonly recognized method of utilizing third party cookies [9]. In fact, this citation is directly followed by the most publicized aspect of the report, a call for a single comprehensive mechanism to opt out of online behavioral advertising commonly known as "Do Not Track."

The FTC's Division of Consumer and Business Education regularly issues reports on current threats and countermeasures through OnGuard Online, a repository for consumer education[4]. "Wise Up about Wi-Fi: Tips for Using Public Wireless Networks" warned consumers about the dangers of unencrypted connections over public wireless networks [10]. The report specifically cited the EFF's HTTPS-Everywhere and Sid Stamm's Force-TLS add-ons for Firefox as mechanisms that can be used to help mitigate this threat.

5 Increasing Influence and Impact

The Commission does not have a technical research arm for privacy matters. In fact, as of this writing there are only two staff technologists for the entire commission, excluding IT staff. There are varying numbers of technically savvy staff or interns at different times but in general it is up to non-technical staff to read and understand much of the technical literature that exists relating to privacy and data security.

Academic influence at the Commission, and policy and enforcement agencies in general, would be improved by being aware of the a non-technical audience of your paper and accompanying technical literature with memos that explain the research and implications to that audience.

An accompanying memo should contain three elements. The first is an understanding of current trends within the target agency and how the work fits in or, if it doesn't fit, why it is important and how it relates to the mandate of the agency. In the case of the FTC, browsing the recent list of actions[5] and closing letters[6] can be a good indicator, as can recent speeches delivered by Commissioners and FTC staff[7].

[2] `www.ftc.gov/bcp/workshops/privacyroundtables`
[3] `http://samy.pl/evercookie`
[4] `www.onguardonline.gov`
[5] `www.ftc.gov/opa/index.shtml`
[6] `www.ftc.gov/os/closings/`
[7] `www.ftc.gov/speeches/speech1.shtm`

The second element is an understanding of the basic legal theories that influence the target organization and how that may fit with your work. It can be difficult for staff to understand how a technical issue translates to an actionable item. For the FTC, understanding the concepts of deception and unfairness, legal jurisdictions, and history of action in the area, if any, will increase the likelihood for impact.

The third is a general focus on the scope and impact of your work in the context of the target audience. How does it fit into the big picture? With the FTC, answer questions like "How does this directly affect consumer privacy choices?" And "why does this matter to my parents?" not "why does this matter to a computer scientist?"

Overall, the most important action to take is to talk to people at the FTC and other relevant agencies. Start with the staff technologists to explain your work to initially. Contact staff members who have worked on previous cases you are interested in or that might be relevant. And finally, remember that this memo is also helpful in widely disseminating your work to journalists and non-technical audiences in general.

6 Conclusions

As researchers we can easily forget that the ultimate audience for our work is not the program committee of the next conference or the members of a tenure committee. This is especially true in the field of privacy where research is often intended to directly inform technically sound policy decisions for everyone from national government to an end user.

The question of broad impact is a fundamental research question in privacy and security. How to get users to make good privacy and security decisions, encourage governments to adopt effective policies, and the pursuit of meaningful regulation are important scientific questions. As such, work in the privacy and security fields being adopted or cited by the Commission and other governmental bodies that have a direct role in these questions should be considered the same, depending on the specific instance, as invited talks, conference publications and journal publications.

The broader impact of important privacy and security research is served by greater communication and cooperation between the academic community and the FTC. Technologists at the FTC are seeking to build upon this bridge by bringing in more academic research and speakers into the decision making processes of the Commission. Building this bridge can also come through the invitation of government participation in conferences by way of speakers and presence on program committees. There are legal conflicts of interest to be aware of but these can be navigated by organizers communicating with government representatives and allowing them extra time to obtain approval for participation.

7 Acknowledgments

Many thanks for the valuable input and insight from Rachel Greenstadt of Drexel University, Ed Felten and Peder Magee of the FTC, Thomas Lowenthal of Princeton, and former FTC technologists Chris Soghoian and Ashkan Soltani.

References

[1] L. D. Berger and C. T. Han. Twitter, inc., file no. 092 3093. Consent Agreement, June 24, 2010.

[2] E. P. I. Center. In the matter of facebook, inc.: Complaint, request for investigation, injunction, and other relief. Complaint, January 14, 2010.

[3] P. L. Chavez. Comments of Google, Inc. The New York Times, February 18, 2011.

[4] D. S. Clark. Twitter, inc., file no. 092 3093. Complaint, March 11, 2010.

[5] D. S. Clark. Chitika, inc., file no. 102 3087. Complaint, June 17, 2011.

[6] D. S. Clark. Facebook, inc., file no. 092 3184. Complaint, November 29, 2011.

[7] Workshop on Privacy in the Electronic Society Call for Papers, 2011. `http://wpes11.rutgers.edu/#cfp`.

[8] S. Egelman, J. Tsai, L. F. Cranor, and A. Acquisti. Timing is everything?: the effects of timing and placement of online privacy indicators. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 319–328, New York, NY, USA, 2009. ACM.

[9] Federal Trade Commission. Protecting consumer privacy in an era of rapid change: A proposed framework for businesses and policymakers. Staff Report, December 2010.

[10] Federal Trade Commission. Wise up about wi-fi: Tips for using public wireless networks. OnGuard Online, February 2011.

[11] E. Felten. Another privacy misstep from facebook, December 14 2009.

[12] Federal Trade Commission Act, September 8, 1914.

[13] D. Jang, R. Jhala, S. Lerner, and H. Shacham. An empirical study of privacy-violating information flows in javascript web applications. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 270–283, New York, NY, USA, 2010. ACM.

[14] A. M. Mcdonald and L. F. Cranor. The cost of reading privacy policies. *I/S: A Journal of Law and Policy for the Information Society*, 4:1–22, 2008.

[15] M. Mithal. Netflix, inc., file no. 102 3027. Staff Closing Letter, March 12, 2010.

[16] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

[17] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy*, pages 173–187, Washington, DC, USA, 2009. IEEE Computer Society.

[18] Front page. The New York Times, December 2, 2010. www.nytimes.com/indexes/2010/12/02/pageone/scan.

[19] P. Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. In *UCLA Law Review*, volume 57, page 1701, 2009.

[20] K. D. Ratté and K. R. Brin. Google, inc., file no. 102 3136. Consent Agreement, March 30, 2011.

[21] S. Sengupta. F.t.c. settles privacy issue at facebook. The New York Times, November 29, 2011.

[22] Stanford Law School. Aleecia mcdonald. http://cyberlaw.stanford.edu/profile/aleecia-mcdonald.

[23] D. Vladeck. Remarks of David C. Vladeck. Consumer Watchdog Conference, December 1, 2010.

[24] D. Vladeck. Remarks of David C. Vladeck. IAPP Practical Privacy Series, December 7, 2010.