Simone Fischer-Hübner
Matthew Wright (Eds.)

LNCS 7384

Privacy Enhancing Technologies

12th International Symposium, PETS 2012
Vigo, Spain, July 2012
Proceedings

Springer

Lecture Notes in Computer Science 7384

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Simone Fischer-Hübner Matthew Wright (Eds.)

Privacy Enhancing Technologies

12th International Symposium, PETS 2012
Vigo, Spain, July 11-13, 2012
Proceedings

Springer

Volume Editors

Simone Fischer-Hübner
Karlstad University
Department of Computer Science
Universitetsgatan 2, 65188, Karlstad, Sweden
E-mail: simone.fischer-huebner@kau.se

Matthew Wright
University of Texas at Arlington
Department of Computer Science and Engineering
500 UTA Blvd., Arlington, TX 76019, USA
E-mail: mwright@uta.edu

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-31679-1 e-ISBN 978-3-642-31680-7
DOI 10.1007/978-3-642-31680-7
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012940986

CR Subject Classification (1998): K.6.5, D.4.6, C.2, E.3, H.3-4, J.1

LNCS Sublibrary: SL 4 – Security and Cryptology

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Privacy and anonymity are increasingly important in the online world. Corporations, governments, and other organizations are realizing and exploiting their power to track users and their behavior. Approaches to protecting individuals, groups, but also companies and governments from profiling and censorship include decentralization, encryption, distributed trust, and automated policy disclosure.

The 2012 Privacy Enhancing Technologies Symposium (PETS 2012) addressed the design and realization of such privacy services for the Internet and other data systems and communication networks by bringing together privacy and anonymity experts from around the world to discuss recent advances and new perspectives.

PETS 2012 was held in Vigo, Spain, during July 11–13, 2012. It was the 12th in the series of events, and the fifth after the transition from workshop to symposium. The PETS symposium remains a premier scientific international event for publishing on both the theory and practice of privacy-enhancing technologies, and it has a broad scope that includes all facets of the field.

PETS 2012 received 72 submissions, which were all reviewed by at least three members of the international Program Committee (PC). Based on an intensive discussion among the reviewers and other PC members, 16 papers were finally accepted for presentation at the PETS symposium. Topics addressed by the accepted papers published in the proceedings include anonymization of statistics, content, and traffic, network traffic analysis, censorship-resistant systems, user profiling, training users in privacy risk management, and privacy for Internet and cloud-based services.

A further highlight of PETS 2012 was the popular HotPETs session, designed as a venue to present exciting but still preliminary and evolving ideas, rather than formal and rigorous completed research results. HotPETs included an invited keynote talk by Moez Chakchouk, the head of the Tunisian Internet Agency. As with the previous four HotPETs in the past, there were no published proceedings for HotPETs. PETS also included a panel on "The impact of upcoming privacy legislation for PETs" organized and moderated by Marit Hansen and a rump session with brief presentations on a variety of topics. Additionally, a workshop on Provable Privacy was held in conjunction with PETS 2012.

We would like to thank all PETS and HotPETs authors, especially those who presented their work selected for the program, as well as all rump session presenters. Moreover, we are very grateful to all PC members and additional reviewers, who contributed with thorough reviews and actively participated in the PC discussions, ensuring a high quality of all accepted papers. We owe special thanks to the following PC members and reviewers, who volunteered to shepherd

some of the accepted papers: Emiliano De Cristofaro, Erman Ayday, Roger Dingledine, Thomas S. Benjamin, Nicholas Hopper, Aaron Johnson, Damon McCoy, and Arvind Narayanan.

We gratefully acknowledge the outstanding contributions of the PETS 2012 General Chair, Carmela Troncoso, the Local Arrangements Chair, Fernando Pérez-González, and of our webmaster since 2007, Jeremy Clark. Moreover, our gratitude goes to the HotPETs 2012 Chairs, Emiliano De Cristofaro and Julien Freudiger, who reviewed all HotPETs submissions and put together an excellent program. Last but not least, we would like to thank all sponsors of PETS 2012, including Ministerio de Economía y Cooperación (Spanish Ministry of Economy), Gradiant (Galician Research and Development Center in Advanced Telecommunications), and Ayuntamiento de Vigo (Vigo City Hall) for their generous support as well as Microsoft for its continued sponsorship.

May 2012

Simone Fischer-Hübner
Matthew Wright

Organization

Program Committee

Kevin Bauer	University of Waterloo, Canada
Thomas S. Benjamin	IBM Research Zurich, Switzerland
Jean Camp	Indiana University, USA
George Danezis	Microsoft Research, UK
Sabrina De Capitani Di Vimercati	DTI - Università degli Studi di Milano, Italy
Emiliano De Cristofaro	Palo Alto Research Center, USA
Roger Dingledine	The Tor Project, USA
Hannes Federrath	University of Hamburg, Germany
Julien Freudiger	EPFL, Switzerland
Simson Garfinkel	Naval Postgraduate School, USA
Rachel Greenstadt	Drexel University, USA
Nicholas Hopper	University of Minnesota, USA
Jean-Pierre Hubaux	EPFL, Switzerland
Renato Iannella	Semantic Identity, Australia
Aaron Johnson	Naval Research Laboratory, USA
Damon McCoy	George Mason University, USA
Aleecia Mcdonald	Carnegie Mellon, USA
Steven Murdoch	University of Cambridge, UK
Shishir Nagaraja	University of Birmingham, UK
Arvind Narayanan	Stanford University, USA
Gregory Neven	IBM Research Zurich, Switzerland
Siani Pearson	HP Labs, UK
Kazue Sako	NEC, Japan
Pierangela Samarati	Università degli Studi di Milano, Italy
Michael Waidner	Fraunhofer SIT, Germany

Additional Reviewers

Acs, Gergely	Clauß, Sebastian	Herrmann, Dominik
Afroz, Sadia	Elahi, Tariq	Huguenin, Kevin
AlSabah, Mashael	Foresti, Sara	Humbert, Mathias
Anirban, Basu	Fuchs, Karl-Peter	Kadianakis, George
Appelbaum, Jacob	Furukawa, Jun	Kalabis, Lukas
Ayday, Erman	Garg, Vaibhav	Kohlweiss, Markulf
Basu, Anirban	Gerber, Christoph	Kreitz, Gunnar
Bilogrevic, Igor	Ghiglieri, Marco	Kuzu, Mehmet
Chothia, Tom	Henry, Ryan	Lindqvist, Janne

Papanikolaou, Nick
Patil, Sameer
Pham, Vinh
Shokri, Reza
Simo, Hervais
Soriente, Claudio
Stopczynski, Martin
Tancock, David
Teranishi, Isamu
Uzun, Ersin
Wang, Qiyan
Zhang, Nan

Table of Contents

Session 5: Privacy Services

Session 6: User-Related Privacy Perspectives

Betrayed by Your Ads!

Reconstructing User Profiles from Targeted Ads

Claude Castelluccia, Mohamed Ali Kaafar, and Minh-Dung Tran

Inria, France
{claude.castelluccia,mohamed-ali.kaafar,minh-dung.tran}@inria.fr

Abstract. In targeted (or behavioral) advertising, users' behaviors are tracked over time in order to customize served ads to their interests. This creates serious privacy concerns since for the purpose of profiling, private information is collected and centralized by a limited number of companies. Despite claims that this information is secure, there is a potential for this information to be leaked through the customized services these companies are offering. In this paper, we show that targeted ads expose users' private data not only to ad providers but also to any entity that has access to users' ads. We propose a methodology to filter targeted ads and infer users' interests from them. We show that an adversary that has access to only a small number of websites containing Google ads can infer users' interests with an accuracy of more than 79% (Precision) and reconstruct as much as 58% of a Google Ads profile in general (Recall). This paper is, to our knowledge, the first work that identifies and quantifies information leakage through ads served in targeted advertising.

Keywords: Targeted Advertising, Privacy, Information leakage.

1 Introduction

Context. Internet users are being increasingly tracked and profiled. Companies utilize profiling to provide customized, i.e. personalized services to their customers, and hence increase revenues. In particular, behavioral advertising takes advantage from profiles of users' interests, characteristics (such as gender, age and ethnicity) and purchasing activities. For example, advertising or publishing companies use behavioral targeting to display advertisements that closely reflect users' interests (e.g. 'sports enthusiasts'). Typically, these interests are inferred from users' web browsing activities, which in turn allows building of users' profiles.

It can be argued that customization resulting from profiling is also beneficial to users who receive useful information and relevant online ads in line with their interests. However, behavioral targeting is often perceived as a threat to privacy mainly because it heavily relies on users' personal information, collected by only a few companies. In this paper, we show that behavioral advertising poses an additional privacy threat because targeted ads expose users' private data to any entity that has access to a small portion of these ads. More specifically, we show

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 1–17, 2012.

that an adversary who has access to a user's targeted ads can retrieve a large part of his interest profile. This constitutes a privacy breach because, as illustrated in Section 2, interest profiles often contain private and sensitive information.

Motivation. This work was largely motivated by the Cory Doctorow's "Scroogled" short story that starts as follows [12]:

Greg landed at San Francisco International Airport at 8 p.m... The officer stared at his screen, tapping...

- "Tell me about your hobbies. Are you into model rocketry?"
- "What?"
- "Model rocketry."
- "No," Greg said, "No, I'm not."
- "You see, I ask because I see a heavy spike in ads for rocketry supplies showing up alongside your search results and Google mail."
- "You're looking at my searches and e-mail?"
- "Sir, calm down, please. No, I am not looking at your searches,... That would be unconstitutional. We see only the ads that show up when you read your mail and do your searching. I have a brochure explaining it ..."

The main goal of this paper is to study whether such scenario would be possible today, and if one can infer a user's interests from his targeted ads. More specifically, we aim at quantifying how much of a user's interest profile is exposed by his targeted ads. However, as opposed to the above story, we do not consider ads that show up when a user reads his email or uses a search engine. These ads are often contextual, i.e. targeted to email contents or search queries. Instead, we consider targeted ads that are served on websites when a user is browsing the web.

Contributions of This Paper. We describe an attack that allows any entity that has access to users' targeted ads to infer these users' interests recovering a significant part of their interest profiles. More specifically, our experiments with the Google Display Network[4] demonstrate that by analyzing a small number of targeted ads, an adversary can correctly infer users' Google interest categories with a high probability of 79% and retrieve as much as 58% of Google Ads profiles.

The attack described in this paper is practical and easy to perform, since it only requires the adversary to eavesdrop on a network for a short period of time and collect a limited number of served ads.

The crux of the problem is that even if some websites use secure connections such as SSL (Secure Socket Layer), ads are almost always served in clear. For example, Google currently does not provide any option to serve ads with SSL[1] [2]. We acknowledge that in some scenarios the adversary can recover a user's profile directly from the websites he visits, i.e. without considering targeted ads. However, we show in this paper that targeted ads can often improve the accuracy of recovered profiles and reduce the recovery time. Furthermore, in

[1] We verified this feature by browsing through several https websites (e.g. `https://www.nytimes.com/`).

some circumstances, the victim has different browsing behaviors according to his environment. For example, a user at work mostly visits websites related to his professional activity, while he visits websites related to his personal interests at home. We show in this paper that an adversary, such as an employer, that can eavesdrop on the victim's computer or network while at work can infer information about his "private" and personal interest profile. In other words, targeted ads constitute a covert channel that can leak private information.

Although there are various targeted advertising networks today, this work focuses on Google advertising system, which is "the most prevalent tracker" according to a survey of The *Wall Street Journal* [17]. However, our methodology is general enough to be extended to other ad networks. The problem of generality will be discussed in Section 3.1.

Structure of the Paper. Section 2 describes the Google targeted advertising system. In section 3, we present our approach to filter targeted ads and describe how we infer Google Ads profiles from them. We then present in Section 4 the performance of our method through some experiments in the Google Display Network. In section 5, we discuss the related work. Section 6 presents possible countermeasures and discusses some relevant problems. Section 7 concludes the paper.

2 Targeted Advertising: The Case of Google

Google Display Network is a network of websites (also called publishers) that serve Google ads. Google receives ads from advertisers and then selects the appropriate publishers using various criteria such as relevant content, bid price and revenue.

In the Google targeted advertising model, Google Display Network sites are also used to track users as they browse the Internet. Each time a user visits a website that contains Google ads, i.e. a website that belongs to the Google Display Network, he sends his *DoubleClick*[2] cookie to Google, along with information about the visited website. As a result, Google collects all the sites within the Google Display Network that have been visited by a user, and builds an interest profile from them. A Google profile is defined as a set of categories and sub-categories (see figure 1). For example, if a user visits a football site several times, Google may assign him the category *Sport*, or more specifically the sub-category *Sport* $\rightarrow$ *Football*. In addition, a Google profile may include location information and some demographic data such as the gender and age of the user. These profiles are then used to target ads to users.

A user can access and modify his Google Ads Preferences by accessing the webpage `http://www.google.com/ads/preferences` [8]. Furthermore, a user can choose to opt out of the Google targeted advertising if he no longer wants to receive targeted ads. Figure 1 displays an example of a Google user profile

[2] In order to keep track of users visiting the Google Display Network, Google uses the DoubleClick cookie issued from the doubleclick.net domain which belongs to Google.

Fig. 1. An Example of a Google Ads Preferences Page

that contains potentially private and sensitive information. For example, the "Job listing" category indicates that the user is probably looking for a job. A user might probably want to keep this information secret, in particular from his current employer. Furthermore, the category "Dating & Personals" indicates that the user is currently actively looking for a relationship, and the subcategories "Baby names" and "Adoption" that he has been recently visiting web sites related to baby adoption.

In its privacy policy, Google states that "We take appropriate security measures to protect against unauthorized access to or unauthorized alteration, disclosure or destruction of data.", and "We restrict access to personal information to Google employees, contractors and agents who need to know that information in order to process it on our behalf" [5]. Nevertheless, in this paper we show that a portion of personal users' profiles could be leaked through targeted ads. Even if Google does not consider users' interests as "personal information", this data which is related to users online activities, can be very private from a user's perspective.

3 Inferring Users' Profiles from Targeted Ads

As targeted ads are personalized to each user based on his profile, they can reveal a lot of information about users' interests. This section describes how an adversary who has access to an user's ads can derive part of his interests from them.

As shown in Figure 2, our approach is composed of two main phases. The first phase collects all ads served to a target user and filters them to only retain targeted ones. In the second phase, targeted ads are classified into categories

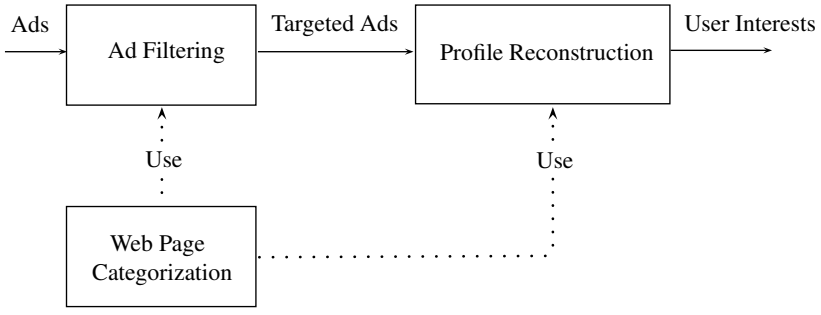


Fig. 2. Filtering targeted ads and inferring user interests

and a profile is re-constructed. These two phases are detailed in the rest of this section. However, we first start by presenting two building blocks used by these both phases. The first one is an algorithm that is used to categorize webpages. The second one is a set of algorithms to compare categories.

3.1 Building Blocks

Web Page Categorization. The web page categorization tool is an algorithm that derives interest categories from a web page. This algorithm relies on the tool used by Google to generate Google Ads Preferences pages. As described in the previous section, each time a user visits a web page, Google derives some interests in the form of categories and updates the user's Google Ads Preferences page accordingly.

We use this tool to build our page categorization algorithm. Given a webpage W, our algorithm operates as follows:

1. W is visited and and the resulting DoubleClick cookie is saved.
2. A request is made to the Google Ads Preferences page with the previously saved cookie. Note that Google does not always update the Google Ads Preferences page upon a single web page visit. Usually, a webpage has to be visited multiple times to update the Google Ads Preferences page. Furthermore, we noticed that users' Ads preferences are updated after a period of time ranging between 1 and 2 minutes. Therefore, this step is repeated 5 times (heuristic value) every 2 minutes.
3. The Google Ads Preferences page is parsed and the corresponding categories are retrieved.

To evaluate the performance of our approach, we scraped 5000 ads from Google search page and 2000 sites from Google Display Network, classified them by the tool, and reviewed the results. We detected that almost all of these pages can be categorized by Google (more than 90%). We also manually reviewed the categorization results and observed that, although there are some irrelevant

Table 1. Ad page categorization example

Ad Url	Categories
http://www.elasticsteel.net ID=156	Beauty & Fitness → Fitness
http://www.livecarhire.com	Travel → Car Rental & Taxi Services
http://www.terracebeachresort.ca	Travel → Hotels & Accommodations Travel → Tourist Destinations → Beaches & Islands
http://www.sanibelbayfronthouse.com	Arts & Entertainment → Entertainment Industry → Film & TV Industry → Film & TV Production Real Estate → Timeshares & Vacation Properties Travel → Tourist Destinations → Zoos-Aquariums-Preserves
http://www.siestakeyaccommodation.com	Real Estate → Timeshares & Vacation Properties Travel

categories, the categorization generally reflects the content of each page. Table 1 presents several examples of ad page categorization.

It should be noted that relying on Google does not reduce the generality of our method. There exist many efficient techniques to categorize the content of web pages. For example [16] uses cosine similarity. This method is very efficient since it relies on social/crowd data (folksonomy) which is continuously updated, and is appropriate for fine-grained categorization. We implemented this method and compared its performance with the Google-based categorization approach we described above. The obtained results were quite similar, with more than 60% of the categories overlapping. We therefore believe that our work can be extended to other ad networks, such as Yahoo! or Microsoft, either by applying their own categorization, or by simply using an existing webpages categorization technique. Note that Yahoo! and Microsoft also build users' behavior-based interest profiles and similarly to Google personalize ads to users according to their interests [9] [10].

Category Comparison Methods. Many of the filtering and evaluation algorithms presented in this paper need to compare categories. We use three methods for this purpose: "Same category", "Same parent" and "Same root":

1. *Same category:* Two categories are considered equivalent in the "Same category" method if they match exactly.
2. *Same parent:* Two categories are considered equivalent in the "Same parent" method if they have the same parent category. For example, the two categories "Arts & Entertainment → Entertainment Industry → Film & TV Industry → Film & TV Awards" and "Arts & Entertainment → Entertainment Industry → Film & TV Industry → Film & TV Production" have

the same parent category "Film & TV Industry", so they are considered equivalent to each other in the "Same parent" method.

3. *Same root:* Two categories with same root category are considered equivalent in the "Same root" method. For example, the two categories "Arts & Entertainment → Entertainment Industry → Recording Industry → Music Awards" and "Arts & Entertainment → Movies → Action & Adventure Films → Superhero Films" have the same root category "Arts & Entertainment" and therefore are equivalent to each other in the "Same root" method. Obviously, if two categories are equivalent in the "Same parent" method, they are also equivalent in the "Same root" method.

3.2 Extracting Targeted Ads

Ads provided by Google are either location-based, content-based (we call hereafter contextual, i.e. related to the visited page's content), generic, or profile-based (we call hereafter targeted, i.e. customized to users' profiles). In this paper, we only consider targeted ads. We therefore need to filter out location-based, content-based and generic ads (see figure 3).

Fig. 3. Filtering targeted ads

We conducted all experiments with users from the same location. As a result, the location-based filter is not used (and therefore not presented here). Furthermore, we consider that an ad is contextual if it shares at least one category with its displaying page (the page on which the ad is delivered). To filter out contextual ads, we therefore categorize, using the categorization technique described in Section 3.1, each ad and their displaying page. If at least one category is in common, the ad is classified as contextual. To filter generic (i.e. not customized) ads, we create a number of non-overlapping user profiles (i.e. profiles without any categories in common), and perform 10 requests to the tested pages[3]. Ads that are served independently of the requesting profile are then deemed generic and filtered out.

3.3 User-Profile Reconstruction

Given the targeted ads from the previous step, there are possibly many approaches to infer user information. In our work, we aim at reconstructing the

[3] The number of 10 requests is considered to be enough to get a sufficient ad collection while resisting well to the ad churn [13].

Google-assigned interest categories which are presented as user profiles. In order to reconstruct a user profile, we categorize all of his targeted ads using our Google-based web page categorization tool. The reconstructed profile is then the set of resulting Google categories.

For example, considering the ads provided in table 1, the reconstructed profile will look as follows:

Table 2. Profile reconstruction example

Reconstructed profile
Beauty & Fitness → Fitness
Travel
Travel → Car Rental & Taxi Services
Travel → Hotels & Accommodations
Travel → Tourist Destinations → Beaches & Islandsl
Arts & Entertainment → Entertainment Industry → Film & TV Industry → Film & TV Production
Real Estate → Timeshares & Vacation Properties

4 Experimental Results

In this section, we evaluate the performance of our profile reconstructing technique.

4.1 Experiment Setup

Figure 4 illustrates the setup of our experiments. Our experiments are composed of two main phases:

Profile Creation. In this phase, we create a set of profiles corresponding to different web users. Each of these profiles, that we call *targeted profiles,* P_t, is obtained by visiting several websites from a user's real web-history (i.e. list of websites that the user has visited). We refer to these websites as *training sites*. Each of them is visited 15 times to make sure it really affects profiles. We then retrieve the generated Google profile from the Google Ads Preferences page (this phase corresponds to the lower part of figure 4).

Profile Re-construction. In this phase, we visit for each targeted profile (P_t) created as described above another set of websites, that we refer to hereafter as *visited websites*. As opposed to the training sites, each visited site is only visited once. The ads are then collected, filtered and the profile reconstructed as described in Section 3. We refer to the set of profiles we obtain as *reconstructed profiles*, P_r (this phase corresponds to the upper part of figure 4).

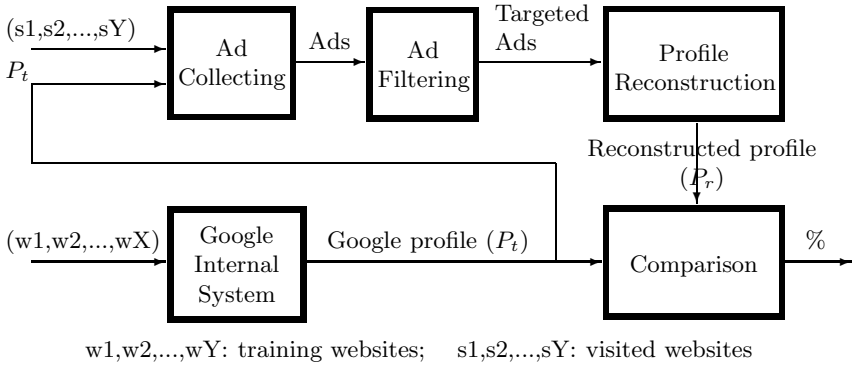


Fig. 4. Filtering targeted ads and inferring user interests

4.2 Evaluation Methodology

Dataset. Our target web-history data comes from a set of 40 volunteers who provided their list of websites they visited during two days. The first X websites in each profile were used as the set of training sites to create P_t. The Y following websites were used to build the reconstructed profiles, P_r, as shown in Figure 5.

In the presented experiments, X was set to 30 and different values of Y were used. The average number of root categories and categories in a targeted profile from X websites is displayed in Table 3.

Table 3. Profile size statistics

	# of root categories	# of categories
X = 30	6.64	18.06

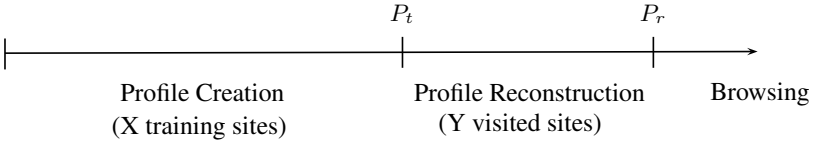


Fig. 5. Profile creation and reconstruction

Performance Evaluation Metrics. To evaluate the results, we compare each reconstructed profile with the corresponding original one. We compare profiles using the "same-category", "same-parent" and "same-root" methodologies described in Section 3.1. We evaluate the performance of our technique by computing the average *Precision*, *Recall* and *F-Measure* values of all reconstructed profiles. Precision is the fraction of rebuilt categories that are correct, while Recall is the

fraction of original categories that are correctly rebuilt. F-Measure is the harmonic mean between Precision and Recall, defined as: $F = \frac{2.precision.recall}{precision+recall}$.

In other words, if we denote by $P_{r,c}$ the categories of the reconstructed profile P_r that are correct, and $P_{r,i}$ the categories of P_r that are incorrect, $Precision = \frac{|P_{r,c}|}{|P_r|} = \frac{|P_{r,c}|}{|P_{r,c}+P_{r,i}|}$ and $Recall = \frac{|P_{r,c}|}{|P_t|}$.

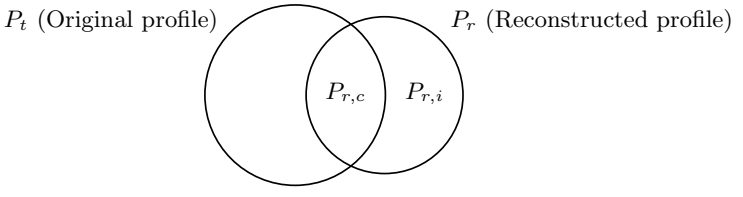


Fig. 6. An illustration of Precision and Recall

Adversary Strategies. In order to evaluate the performance gain obtained by using targeted ads as opposed to only using visited websites, we consider the following three strategies:

- the adversary only uses visited websites ("Sites only").
- the adversary only uses targeted ads ("Ads only").
- the adversary uses visited websites and targeted ads ("Ads & Sites")

Tested Scenarios. Finally, we consider two different scenarios, corresponding to two different browsing behaviors:

1. *HotSpot Scenario*: This scenario corresponds to the case where the victim is connecting to an open network and browses the Internet according to his interests. In this scenario, the X training sites and the Y visited sites are related to each others, i.e. generated from the same interest profiles. The goal of this scenario is to show that targeted ads can be used to boost the accuracy of profile reconstruction.
2. *Workplace Scenario*: This scenario corresponds to the case where the victim changes his browsing behavior during the reconstruction phase. In other words, profiles used to generate the training sites and the visited sites are significantly different. This situation happens, for example, when the victim is moving from his home to his work environment. The goal of this scenario is to study how much of the home profile leaks from the targeted ads shown at work.

In the following, we present, for the workplace scenario, how we select the visited websites so that they are largely separated from a user's interests. We first randomly select a set of Google root categories, namely "Autos & Vehicles", "Law & Government", "Books & Literature", "Beauty & Fitness", "Jobs & Education" and "Business & Industrial". We then get for each of these categories

500 websites using the Google Adwords Placement Tool [3]. This tool aims at helping advertisers to select websites to publish their ads. We then get for each user all of his root categories, and select a root category C that does not belong to them. The user's visited sites are then randomly selected from the 500 websites corresponding to category C. For example, if a profile contains 4 root categories: "Law & Government", "Books & Literature", "Beauty & Fitness", "Jobs & Education", then one of the remaining categories, "Autos & Vehicles" or "Business & Industrial", will be chosen for visited websites. We verified that none of our test profiles contains all the six visited categories.

Note that a website classified in a Google category according to Google Adwords Placement Tool may result in another category in Google Ads Preferences. For instance, Google may assign a website W to category "Arts & Entertainment". However, when categorizing this website using Google Ads Preferences, the result may include, in addition to "Arts & Entertainment", another root category, say "Books & Literature". Therefore, we cannot completely guarantee that the visited websites are totally separated from the training ones.

4.3 Result Analysis

Tables 4, 5, 6 and 7 represent the achieved Precision, Recall and F-Measure values in percentage with ($X = 30$, $Y = 10$) and ($X = 30$, $Y = 15$) for the hotspot and workplace scenarios respectively. The rows in these tables specify the category comparison methods used to filter out contextual ads[4]. This comparison method is also used to evaluate the results (i.e. to compare the reconstructed profiles with the original ones)[5]. We remind the reader that these comparison methods are described in Section 3.1. The columns of the table specify the three different cases of profile reconstruction, using "Sites only", "Ads only" and "Ads & Sites", respectively. The tables show that the Ads-based information leakage is significantly high, with precision values ranging from 73 to 82% for reconstructed profiles evaluation based on recovering the root of categories solely from Ads. For example, in case ($X = 30$, $Y = 15$) in the workplace scenario, with "Ads only" and the "Same root" comparison method (used for both filtering and evaluation processes), we achieve Precision, Recall and F-Measure of more than 79%, 58% and 67% respectively (Table 7). The average number of targeted ads we observed accounts for approximately 30% of all collected ads in each case. We note that the results of the row "Same Category" show in general a relatively lower precision and recall values than the results of the "Same Parent" and "Same Root" rows.

Figures 7 and 8 display the variation of Precision, Recall and F-Measure when varying the number Y of visited web sites for each targeted profile, for different comparison methods. We observe that, for a given profile (i.e. when X and therefore $|P_t|$ are fixed), the recall increases noticeably with Y, the number

[4] For example, the row "same parent" displays results when ads are considered contextual if they share the same parent categories with the pages that display them.

[5] For example, the column "same parent" means that two categories are deemed identical if they share the same parent.

Table 4. Reconstructing Google profiles performance in Hotspot scenario ($X = 30$ and $Y = 10$)

	Av.# of targ. ads	Sites only Prec./Recall /F	Ads only Prec./Recall /F	Ads & Sites Prec./Recall /F
Same cat.	14.29	19.66/7.6 /10.96	18.04/7.06 /10.15	18.3/14 /15.86
Same parent	10.94	58.25/29 /38.72	53.67/19.38 /28.48	55.98/42.29 /48.18
Same root	9.24	79.26/51.44 /62.39	73.08/30.06 /42.6	79.6/68.33 /73.54

Table 5. Reconstructing Google profiles performance in Hotspot scenario ($X = 30$ and $Y = 15$)

	Av.# of targ. ads	Sites only Prec./Recall /F	Ads only Prec./Recall /F	Ads & Sites Prec./Recall /F
Same cat.	21.53	19.67/10.28 /13.50	15.71/8.47 /11.01	17.07/17.66 /17.36
Same parent	16.67	54.46/34.44 /42.2	51.26/23.54 /32.26	52.73/50.16 /51.41
Same root	14.4	75.57/61.13 /67.59	82.24/40.3 /54.09	78.5/80.52 /79.5

Table 6. Reconstructing Google profiles performance in Workplace scenario ($X = 30$ and $Y = 10$)

	Av.# of targ. ads	Sites only Prec./Recall /F	Ads only Prec./Recall /F	Ads & Sites Prec./Recall /F
Same cat.	19.23	2.92/1.05 /1.54	14.73/10.28 /12.11	11.84/10.94 /11.37
Same parent	13.6	9.09/3.99 /5.55	46.31/30.31 /36.64	34.39/31.56 /32.91
Same root	11.2	12.65/6.43 /8.53	78.07/53.96 /63.81	56.49/55.94 /56.21

Table 7. Reconstructing Google profiles performance in Workplace scenario ($X = 30$ and $Y = 15$)

	Av.# of targ. ads	Sites only Prec./Recall /F	Ads only Prec./Recall /F	Ads & Sites Prec./Recall /F
Same cat.	28.11	2.99/1.31 /1.82	13.44/11.95 /12.65	10.89/12.62 /11.69
Same parent	20.3	9.13/5.06 /6.51	44.95/33.8 /38.59	32.75/35.45 /34.05
Same root	17.13	14/8.61 /10.66	79.37/58.12 /67.10	55.85/60.1 /57.9

of visited web sites, while the precision is steady. This shows that the number of correctly reconstructed categories, i.e. $|P_{r,c}|$, increases with Y. This result is expected since when Y increases the number of collected ads also increases and as such the amount of available information is higher. However for a given X, the precision is not notably affected by Y, which means that the number of incorrectly reconstructed categories, i.e. $|P_{r,i}|$, also increases with Y.

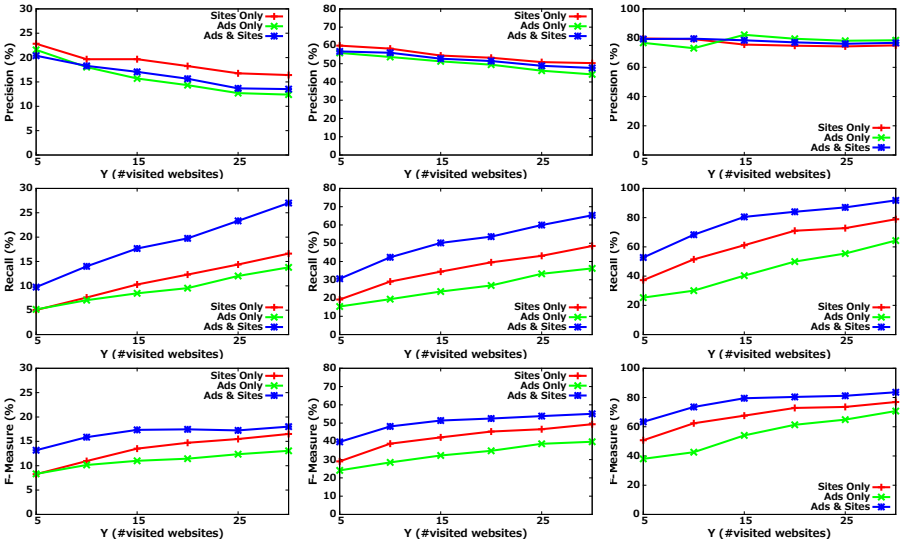


Fig. 7. Precision, Recall and F-Measure with the "Same category", "Same parent" and "Same Root" comparison methods (from left to right respectively) used in both filtering and evaluation processes (**In hotspot scenario with** $X = 30$)

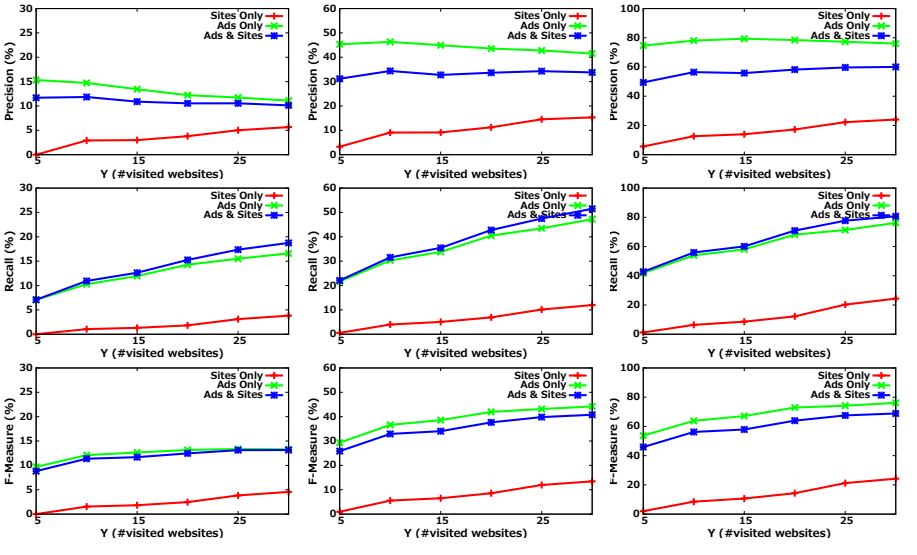


Fig. 8. Precision, Recall and F-Measure with the "Same category", "Same parent" and "Same Root" comparison methods (from left to right respectively) used in both filtering and evaluation processes (**In workplace scenario with** $X = 30$)

In the hotspot scenario, the visited websites are largely relevant to the training websites, therefore reconstructing profiles from "Sites only" achieves the results as good as, if not better than, the results obtained from "Ads only" (see figure 7). However, when we combine both sites and ads in the reconstruction process, we get nearly the same Precision, while increasing Recalls remarkably (almost the sum of the two cases "Sites only" and "Ads only"). In this scenario, Ads are very useful to boost the performance since they allow the recovery of a larger part of the targeted profiles.

In the workplace scenario, the visited websites are considerably separated from training websites. Therefore, reconstructing profiles from "Sites only" leads to very poor results, whereas the "Ads only" technique achieves significantly better results (see figure 8). By combining sites and ads, we slightly increase the Recall while reducing the Precision. In this scenario, we observe that ads do indeed constitute a "hidden" channel that leaks private information about users.

While the performance of our method when evaluating the recovery of "root" and "parent" categories is notably high, we acknowledge that it can only recover a small proportion of a user's actual categories (Precision varies between 10 and 18% when using the same category method for evaluation, and Recall ranges from 10 to 17%). We believe there are several explanations for this result. First, Google might not be using the user's actual categories to deliver ads, and instead might use the root or parent categories for a broader coverage of users' interests. Furthermore, our ads classification method is probably not optimal and could be improved in many ways. In particular, we took a simple and conservative approach in the filtering step by ignoring location-based ads. In addition, we did not consider remarketing ads that, as discussed in Section 6, may contain additional information about the users recent online activities.

However, even only 10 to 15% of an user's interest categories can constitute a severe privacy breach. To illustrate this statement, we ran our technique on the profile shown in Figure 1, and using targeted ads *only*, we recovered the profile shown in Figure 9. Among the recovered categories, the category "Online Communities → Dating & Personals" may constitute a private piece of information which a user might not be willing to share.

Fig. 9. Reconstructed Profile

5 Related Work

To the best of our knowledge, this work is the first work to quantify the private information leakage from targeted ads content. In the following, we present the most relevant work to our paper:

Privacy-Preserving Advertising Systems. Some initial efforts have been put in designing targeted advertising models yet preventing users from being tracked by ad networks. Among them are Privad [14] and Adnostic [16]. Their main idea is to keep behavioral information at the client side and then to perform the ad selection process locally. The proposed models provide a stronger protection for user privacy than current systems do, but their feasibility is in turn still open to debate. Our work considers a different adversary model. While these schemes try to prevent ad networks from tracking users, we assume that the ad network is trusted and aim at protecting users privacy from eavesdroppers.

Privacy Violations Using Microtargeted Ads. Korolova has recently presented attacks that could be used by advertisers to obtain private user information on Facebook [15]. The author showed that an advertiser can manipulate its served ads in order to learn information about users' profiles. This work is complementary to ours, since it considers a different adversary model.

Retrieving User's Profile. [11] presented an attack to infer user search history from Google Web search suggestions. While the webhistory webpage is protected by SSL and Google account authentication, the authors showed that a large part of a user's search history can be reconstructed by a simple session hijacking attack.

6 Discussion

Countermeasures. In order to protect against this information leakage, the easiest solution today is to simply opt out of targeted advertising, frequently delete cookies or use ad-blocking software. Initiatives such as NAI (Network Advertising Initiative) [6], DNT (Do Not Track) [1] or TPLs (Tracking Protection Lists) [7] that aim to provide users with tools to restrict tracking and/or behavioral advertising could also mitigate the identified privacy threat. However, these solutions often prevent to target ads or even to serve ads to users.

There exist several possible countermeasures that could be used to target ads to users and mitigate the information leakage identified in this paper. In particular, there are ongoing efforts to design and deploy privacy-preserving ad systems (e.g. Privad [14] and Adnostic [16]) whose main principle is to select ads locally. These solutions make the eavesdropping and filtering of targeted ads, and therefore our inferring attack, much more difficult. Another possible solution would be to send all ad requests and responses (containing DoubleClick cookies and ads content) over secure channels (SSL). However, we believe that this solution

needs deeper analysis from the research community and the advertising industry since it might be too costly, not practical or altogether hard to deploy.

Stealing Ads Preferences via an Active Attack. The attack presented in this paper is *passive*, i.e. completely transparent to the victim and to the ads providers. We note that a user's preferences can also be stolen by a simple active attack. If fact, if an adversary is able to steal the victim's DoubleClick cookie, it can connect to his Google Ads preference page and retrieve his preferences. We examined the top 100 commercial websites from Alexa and found that at least 71% of them exchange DoubleClick cookie in clear with remote servers. Stealing a Double Click cookie is then quite easy. We implemented and tested this cookie hijacking attack, and were always able to retrieve the victim's Ads preferences page with a simple request to Google Ads servers. This attack is simple, however as opposed to our scheme, it is active and intrusive.

Remarketing Ads. This paper did not consider "remarketing ads", which advertise the services or products of a site that a user has visited. Consider a user who is looking for a hotel in Vigo, Spain and performs some searches on the site `www.hotels.com`. It is very likely that he will consequently receive frequent ads advertising hotels in Vigo while browsing the Internet. Remarketing ads are not only targeting a particular user's interests, but specifically aim to match an exact intention or previous online action. Remarketing ads actually leak much more information about the user. In fact, in our example, they will not only leak that the user is interested in travelling, but also his actual destination i.e. Vigo, Spain. Note that remarketing ads are served independently of Google Ads Preferences profiles. A user will receive remarketing ads even if he empties his ads preferences profile. The only way to stop receiving remarketing ads is to clear his cookies or to completely opt out of targeted ads.

7 Conclusion

In this paper, we showed that targeted ads contain valuable information that allows accurate reconstruction of users' interest profiles. We presented a methodology to categorize and filter targeted ads, which are in turn used to infer users' profiles. Based on both real users' web histories and synthetic users' profiles, we showed that our technique achieves a high accuracy in predicting general topics of users' interests. Additionally, using only a limited number of collected targeted ads we demonstrated that an adversary can capture on average more than half of targeted profiles. The algorithms described in this paper are simple and probably not optimal. We believe they could be improved in many ways.

Many people claim that the main issue in online behavioral advertising is not related to ads personalization itself, which allows users to receive useful ads, but rather to the fact that it requires users' activities tracking. In this paper, we show that ads personalization can also be harmful to users' privacy and does actually leak sensitive information such as users' profiles. We also note that this information leakage is not specific to online behavioral advertising, but in fact

exists in any personalized content (news, searches, recommendations, etc.). As the web is moving toward services personalization almost everywhere, special attention should be paid to these privacy threats. This paper contributes in the understanding of possible privacy risks of content personalization.

Acknowledgments. The authors are grateful to numerous colleagues for thought-provoking discussions on an earlier version of this paper, and to the anonymous reviewers for their valuable comments.

References

1. Do Not Track (2011), http://donottrack.us/
2. Google AdSense Help (2011), https://www.google.com/adsense/support/bin/answer.py?hl=en&answer=10528
3. Google Adwords Placement Tool (2011), http://adwords.google.com/support/aw/bin/answer.py?hl=en&answer=179238/
4. Google Display Network (2011), http://www.google.com/ads/displaynetwork/
5. Google Privacy Policy (2011), http://www.google.com/intl/en/policies/privacy/
6. Network Advertising Initiative (2011), http://www.networkadvertising.org/
7. Tracking Protection Lists (2011), http://www.privacyonline.org.uk/
8. Google Ads Preferences (2012), http://www.google.com/ads/preferences/
9. Personalized Advertising from Microsoft (2012), http://choice.live.com/AdvertisementChoice/Default.aspx
10. Yahoo! Ad Interest Manager (2012), http://info.yahoo.com/privacy/us/yahoo/opt_out/targeting/
11. Castelluccia, C., De Cristofaro, E., Perito, D.: Private Information Disclosure from Web Searches. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 38–55. Springer, Heidelberg (2010)
12. Doctorow, C.: Scroogled (2007), http://blogoscoped.com/archive/2007-09-17-n72.html
13. Guha, S., Cheng, B., Francis, P.: Challenges in measuring online advertising systems. In: Internet Measurement (2010)
14. Guha, S., Cheng, B., Francis, P.: Privad: Practical privacy in online advertising. In: NSDI (2011)
15. Korolova, A.: Privacy violations using microtargeted ads: A case study. In: ICDM Workshops (2010)
16. Toubiana, V., Narayanan, A., Boneh, D., Nissenbaum, H., Barocas, S.: Adnostic: Privacy preserving targeted advertising. In: NDSS (2010)
17. Valentino-Devries, J.: What they know about you. The Wall Street Journal (July 31, 2010)

Private Client-Side Profiling with Random Forests and Hidden Markov Models

George Danezis[1], Markulf Kohlweiss[1], Benjamin Livshits[1], and Alfredo Rial[2]

[1] Microsoft Research
{gdane,markulf,livshits}@microsoft.com
[2] IBBT and KU Leuven, ESAT-COSIC, Belgium
alfredo.rial@esat.kuleuven.be

Abstract. Nowadays, service providers gather fine-grained data about users to deliver personalized services, for example, through the use of third-party cookies or social network profiles. This poses a threat both to privacy, since the amount of information obtained is excessive for the purpose of customization, and authenticity, because those methods employed to gather data can be blocked and fooled.

In this paper we propose privacy-preserving profiling techniques, in which users perform the profiling task locally, reveal to service providers the result and prove its correctness. We address how our approach applies to tasks of both classification and pattern recognition. For the former, we describe client-side profiling based on random forests, where users, based on certified input data representing their activity, resolve a random forest and reveal the classification result to service providers. For the latter, we show how to match a stream of user activity to a regular expression, or how to assign it a probability using a hidden Markov model. Our techniques, based on the use of zero-knowledge proofs, can be composed with other protocols as part of the certification of a larger computation.

1 Introduction

Many popular business models rely on profiling users to deliver customised services. Currently, privacy-sensitive fine-grained data about each user's activities is gathered on the server side to perform this personalization. As an alternative strategy, we propose a set of techniques that allow for user profiling to be performed in a privacy-preserving manner.

By way of motivation, consider a retailer of home heating equipment that wishes to tailor prices or rebates according to a household's occupancy patterns. Similarly, a soft drink manufacturer may wish to provide free drink samples to loyal customers of its competitors. More generally, a marketing company may wish to classify customers according to their lifestyle to better target reduction coupons. A news site may adjust its content according to user preferences on a social network. These are all examples of *personalization*, a strategy that necessitates knowing more about the user, or user *profiling*.

In recent years, several mechanisms have emerged for discerning user preferences on a large scale. The most notable of them is third-party advertising, which

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 18–37, 2012.

involves partially observing user browsing history through third-party cookies to understand user preferences. Similarly, social network platforms allow third-party apps to query user profiles and "likes" for personalization purposes. If this information were only to be used for the purposes of personalization, in both settings the amount of detailed information relating to the user would seem to be excessive. To summarize, the current approach has the following drawbacks:

- **Privacy:** service providers not only learn the profiles of customers, which might in themselves be privacy-sensitive, but also acquire a mass of fine-grained data about users, including click streams, previous purchases, social contacts, etc.
- **Authenticity:** the correctness of profiling is doubtful as the data has been gathered in an ad-hoc manner, often through third party networks or web "bugs" or beacons, which can be blocked and fooled. Moreover, a determined user may interfere with cookie-based profile gathering by geting involved in so-called cookie swaps — rings of people exchanging tracking cookies to confuse the trackers.

In this work, we propose techniques that perform profiling on the client side using state-of-the-art machine learning procedures, namely random forests [11]. Clients can conclusively prove that they have performed the classification task correctly, with reference to certified data representing user activity. These certificates serve as a *root of trust*: a service provider can unreservedly rely on the computed profile for business decisions, such as giving free samples, coupons or rebates, without learning anything more than the decision or fearing that the user is cheating.

In addition to resolving random forests, we show how to match a stream of user activity against a regular expression. If desired, we can also assign a probability to a stream of user activity using a hidden Markov model [4]. This has applications in matching against streams of user activity, such as finding patterns in bank transactions, for example. The same guarantees relating to privacy and authenticity hold in both of these cases.

Previous private data mining techniques are too specific to the problem they solve and can only be used in isolation [44,36]. In contrast, the techniques presented in this paper are *fully composable* with other zero-knowledge protocols. As a result, they may be used as part of certifying a larger computation.

2 System Overview

At a high level, we present cryptographic mechanisms that allow a user to prove some certified features match a class or a sequence, defined by a random forest, regular expression, or a hidden Markov model, respectively.

Our protocol takes place between a prover, a verifier, and two authorities: A and A'. The basic interactions between those entities are illustrated in Figure 1 and described in detail below. We assume that there is an implicit setup phase, during which all principals in the system generate their public/private key pairs and distribute them securely to all other entities. The prover and the verifier also generate signature and verification keys for purposes of authentication.

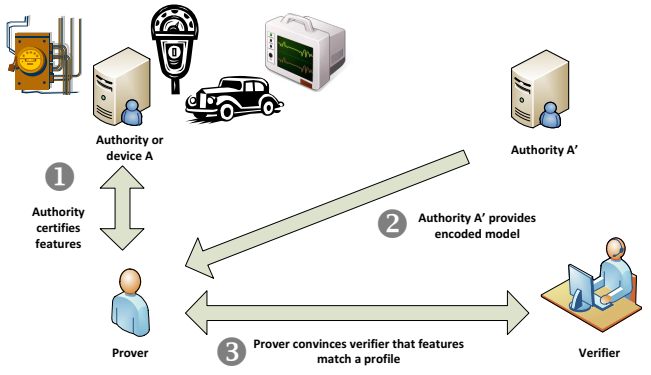


Fig. 1. An overview of the key interactions between all parties in the system

For the purposes of the protocol, a prover must be provided with a set of certified features (Figure 1, step 1). Abstractly, these features are signed by an authority A that is trusted by the verifier to authoritatively associate features with the prover. In practice, this authority could be a third party service, such as a trusted social network platform that observes user actions and provides a signed account of them for third parties to use. Or it could be a trusted hardware device, such as a smart meter that is installed on user premises to provide certified readings. Similarly an automotive on-board unit could provide trusted readings, such as time and vehicle speed, that need to be matched or classified for the purposes of billing for pay-per-mile insurance. Finally, the user could self-certify some features, and even keep them secret, and simply attest them with a third party using a blind signing protocol.. The features need to be encoded in a specific manner as described in Section 5 and Section 6.

A separate entity A', that could be the verifier, cryptographically encodes a decision forest or a hidden Markov model and provides it to the prover (Figure 1, step 2). In Section 5 and Section 6 we describe how to encode random forest and hidden Markov model classifiers to support efficient proof of their operation. In theory, the classifiers themselves could be kept secret from the verifier, since the verification does not require them. This may enable some business models, but is fragile due to the fact that the prover has to know the full details of the classifiers to construct the necessary proofs.

Finally, the prover and verifier interact, in such a way that the verifier is provided with the outcome of the random forest classification or hidden Markov model match probability, along with a proof that it was performed correctly (Figure 1, step 3). No other information about the features is revealed. The subsequent sections of this paper present the details of these proofs. Standard techniques can be used to turn the interactive protocol into a non-interactive one. In that setting, the prover simply constructs the proof and sends it to the verifier that can either accept or reject it.

Our protocols protect privacy in the sense that they leak no more information about the detailed features than what becomes known through the decision of a classifier or a match with a hidden Markov chain. We note that such classifiers could be build to leak specific features, and therefore a user needs to know what being subjected to such profiling entails. Since the classifier is public it can be executed, or not, according to the privacy preference of the prover.

The interactions between the authority certifying the features, the prover and the verifier are reminiscent of anonymous credential protocols. The key difference of our approach from such protocols is that we do not attempt to hide the identity of the prover, but rely on hiding the detailed user features to preserve privacy. The protocols can be extended with relative ease to protect the identity of the user but this goes beyond the scope of this work.

3 Applications

This section outlines some of the applications of the theory presented in this paper. The application focus for this paper is on end-user settings, many of which pertain to the context of everyday life, rather than enterprise or business-to-business scenarios.

Behavioural Advertising. Perhaps, the most common application setting on people's minds today is that of behavioural profiling – web users are classified based on their pattern of visiting web pages, according to either a taxonomy or by associating keywords with their profile, typically, for more accurate ad targeting. Our techniques allow the user to run the classification algorithm *locally* using fine-grained behavioural information — usually, on their URL stream — and only reveal their aggregate profile to an advertiser or web site for the purposes of customization or ad targeting. Our schemes could make use of features certified by social networking sites, or even certified cookies gathered by browsing on the internet. As has been proposed in the literature [40,30,43,26], in this case, a set of ads can be matched against the user profile without revealing precise user web browsing history.

P2P Dating and Matchmaking. A common functionality of on-line social sites is the ability to filter users based on one or more criteria. More specifically, imagine a P2P dating site. The site is decentralized, i.e. P2P in order to avoid sharing rather personal details with the site owner. The process of matchmaking, i.e. filtering of the users with respect to criteria such as music tastes or hair color, would be done locally, on each of the P2P nodes. Of course, we need to make sure that this local filter is not compromised to always cheerfully, but deceitfully answer "yes, I am a match." We can prevent this by requiring users to commit to their preferences and attributes at most once, and then use the same, committed, profile when answering matching queries from different parties.

In this scenario, trust is rooted in the fact that users cannot have their attributes certified at will, and can only do so in a restricted manner: either through

paying some money to get their subscription to the dating site to establish an identity, or any other Sybil defence mechanism [35].

Financial Logs. In banking applications, it is common to look for patterns over user transaction data. One common application is fraud detection. For example, two or more bank charges in a stream of account transactions per month or too many charges by a particular merchant may be indicative of fraud. Banks themselves are often interested in flagging certain kinds of transactions, primarily for fraud prevention reasons. Governments are keen to oversee transactions to avoid tax-evasion or money laundering.

A more specific application using our techniques is a lender who is interested in querying your transaction history for determining if you are a reliable borrower, which might include checking for more than two late fees within a month. Our techniques could use transaction financial logs provided by financial institutions or shops to prove certain properties to third parties, such as a lending institution.

Insurance. Metered automotive insurance is an example of an application where revealing too much about the user's driving habits can be privacy-compromising, as it will leak excessive data about their whereabouts. However, certain behavioural patterns on the part of the user can be utilized to flag dangerous or reckless behaviour, which are legitimate to inquire about for the purpose of providing and charging for automotive insurance.

For instance, a driver that alternates prolonged day and night driving within the same 24- or 48-hour period could be at risk for an accident. Someone who starts the car more than, say, ten times a day does a fair bit of city driving. Such patterns may be mined to determine the right insurance premium. To support such settings, a trusted on-board unit in the car can provide certified readings of speed and time that are mined to detect the presence of certain driving patterns, without revealing the rest.

Bio-medical and Genetic. A number of services have been proposed on the basis of individual genetic or biological profiles. These can include claiming a state benefit due to some debilitating genetic condition or even testing a genetic fingerprint as part of a criminal investigation. Our techniques could be used to support privacy in such settings: an authority could provide a user with a certified generic profile as detailed as their DNA profile. Third parties can then provide certified models that match specific genes or other biological profiles to tune their services. The results computed by these third parties can be shared, yet the detailed biological profile will never be revealed.

3.1 Related Work

Random forests are a very popular classification algorithm first proposed by Breiman [11]. Their performance, efficiency and potential for parallelization

has made random forests popular for image classification [8], and they have been widely used in bio-informatics [24]. Hidden Markov models [31] have found applications in speech recognition [39] as well as bio-informatics [32]. The fact that both techniques have serious medical applications motivates us further to perform classification and matching tasks without revealing raw data.

Previous works have considered privacy in the context of classification tasks. Pinkas [38] argues that arbitrary multi-party computation can be used to jointly perform privacy-preserving data mining when different parties have secrets. The key building block of the presented protocol is oblivious transfer, which requires interactions among parties. Our techniques differ in that the only party with secrets is the user, and all other parties only care about the integrity and authenticity of the classification procedure. As a result, our protocols are not interactive, and are more efficient than generic secure multi-party computations.

Vaideep et al. [44] consider privacy friendly training of decision trees and forests. These techniques are orthogonal and complementary to our algorithms. We provide a zero-knowledge proof that a trained random forest classifier was applied on a secret data set correctly, and do not discuss the privacy of the training phase. Similarly, Magkos et al. [36] use protocols based on secure electronic election to implement a privacy friendly-frequency counter. They discuss how multiple rounds of their protocol can be used to privately train a random forest classifier.

Our cryptographic constructions build upon a number of previous cryptographic techniques, mostly involving zero-knowledge proofs, that we describe in detail in the next section.

4 Cryptographic Foundations

The novel techniques for resolving random forests and matching regular languages make use of modern cryptographic primitives, and in particular efficient zero-knowledge proofs of knowledge. We present here the basic building blocks, namely commitments, zero-knowledge proofs and signatures with efficient proof protocols (P-signatures). We also dwell deeper into how to use P-signatures to perform lookups into a public indexed table without revealing the lookup key or row retrieved.

4.1 Zero-Knowledge, Commitments and P-Signatures

Zero-Knowledge proofs. A zero-knowledge proof of knowledge [6] is a two-party protocol between a prover and a verifier. The prover demonstrates to the verifier her knowledge of some secret input (witness) that fulfills some statement without disclosing this input to the verifier. The protocol should fulfill two properties. First, it should be a proof of knowledge, i.e., a prover without knowledge of the secret input convinces the verifier with negligible probability. Second, it should be zero-knowledge, i.e., the verifier learns nothing but the truth of the statement.

We make use of classical results for efficiently proving knowledge of discrete logarithm relations [42,21,19,12,10,22]. To avoid common pitfalls, and to be compatible with the newest results in the literature, we use the language proposed by Camenisch et al. [15]. Their language is inspired by the CKY-language [14], which formalized and refined the PK notation for proofs of knowledge by Camenisch and Stadler [20]. While proofs of knowledge for the standard model are addressed in [14], Camenisch et al. [15] show how to realize proofs for their language in the UC-framework. As a third option, these proofs can be compiled [25] into non-interactive proofs in the random oracle model [7].

In the notation of [15], a protocol proving knowledge of integers $w_1, \ldots, w_n$ satisfying the predicate $\phi(w_1, \ldots, w_n)$ is described as

$$\text{ℵ}\, w_1 \in \mathcal{I}_1, \ldots, w_n \in \mathcal{I}_n : \phi(w_1, \ldots, w_n) \tag{1}$$

Here, we use the symbol "ℵ" instead of "∃" to indicate that we are proving "knowledge" of a witness, rather than just its existence.

The predicate $\phi(w_1, \ldots, w_n)$ is built up from "atoms" using arbitrary combinations of ANDs and ORs. An atom expresses *group relations*, such as $\prod_{j=1}^{k} g_j^{\mathcal{F}_j} = 1$, where the g_j are elements of an abelian group and the $\mathcal{F}_j$'s are integer polynomials in the variables $w_1, \ldots, w_n$.

Instead of using group relations directly, we rely on classical results that show how to reduce the following proof components to group relations: (1) linear relations and equality, (2) inequalities, and (3) proofs about commitments and signatures.

For the prime-order and the hidden-order setting there are different techniques for dealing with inequalities and P-signature possession. We refer to the literature for more details. We consciously keep our presentation independent of the lower-level cryptography by using an abstract notation for zero-knowledge statements.

Inequalities. A couple of techniques are available to prove that a value is positive. Groth [29], who builds on [9], relies on the fact that a number can be expressed using the sum of 3 squares. Alternatively, the value can be shown to be within the set of positive integers as provided by a trusted authority, using set membership techniques by Camenisch et al. [13].

Commitment schemes. A non-interactive commitment scheme consists of the algorithms $\mathsf{ComSetup}$, Commit and Open. $\mathsf{ComSetup}(1^k)$ generates the parameters of the commitment scheme par_c. $\mathsf{Commit}(par_c, x)$ outputs a commitment C_x to x and auxiliary information $open_x$. A commitment is opened by revealing $(x, open_x)$ and checking whether $\mathsf{Open}(par_c, C_x, x, open_x)$ outputs accept. The hiding property ensures that a commitment C_x to x does not reveal any information about x, whereas the binding property ensures that C_x cannot be opened to another value x'. Our scheme requires commitment schemes that support an efficient proof predicates $\mathrm{POpen}_{C_x}(x, open_x)$ for the opening algorithm of commitment C_x [37,27].

P-signatures. A signature scheme consists of algorithms (Keygen, Sign, Verify). $\mathsf{Keygen}(1^k)$ outputs a key pair (sk, pk). $\mathsf{Sign}(sk, m)$ outputs a signature s on message m. $\mathsf{Verify}(pk, s, m)$ outputs accept if s is a valid signature on m and reject otherwise. This definition can be extended to support multi-block messages $\boldsymbol{m} = \{m_1, \ldots, m_n\}$. Existential unforgeability [28] requires that no probabilistic polynomial time (p.p.t.) adversary should be able to output a message-signature pair (s, m) unless he has previously obtained a signature on m.

One important proof component that we use in many of our constructions is a predicate $\mathrm{PVerify}_{pk}(s, m_1, \ldots m_n)$ for the verification algorithm of a signature scheme. This allows to prove possession of a signature s, while keeping the signature itself and parts of the signed message secret. Signature schemes for which verification is efficiently provable are thus very attractive for protocol design, and are variously referred to as CL-signatures or P-signatures, i.e., signatures with efficient protocols [17,18,5].

Extended notation for zero-knowledge statements. We extend the notation above to abstract the details of proofs of knowledge that are used to verify private computations. In particular we allow the definition of named proof components that can be reused as sub-components of larger proofs.

We introduce the special notation $\mathrm{F}(b) \rightarrow (a)$ to abstract details of proof components. We call this a proof component declaration. For example we may name a proof F on some secret inputs and outputs as being equivalent to a statement in zero-knowledge in the following manner: $\mathrm{F}(a) \rightarrow (b) \equiv \nexists\, a, b : b = a + 1$. Semantically, the function F represents the proof that a counter was incremented. Secret a is the initial value and secret b the new value. Whether a variable appears on the left or the right is somewhat arbitrary and primarily meant to give useful intuition to users of the component. In terms of cryptography, it is simply syntactic sugar for the equivalent statement above.

Named proof components can be used in further higher-level proofs without their details being made explicit. For example, the proof $\nexists\, c, d : d = \mathrm{F}(c)$ is equivalent to the two statements above. All variables within the component declaration (e.g. variables a, b in $\mathrm{F}(a) \rightarrow (b)$) can be re-used in the high level proof. Any variables whose knowledge is proved, but that are not in the declaration, are considered inaccessible to the higher-level proof.

4.2 Direct Lookups

A P-signature on a sequence of messages, representing a number of keys and values, can be used to prove the equivalent of a table look-up [41].

Consider a public table T of keys k_i each mapping to the corresponding values v_i. Keys and values themselves can have multiple components $k_i = (k_{(i,0)}, \ldots, k_{(i,n-1)})$ and $v_i = (v_{(i,0)}, \ldots, v_{(i,m-1)})$.

A trusted authority A can encode this table to facilitate zero-knowledge look-ups by providing a set of P-signatures t_i:

$$\forall i. \quad t_i = \mathsf{Sign}(\mathrm{sk}_A, \langle T_{id}, n, m, k_{(i,0)}, \ldots, k_{(i,n-1)}, v_{(i,0)}, \ldots, v_{(i,m-1)} \rangle) \qquad (2)$$

To prove that a set of secrets (k', v') corresponds to a lookup in the table it is sufficient to show that:

$$\text{LOOKUP}_T(k'_0, \ldots, k'_{n-1}) \rightarrow (v'_0, \ldots, v'_{m-1}) \equiv \tag{3}$$

$$\nexists\, t, k'_0, \ldots, k'_{n-1}, v'_0, \ldots, v'_{m-1} : \tag{4}$$

$$\text{PVerify}_{pk_A}(t, T_{id}, n, m, k'_0, \ldots, k'_{n-1}, v'_0, \ldots, v'_{m-1}) \tag{5}$$

The predicate PVerify corresponds to the verification equations of the signature scheme. The proof hides the keys k' and corresponding values v', but not the identity of the table used to perform the lookup. To avoid any confusion, we always specify the table as part of the name of the lookup.

A variant of the proof allows for range lookups. In this case, a table T' contains as key a pair of values $(k_{(i,\min)}, k_{(i,\max)})$ defining a range, corresponding to a set of values $(v_{(i,0)}, \ldots, v_{(i,m-1)})$. We assume that the ranges defined by the keys do not overlap.

We denote the creation of the table with columns a, b and c, signed by authority A, by an algorithm called $\text{ZKTABLE}_A([a_i, b_i, c_i])$. The cost of performing a lookup in terms of computation and communication is $O(1)$ and constructing a table requires $O(n)$ signatures in the size of the table.

5 Random Forests

Random forests are a state-of-the-art classification technique. A random forest is a collection of decision trees. Each decision tree is trained on a subset of a training dataset. Once the set of trees is trained, it can be used to classify unseen data items. Each decision tree is used separately to classify the item using its features and the majority decision is accepted as valid.

The aim of our protocol is to apply the random forest classification algorithm to a feature set that remains secret. At the same time, we wish to provide a proof that the classification task was performed correctly. We assume that the random forest classifier is already trained and public.

5.1 Vanilla Training and Resolution of Random Forests

This section provides a short overview of how a random forest is grown from a training set, as well as how it is used to classify data items, without any security considerations. Full details are available in Breiman [11]. The notation and resolution algorithms will be used as part of the privacy friendly protocols.

Consider a labelled training data set comprising items d_i. Each item has a set of M features, denoted as $F_m(d_i)$ for m in $[0, M-1]$. Each item is labelled with l_i into one of two classes c_0 or c_1.

The training algorithm takes two parameters: the depth D and the number of items N to be used to train each tree. Training each tree proceeds by sampling N items, with replacement, from the available data items d_i. Then, at each branch, starting with the root, a random feature is selected. The value of the

feature that best splits the data items corresponding to this branch into the two separate classes is used to branch left or right. The algorithm proceeds to build further branches until the tree has reached the maximum depth D. Leaves store the decision or the relative number of items in each class.

A data item d that has not been previously seen is assigned a class by resolving each tree t in the forest. Starting at the root, the branching variable b and its threshold τ is used to decide whether to follow the right or left branch according to the corresponding feature $F_b(d')$. The process is repeated until a leaf is reached and the relative weight of belonging to two classes is stored as (c_{t0}, c_{t1}).

The sum of the weights corresponding to the two classes can be computed as $(\sum_t c_{t0}, \sum_t c_{t1})$. These sums represent the relative likelihood of the item belonging to the corresponding classes.

5.2 Encoding of Features and Trees

The key objective of our algorithm is to correctly execute the random forest classification algorithm on certified features without revealing the features. To enable this approach, both the user data and the random forest need to be cryptographically encoded in a specific manner.

First, the profile features are encoded as a table supporting zero knowledge lookups (as described in Section 4.2). A user u, with a profile with features k_i taking values v_i, has to be provided by an authority A with a table: $\mathrm{F}_u \equiv \mathrm{ZKTABLE}_A([k_i, v_i, u])$.

Second, we need to cryptographically encode each tree of the random forest. Each decision tree is individually encoded by building two tables supporting zero-knowledge lookups: one table for *branches* and one table for *leaves*.

Each non-leaf node of the decision tree is encoded as two separate left and right branches. Branches are encoded by an entry containing a node id n_i, a feature id k_i, and a range for the value $[v_{\min}, v_{\max}]$ as well as target node id n'_i. The node id n_i represents the node of the tree, while k_i is the feature used at this node to make a decision. If the branch feature is indeed in $[v_{\min,i}, v_{\max,i}]$, then the algorithm should proceed to n'_i. Representing an interval that has to be matched represents a uniform way to encode both left and right branches: left branches are encoded with an interval $[v_{\min}, v_{\max}] \equiv [\mathrm{MIN}_i, \tau_i - 1]$ and right branches with an interval $[v_{\min}, v_{\max}] \equiv [\tau_i, \mathrm{MAX}_i]$, where τ_i is the threshold of feature k_i that has a domain $[\mathrm{MIN}_i, \mathrm{MAX}_i]$. An authority A' encodes branches as: $\mathrm{B}_t \equiv \mathrm{ZKTABLE}_{A'}([n_i, k_i, v_{\min,i}, v_{\max,i}, n'_i])$.

Leafs are encoded separately to store the decision for each decision tree. Each leaf has a node id n_i and two values $c_{0,i}, c_{1,i}$ representing the relative likelihood of the item being in two classes respectively. The table representing leaves signed by A' is $\mathrm{L}_t \equiv \mathrm{ZKTABLE}_{A'}([n_i, c_{0,i}, c_{1,i}])$.

Branches and leaves of decision trees are encoded in order to certify its correctness when verifying proofs of signature possession. It is worth noting that they are public vis-a-vis the prover. In particular, the node id of the root node of each tree is known to all as is the fixed depth D of trees.

5.3 Private Resolution for Random Forests

Given a table of user features F_u, and a set of $|t|$ trees given by the sets of branch and leaf tables $\mathrm{B}_t, \mathrm{L}_t$, we present the proofs of correct classification.

In a nutshell, we can decompose the zero-knowledge proof required into performing the classification using the separate trees, and then aggregating and revealing the forest decision (without revealing the intermediate decisions). Each decision tree is matched by proving sequentially that the feature considered at each branch satisfies the prescribed condition.

The zero knowledge proof of correct resolution of a single tree is:

$$\mathrm{RESOLVETREE}_t(F_u, B_t, L_t) \rightarrow (c_{0,t}, c_{1,t}) \equiv \tag{6}$$
$$\not\exists\, \forall j. n_j, k_j, v_{\min,j}, v_{\max,j}, v_j, c_{0,t}, c_{1,t} : \tag{7}$$
$$n_0 = \mathrm{start} \wedge \tag{8}$$
$$\text{for } j = 0 \ldots D-2 : \{ \tag{9}$$
$$n_j, k_j, v_{\min,j}, v_{\max,j}, n_{j+1} = \mathrm{LOOKUP}_{B_t}(n_j) \wedge \tag{10}$$
$$v_j, u = \mathrm{LOOKUP}_{F_u}(k_j) \wedge \tag{11}$$
$$v_{\min,j} \leq v_j \wedge v_j \leq v_{\max,j} \} \wedge \tag{12}$$
$$(c_{0,t}, c_{1,t}) = \mathrm{LOOKUP}_{L_t}(n_{D-1}) \tag{13}$$

The RESOLVETREE proof works by starting with a record of the known root node, and following $D-2$ branches to a leaf. For each branch, the proof demonstrates that it follows the previous branch, and that the feature variable concerned falls within the valid branch interval. The appropriate feature is looked-up in the feature table without revealing its identity or value. Note that n_0, u, D and j are not secret, which leads to some efficiency improvement when implementing the proof for the root branch.

To resolve a forest of $|t|$ trees, the following proof is required:

$$\mathrm{RESOLVEFOREST}_t(F_u, \forall t. B_t, L_t) \rightarrow (c_0, c_1) \equiv \tag{14}$$
$$\not\exists\, \forall t. c_{0,t}, c_{1,t}, c_0, c_1 : \tag{15}$$
$$\text{for } t = 0 \ldots |t| - 1 : \{ \tag{16}$$
$$(c_{0,t}, c_{1,t}) = \mathrm{RESOLVETREE}_t(F_u, B_t, L_t) \} \wedge \tag{17}$$
$$(c_0, c_1) = (\sum_t c_{0,t}, \sum_t c_{1,t}) \wedge \tag{18}$$
$$\mathrm{POpen}_{C_{c_0}}(c_0, open_{c_0}) \wedge \mathrm{POpen}_{C_{c_1}}(c_1, open_{c_1}) \tag{19}$$

The RESOLVEFOREST proof uses the tree resolution as a sub-component. The secret values returned from each tree are then aggregated into committed values (c_0, c_1). These values can be revealed, or kept secret and used as part of a further protocol (for example to prove that the probability of the user belonging to a certain profile is higher than a set threshold).

5.4 Extensions

Decision forests can be used to classify items into more than two classes. To achieve this, the leaf table can encode likelihoods of multiple classes, and the aggregation algorithm can sum the evidence for more than two classes in parallel.

Similarly, more than one feature can be used to branch at each node. In such cases, each branch needs to encode the identity and interval for all necessary features. To maintain indistinguishability among all branches, the same number of features must be tested for all of them.

6 HMMs, Probabilistic Automata and Regular Languages

Regular languages can be used to match streams of symbols to a set of rules based on an initial state and possible transitions between states. Probabilistic automata extend regular languages to consider specific probabilities associated with initial states and transitions. Finally, hidden Markov models dissociate states from emitted symbols. We show how a client can match a sequence of certified actions or logs to a given model without revealing the specific actions or log entries. We illustrate our techniques using hidden Markov models (HMM), as both probabilistic automata and regular languages can be considered special cases of such models. We also discuss how matching simpler models can be implemented more efficiently.

6.1 Vanilla Matching for Hidden Markov Models

We first review the HMM matching algorithms in the absence of any privacy protection. A complete tutorial on hidden Markov model based techniques can be found in [39].

A hidden Markov model is fully defined by three tables: an *initial state* table, a *transition* table, and a *symbol emission* table. All tables refer to a finite set of hidden states and observed symbols. The initial state table maps possible initial states of the model to a probability. The state transition table maps state pairs to a probability. Finally, the symbol emission table maps state and symbol tuples to a probability. We assume that any entry missing from a table is implicitly assigned a probability of zero. By convention, a specific state is considered to be the final state of a model, and matching ends there.

Matching a sequence of symbols starts with assigning the first symbol both a state and a probability from the initial state table, and, given the initial state, a probability of emitting the observed first symbol from the symbol emission table. Matching then proceeds through the sequence of observed symbols by selecting a new hidden state for each symbol. The probability of state transition and the probability of emitting the subsequent symbol from the new state can be retrieved from the state transition table and the symbol emission table respectively.

The total probability of a match is the product of all the probabilities from the initial state to the final state. This product can also be expressed as the sum of the logarithms of all the probabilities.

6.2 Encoding of Hidden Markov Models

To enable a user to prove that a sequence of their actions or log is matched by a hidden Markov model, both their actions and the HMM need to be encoded in a specific manner.

First, we consider how an authority A encodes a sequence of symbols relating to a user. This sequence can represent specific user actions or generic log entries relating to a user u. Each symbol in the sequence is represented by an individual entry within a table supporting zero-knowledge lookups $\mathrm{S}_u \equiv \mathrm{ZKTABLE}_A([t_i, e_i, u])$, where t_i is a sequence number and e_i represents the symbol observed.

Each table of the hidden Markov model is also encoded using tables that allow for zero-knowledge lookups signed by authority A'. The initial symbol table I_t contains: $\mathrm{I}_t \equiv \mathrm{ZKTABLE}_{A'}([s_i, p_i])$, where s_i is a possible initial state and p_i the logarithm of its probability. The state transition table is represented as: $\mathrm{T}_t \equiv \mathrm{ZKTABLE}_{A'}([s_i, s_j, p_i])$, where s_i and s_j are states between which there is a transition with log-probability p_i. Finally, the emission table for each state s_i and potential emitted symbol e_j is encoded as: $\mathrm{E}_t \equiv \mathrm{ZKTABLE}_{A'}([s_i, e_j, p_i])$.

We implicitly assign a zero probability to any entries that are not encoded in the above tables. No proof of a match can be produced for zero-probability matches. Furthermore, we store all probabilities as a positive integer representing the quantised version of their negative logarithm. Therefore, the expression p_i used above is a shorthand for $-\lceil 10^{\psi} \log \pi_i \rceil$ in case ψ decimal digits of the logarithm are to be used where π_i is the raw probability.

6.3 Private Matching

A user with a certified sequence of actions S_u can prove that it matches a hidden Markov model described by a triplet of tables (I_t, T_t, E_t). This can be done without revealing the symbols in the sequence of actions, their position within the sequence and without revealing the matched sequence or states or matching probability. In practice, it might be necessary to reveal the position of the match, the probability of the match or some hidden states for subsequent processing. We present the algorithms where only commitments to these are produced, which can be selectively opened to reveal interesting values.

The MATCH function proves that a sequence of hidden states $s_0 \ldots s_{l-1}$ match the sequence of symbols $e_0 \ldots e_{l-1}$ present, at the offset t_0 of the user sequence, with probability p.

$$\text{MATCH}(S_u, I_t, T_t, E_t) \rightarrow (s_0 \dots s_{l-1}, e_0 \dots e_{l-1}, t_0, p) \equiv \tag{20}$$
$$\nvDash \forall i \in [0, l-1].t_i, s_i, e_i, p_i, p'_i, p : \tag{21}$$
$$e_0, u = \text{LOOKUP}_{S_u}(t_0) \wedge \tag{22}$$
$$p_0 = \text{LOOKUP}_{I_t}(s_0) \wedge \tag{23}$$
$$p'_0 = \text{LOOKUP}_{E_t}(s_0, e_0) \wedge \tag{24}$$
$$\text{for } i = 1 \dots |i| - 1 : \{ \tag{25}$$
$$t_i = t_0 + i \wedge \tag{26}$$
$$e_i, u = \text{LOOKUP}_{S_u}(t_i) \wedge \tag{27}$$
$$p_i = \text{LOOKUP}_{T_t}(s_{i-1}, s_i) \wedge \tag{28}$$
$$p'_i = \text{LOOKUP}_{E_t}(s_i, e_i)\} \wedge \tag{29}$$
$$p = \sum_{0}^{l-1}(p_i + p'_i) \wedge \tag{30}$$
$$\text{POpen}_{C_p}(p, open_p) \wedge \text{POpen}_{C_{s_{l-1}}}(s_{l-1}, open_{s_{l-1}}) \tag{31}$$

The main result of the protocol is a commitment $C_{s_{l-1}}$ to the final state s_{l-1} and a commitment C_p to the probability p of reaching this state. The prover can open $C_{s_{l-1}}$ and C_p. The raw probability of a match can be computed as $\pi = e^{10^{-\psi} \cdot -p}$. Instead of opening the commitment corresponding to the probability of the match p the user could prove in zero knowledge that it is within a certain range.

We note that we do not hide the length of the match denoted by l. The user is also free to commit to and open commitments to any of the values $s_0 \dots s_{l-1}$, $e_0 \dots e_{l-1}$, t_0 to reveal more information to the verifier.

We note that the above procedure proves there is a match and returns its probability, but does not guarantee in itself that this is the best match, i.e. the match with the highest probability. Such a procedure is likely to be computationally more expensive, and thus not really practical for real-world applications. In comparison, the above match algorithm requires only a linear number of proofs in the length of the match.

6.4 Simplifications for Finite Automata and Regular Languages

Using the match algorithm for HMMs, we can trivially match regular languages and probabilistic automata. For regular languages, we assume that each state only emits one symbol with probability one, and that all probability values are equal. For finite automata, we extend the encoding for regular languages to allow for different probabilities for the initial state and transitions between states. In both cases, the matching algorithm can be simplified to avoid proving the link between symbols and states, as they are in fact the same entities.

Similarly, for regular languages there is no need to store or process probabilities as the length of the match contains all the necessary information.

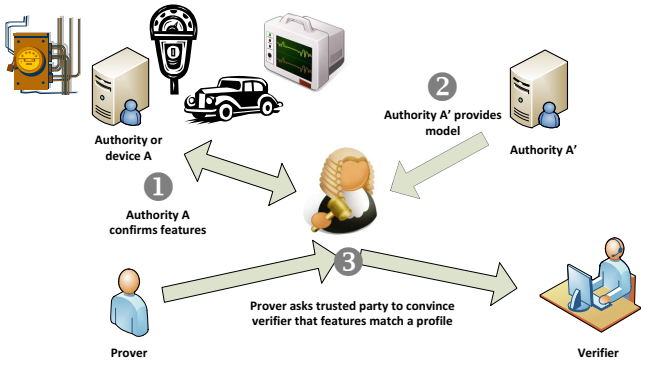


Fig. 2. An ideal protocol with a trusted party

7 Discussion

7.1 Security

Security and privacy properties. The security and privacy properties of our client-side profiling schemes can be described via an ideal protocol depicted in Figure 2 that carries out the same task. In the ideal protocol, a trusted party T receives all the inputs of the parties involved in the protocol, performs the required computations, and provides each party with its output.

As mentioned in Section 2, we have a setting with four parties: provers P, verifiers V, an authority A that certifies prover's profiles, and an authority A' that certifies the model. In the ideal protocol, first, A' sends T the model, which is either the random forest or the hidden Markov model. P sends T its features, which forwards them to A. A replies T whether the features are correct, and, if it is the case, T stores them. When P tells T to send the result of the profiling task to V, T employs P's features and the received model to perform the profiling task and sends V the result.

As can be seen, in the ideal protocol V only learns the result, which protects P's privacy. The resolution of the profiling task is performed by T, which is trusted, and thus the correctness of the result is guaranteed.

Our scheme mirrors the security and privacy properties of the ideal protocol while limiting the involvement of trusted parties. Namely, our protocols require A and A' to be trusted when certifying the correctness of provers' profiles and of the model respectively. Once this is done, A and A' do not participate in the protocol anymore. P and V can be adversarial, and yet our scheme guarantees that, as in the ideal protocol, the result computed by P is correct and V does not learn more information on P's features. We note that our protocols reveal to V a small amount of information on the model, such as the depth of the decision trees. Such information is also revealed to V by the ideal functionality in the ideal protocol.

Secure implementation of our protocols. It may seem that the provable guarantees afforded by zero-knowledge proofs of knowledge, particularly when using the compilation techniques of [15] to generate a universally composable protocol, are all that is needed to implement such an ideal functionality, and thus to automatically guarantee the authenticity and privacy of the classification result. The soundness of proofs guarantees the correctness and authenticity of the classification task towards verifiers, and the fact that the protocols are zero-knowledge guarantees to users that nothing besides the result of the classification is revealed.

Our system is, however, not only a zero-knowledge proof of knowledge. It also consists of input generators that provide auxiliary information for the proofs, e.g., signed lookup tables. Unfortunately, as noted by [14] many zero-knowledge proof protocols designed with only honestly generated input in mind, are not portable, i.e., they do not necessarily retain their properties when executed on malicious input. In our system the input generation stage involves the prover P, verifiers V, an authority A that certifies users profiles, and an authority A' that certifies the tables and algorithms used for classification. When considering *authenticity* A and A' are considered as being fully trusted by the verifier. For authenticity, portability is thus naturally maintained. For *privacy* we consider A and A', however, as potentially colluding with the verifier. To prevent the attacks of [33] the prover needs to be able to check that the cryptographic group parameters provided by A and A' are well formed as required in [14]. In our case these parameters primarily consist of P-signature public keys, and commitment parameters. To meet their security properties these schemes should already be carefully designed to be portable. Note that provers need to verify the correct generation of these public keys and commitment parameters only once in the *setup* phase.

Portability is significantly easier for prime order groups. Also in light of [34], using a single pairing-based elliptic curve could thus be a conscious design decision, if one is willing to live without Strong RSA based CL-signatures (there are pairing-based alternatives [18]) and sum-of-squares based range proofs [9]. This may, however, exclude credential issuing authorities [16] that only support the more mature Strong RSA based CL-signatures.

We do not provide a concrete implementation and leave many low-level design decisions underspecified. We advice against adhoc implementations, and recommend the use of zero-knowledge compilation frameworks such as [3,1] that are approaching maturity.

Systems security issues. A malicious service provider could utilize our client-side profiling schemes and yet employ covertly web-bugs, cookies or other techniques to obtain further information about users. We point out that existing countermeasures, e.g., [2] for web-bugs, can be employed to prevent that. When used in combination with out scheme, service providers still get the result of the profiling task.

7.2 Efficiency of Proposed Schemes

Encoding a table for zero-knowledge lookups using $\text{ZKTABLE}_A(\cdot)$ requires one re-ranomizable signature per encoded row. The LOOKUP operation requires the proof of possession of a single P-signature, and RLOOKUP additionally requires a range proof to ensure the key is within a secret range.

To prove that a value x lies within an interval $[a, b]$, it is necessary to show that $x - a \geq 0$ and $b - x \geq 0$. The techniques of Groth [29] requiring 3 additional commitments and 3 zero-knowledge proofs of multiplication. Using set membership techniques by Camenisch et al. [13] requires two proof of possession of a signature per range proof.

The RESOLVETREE procedure relies heavily on zero-knowledge lookups and range proofs: two LOOKUP and a range proof are needed per branch followed in the tree. A single lookup is needed to retrieve the leaf with the decision. In total $4 \cdot (D-1)+1$ proofs of possession of a signature are needed to resolve each decision tree, where D is the depth of the tree. Aggregating the decisions of multiple trees in RESOLVEFOREST can be done very efficiently using homomorphic properties of commitments.

The MATCH procedure to resolve hidden Markov models requires 3 lookups per symbol matched in the general case. This can be reduced to two lookups if only regular automata are used (and the emitted symbols are equivalent to the matched states). In case the location of the sequence into the user stream is not secret, the stream can be encoded as a sequence of commitments, and a single lookup is necessary per matched symbol.

8 Conclusion

A rule for building secure system states that all user input must be positively validated to be correct. Traditionally, any computation performed on the client-side cannot be trusted. As a result, personal information must be available to be processed on the server side, which leads to a number of privacy problems.

In this paper, we extend the paradigm of secure client-side computation [23], and show that complex classification and pattern recognition tasks on user secrets can be proved correct. We heavily rely on the efficiency of proving possession of P-signatures to efficiently implement random forest and hidden Markov model matching.

Our techniques allow for classification and matching without any unnecessary leakage of other information. Matching and classification are often necessary to transform unstructured data into information that can be acted upon: to perform authentication, to tune a business decision, to detect intrusion, to reason about a stream of activity.

Yet, in many contexts *the mere act of profiling* in itself can violate privacy. Recent news stories, for example, describe how data mining techniques can be used to detect whether one is pregnant for the purposes of marketing from shifts

in their buying habits[1]. Our techniques could deliver comparative functionality without the need to reveal the detailed shopping list of a customer. The extent to which one should be subjected to such profiling at all is highly debatable.

Providing a definite answer as to the necessity or oppression inherent in profiling is beyond the scope of this work. Those applying it to specific settings would be advised to reflect on the remarks Roland Barthes made at his inaugural lecture at the *College de France*: "*We do not see the power which is in speech because we forget that all speech is a classification, and that all classifications are oppressive.*"[2]

Acknowledgements. The authors would like to thanks the anonymous reviewers and the paper shepherd, Ayday Erman, for suggestions that improved this work.

References

1. Almeida, J.B., Bangerter, E., Barbosa, M., Krenn, S., Sadeghi, A.-R., Schneider, T.: A Certifying Compiler for Zero-Knowledge Proofs of Knowledge Based on Σ-Protocols. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 151–167. Springer, Heidelberg (2010)
2. Alsaid, A., Martin, D.: Detecting Web Bugs with Bugnosis: Privacy Advocacy through Education. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 13–26. Springer, Heidelberg (2003)
3. Bangerter, E., Briner, T., Henecka, W., Krenn, S., Sadeghi, A.-R., Schneider, T.: Automatic Generation of Sigma-Protocols. In: Martinelli, F., Preneel, B. (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 67–82. Springer, Heidelberg (2010)
4. Baum, L.E., Petrie, T.: Statistical Inference for Probabilistic Functions of Finite State Markov Chains. The Annals of Mathematical Statistics 37(6), 1554–1563 (1966)
5. Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and Noninteractive Anonymous Credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)
6. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
7. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: First ACM Conference on Computer and Communication Security, pp. 62–73. Association for Computing Machinery (1993)
8. Bosch, A., Zisserman, A., Muoz, X.: Image classification using random forests and ferns. In: IEEE 11th International Conference on Computer Vision, ICCV 2007, pp. 1–8. IEEE (2007)
9. Boudot, F.: Efficient Proofs that a Committed Number Lies in an Interval. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer, Heidelberg (2000)

[1] Charles Duhigg. How Companies Learn Your Secrets. The New York Times. February 16, 2012.

[2] In Barthes: Selected Writings (1982). Lecon (1978).

10. Brands, S.: Rapid Demonstration of Linear Relations Connected by Boolean Operators. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 318–333. Springer, Heidelberg (1997)
11. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
12. Camenisch, J.: Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem. PhD thesis, ETH Zürich (1998)
13. Camenisch, J.L., Chaabouni, R., Shelat, A.: Efficient Protocols for Set Membership and Range Proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008)
14. Camenisch, J., Kiayias, A., Yung, M.: On the Portability of Generalized Schnorr Proofs. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 425–442. Springer, Heidelberg (2009)
15. Camenisch, J., Krenn, S., Shoup, V.: A Framework for Practical Universally Composable Zero-Knowledge Protocols. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 449–467. Springer, Heidelberg (2011)
16. Camenisch, J.L., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
17. Camenisch, J.L., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
18. Camenisch, J.L., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
19. Camenisch, J., Michels, M.: Proving in Zero-Knowledge that a Number Is the Product of Two Safe Primes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 107–122. Springer, Heidelberg (1999)
20. Camenisch, J.L., Stadler, M.A.: Efficient Group Signature Schemes for Large Groups. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 410–424. Springer, Heidelberg (1997)
21. Chaum, D., Pedersen, T.P.: Wallet Databases with Observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
22. Cramer, R., Damgård, I.B., Schoenmakers, B.: Proof of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
23. Danezis, G., Livshits, B.: Towards ensuring client-side computational integrity. In: Cachin, C., Ristenpart, T. (eds.) CCSW, pp. 125–130. ACM (2011)
24. Díaz-Uriarte, R., De Andres, S.: Gene selection and classification of microarray data using random forest. BMC Bioinformatics 7(1), 3 (2006)
25. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987)
26. Fredrikson, M., Livshits, B.: Repriv: Re-imagining content personalization and in-browser privacy. In: IEEE Symposium on Security and Privacy, pp. 131–146. IEEE Computer Society (2011)
27. Fujisaki, E., Okamoto, T.: Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
28. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)

29. Groth, J.: Non-interactive Zero-Knowledge Arguments for Voting. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 467–482. Springer, Heidelberg (2005)
30. Guha, S., Cheng, B., Francis, P.: Privad: Practical Privacy in Online Advertising. In: Proceedings of the 8th Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA (March 2011)
31. Juang, B.: Hidden markov models. Encyclopedia of Telecommunications (1985)
32. Karplus, K., Barrett, C., Hughey, R.: Hidden markov models for detecting remote protein homologies. Bioinformatics 14(10), 846–856 (1998)
33. Kunz-Jacques, S., Martinet, G., Poupard, G., Stern, J.: Cryptanalysis of an Efficient Proof of Knowledge of Discrete Logarithm. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 27–43. Springer, Heidelberg (2006)
34. Lenstra, A.K., Hughes, J.P., Augier, M., Bos, J.W., Kleinjung, T., Wachter, C.: Ron was wrong, whit is right. Cryptology ePrint Archive, Report 2012/064 (2012), http://eprint.iacr.org/
35. Levine, B.N., Shields, C., Margolin, N.B.: A survey of solutions to the sybil attack (2006)
36. Magkos, E., Maragoudakis, M., Chrissikopoulos, V., Gritzalis, S.: Accurate and large-scale privacy-preserving data mining using the election paradigm. Data & Knowledge Engineering 68(11), 1224–1236 (2009)
37. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
38. Pinkas, B.: Cryptographic techniques for privacy-preserving data mining. SIGKDD Explorations 4(2), 12–19 (2002)
39. Rabiner, L.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
40. Reznichenko, A., Guha, S., Francis, P.: Auctions in Do-Not-Track Compliant Internet Advertising. In: Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS), Chicago, IL (October 2011)
41. Rial, A., Danezis, G.: Privacy-Preserving Smart Metering. In: Proceedings of the 11th ACM Workshop on Privacy in the Electronic Society (WPES 2011). ACM, Chicago (2011)
42. Schnorr, C.: Efficient signature generation for smart cards. Journal of Cryptology 4(3), 239–252 (1991)
43. Toubiana, V., Narayanan, A., Boneh, D., Nissenbaum, H., Barocas, S.: Adnostic: Privacy preserving targeted advertising. In: NDSS (2010)
44. Vaidya, J., Clifton, C., Kantarcioglu, M., Patterson, A.S.: Privacy-preserving decision trees over vertically partitioned data. TKDD 2(3) (2008)

Understanding Statistical Disclosure: A Least Squares Approach

Fernando Pérez-González[1,2,3] and Carmela Troncoso[4]

[1] Signal Theory and Communications Dept., University of Vigo
[2] Gradiant (Galician R&D Center in Advanced Telecommunications)
[3] Electrical and Computer Engineering Dept., University of New Mexico
fperez@gts.uvigo.es
[4] K.U. Leuven/IBBT, ESAT/SCD-COSIC
carmela.troncoso@esat.kuleuven.be

Abstract. It is widely accepted that Disclosure Attacks are effective against high-latency anonymous communication systems. A number of Disclosure Attack variants can be found in the literature that effectively de-anonymize traffic sent through a threshold mix. Nevertheless, these attacks' performance has been mostly evaluated through simulation and how their effectiveness varies with the parameters of the system is not well-understood. We present the LSDA, a novel disclosure attack based on the Maximum Likelihood (ML) approach, in which user profiles are estimated solving a Least Squares problem. Further, contrary to previous heuristic-based attacks, our approach allows to analytically derive formulae that characterize the profiling error of the LSDA with respect to the system's parameters. We verify through simulation that our predictors for the error closely model reality, and that the LSDA recovers users' profiles with greater accuracy than its predecessors.

1 Introduction

Mixes, relaying routers that hide the relation between incoming and outgoing messages [2], are one of the main building blocks for high-latency anonymous communications [4,6,9,16]. A variety of Disclosure or Intersection Attacks [1, 3,7,8,11,12,18,19,21] have been proposed to uncover persistent and repeated patterns of communication taking place through a mix. In a nutshell, these attacks find a target user's likely set of friends, also known as user profile, by intersecting the recipient anonymity sets of the messages this user sends.

Even though all attacks operate on the same principle they differ on how they exploit the observations in order to obtain user profiles. Statistical variants [3,7, 21] rely on heuristics to operate, while the Bayesian inference-based method by Danezis and Troncoso [8] use Bayesian sampling techniques to recover accurately users' profiles in more complex systems than its predecessors. For any of these attacks it is difficult to obtain analytic results that characterize the dependence of their effectiveness on the parameters of the system, and hence they (as well as their sequels [5,14,15,18]) have been mostly evaluated through simulation.

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 38–57, 2012.

In this paper we propose a novel profiling attack based on the Maximum Likelihood (ML) approach. The attack estimates user profiles by solving a Least Squares problem, ensuring that the mean squared error between the real and estimated profiles is minimized. We empirically show that our attack indeed minimizes the mean squared error with respect to heuristic disclosure attack variants [3, 7, 21], although it performs slightly worse than the Bayesian approach [8] in the scenarios considered in this paper.

Nevertheless, we note that the most outstanding feature of the Least Squares approach is that, contrary to its predecessors, it allows us to derive analytical expressions that describe the evolution of the profiling error with the parameters of the system. This is a key property, as it allows designers to choose system parameters that provide a certain level of protection without the need to run simulations. We thoroughly validate our results through simulation, proving that our formulae reliably predict the evolution of our attack's error as the parameters of the system change.

The rest of the paper is organized as follows: in the next section we revisit previous work on Disclosure Attacks and we describe our system and adversarial models in Sect. 3. We introduce the Least Squares approach to disclosure in Sect. 4, where we derive equations that characterize its error with respect to the system parameters which we validate in Sect. 5. Finally, we discuss future lines of work in Sect. 6, and we conclude in Sect. 7.

2 Related Work

We can find different flavors of disclosure attacks in the literature [1,3,7,8,10–13, 18,19,21] which we now proceed to revisit. A first family of intersection attacks are the so-called Disclosure Attack [1,10] and its sequels [11–13,18]. These attacks rely on Graph Theory in order to uncover the recipient set of a target user Alice. They seek to identify mutually disjoint sets of receivers amongst the recipient anonymity sets of the messages sent by Alice, which are intersected with the anonymity sets of Alice's sent messages to find her communication partners. The main drawback of the Disclosure attack is that it is equivalent to solving a Constraint Satisfaction Problem which is well-known to be NP-complete.

The subfamily of Hitting Set Attacks [11,12,18] speeds up the search for Alice's messages recipients by looking for unique minimal hitting sets. An evaluation of this attack is provided in [18], where the relationship between the number of rounds the adversary needs to observe to uniquely identify the set of receivers is analyzed. The study by Pham et al. is similar to our work in spirit, but different in that they focus on attacks that unambiguously identifying recipient sets while our focus is on statistical attacks that only provide an estimation of such sets as the ones discussed below.

The series of statistical attacks was started by Danezis in [3] where he introduced the Statistical Disclosure Attack (SDA). Danezis observed that for a large enough number of observed mixing rounds the average of the probability distributions describing the recipient anonymity set [20] of Alice's messages offers a

very good estimation of her sending profile. Danezis considers that in each round where Alice sends a message, the recipient anonymity set of this message is uniform over the receivers present in the round (and zero for the rest of users). The SDA was subsequently extended to more complex mixing algorithms [7], to traffic containing replies [5], to consider other users in order to improve the identification of Alice's contacts [14], and to evaluate more complex user models [15].

Troncoso et al. proposed in [21] two attacks that outperform the SDA, the Perfect Matching Disclosure Attack (PMDA) and the Normalized Statistical Disclosure Attack (NSDA). Under the observation that in a round of mixing the relationships between sent and received messages must be one-to-one, the attacks consider interdependencies between senders and receivers in order to assign most likely receivers to each of the senders: the PMDA searches for perfect matchings in the underlying bipartite graph representing a mix round, while the NSDA normalizes the adjacency matrix representing this graph. The recipient anonymity set of each sender's message in a round is built taking into consideration the result of this assignment, instead of assigning uniform probability amongst all recipients.

Last, Danezis and Troncoso propose to approach the estimation of user profiles as a Bayesian inference problem [8]. They introduce the use of Bayesian sampling techniques to co-infer user communication profiles and de-anonymize messages. The Bayesian approach can be adapted to analyze arbitrarily complex systems and outputs reliable error estimates, but it requires the adversary to repeatedly seek for perfect matchings increasing the computational requirements of the attack.

We note that previous authors evaluated the attacks either from mostly a de-anonymization of individual messages perspective (e.g., [8,21]), or from the point of view of the number of rounds necessary to identify a percentage of Alice's recipients (e.g., [14,15]). In this work we are interested in the accuracy with which the adversary can infer the sender (respectively, receiver) profile of Alice, i.e., we not only seek to identify Alice's messages receivers, but also to estimate the probability that Alice sends (or receives) a message to (from) them.

3 System Model

In this section we describe our model of an anonymous communication system an introduce the notation we use throughout the paper, summarized in Table 3.

System Model. We consider a system in which a population of N_{users} users, designated by an index $i \in \{1, \ldots, N_{\text{users}}\}$, communicate through a threshold mix. This mix operates as follows. In each round of mixing it gathers t messages, transforms them cryptographically, and outputs them in a random order; hence hiding the correspondence between incoming and outgoing messages.

We model the number of messages that the ith user sends in round r as the random variable X_i^r; and denote as x_i^r the actual number of messages i sends in that round. Similarly, Y_j^r is the random variable that models the number of messages that the jth user receives in round r; and y_j^r the actual number of

messages j receives in that round. Let $\mathbf{x}^r$ and $\mathbf{y}^r$ denote the column vectors that contain as elements the number of messages sent or received by all users in round r: $\mathbf{x}^r = [x_1^r, \cdots, x_{N_{\text{users}}}^r]^T$, and $\mathbf{y}^r = [y_1^r, \cdots, y_{N_{\text{users}}}^r]^T$, respectively. When it is clear from the context, the superindex r is dropped.

Users in our population choose their recipients according to their sending profile $\mathbf{q}_i \doteq [p_{1,i}, p_{2,i}, \cdots, p_{N_{\text{users}},i}]^T$; where $p_{j,i}$ models the probability that user i sends a message to user j. We consider that a user i has f friends to whom she sends with probability $p_{j,i}$, and assign $p_{j,i} = 0$ for each user j that is not a friend of i. Conversely, $\mathbf{p}_j$ is the column vector containing the probabilities of those incoming messages to the jth user, i.e., $\mathbf{p}_j \doteq [p_{j,1}, p_{j,2}, \cdots, p_{j,N_{\text{users}}}]^T$. (This vector can be related to the receiving profile of user j through a simple normalization, i.e., by dividing its components by $\sum_{i=1}^{N_{\text{users}}} p_{j,i}$.) We denote as f_j the number of senders that send messages to receiver j, i.e., the cardinality of the set $\mathcal{F}_j = \{i | p_{j,i} > 0, p_{j,i} \in \mathbf{p}_j\}$; and define $\tau_f \doteq \sum_{i=1}^{N_{\text{users}}} f_i^2/(f^2 N_{\text{users}})$, which shall come handy in the performance evaluation performed in Sect. 5.

Adversary Model. We consider a global passive adversary that observes the system during ρ rounds. She can observe the identity of the senders and receivers that communicate through the mix. As our objective is to illustrate the impact of disclosure attacks on the anonymity provided by the mix we assume that the cryptographic transformation performed during the mixing is perfect and thus the adversary cannot gain any information from studying the content of the messages.

The adversary's goal is to uncover communication patterns from the observed flow of messages. Formally, given the observation $\mathbf{x}^r = \{x_i^r\}$ and $\mathbf{y}^r = \{y_j^r\}$, for $i, j = 1, \ldots, N_{\text{users}}$, and $r = 1, \ldots, \rho$, the adversary's goal is to obtain estimates $\hat{p}_{j,i}$ as close as possible to the probabilities $p_{j,i}$, which in turn can be used to recover the users' sender and receiver profiles.

4 A Least Squares Approach to Disclosure Attacks

We aim here at deriving a profiling algorithm based on the Maximum Likelihood (ML) approach to recover the communication patterns of users anonymously communicating through a threshold mix. The general idea is to be able to estimate the probabilities $p_{j,i}$ that user i sends a message to user j, which allow to simultaneously determine the sender and receiver profiles of all users.

We make no assumptions on the user's profiles (i.e., we impose no restrictions on the number of friends a user may have, nor on how messages are distributed amongst them). Nevertheless, we follow the standard assumptions regarding users' behavior and consider that they are memoryless (i.e., for a user the probability of sending a message to a specific receiver does not depend on previously sent messages), independent (i.e., the behavior of a certain user is independent from the others), with uniform priors (i.e., any incoming message to the mix is a priori sent by any user with the same probability), and stationary (i.e., the parameters modeling their statistical behavior do not change with time).

Table 1. Summary of notation

Symbol	Meaning
N_{users}	Number of users in the population, denoted by $i = \{1, \cdots, N_{\text{users}}\}$
f	Number of friends of each sender i
t	Threshold mix
f_j	Number of senders sending messages to receiver j
τ_f	$\sum_{j=1}^{N_{\text{users}}} f_j^2/(f^2 N_{\text{users}})$
$p_{j,i}$	Probability that user i sends a message to user j
$\mathbf{q}_i$	Sender profile of user i, $\mathbf{q}_i = [p_{1,i}, p_{2,i}, \cdots, p_{N_{\text{users}},i}]^T$
$\mathbf{p}_j$	Unnormalized receiver profile of user j, $\mathbf{p}_j = [p_{j,1}, p_{j,2}, \cdots, p_{j,N_{\text{users}}}]^T$
ρ	Number of rounds observed by the adversary
x_i^r (y_j^r)	Number of messages that the ith (jth) user sends (receives) in round r
$\mathbf{x}^r$ $(\mathbf{y}^r)$	Column vector containing elements x_i^r (y_j^r), $i = 1, \cdots, N_{users}$
$\hat{p}_{j,i}$	Adversary's estimation of $p_{j,i}$
$\hat{\mathbf{q}}_\mathbf{i}$	Adversary's estimation of user i's sender profile $\mathbf{q}_i$
$\hat{\mathbf{p}}_\mathbf{j}$	Adversary's estimation of user j's unnormalized receiver profile $\mathbf{p}_j$

4.1 Analysing One Round of Mixing

For simplicity of notation we will consider first a single round of observations, and later explain how to extend the derivation to an arbitrary number of rounds. Hence, for the moment, we will drop the superindex r. Let $Y_{j,i}$ be the random variable that models the number of messages received by user j that were sent by user i in the round under consideration. Then the number of messages that the jth user receives in this round can be computed as:

$$Y_j = \sum_{i=1}^{N_{\text{users}}} Y_{j,i}\,.$$

Recall that $p_{j,i}$ represents the probability that user i sends a message to user j. Then, the probability of user j receiving $y_{j,i}$ messages when the number of messages sent by user i is $X_i = x_i$ is given by a binomial distribution:

$$\Pr(Y_{j,i} = y_{j,i} | X_i = x_i) = \binom{x_i}{y_{j,i}} p_{j,i}^{y_{j,i}} (1 - p_{j,i})^{x_i - y_{j,i}}\,, \tag{1}$$

whose mean is $x_i \cdot p_{j,i}$ and variance $x_i \cdot p_{j,i}(1 - p_{j,i})$. This probability can be approximated by a Gaussian with the same mean and variance.

It is important to notice that the variables $Y_{j,i}$, $j = 1, \cdots, N_{\text{users}}$ are not independent, and rather they are jointly modeled by a multinomial distribution. However, the covariance $\text{cov}(Y_{j,i}, Y_{k,i}) = -x_i \cdot p_{j,i} \cdot p_{k,i}$, $k \neq j$, is small (in comparison with diagonal terms of the covariance matrix) if the transition probabilities are also small. Moreover, in such case the variance of the binomial can be approximated by $x_i \cdot p_{j,i}$. Therefore, when the transition probabilities are small, and recalling that the sum of independent Gaussian random variables

is itself Gaussian, we can approximate the conditional distribution of Y_j by a normal:

$$Pr(Y_j|\mathbf{X}=\mathbf{x}) \sim \mathcal{N}\left(\sum_{i=1}^{N_{\text{users}}} x_i p_{j,i}, \sum_{i=1}^{N_{\text{users}}} x_i p_{j,i}\right) ,$$

and consider that $\text{cov}(Y_j, Y_k) \approx 0$, whenever $k \neq j$.

Under the hypothesis above, since the random variables Y_j are approximately independent, we can write the joint probability of $\mathbf{Y}$ as

$$Pr(\mathbf{Y}|\mathbf{X}=\mathbf{x}) \sim \mathcal{N}\left(\mathbf{Hp}, \mathbf{\Sigma}_y\right) ,$$

where $\mathbf{p}^T \doteq [\mathbf{p}_1^T, \mathbf{p}_2^T, \cdots, \mathbf{p}_{N_{\text{users}}}^T]$, $\mathbf{\Sigma}_y \doteq \text{diag}(\mathbf{Hp})$, and $\mathbf{H}^T \doteq \mathbf{x} \otimes \mathbf{I}_{N_{\text{users}}}$. Here, $\mathbf{I}_n$ denotes the identity matrix of size $n \times n$, and $\otimes$ denotes the Kronecker product.

For a ML solution to the profiling problem, after observing $\mathbf{Y} = \mathbf{y}$, we seek that vector $\hat{\mathbf{p}}$ of probabilities that maximizes $Pr(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$.[1] This can be explicitly written as follows:

$$\hat{\mathbf{p}} = \arg\max_{\mathbf{p}\in\mathcal{P}} \frac{1}{\sqrt{\det(\mathbf{\Sigma}_y)}} \cdot \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{Hp})^T \mathbf{\Sigma}_y^{-1}(\mathbf{y}-\mathbf{Hp})\right) , \tag{2}$$

where $\mathcal{P}$ denotes the set of valid probability vectors.[2]

For the unconstrained problem in (2) it is possible to uncouple the different terms and show that the solution must satisfy

$$\mathbf{x}^T \hat{\mathbf{p}}_j = \frac{1}{2}\left(\sqrt{1+4y_j^2} - 1\right), \quad j = 1, \cdots, N_{\text{users}} , \tag{3}$$

where $\hat{\mathbf{p}}_j$ is the estimated unnormalized receiver profile of user j.

The right hand side of (3) is smaller than y_j; however, it can be well approximated by y_j when the latter is large. Notice that (3) becomes an underdetermined linear system of equations.

4.2 Analysing ρ Rounds

A different situation arises when the number of observed rounds is larger than the number of users. In this case, we form the following vectors/matrices:

$$\mathbf{Y}^T \doteq [Y_1^1, Y_1^2, \cdots, Y_1^\rho, Y_2^1, Y_2^2, \cdots, Y_2^\rho, \cdots, Y_{N_{\text{users}}}^1, Y_{N_{\text{users}}}^2, \cdots, Y_{N_{\text{users}}}^\rho]$$
$$\mathbf{U}^T \doteq [\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^\rho]$$
$$\mathbf{H} \doteq \mathbf{U} \otimes \mathbf{I}_{N_{\text{users}}}$$

[1] Notice that since the random variable $\mathbf{X}$ does not depend on the probabilities $\mathbf{p}$, the maximization of $Pr(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$ is equivalent to that of $Pr(\mathbf{Y} = \mathbf{y}; \mathbf{X} = \mathbf{x})$.

[2] Without further constraints, that may be furnished when there is partial knowledge about the transition probabilities, $\mathcal{P}$ is simply given by the constraints $0 \leq p_{j,i} \leq 1$ for all j, i, and $\sum_{j=1}^{N_{\text{users}}} p_{j,i} = 1$, for all i.

The ML solution must satisfy (2). However, notice that in the case of ρ rounds, the involved matrices and vectors are larger than those found in the case of a single observation.

Unlike (3), a closed-form solution seems not exist (even for the unconstrained case, i.e., when no constraints are imposed upon $\mathcal{P}$). We examine next some approximate solutions to the unconstrained problem that satisfy that $\hat{\mathbf{p}} \to \mathbf{p}$ as $\rho \to \infty$. To make this possible, we disregard the dependence of the covariance matrix $\mathbf{\Sigma}_y$ with $\mathbf{p}$ making the following approximation $\mathbf{\Sigma}_y \approx \mathrm{diag}(\mathbf{y})$.

In such case, the approximate ML estimator is given by

$$\hat{\mathbf{p}} = \arg \min_{\mathbf{p} \in \mathcal{P}} ||\mathbf{\Sigma}_y^{-1/2}(\mathbf{y} - \mathbf{H}\mathbf{p})||^2 , \tag{4}$$

which is nothing but a constrained weighted least squares (WLS) problem.

For simplicity, we consider here the unweighted least squares (LS) case, i.e.,

$$\hat{\mathbf{p}} = \arg \min_{\mathbf{p} \in \mathcal{P}} ||\mathbf{y} - \mathbf{H}\mathbf{p}||^2 , \tag{5}$$

which, for the unconstrained case, has the well-known Moore-Penrose pseudoinverse solution:

$$\hat{\mathbf{p}} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{y} . \tag{6}$$

At first sight, it might look that the matrix inversion needed in (6) is formidable: the matrix $\mathbf{H}^T\mathbf{H}$ has size $N_{\text{users}}^2 \times N_{\text{users}}^2$. However, its block-diagonal structure allows for a much more efficient solution; indeed,

$$\mathbf{H}^T\mathbf{H} = (\mathbf{U} \otimes I_{N_{\text{users}}})^T \cdot \mathbf{U} \otimes I_{N_{\text{users}}} = (\mathbf{U}^T\mathbf{U}) \otimes I_{N_{\text{users}}}$$

and, hence,

$$(\mathbf{H}^T\mathbf{H})^{-1} = (\mathbf{U}^T\mathbf{U})^{-1} \otimes I_{N_{\text{users}}}$$

where now $\mathbf{U}^T\mathbf{U}$ has size $N_{\text{users}} \times N_{\text{users}}$.

The decoupling above allows us to write a more efficient solution as follows. Let $\mathbf{y}_j \cdot [y_j^1, y_j^2, \cdots y_j^\rho]^T$. Then, the LS estimate $\hat{\mathbf{p}}_j$ for the jth probability vector can be written as

$$\hat{\mathbf{p}}_j = (\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{y}_j, \quad j = 1, \cdots, N_{\text{users}} .$$

The decoupling above is possible only in the unconstrained case; this consideration, together with the simplicity of the performance analysis, make us focus on the unconstrained LS approach. Notice, however, that, as a consequence, the obtained solution is not guaranteed to meet the constraints on the transition probabilities. This can be overcome by projecting the solution onto the set $\mathcal{P}$. In any case, the fact that the error $\mathbf{p} - \hat{\mathbf{p}}$ tends to zero as $\rho \to \infty$, ensures that $\hat{\mathbf{p}}$ can be made arbitrarily close to $\mathcal{P}$ by increasing the number of observed rounds. Finally, note that when $\hat{\mathbf{p}}_j$ is computed for all users, it is also possible to recover the sender profiles $\mathbf{q}_i$ by taking the rows of the matrix $\hat{\mathbf{p}}$.

In any case, it is worth remarking that there are many iterative algorithms for solving (constrained) least squares problems, which do not require matrix

inversion. We leave the discussion on how they can be adapted to the problem as subject for future work. It is also worth stressing that we could have arrived at the LS estimate from the perspective of minimizing the mean square error between the observed $\mathbf{y}$ and a predictor based on a linear combination of the given inputs $\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^\rho$.

Finally, we note that the original Statistical Disclosure Attack (SDA) corresponds to a particular case of the proposed LS estimator. The SDA model assumes that the first user (Alice) sends only one message to an unknown recipient chosen uniformly from a set f friends. The other users send messages to recipients chosen uniformly from the set of all users $p_{j,i} = 1/N_{\text{users}}, \forall i \neq 1$. From this considerations, for a given round r where Alice does send a message, we have that $x_1^r = 1$ and $\sum_{j=2}^{N_{\text{users}}} x_i^r = (t-1)$, and all the transition probabilities of the form $p_{j,i}$, for $i \geq 2$ are known to be equal to $1/N_{\text{users}}$. If we suppose that in all rounds Alice transmits a message, we will have a vector $\mathbf{y}$ which contains the $\rho \cdot N_{\text{users}}$ observations, $\mathbf{p}_1$ is unknown and all $\mathbf{p}_i$, $i = 2, \cdots, N_{\text{users}}$ are known. From here, it is possible to find that the LS estimate of the unknown probabilities is

$$\hat{p}_{j,1} = \frac{1}{\rho} \sum_{r=1}^{\rho} y_j^r - \frac{(t-1)}{N_{\text{users}}}, \quad j = 1, \cdots, N_{\text{users}}$$

which coincides with the SDA estimate. (We leave a more detailed derivation of this equation for an extended version of this paper [17].)

4.3 Performance Analysis with Respect to the System Parameters

The Least Squares estimate in (6) is unbiased: it is straightforward to show that $\mathrm{E}[\hat{\mathbf{p}}] = \mathbf{p}$. On the other hand, the covariance matrix of $\hat{\mathbf{p}}$, for a fixed matrix $\mathbf{H}$, is given by

$$\mathrm{E}[(\mathbf{p} - \hat{\mathbf{p}})(\mathbf{p} - \hat{\mathbf{p}})^T] = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\boldsymbol{\Sigma}_y\mathbf{H}(\mathbf{H}^T\mathbf{H})^{-1}\,. \tag{7}$$

Notice that the performance will depend on the actual input matrix $\mathbf{H}$; however, when the input process is wide-sense stationary, and $\rho \to \infty$ then $\mathbf{U}^T\mathbf{U}$ will converge to the input autocorrelation matrix $\mathbf{R}_x$. Then, when the number of observations is large, approximating $\mathbf{U}^T\mathbf{U} \approx \mathbf{R}_x$ will allow us to extract some quantitative conclusions that are independent of $\mathbf{U}$. To this end, notice that if $\mathrm{Cov}(Y_i, Y_j) \approx 0$ for all $i \neq j$, then

$$\boldsymbol{\Sigma}_y \approx \mathrm{diag}(\xi_y) \otimes \mathbf{I}_{N_{\text{users}}}\,,$$

with $\xi_y = [\mathrm{Var}\{Y_1\}, \cdots, \mathrm{Var}\{Y_{N_{\text{users}}}\}]$.

In this case, (7) becomes

$$\mathrm{E}[(\mathbf{p} - \hat{\mathbf{p}})(\mathbf{p} - \hat{\mathbf{p}})^T] = \mathrm{diag}(\xi_y) \otimes (\mathbf{U}^T\mathbf{U})^{-1}\,. \tag{8}$$

Still we would need to quantify how large $(\mathbf{U}^T\mathbf{U})^{-1}$ is. Since $\mathbf{U}^T\mathbf{U}$ is symmetric, we can write the following eigendecomposition

$$\mathbf{U}^T\mathbf{U} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}\,, \tag{9}$$

where $\mathbf{Q}$ is orthonormal and $\mathbf{\Lambda}$ is diagonal. In this case, $(\mathbf{U}^T\mathbf{U})^{-1} = \mathbf{Q}^{-1}\mathbf{\Lambda}^{-1}\mathbf{Q}$. Then, if we define the *transformed probability space* where $\mathbf{p}'_j \doteq \mathbf{Q}\mathbf{p}_j$ and $\hat{\mathbf{p}}'_j \doteq \mathbf{Q}\hat{\mathbf{p}}_j$ we have

$$\mathrm{E}[(\mathbf{p}' - \hat{\mathbf{p}}')(\mathbf{p}' - \hat{\mathbf{p}}')^T] = \mathrm{diag}(\xi_y) \otimes \mathbf{\Lambda}^{-1} \tag{10}$$

A measure of the total error variance made with the proposed estimator is given by the trace. Notice that

$$\mathrm{E}\left[\mathrm{tr}\left((\mathbf{p}' - \hat{\mathbf{p}}')(\mathbf{p}' - \hat{\mathbf{p}}')^T\right)\right] = \mathrm{E}\left[\mathrm{tr}\left((\mathbf{p} - \hat{\mathbf{p}})(\mathbf{p} - \hat{\mathbf{p}})^T\right)\right] = \sum_{i=1}^{N_{\mathrm{users}}} \sigma^2_{y_i} \cdot \sum_{j=1}^{N_{\mathrm{users}}} \lambda^{-1}_{u,j} \tag{11}$$

where $\lambda_{u,j}$, $j = 1, \cdots, N_{\mathrm{users}}$ denote the eigenvalues of $\mathbf{U}^T\mathbf{U}$.

Equation (11) can be interpreted as having two terms that depend on the output covariance and input autocorrelation, respectively. In fact, for some cases of interest, it is possible to derive explicit expressions, as we discuss next.

Consider the case where each user has exactly the same probability $1/N_{\mathrm{users}}$ of sending a message to one of her friends and that each message is sent independently. Then, if t messages are sent per round, the observed input vector at the jth round $\mathbf{x}^j$ will follow a multinomial distribution for which

$$\mathrm{E}\{X_i^2\} = t^2p_x^2 + tp_x(1 - p_x)), \;\text{ and }\; \mathrm{E}\{X_iX_k\} = t^2p_x^2 - tp_x^2, \;\; i \neq k$$

where $p_x = 1/N_{\mathrm{users}}$. Then, the autocorrelation matrix $\mathbf{R}_x$ can be shown to have $(N_{\mathrm{users}} - 1)$ identical eigenvalues which are equal to $\rho \cdot t \cdot p_x$ and the remaining eigenvalue equal to $\rho \cdot t \cdot p_x + \rho \cdot t \cdot p_x^2(t-1)N_{\mathrm{users}}$. Therefore,

$$\sum_{j=1}^{N_{\mathrm{users}}} \lambda^{-1}_{u,j} = \frac{N_{\mathrm{users}}}{\rho t}\left(N_{\mathrm{users}} - 1 + \frac{1}{t}\right) \tag{12}$$

Next we focus on the output variance. We consider the case where each user has f friends in her sending profile to whom she sends messages with probability $1/f$ each. Let $\mathcal{F}_j$ be the set of users that send messages to the jth user with non-zero probability, and let f_j be its cardinality. Then, for the input conditions discussed in the previous paragraph (i.e., i.i.d. uniform users), the probability that one given message is sent by one user in $\mathcal{F}_j$ is f_j/N_{users}. In turn, the probability that one message originating from a user in $\mathcal{F}_j$ is sent to the jth user is $1/f$. Therefore, we can see Y_j^k as the output of a binomial process with probability

$$p_{y_j} = \frac{f_j}{fN_{\mathrm{users}}}\,,$$

and with t messages at its input. Hence, the variance of Y_j is

$$\sigma^2_{y_j} = t \cdot p_{y_j}(1 - p_{y_j}) = \frac{t \cdot f_j}{f \cdot N_{\text{users}}} \cdot \left(1 - \frac{f_j}{f \cdot N_{\text{users}}}\right) ,$$

so the sum of variances becomes

$$\sum_{j=1}^{N_{\text{users}}} \sigma^2_{y_j} = t \left(1 - \frac{\sum_{j=1}^{N_{\text{users}}} f_j^2}{f^2 N_{\text{users}}^2}\right) = t \left(1 - \frac{\tau_f}{N_{\text{users}}}\right) , \tag{13}$$

where we have used the fact that $\sum_{i=1}^{N_{\text{users}}} f_j = f \cdot N_{\text{users}}$.

Combining (12) and (13) we can write the MSE as

$$\mathrm{E}\left[\mathrm{tr}\left((\mathbf{p} - \hat{\mathbf{p}})(\mathbf{p} - \hat{\mathbf{p}})^T\right)\right] = \frac{1}{\rho}\left(N_{\text{users}} - 1 + \frac{1}{t}\right) \cdot (N_{\text{users}} - \tau_f) . \tag{14}$$

It is useful to interpret (14) in terms of the number of friends of each receiver. We will consider two particular cases of interest: 1) If each receiver has exactly f friends, then $\tau_f = \tau_{f,1} = 1$; 2) If only f receivers have N_{users} friends, and the remaining $N_{\text{users}} - f$ receivers have no friends, then $\tau_f = \tau_{f,2} = N_{\text{users}}/f$. The second case models a situation where f receivers act as hubs (i.e., f users concentrate the traffic of all the population), while in the first there is absolutely no skew in the distribution. In fact, using the Lagrange multipliers technique, it can be shown that for all other cases, including random connections (but always keeping the constraint that each sender has exactly f friends), the parameter τ_f satisfies that $\tau_{f,1} \leq \tau_f \leq \tau_{f,2}$. Since (14) monotonically decreases with τ_f, we can conclude that for the symmetric case (i.e., $\tau_f = 1$) the MSE is larger, revealing that it will be harder to learn the transition matrix.

When N_{users} is large, we can approximate (14) as follows

$$\mathrm{E}\left[\mathrm{tr}\left((\mathbf{p} - \hat{\mathbf{p}})(\mathbf{p} - \hat{\mathbf{p}})^T\right)\right] \approx \frac{N_{\text{users}}^2}{\rho} . \tag{15}$$

If we recall that there are N_{users}^2 probabilities to estimate from the transition matrix, we can conclude that the variance *per transition element* $p_{j,i}$ is approximately $1/\rho$. The total MSE decreases as $1/\rho$ with the number of rounds ρ; this implies that the unconstrained, unweighted LS estimator is asymptotically efficient as $\rho \to \infty$. Even though this is somewhat to be expected, notice that other simpler estimators might not share this desirable property, as we will experimentally confirm in Sect. 5.

5 Evaluation

5.1 Experimental Setup

We evaluate the effectiveness of the Least Squares approach to Disclosure Attacks (LSDA) against synthetic anonymized traces created by a simulator written in

the Python language.[3] We simulate a population of N_{users} users with f contacts each, to whom they send messages with equal probability (i.e., $p_{j,i} = 1/f$ if i is friends with j, zero otherwise). In order to easily study the influence of the system parameters on the success of the attack, in our simulations we further fix the senders that send messages to each receiver to be $f_j = f$. In other words, every sender (receiver) profile has the same number of non-zero elements, and hence $\tau_f = 1$. Messages are anonymized using a threshold mix with threshold t, and we consider that the adversary observes ρ rounds of mixing. Table 2 summarizes the values of the parameters used in our experiments, where bold numbers indicate the parameters of the baseline experiment.

Table 2. System parameters used in the experiments

Parameter	Value
N_{users}	$\{50, \mathbf{100}, 150, 200, 250, 300, 350, 400, 450, 500\}$
f	$\{5, 10, 15, 20, \mathbf{25}, 30, 35, 40, 45, 50\}$
t	$\{2, 5, \mathbf{10}, 20, 30, 40\}$
ρ	$\{\mathbf{10\,000}, 20\,000, \ldots, 100\,000\}$
τ_f	$\{\mathbf{1.0}, 1.76, 2.44, 3.04, 3.56, 4.0\}$

The parameters' values used in our experiments, though rather unrealistic, have been chosen in order to cover a wide variety of scenarios in which to study the performance of the attack while ensuring that experiments could be carried out in reasonable time. We note, however, that the results regarding the LSDA can be easily extrapolated to any set of parameters as long as the proportion amongst them is preserved. Unfortunately, we cannot make a similar claim for the other attacks. Their heuristic nature makes it difficult to obtain analytical results that describe the dependence of their success on the system parameters, and the evolution of their error difficult to predict as we will see throughout this section.

Besides testing the effectiveness of the LSDA when profiling users, we also compare its results to those obtained performing the Statistical Disclosure Attack (SDA) [3,7], the Perfect Matching Disclosure Attack (PMDA) [21], the Normalized Statistical Disclosure Attack (NSDA) [21], and the Bayesian inference-based attack Vida [8].

5.2 Success Metrics

We recall that the goal of the adversary is to estimate the values $p_{j,i}$ with as much accuracy as possible. The LSDA, as described in Sect. 4, is optimized to minimize the Mean Squared Error (MSE) between the actual transition probabilities $p_{j,i}$ and the adversary's estimated $\hat{p}_{j,i}$. We define two metrics to illustrate the profiling accuracy of the attacks. The *Mean Squared Error per transition probability* (MSE_p) measures the average squared error between the elmements

[3] The code will be made available upon request.

of the estimated matrix $\hat{\mathbf{p}}$ and the elements of the matrix $\mathbf{p}$ describing the actual behaviour of the users (see (6)):

$$\mathrm{MSE}_p = \frac{\sum_{i,j}(\hat{p}_{j,i} - p_{j,i})^2}{N_{\text{users}}^2}.$$

Secondly, we define the *Mean Squared Error per sender profile* (MSE_{q_i}):

$$\mathrm{MSE}_{q_i} = \frac{\sum_j(\hat{p}_{j,i} - p_{j,i})^2}{N_{\text{users}}}\ ,\ i = 1, \ldots, N_{\text{users}}$$

which measures the average squared error between the probability of the estimated $\hat{\mathbf{q}}_i$ and actual $\mathbf{q}_i$ user i's sender profiles. Both MSEs measure the amount by which the values output by the attack differ from the actual value to be estimated. The smaller the MSE, the better is the adversary's estimation of the users' actual profiles.

For each of the studied set of parameters (N_{users}, f, t, ρ, τ_f) we record the sets of senders and receivers during ρ rounds and compute the MSE_p (or the MSE_{q_i}) for each of the attacks. We repeat this process 20 times and plot the average of the result in our figures.

5.3 Results

Estimating Sender and Receiver Profiles with the LSDA. We first illustrate how the LSDA can simultaneously estimate sender and receiver profiles. Traditionally, Disclosure Attacks focus in estimating the sender profiles; and receiver profiles can be inferred by resolving the inverse problem (i.e., performing the same attack inverting the role of senders and receivers). Even the Reverse Statistical Disclosure Attack [14], that explicitly requires receiver profiles to improve the estimation of the sender profiles, includes a step in which the SDA is applied in the reverse direction before results can be obtained.

The LSDA estimates the full matrix $\mathbf{p}$ in one go. By either considering the rows or columns of this matrix the adversary can recover the unnormalized receiver profile $\mathbf{p}_j = [p_{j,1}, p_{j,2}, \cdots, p_{j,N_{\text{users}}}]^T$, or the sender profile $\mathbf{q}_i = [p_{1,i}, p_{2,i}, \cdots, p_{N_{\text{users}},i}]^T$ without any additional operation. Fig. 1, left, shows box plots[4] describing the distribution of the MSE_p over 20 experiments for senders and receiver profiles, respectively. The right-hand side of the figure shows the sender and the receiver profiles computed performing the LSDA in the reverse direction, considering the receivers as senders, and vice versa.

We can see that the results obtained in the "forward" and reverse direction are not the same. In fact, we have observed that in each instance there is a direction

[4] The line in the middle of the box represents the median of the distribution. The lower and upper limits of the box correspond, respectively, to the distribution's first (Q1) and third quartiles (Q3). We also show the outliers, represented with +: values x which are "far" from the rest of the distribution ($x > Q3 + 1.5(Q3 - Q1)$ or $x < Q1 - 1 : 5(Q3 - Q1)$).

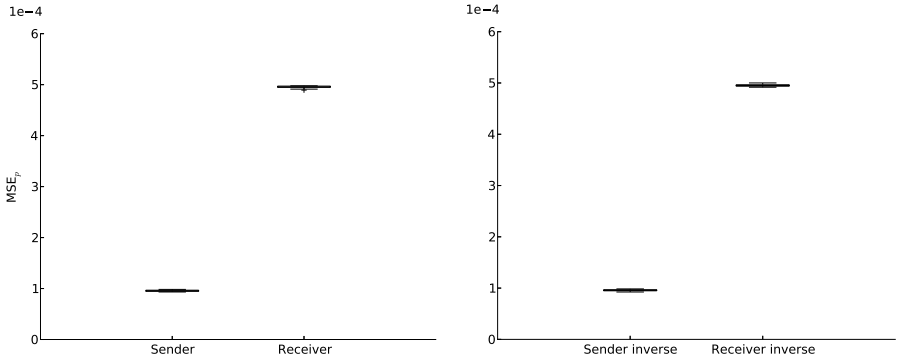


Fig. 1. LSDA's MSE_p per transition probability when inferring sender and receiver profiles in the forward (left) and reverse (right) directions ($N_{\text{users}} = 100$, $f = 25$, $t = 10$, $\rho = 10\,000$, $\tau_f = 1$)

that is better than the other in terms of MSE_p. While it is not possible to decide which side is going to provide better results, because a priori all profiles are equally likely, it is easy to see that the average of the estimations $\hat{\mathbf{p}}$ in both directions will yield a MSE per transition probability smaller than the worst case.

Performance with Respect to the Number of Rounds ρ. As we discuss in Sect. 4.3, the number of observed rounds ρ has a dominant role in the estimation error incurred by the LSDA. We plot in Fig. 2, left, the MSE per transition probability MSE_p for the SDA, NSDA, PMDA and LSDA.

The LSDA, optimized to minimize the MSE_p, obtains the best results. Further, we can see how the approximation in Eq. (15), represented by $\bullet$ in the figure, reliably describes the decrease in the profile estimation error as more information is made available to the adversary.

It is also interesting to notice how the different attacks take advantage of the information procured by additional rounds. The naive approach followed by the SDA soon maxes out in terms of information extracted from the observation and its MSE_p does not decrease significantly as more rounds are observed, confirming the results in [21]. The NSDA and PMDA perform slightly better in this sense, although their MSE_p also decreases slowly. The LSDA, on the other hand, is able to obtain information from each new observed round reducing significantly the MSE_p, that tends to zero as $\rho \to \infty$. This is because, as opposed to its predecessors which process the rounds one at a time, the LSDA considers all rounds simultaneously (by means of the matrices $\mathbf{Y}$ and $\mathbf{U}$).

Performance with Respect to the Mix Threshold t. By observing Eq. (14) one can see that the threshold t of the mix has little influence on the MSE_p of the LSDA, becoming negligible as t increases and $t >> 1$. This is reflected by our experiments, shown in Fig. 2, left, where the error of the LSDA soon becomes stable as the threshold of the mix grows.

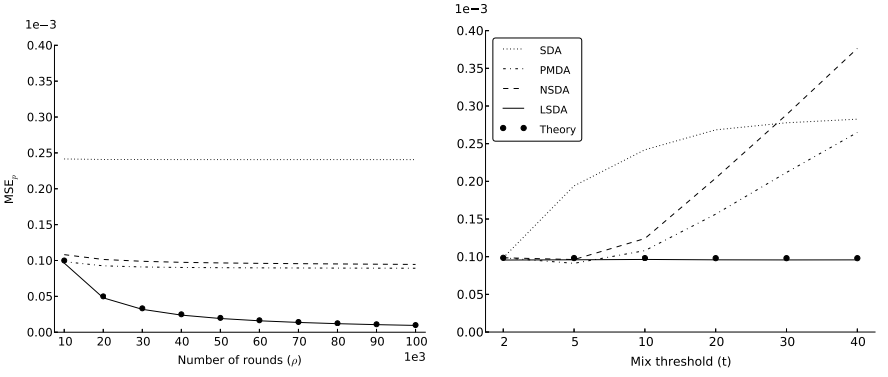


Fig. 2. MSE$_p$ evolution with the number of rounds in the system ρ ($N = 100$, $f = 25$, $t = 10$, τ_f=1), and with the threshold mix t ($N = 100$, $f = 25$, $\rho = 10\,000$, $\tau_f = 1$) (left and right, respectively).

This desirable property does not hold for the other approaches. As expected, increasing the threshold has a negative effect on the three attacks. Nevertheless this effect differs depending on the approach used. The SDA's, surprisingly, seems to grow proportionally to $(1 - 1/t)$ and thus the increase of the error with the threshold is greatly reduced as t increases. This is not the case for the NSDA and PMDA, based on solving an optimization problem on the underlying bipartite graph representing a mix round. This problem becomes harder as the threshold grows, thus their MSE$_p$ significantly increases with the number of messages processed in each mix round.

Performance with Respect to the Number of Users N_{users}. Next, we study the influence of the number of users in the system on the estimation error. The results are shown in Fig. 3 for $\rho = 10\,000$ (left) and $\rho = 100\,000$ (right). As expected (see 15), the LSDA's MSE$_p$ grows slowly with the number of users. The other three attacks, on the other hand, improve their results when the number of users increase. When the number of users increases, and the mix threshold does not vary, the intersection between the senders of different mixing rounds becomes smaller, and thus the SDA can better identify their sender profiles. The PMDA and the NSDA use the result of the SDA as attack seed. Hence, the better estimations output by the SDA, the better results obtained by the PMDA and the NSDA.

Even though N_{users} has some effect on the MSE$_p$ of the LSDA the results in Fig. 3 reinforce the idea that the number of rounds ρ is the main component of the error. When $\rho = 10\,000$ rounds are observed the LSDA does not provide better results than the other attacks. Nevertheless, as the number of rounds increases, the LSDA outperforms the other attacks regardless of the growth of the MSE with N_{users}.

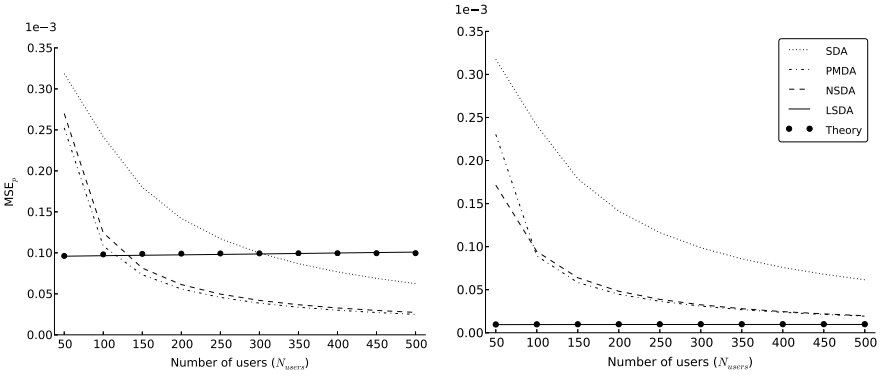


Fig. 3. MSE$_p$ evolution with the number of users in the system N_{users} ($f = 25$, $t = 10$, $\rho = 10\,000, 100\,000$, $\tau_f = 1$) (left and right, respectively)

Performance with Respect to the Output Variance $\sigma^2_{y_j}$. The influence on the LSDA's MSE of the output variance $\sigma^2_{y_j}$ can be studied by varying the value of the parameters f and τ_f, while maintaining N_{users} and t constant (see Eq. (13)). We first vary the number of friends of the senders f while keeping $f_j = f$ for all receivers j, ensuring that $\tau_f = 1$. We observe in Fig. 4, left, that the LSDA's MSE$_p$ closely follows the prediction in formula (14).

In a second experiment, we fix the parameter f vary τ_f to represent different degrees of "hubness" in the population. We construct populations such in which there are $\alpha = 0, \cdots, f$ hub receivers that have N_{users} friends, while the remaining $N_{\text{users}} - \alpha$ receivers are assigned small amounts of friends in order to obtain different τ_f arbitrarily chosen between $\tau_{f,1} = 1$ and $\tau_{f,2} = N_{\text{users}}/f$. The result is shown in Fig. 4, right. It is worthy to note that the SDA significantly benefits from the hubness of the population. As some users concentrate the traffic, and the sending profiles become more uniform all users tend to send their messages to the same set of receivers. In this scenario the strategy of the SDA, that assigns equal probability to every receiver in a mix batch, closely models reality and the error tends to zero. While the error of the SDA is very small, the estimated profiles still have small biases toward some users. This effect is amplified by the NSDA and PMDA, significantly increasing their estimation error.

Performance with Respect to the User Behaviour. Our experiments so far considered a very simplistic population in which users choose amongst their friends uniformly at random (which we denote as SDA). As it has been discussed in the past [8, 21] this population is unlikely to represent real users. We now evaluate the four attacks against two more realistic populations in which users choose the recipients according to an arbitrary multinomial distribution, more (SKW) or less (ARB) skewed depending on the experiment.

We show in Fig. 5 (left) box plots representing the distribution of the MSE per sender profile MSE$_{q_i}$ for all users in the population. We also plot the MSE$_p$ for

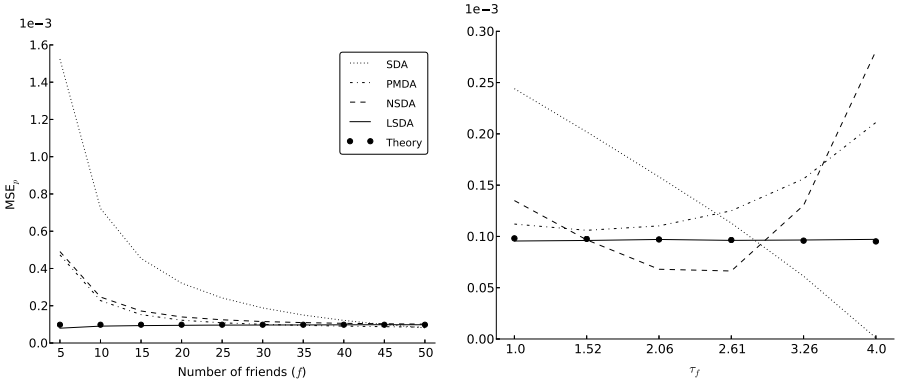


Fig. 4. MSE_p evolution with the number of friends f ($N = 100$, $f = 25$, $\rho = 10\,000$, τ_f=1), and with τ_f ($N = 100$, $f = 25$, $t = 10$, $\rho = 10\,000$) (left and right, respectively)

each attack in the figure, representing it with $\star$ (note that the MSE_p is also the mean of MSE_{q_i} for all i). We recall that, as the PMDA and NSDA, the LSDA makes no assumptions on the users' profiles, while the SDA assumes uniform behavior. Hence, as expected when the profiles become increasingly skewed the SDA performs the worst, obtaining the LSDA the smallest MSE_p. Furthermore, it is worthy to notice that the user behaviour has a strong influence on the variance of the MSE_{q_i}. The fact that users have favorite friends who receive a large fraction of their messages makes the probability of these receivers easy to estimate, while for receivers that are not often chosen the attacks' estimations are poor. This explains the large variance in the SKW population with respect to the other population types.

Comparison between Attack Principles. Throughout the evaluation section we have considered four disclosure attacks that estimate users profiles using statistics and optimization techniques. We now compare these attacks to Vida, the Bayesian inference-based machine learning algorithm proposed by Danezis and Troncoso in [8]. We can see in Fig. 5 (right), which shows box plots representing the distribution of the MSE_{q_i} for all users under observation, that Vida outperforms the statistical variants. In order to simplify the figure, we have not plotted the the MSE_p, that lies extremely close to the median in all cases.

We have already discussed that the LSDA obtains an advantage over the SDA, PMDA, and NSDA by considering all observed rounds simultaneously, but does not account for the one-to-one relationship between send and received messages in the individual rounds of mixing. Vida, on the other hand, not only considers all rounds, but searches for perfect matchings in each round improving the profile estimation considerably. These results seemingly contradict the performance evaluation in [8]. This is because the comparison performed by Danezis and Troncoso was with respect to the message de-anonymization success rate, while we focus on the estimation of profiles. In fact, the results reported by Danezis

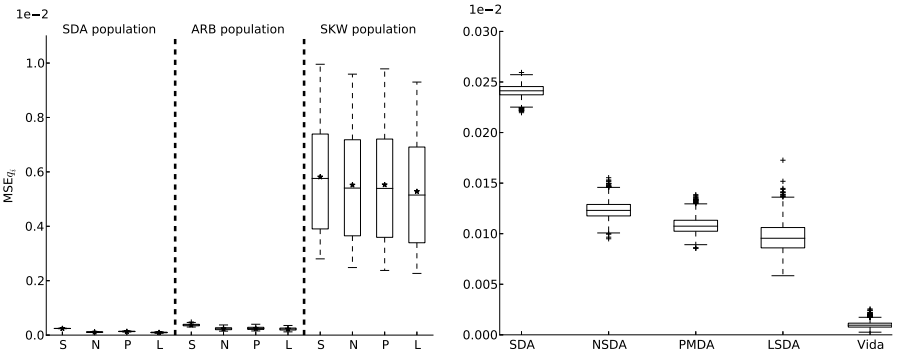


Fig. 5. MSE$_{q_i}$ evolution with respect to the population type for all attacks (left) and only comparison between attack principles (right) ($N = 100$, $f = 25$, $t = 10$, $\rho = 10\,000$, $\tau_f = 1$). (We represent MSE$_p$ with a $\star$.)

and Troncoso show that when 512 rounds of mixing are observed the profiling accuracy of the algorithm is excellent.

While the effectiveness of Vida is desirable, it comes at a high computational cost because each iteration of the algorithm requires finding a perfect matching in all the ρ rounds observed. We note however that, as in [8], we have used the SDA's result as seed for the machine learning engine. Intrestingly, using the LSDA, which yields better estimation of the real profiles than the the SDA, instead may significantly speed up the learning time.

6 Discussion

We have shown that the LSDA is more effective than its statistical predecessors. Further, the matrix operations performed by the LSDA have much smaller computational requirements than the round-by-round processing carried out by the PMDA or the NSDA. This decrease in computation comes at the cost of memory: the LSDA operates with big matrices that have to be loaded to the RAM. The parameters we have used in this paper generated matrices that fitted comfortably in a commodity computer, but larger mix networks may need extra memory. When memory is an issue a gradient-based approach can be used to iteratively process the rounds obtaining the same result while reducing the computational requirements of the attack, that would deal with smaller matrices. This iterative approach can be further adapted to account for temporal changes in the profiles. Extending the LSDA to accommodate such evolution is a promising line of future work.

The fact that we have considered user profiling as an unconstrained problem (see Eq. (2)) resulted in some of the probabilities $\hat{p}_{j,i}$ estimated by the LSDA being negative, corresponding to receivers j that are not friends of user i. When $p_{j,i} = 0$ the algorithm returns $\hat{p}_{j,i}$ that lie near zero, but as the solution is

unconstrained it is not guaranteed that $\hat{p}_{j,i} \geq 0$. One could reduce the error by just setting those probabilities to zero, disregarding that $\sum_j p_{j,i} = 1$ for all i. Alternatively, it is possible to establish constraints on Eq. (2) to ensure that the profiles recovered by the LSDA are well-defined. However, enforcing such constraints will no longer guarantee the decoupling of the unnormalized receiver profiles, and hence the solution is likely to be quite cumbersome. The development and analysis of such solution is left as subject for future research.

Threshold mixes are well fitted to analyse in theory, however deployed systems use pool mixes, which offer better anonymity. Up to know only the SDA has been adapted to traffic analysis of anonymous communications carried out through a pool mix [16]. This is because the internal mechanism of this mix, that may delay messages for more than one round, hinders the construction of a bipartite graph between senders and receivers. Hence, adapting the PMDA, the NSDA, or Vida to such scenario is non-trivial. The independence of the LSDA from the mix threshold makes it an ideal candidate for the analysis of pool mixes. In order to adapt the attack to this mix it is necessary to estimate the matrices $\mathrm{E}\{\mathbf{U}^T\mathbf{U}\}$ and $\mathrm{E}\{\mathbf{U}^T\mathbf{y}_j\}$, for all j.

Finally, in some cases it might be possible that some of the transition probabilities are known. It is possible to modify the machine learning approach [8] to account for this extra knowledge, but this is non-trivial for the SDA, PMDA or NSDA. The Least Squares formulation can be easily adapted to consider this additional information. Without loss of generality let us assume that $p_{1,1}$ is known. As this corresponds to the first element of $\mathbf{p}$, one can work instead with an equivalent problem in which we remove the first column of $\mathbf{H}$ and $p_{1,1}$ from $\mathbf{p}$; consequently, the observation vector $\mathbf{y}$ is replaced by $\mathbf{y} - p_{1,1}\mathbf{h}_1$. This procedure can be repeated for every known transition probability. Similar considerations can be made for the case where the transition probabilities $p_{j,i}$ depend on a smaller set of parameters (e.g., when some of the probabilities are known to be identical).

7 Conclusion

Since Kesdogan and Agrawal [1,12] introduced the Disclosure Attack to profile users sending messages through an anonymous network, a stream of efficient statistical variants have been proposed [3,5,7,8,14,15,21]. Nevertheless, their heuristic nature hinders the search for analytical formulae describing the dependence of their success on the system parameters, which is difficult to characterize and predict as we have shown in our results.

We have introduced the LSDA, a new approach to Disclosure based on solving a Least Square problem, that minimizes the mean squared error between the estimated and real profiles. Further, the LSDA is the first disclosure attack able to simultaneously estimate sender and receiver profiles. The main advantage of our approach is that it allows the analyst to predict the profiling error given the system parameters. This capability is essential at the time of designing high-latency anonymous communication systems, as it permits the designer to choose

the system parameters that provide a desired level of protection depending on the population characteristics without the need to perform simulations, which may require a large computational effort as in the case of Vida. We have empirically evaluated the LSDA and we have proved that our formulae closely model its error.

Acknowledgements. Research supported by the European Regional Development Fund (ERDF); by the Galician Regional Government under projects Consolidation of Research Units 2010/85 and SCALLOPS (10PXIB322231PR); by the Spanish Government under project COMONSENS (CONSOLIDER-INGENIO 2010 CSD2008-00010); by the Iberdrola Foundation through the Prince of Asturias Endowed Chair in Information Science and Related Technologies; by the Concerted Research Action (GOA) Ambiorics 2005/11 of the Flemish Government; and by the IAP Programme P6/26 BCRYPT. C. Troncoso is a research assistant of the Flemish Fund for Scientific Research (FWO). The authors thank G. Danezis and C. Diaz for their comments on earlier versions of the manuscript.

References

1. Agrawal, D., Kesdogan, D.: Measuring anonymity: The disclosure attack. IEEE Security & Privacy 1(6), 27–34 (2003)
2. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24(2), 84–90 (1981)
3. Danezis, G.: Statistical disclosure attacks: Traffic confirmation in open environments. In: Gritzalis, Vimercati, Samarati, Katsikas (eds.) Proceedings of Security and Privacy in the Age of Uncertainty (SEC 2003), Athens. IFIP TC11, pp. 421–426. Kluwer (May 2003)
4. Danezis, G., Diaz, C., Syverson, P.: Systems for anonymous communication. In: Rosenberg, B. (ed.) Handbook of Financial Cryptography and Security. Cryptography and Network Security Series, pp. 341–389. Chapman & Hall/CRC (2009)
5. Danezis, G., Diaz, C., Troncoso, C.: Two-Sided Statistical Disclosure Attack. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 30–44. Springer, Heidelberg (2007)
6. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a Type III Anonymous Remailer Protocol. In: IEEE Symposium on Security and Privacy (S&P 2003), pp. 2–15. IEEE Computer Society (2003)
7. Danezis, G., Serjantov, A.: Statistical Disclosure or Intersection Attacks on Anonymity Systems. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 293–308. Springer, Heidelberg (2004)
8. Danezis, G., Troncoso, C.: Vida: How to Use Bayesian Inference to De-anonymize Persistent Communications. In: Goldberg, I., Atallah, M.J. (eds.) PETS 2009. LNCS, vol. 5672, pp. 56–72. Springer, Heidelberg (2009)
9. Edman, M., Yener, B.: On anonymity in an electronic society: A survey of anonymous communication systems. ACM Computing Surveys 42(1) (2010)
10. Kesdogan, D., Agrawal, D., Penz, S.: Limits of Anonymity in Open Environments. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 53–69. Springer, Heidelberg (2003)

11. Kesdogan, D., Mölle, D., Richter, S., Rossmanith, P.: Breaking Anonymity by Learning a Unique Minimum Hitting Set. In: Frid, A., Morozov, A., Rybalchenko, A., Wagner, K.W. (eds.) CSR 2009. LNCS, vol. 5675, pp. 299–309. Springer, Heidelberg (2009)
12. Kesdogan, D., Pimenidis, L.: The Hitting Set Attack on Anonymity Protocols. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 326–339. Springer, Heidelberg (2004)
13. Liu, J., Xu, H., Xie, C.: A new statistical hitting set attack on anonymity protocols. In: Computational Intelligence and Security, International Conference (CIS 2007), pp. 922–925. IEEE Computer Society (2007)
14. Mallesh, N., Wright, M.: The Reverse Statistical Disclosure Attack. In: Böhme, R., Fong, P.W.L., Safavi-Naini, R. (eds.) IH 2010. LNCS, vol. 6387, pp. 221–234. Springer, Heidelberg (2010)
15. Mathewson, N., Dingledine, R.: Practical Traffic Analysis: Extending and Resisting Statistical Disclosure. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 17–34. Springer, Heidelberg (2005)
16. Möller, U., Cottrell, L., Palfrader, P., Sassaman, L.: Mixmaster Protocol — Version 2. IETF Internet Draft (July 2003)
17. Pérez-González, F., Troncoso, C.: Understanding statistical disclosure: A least squares approach. IEEE Transactions on Information Forensics and Security (under submission, 2012)
18. Pham, D.V., Wright, J., Kesdogan, D.: A Practical Complexity-Theoretic Analysis of Mix Systems. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 508–527. Springer, Heidelberg (2011)
19. Raymond, J.-F.: Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems. In: Federrath, H. (ed.) Anonymity 2000. LNCS, vol. 2009, pp. 10–29. Springer, Heidelberg (2001)
20. Serjantov, A., Danezis, G.: Towards an Information Theoretic Metric for Anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 41–53. Springer, Heidelberg (2003)
21. Troncoso, C., Gierlichs, B., Preneel, B., Verbauwhede, I.: Perfect Matching Disclosure Attacks. In: Borisov, N., Goldberg, I. (eds.) PETS 2008. LNCS, vol. 5134, pp. 2–23. Springer, Heidelberg (2008)

Website Detection Using Remote Traffic Analysis

Xun Gong[1], Nikita Borisov[1], Negar Kiyavash[2], and Nabil Schear[3]

[1] Department of Electrical and Computer Engineering, UIUC
[2] Department of Industrial and Enterprise Systems Engineering, UIUC
[3] Department of Computer Science, UIUC
{xungong1,kiyavash,nikita,nschear2g}@illinois.edu

Abstract. Recent work in traffic analysis has shown that traffic patterns leaked through side channels can be used to recover important semantic information. For instance, attackers can find out which website, or which page on a website, a user is accessing simply by monitoring the packet size distribution. We show that traffic analysis is even a greater threat to privacy than previously thought by introducing a new attack that can be carried out remotely. In particular, we show that, to perform traffic analysis, adversaries do not need to directly observe the traffic patterns. Instead, they can gain sufficient information by sending probes from a far-off vantage point that exploits a queuing side channel in routers.

To demonstrate the threat of such remote traffic analysis, we study a remote website detection attack that works against home broadband users. Because the remotely observed traffic patterns are more noisy than those obtained using previous schemes based on direct local traffic monitoring, we take a dynamic time warping (DTW) based approach to detecting fingerprints from the same website. As a new twist on website fingerprinting, we consider a website detection attack, where the attacker aims to find out whether a user browses a particular web site, and its privacy implications. We show experimentally that, although the success of the attack is highly variable, depending on the target site, for some sites very low error rates. We also show how such website detection can be used to deanonymize message board users.

1 Introduction

Traffic analysis is the practice of inferring sensitive information from patterns of communication. Recent research has shown that traffic analysis applied to network communications can be used to compromise users' secrecy and privacy. By using packet sizes, timings, and counts, it is possible to fingerprint websites visited over an encrypted tunnel [2,4,11,17], infer keystrokes sent over a secure interactive connection [27,34] and even detect phrases in VoIP sessions [31–33]. These attacks have been explored in the context of a *local adversary* who can observe the target traffic directly on a shared network link or can monitor a wireless network from a nearby vantage point [25].

We consider an alternate traffic analysis approach that is available to *remote adversaries*. We notice that it is possible to infer the state of a router's queue through the observed queueing delay of a probe packet. By sending frequent probes, the attacker can measure the dynamics of the queue and thus learn an approximation of the sizes, timings, and counts of packets arriving at the router. In the case of home broadband

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 58–78, 2012.

networks, in particular, DSL lines, the attacker can send probe packets from a geographically distant vantage point, located as far away as another country; the large gap between the bandwidth of the DSL line and the rest of the Internet path makes it possible to isolate the queueing delay of the "last-mile" hop from that experienced elsewhere.

To demonstrate the feasibility of using remote traffic analysis for real malicious attack to learn sensitive information, we adapt the website fingerprinting attack [2, 11, 17], previously targeted at local victims, to a new scenario and introduce a remote website detection attack. Our attack can find out when a victim user under observation visits a particular target site without directly monitoring the user's traffic. This would allow, for example, a company to find out when its employees visit its competitors' sites *from their home computers* or deanonymize users of web boards.

In our adaptation, we encountered two challenges: the information obtained through remote traffic analysis is more noisy than in the local case, and there is no easily available training set from which to create a fingerprint. To address the former problem, we improved on the previous inference methodology, which used the distribution of packet sizes and inter-arrival times, and developed a fingerprint detection technique that makes use of ordered packet size sequences and the dynamic time warping (DTW) distance metric. To create a training set, we designed a testbed that uses an emulated DSL link and a virtual execution environment to replicate the victim's home environment.

To evaluate our work, we sent probes to a home DSL line in the United States from a rented server in a data center near Montreal, Canada; we chose this set up to demonstrate the low cost and barrier to entry to conduct the attack. We then compared the probe results with profiles of website fetches generated in a virtual testbed at our university. We tested our attack on detecting each of a list of 1 000 popular websites. We found that detection performance was highly variable; however, for a significant fraction of sites, it was possible to obtain very low false-positive rates without incurring significant false-negative rates. We also found that there is some accuracy loss due to the discrepancies in the test and training environments (distant from each other) that we were not (yet) able to eliminate. If the training and test data are both collected from the same location, a much larger fraction of sites can be accurately detected with low error rates. We find that despite working with a much noisier information source than previous web fingerprinting work [2, 11, 17], our website detection attack nevertheless shows that remote traffic analysis is a serious threat to Internet privacy.

The rest of the paper is organized as follows. We describe our approach to remote traffic analysis in §2. In §3 we describe our adaptation of previous website fingerprinting attack to remotely confirming user's browsing activities. We evaluate our website detection attack in §4. We then discuss further extensions and the limitations of our technique in §5 and present related work in §6, concluding in §7.

2 Remote Traffic Analysis

Traffic analysis attacks have been known to be effective for quite some time. And yet, for most Internet users, they represent a minor concern at best. Although a dedicated attacker could always intercept traffic by, say, bribing a rogue ISP employee, or tapping a switch box, he would run the risk of being caught and potentially incurring criminal

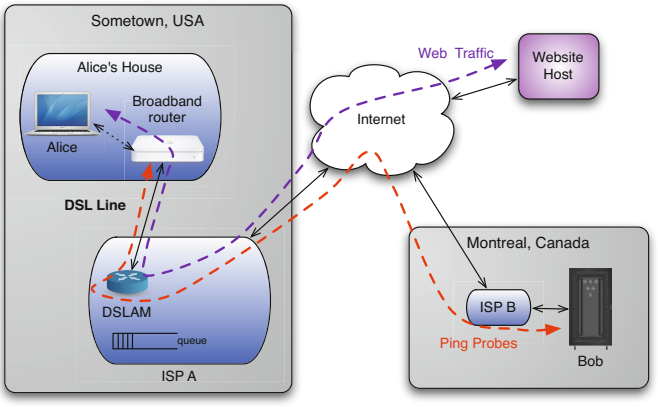


Fig. 1. Queueing side channel. Bob remotely sends probes to Alice's router to infer her activities.

charges. In any case, this level of effort seems justified only for highly sensitive material, rather than casual snooping; therefore, as long as sensitive data are protected by encryption or other techniques, a user may feel relatively safe.

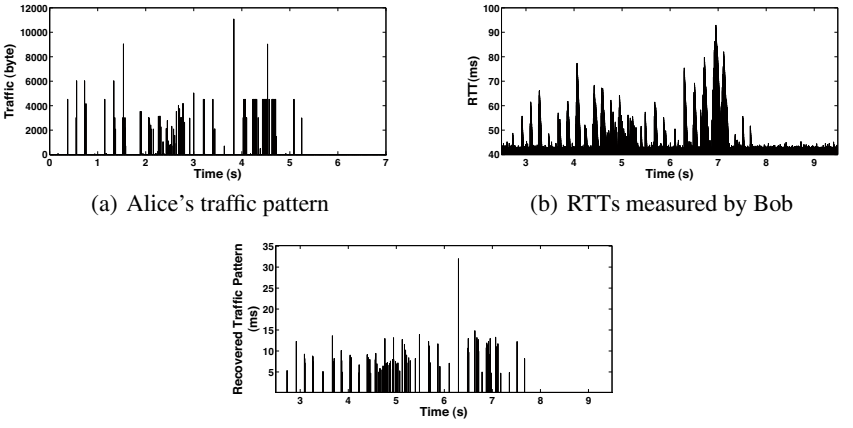
We show, however, that traffic analysis can be carried out at a significantly lower cost, and by attackers who never come into physical proximity with the user. In fact, the attackers can launch their attacks from another state or country, as long as they have access to a well-provisioned Internet connection. This, in turn, is very easy to obtain due to the highly-competitive Internet hosting business sector: a virtual private server in a data center can cost as little as a few dollars a month.[1] We show that the attacker's traffic has very low rate, thus attackers do not need to incur high bandwidth costs. Furthermore, users who are being spied upon are unlikely to notice the small amount of performance overhead. Thus, anyone with a credit card[2] can carry out the attack and leave little trace.

In this section, we describe our approach to remote traffic analysis. We first introduce the queueing side channel, which is the basis of the attack. Then we design an algorithm to recover users' traffic patterns from the information leaked through this side channel.

2.1 Queuing Side Channel

We consider the following scenario as depicted in Figure 1. Alice is a home user at Sometown USA, browsing a website via her DSL Internet connection. Her computer is connected to a broadband router, using a wireless or wired LAN connection.[3] The router is connected via a DSL line to a DSLAM[4] or similar device operated by her ISP, which is then (eventually) connected to the Internet. Unbeknownst to Alice, Bob, who is located in another state, or another country wishes to attack Alice's privacy. If Bob

[1] See, for example, `www.vpslink.com` (retrieved February 2011).
[2] Working stolen credit cards are an easily acquired commodity on the black market [10].
[3] In some cases, Alice's computer might be connected to the DSL line directly.
[4] DSL access multiplexer.



(a) Alice's traffic pattern

(b) RTTs measured by Bob

(c) Recovered traffic pattern

Fig. 2. Real traffic on a DSL vs. probe RTTs. Alice resides in Champaign, IL, while Bob is located in Montereal, Canada.

knows Alice's IP address (for example, if Alice visited a site hosted by Bob), he can use his computer to send a series of ICMP echo requests (pings) to the router in Alice's house and monitor the responses to compute the round-trip times (RTTs). One component of the RTTs is the queueing delay that the packets experience at the DSLAM prior to being transmitted over the DSL line; thus the RTTs leak information about the DSLAM queue sizes. This leakage in turn reveals traffic patterns pertaining to Alice's activities.

Since the probe packets traverse many Internet links, and the queuing delays on Alice's DSL link are but one component of the RTT, the question is, how much information is leaked by this side channel? Furthermore, can it be used to infer any information about Alice's activities? To evaluate the potential of this attack, we carried out a test on a home DSL link located in Champaign, IL, USA. In the test, Alice opens a Web page www.yahoo.com on her computer. Simultaneously, Bob in Montreal, QC, Canada sends a ping request every 10 ms to Alice's home router. Figure 2(a) depicts the traffic pattern of Alice's download traffic. The height of each peak in the figure represents the total size of packets that are downloaded during each 10 ms interval. Figure 2(b) plots the RTTs of Bob's ping requests. We can see a visual correlation between the traffic pattern and observed RTTs; whenever there is a large peak in the user's traffic, the attacker observes a correspondingly large RTT.

The correlation between Alice's traffic and Bob's observed probe RTTs can be explained as follows. The RTTs include both the queuing delay incurred on the DSL link and delays on intermediate routers, which sit between Bob's computer and Alice's router. The intermediate routers are typically well provisioned and are unlikely to experience congestions [1, 16]; furthermore, the intermediate links have high bandwidth and thus queueing delays will be small in all cases. We validate this using our own measurements in §4.1.

On the other hand, Alice's DSL link is, by far, the slowest link that both her traffic and Bob's probes are likely to traverse. The queue at Alice's router can grow to be quite

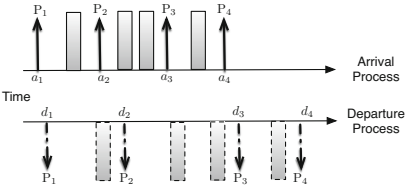


Fig. 3. FIFO queuing in the DSL router

long (in relative terms), due to TCP behaviors, which cause the `www.yahoo.com` server to send a batch of TCP packets at a fast rate. As most routers schedule packets in a First In First Out (FIFO) manner, this congestion will lead to large queuing delays of Bob's ping packets. We saw that the additional delay caused by Alice's incoming traffic could be as high as over 100 ms. Thus, Alice's traffic patterns are clearly visible in the RTTs seen by Bob.

2.2 Traffic Pattern Recovery Algorithm

We now show how the attacker can analyze the information leaked through this queueing side channel. We model the incoming DSL link as a FIFO queue. As most traffic volume in an HTTP session occurs on the download side, we will ignore the queuing behavior on the outgoing DSL link, though it could be modeled in a similar fashion.

Figure 3 depicts the arrival and departure process in this queuing system. The arrows are Bob's ping packets, denoted by P_i's, and the blocks represent HTTP packets downloaded by Alice. The DSLAM serves packets in FIFO manner and at a constant service rate; i.e., the service time is proportional to the packet size. As most HTTP packets are more than an order of magnitude larger than ping packets, we ignore the service time for pings.

Assume that ping packet P_i arrives in the queue at time a_i, waits for the router to serve all the packets currently in the router, and then departs at time d_i. Let us consider the observed RTT of the ping packet P_i; we can represent it as:

$$RTT_i = \sum_{l \in \text{links on path}} q_i^l + p_i^l + t_i^l \tag{1}$$

where q_i^l, p_i^l, and t_i^l are the queueing, propagation, and transmission delays incurred by packet P_i on link l. Note that the propagation and transmission delays are mostly constant, and in fact we can approximate:

$$\sum_{l \in \text{links on path}} p_i^l + t_i^l \approx \min_j RTT_j \tag{2}$$

since Bob is likely to experience near-zero queueing delays for some of the pings. Furthermore, as argued in §2.1, the queueing delay on links other than the DSL line are going to be minimal, thus we can further approximate:

$$RTT_i \approx \min_j RTT_j + (d_i - a_i) \tag{3}$$

Algorithm 1. Traffic pattern recovery algorithm

let $d_0 = 0$
for $i = 1$ to *the probe sequence length* **do**
  # reconstruct the arrival and departure times using the ping interval
  $a_i = t_{ping} \cdot i$
  $d_i = RTT_i - RTT_{min} + a_i$
  # estimate the total size of packets arriving in $[a_{i-1}, a_i]$
  $\widehat{s_i} = d_i - max(d_{i-1}, a_i)$
  # discard noise
  if $\widehat{s_i} < \eta$ **then**
    $\widehat{s_i} = 0$
  end if
end for

Making use of the queuing delay $d_i - a_i$ from (3), the attacker Bob can further infer the total size of HTTP packets arriving during the interval $[a_{i-1}, a_i]$'s, which produces a similar pattern as Alice's traffic in Figure 2(a). For this purpose, two cases need to be considered.

1. $a_i \geq d_{i-1}$. In this case, when P_i enters the queue, the DSLAM is either idle or serving packets destined for Alice. The delay $d_i - a_i$ reflects the time required to finish serving the HTTP packets currently in the buffer, and is thus approximately proportional to the total size of Alice's arrivals during the interval $[a_{i-1}, a_i]$. P_2 in Figure 3 is one example of this case.
2. $a_i < d_{i-1}$. In this case, P_{i-1} is still in the queue when P_i arrives. Only after P_{i-1} departs at d_{i-1}, the router can start to serve packets that arrived in the interval $[a_{i-1}, a_i]$. Thus the delay $d_i - d_{i-1}$ is the service time for those packets and can be used to recover the total size. P_4 in Figure 3 is one example of this case.

Algorithm 1 summarizes the traffic pattern recovery procedure based on these observations. To account for minor queueing delays experienced on other links, we define a threshold η such that RTT variations smaller than η are considered noise and do not correspond to any packet arrival at the DSLAM. Figure 2(c) plots the pattern extracted from RTTs in Figure 2(b). After processing, the resulting time series proportionally approximate the packet size sequence of the original traffic in Figure 2(a). As will be shown in the next section, it can be applied to infer more information about Alice's activities, e.g., website fingerprinting.

Note that in case 1, the attacker may underestimate the size of the HTTP packets arriving in the period $[a_{i-1}, a_i]$ because a portion of them will have already been served by time a_i. The error depends both on the frequency of the probes and the bandwidth of the DSL link. Since most HTTP packets are of maximal size (MTU), we can ensure that all such packets are observed by setting the ping period to be less than: $\frac{\text{MTU}}{\text{DSL bandwidth}}$. Thus the adversary must tune the probe rate based on the DSL bandwidth and faster links will require a higher bandwidth overhead (but the pings will form a constant, small fraction of the overall DSL bandwidth.)

3 Website Fingerprinting

Previous work on traffic analysis has shown that it is often possible to identify the website that someone is visiting based on traffic timings and packet sizes [2, 11, 17], namely, *website fingerprinting*. We consider whether it is possible to carry out a similar attack using our remote traffic analysis. We first review the three basic steps in previous work when conducting a website fingerprinting attack.

1. First, the attacker decides some feature of web traffic used to distinguish websites. The feature needs to stay relatively stable for accesses to the same single website, but has significant diversity across different sites. For example, Herrmann et al. use the size distribution of HTTP packets [11].
2. The next step is the *training* procedure. The attacker needs a training data set of fingerprint samples labeled with corresponding destination websites. Usually, these feature profiles are obtained by the attacker browsing websites himself/herself from the same (or similar) network connection as the user.
3. In the final step, the attacker *tests* his/her knowledge from training on the victim user. He/She monitors traffic going to the user and matches extracted features with the profiles in his/her database. The one with most similarity is chosen as the website browsed by the user.

As compared with previous work, using our remote traffic analysis technique for identifying websites introduces two additional challenges. First, previous work used fine-grained information like exact packet size distributions to create features, whereas in our setting this information is not available directly, since the queueing side channel produces only approximate sums of packet sizes. Second, previous work created a training set from *the same vantage point* that was then used for fingerprinting tests. An attacker performing remote traffic analysis must, of course, use a different environment for collecting the training set, potentially affecting the measured features. We describe our approaches to solving these two challenges next.

3.1 Time Series–Based Feature

Since it is hard to infer information about each single packet from our recovered pattern time series, we use the entire time series, which contains the estimated size of all HTTP packets downloaded during each probe period, to create one fingerprint trace. Identification of websites is based on the similarity between the observed fingerprints and samples in the training set.

The challenge is to find a meaningful distance metric between fingerprint traces. Note that pointwise comparisons will produce poor results. This is because parts of the fingerprint may be impacted by the noise from a small queueing delay on a core Internet link. Additionally, the fingerprint could miss some packets contained in the original traffic due to pattern recovery errors. Finally, even fingerprints of the same website are not strictly synchronized in time due to the inter-packet delay variations. To deal with these issues, we turn to the Dynamic Time Warping (DTW) distance [24]. DTW was developed for use in speech processing to account for the fact that when

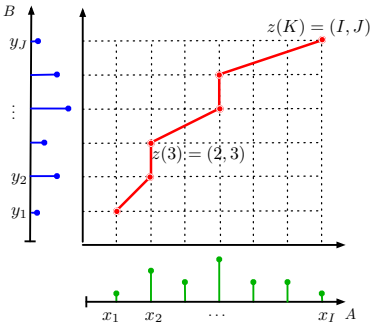


Fig. 4. Warping function in DTW

people speak, they pronounce various features of the phonemes at different speeds, and do not always enunciate all of the features. DTW attempts to find the best alignment of two time series by creating a non-linear time warp between the sequences. Figure 4 visualizes the DTW-based distance between two time series: $X = \{x_1, x_2 \ldots, x_I\}$ and $Y = \{y_1, y_2 \ldots, y_J\}$. Let function $F(z) = \{z(1), \ldots, z(K)\}$ be a mapping from series X to series Y where $z(k) = (x(i), y(j))$. For every pair of matched points based on the mapping, we define the distance as $d(z(k)) = d(i, j) = |x_i - y_j|$. The final distance between the X and Y can then be defined as a weighted and normalized sum over all matched point pairs as $D(X, Y) = \min_F \left\{ \frac{\sum_{k=1}^{K} d(z(k))w(k)}{\sum_{k=1}^{K} w(k)} \right\}$. The weights $w(k)$'s are flexible parameters picked based on the specific application scenario. Applying dynamic programming, one can find the warping function with minimum distance, which captures the similarity between the two time series under best matched alignment.

In our attack, we apply DTW-based distance to account for the estimation errors and time desynchronizations in fingerprints. Based on the distances with the training data set, the attacker will know if a test sample indicates the activity that the user browsed the website of interest.

3.2 Training Environment

To obtain an accurate training fingerprint for a particular user's traffic, the attacker must be able to replicate the network conditions on that user's home network. The approach we use is to set up a virtual machine running a browser that is connected to the Internet via a virtual Dummynet link [23]. The virtual machine is then scripted to fetch a set of web pages of interest; at the same time, an outside probe is sent across the Dummynet link, simulating the attack conditions on a real DSL link.

A number of parameters of the link need to be carefully decided. We found that the most important parameter for the attacker to replicate was the link bandwidth. First, as discussed in §2.2, the probe frequency should be adjusted based on the link bandwidth. Bandwidth also affects the magnitude of observed queuing delays. Additionally, it can significantly alter the traffic pattern itself, as TCP congestion control mechanisms are affected by the available bandwidth. Fortunately, estimating the bandwidth on a link is a well-studied problem [20, 22, 28]. In our tests, we use a packet-train technique by sending a burst of probe packets and measuring the rate at which responses are returned.

Since most DSL lines have asymmetric bandwidth, we used TCP ACK packets with 1000 data bytes to measure the download bandwidth on the link. The target would send a short TCP reset packet for each ACK that it received, with the spacing between resets indicating the downstream bandwidth; we found this method to be fairly accurate.

The round-trip time between the home router and the website hosts also affects the fingerprint. When opening a webpage, the browser can download objects from several host servers. The traffic pattern is the sum of all download connections, hence the shape of observed fingerprint does depend on the RTTs to these servers. However, we did not explicitly model this parameter considering the difficulty to accurately tune up the link delays to multiple destinations. The effects to the attack will be further discussed in §4.

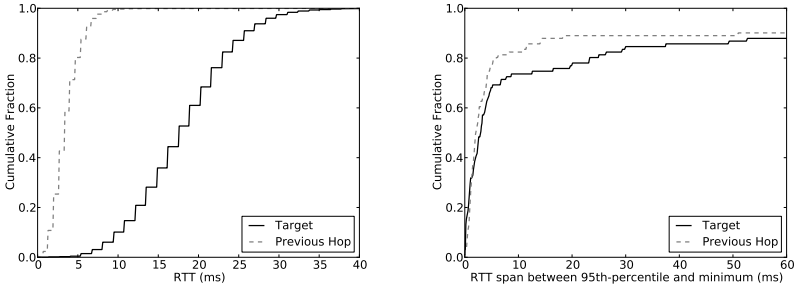
The fingerprint may be affected by the choice of browsers and operating systems as well; for best results, the training environment should model the target as closely as possible. Information about browser and operating system versions can be easily obtained if the target can be convinced to visit a website run by the attacker; additionally, fingerprinting techniques in [18] may be used to recover some of this information.

3.3 Attack Scenarios

We consider several attack scenarios that make use of website fingerprinting. We can first consider the classic website fingerprinting scenario: Bob obtains traces from Alice's computer by sending probes to her DSL router and compares them to fingerprints of websites that he has generated, in order to learn about her browsing habits. Note that this can be seen as a *classification* task: each web request in Alice's trace is classified as belonging to a set of sites. This scenario has been used in most of the previous work on website fingerprinting, but it introduces the requirement that Bob must know the set of potential sites that Alice may visit. Without some prior information about Alice's browsing habits, this potential set includes every site on the Internet, making it infeasible to generate a comprehensive set of fingerprints. One could create fingerprints for popular sites only, but this reduces the accuracy of the classification task [6,11,29]. For example, the top 1 000 US sites, as tracked by Alexa, are responsible for only 56% of all page views, therefore, even a perfect classifier trained on the 1 000 sites would give the wrong result nearly half the time.[5]

We therefore consider a different scenario, where Bob wants to *detect* whether Alice visits a *particular* site. For example, if Bob is Alice's employer, he may wish to check to see if she is considering going to work for Bob's competitor, Carol. To carry out this attack, Bob would create a fingerprint for Carol's jobs site; he would then perform a binary classification task on Alice's traffic, trying to decide whether a trace represents a visit to the target site or some other site on the Internet. As we will see, such binary classification can be performed with relatively high accuracy for some choices of sites. Note that, as Alice's employer, Bob has plenty of opportunities to learn information about Alice's home network, such as her IP address, browser and operating system versions, and download bandwidth, by observing Alice when she connects to a password-protected Intranet site, and can therefore use this information to create accurate training data for building fingerprints.

[5] In fact, the situation is even worse, since Alexa counts *all* page views within a certain top-level domain, whereas fingerprints must be created on each individual URL.



(a) Empirical CDF of ping RTTs from a typical host.

(b) Empirical CDF of 95th percentile minus minimum RTTs.

Fig. 5. Measurement of DSL probe variances

As another example, Bob may be trying to identify an employee who makes posts to a web message board critical of Bob.[6] Bob can similarly build profiles, tailored for each employee's home computer, of the web board and perform remote traffic analysis. He can then correlate any detected matches to the times of the posts by the offending pseudonym; note that this *deanonymization* attack is able to tolerate a significant number of false-positive and false-negative errors by combining observations over many days to improve confidence [7].

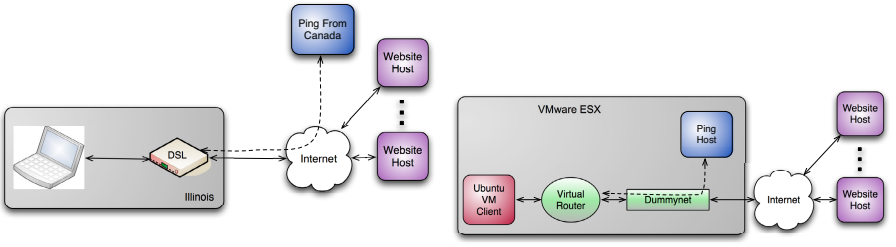
4 Evaluation

We next present our results of website detection attack. First, through measurements of DSL probe variances, we demonstrate the feasibility of the RTT based remote traffic analysis, which is the basis of our attack.

4.1 Measurement of DSL Probe Variance

To further confirm that the fluctuation of user's RTTs are primarily determined by the congestion at the DSL link, we conducted a small Internet measurement study. We harvested IP addresses from access logs of servers run by the authors. We noted that many DSL providers assign a DNS name containing "dsl" to customers. Using reverse DNS lookups, we were able to locate 918 potential DSL hosts. To determine each host's suitability for measurement, we first determine if it responds to ping requests. We then use traceroute to locate the hop before the target DSL host (e.g., the DSLAM). Next, we ensure this previous hop also responds to ping requests. Lastly, we measure the minimum RTT of several hundred ping probes. We exclude any host with a minimum RTT of greater than 100ms to bound the study to hosts in a wide geographical area around our Montreal, Canada probe server. Using this method, we identified 189 DSL

[6] This example is motivated by several actual cases of companies seeking to do this; see https://www.eff.org/cases/usa-technologies-v-stokklerk and https://www.eff.org/cases/first-cash-v-john-doe.



(a) The DSL-Setup for test (b) The VM-Setup for training

Fig. 6. Experimental setups for website detection

hosts to measure.[7] The measurement consists of sending ping probes every 2 ms for 30 seconds to the target DSL host and then to its previous hop. We collected these traces in a loop over a period of several hours.

We found that, on average, the target host RTT was ∼10 ms greater than the previous hop. We frequently observed the pattern in Figure 5(a) where the previous hop RTT was very stable and target RTT variations greater than 10 ms. We then measured the span between the 95th percentile and the minimum observation in each sample. Figure 5(b) shows the CDFs of this data for the each target DSL host and its previous hop from the measurement set. We see that the previous hop RTT span shows more stability than the end host, confirming the one of the primary assumptions of our work.

4.2 Attack Setups and Data Collection

We built a *DSL-Setup* consisting of a target system and a ping server, as shown in Figure 6(a). The target system captured the real environment of a home user. It ran on a laptop, located in Champaign, IL, connected to DSL line with 3 Mbps download and 512 Kbps upload speeds. On the laptop, we used a shell script to automatically load websites using Firefox 4.0[8]. The ping server was a commercial hosting system, located in Montreal, QC, Canada, acting as the remote attacker. It was scripted to send pings at precise time intervals with hping[9] and record ping traces with tcpdump[10]. We set the ping interval to 2 ms.

To emulate the attacker's training procedure, we also built a *VM-Setup*, a VMware ESX host testbed located in our lab, as shown in Figure 6(b). On this machine, we ran several VMware guest operating systems: a Ubuntu VM Client, a virtual router and a host implementing a transparent Dummynet link. The Ubuntu VM Client acted as a virtual target, and was scripted to browse websites using Firefox, similar to the real home user. The virtual router provided NAT service for the client, and was connected

[7] This number is underestimated as many of the IPs on our server log already became out-of-date (the hosts hooked went offline) by the time of ping tests. More accurate number can be obtained if a list of valid DSL IPs is provided in real time.

[8] http://www.mozilla.com/firefox/

[9] http://www.hping.org

[10] http://www.tcpdump.org

to the Internet through the Dummynet link. The Dummynet bridge was configured to replicate the network conditions of the target DSL link (i.e., the bandwidths). As in the DSL-Setup, we sent probes from another host outside the constrained Dummynet link to the virtual NAT router periodically. The attacker then collected training fingerprints while the virtual client was browsing websites through this virtual 'DSL' link. Note the virtual router and ping host were connected to the same dedicated high-speed LAN minimizing the impact of additional noise added by intermediate routers or network congestion caused by other hosts.

We collected fingerprints of the front pages for 1000 websites on the top list on Alexa[11]. For websites which have multiple mirrors in different countries like google.com, we only considered the site with the highest rank. We excluded websites with extremely large loading time (greater than 60 s). For each website, we collected 12 fingerprint samples from both the DSL and VM setups. The delay between collecting two samples is half an hour. Following the same assumptions in previous papers [11,17], the browsers were configured appropriately (no caching, no automatic update checks and no unnecessary plugins). This makes our results comparable with previous work.

4.3 Website Detection

We first analyze the ability of an attacker to detect whether a user visits a particular site. To do so, the attacker checks whether the distance between the user trace and the target web site is smaller than some threshold, and if so, the web site is considered detected. This is a binary classification task and its performance can be characterized by the rates of false positives—a different website incorrectly identified as the target–and false negatives–the target website not being identified. The choice of threshold ν creates a tradeoff between the two rates: a smaller threshold will decrease false positives at the expense of false negatives.

To estimate false-positive rate given a particular threshold ν, we fix a target site w and use the 12 samples $T = \{s_{w,1}, \ldots, s_{w,12}\}$ as the training set. We use the samples from the other sites as a test set; i.e., $U = \{s_{i,j}\}$ for $i \neq w, j \in \{1, \ldots, 12\}$. Given a sample $s_{i,j} \in U$, we calculate the minimum distance from it to the training samples:

$$d_w(s_{i,j}) = \min_k D(s_{i,j}, s_{w,k}) \tag{4}$$

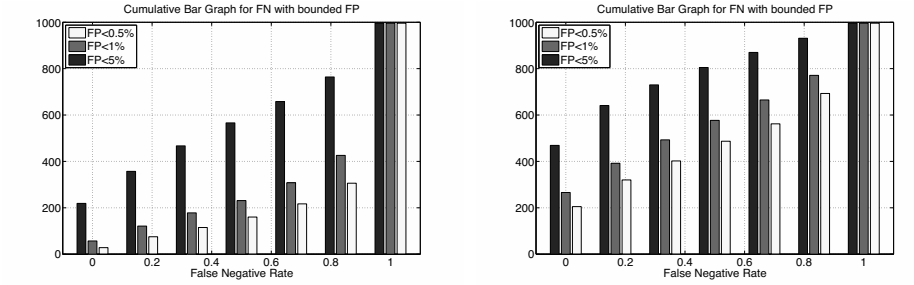
where $D(\cdot, \cdot)$ is the DTW-based distance function defined in §3.1. We then consider every sample $s_{i,j} \in U$ such that $d_w(s_{i,j}) < \nu$ to be a false positive and therefore estimate the false-positive rate:

$$\hat{p}_w = \frac{|\{s_{i,j} \in U | d_w(s_{i,j}) < \nu\}|}{|U|} \tag{5}$$

To estimate the false-negative rate, we pick one of the sample $s_{w,i}$ and calculate its minimum distance to the other 11 samples $s_{w,j}, j \neq i$, and count it as a false negative if the distance is at least ν. We then repeat this process for each $i = 1, \ldots, 12$:

$$\hat{q}_w = \frac{|\{s_{w,i} | i \in \{1, \ldots, 12\}, \min_{j \neq i} d(s_{w,i}, s_{w,j}) \geq \nu\}|}{12} \tag{6}$$

[11] http://www.alexa.com



(a) VM–DSL case (b) DSL–DSL case

Fig. 7. Number of sites with a given false negative rate or smaller

Given a target false positive rate of p_w^*, we can calculate the threshold ν_w^* that would ensure $p_w < p_w^*$. Note that because $\hat{p_w}$ is only an estimate of p_w, we calculate a 95% confidence interval for p_w and chose ν such that the upper limit of the CI is below p_w^*[12]. Note that this threshold will be different for each site. We can then estimate the corresponding false negative rate $\hat{q}_w^*$ that corresponds to ν_w^*.

The target false positive rate will largely depend on the prior knowledge the attacker has. Typically, we will want to aim for a small false-positive rate, since even if Bob considers it likely that Alice does in fact visit the target site w *at some point*, most of the web browsing in any trace will still be to other sites; thus a low false-positive rate is needed for the test to have high positive predictive value. On the other hand, Bob can easily tolerate a moderate false-negative rate, since even if he only finds out about employees searching for other jobs 90%, or even 50% of the time, this information is useful nevertheless. Likewise, perfect detection is not needed for the potential attack to have a chilling effect on Alice's behavior.

Figure 7 shows the false negative rates that can be achieved given a target false-positive rate below 0.5%, 1%, and 5%. Each bar represents a cumulative number of websites, i.e., websites for which $\hat{q}_w^*$ is at or below the x-axis value. We show two sets of results; one using the VM setup for training and DSL for testing (Figure 7(a)) and one using the DSL samples for both training and test data sets (Figure 7(b)). Note that there is a significant difference between the two graphs, resulting from the discrepancies between the simulated (VM) and the test environment. We expect that, with some work, an attacker may be able to reduce such discrepancies by more carefully tuning the parameters of the virtual machine and the simulated link, or by using actual hardware and a real DSL line that mimics Alice's setup. The DSL–DSL case therefore shows the limits of what can be achieved by improving the training environment.

An important observation is that, in both cases, the success of the web detection is highly dependent on the target site. For a small number of sites—75 in the VM–DSL case and 320 in the DSL–DSL case—the web detection attack works very well: we are

[12] We use a binomial proportion confidence interval here. This is slightly imprecise, as the $12{\cdot}999$ samples are not independent; we leave computation of confidence intervals that take this into account for future work.

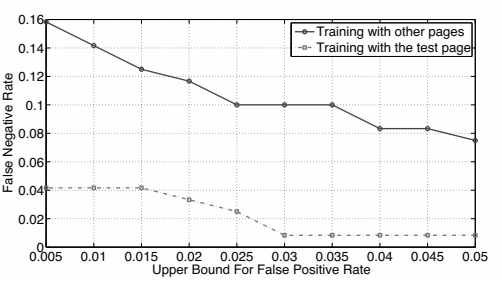


Fig. 8. Detection performance for Warrior Forum when training with the correct post page and when using other pages

able to maintain a very low false-positive rate of 0.5% while experiencing few false negatives (17% or below). On the other hand, some sites are virtually invulnerable to our attack: for 65 of the sites we tested, we were unable to observe *any* true positives with a target false-positive rate of 5% (i.e., $\hat{q}_w^* = 100\%$), even in the best-case DSL–DSL scenario. We found these sites to have either very short traces, making it difficult to distinguish them from other such sites, or highly variable traffic patterns due, for example, to dynamic content, making it difficult to create a useful fingerprint. Note the attacker can predict false-positive rates for different websites ahead of time using experiments carried on VM setups. Making use of this knowledge would make the attack even stronger.

4.4 Deanonymization

We next consider the deanonymization attack described in §3.3. As a case study, we considered the site www.warriorforum.com, a popular Internet marketing forum. It uses the vBulletin software, which was, as of August 2011, the most popular bulletin board software[13], and thus should be representative of a number of other sites. In our attack scenario, Bob wishes to find out if Alice is using a particular pseudonym (say, "diane123") to post on the site. To accomplish this, he first collects traces from Alice's home computer for a period of time. He then waits for posts to the forum from diane123 and performs a detection attack to see if Alice was visiting the site at the time of post. Repeated successful matches can then be used to obtain increasing confidence in tying Alice to diane123.Note that Bob will need to build a profile that targets internal pages of `www.warriorforum.com`, rather than the front page. Alice's post requests will be too small to create an easily-observable feature; however, vBulletin displays the forum thread after a post has been made. Therefore, Bob can collect samples visiting threads where diane123 has posted to create a fingerprint. Note that, in this attack, fingerprint creation happens *after* the trace collection. A problem facing Bob is that different post pages on Warrior Forum will have similar features in their RTT profile. A match, therefore, can show that Alice visited *some* Warrior Forum page with high confidence, but it may not have been the correct thread. Even this information, however, is likely to

[13] http://www.big-boards.com/statistics/

be enough for deanonymization. For example, Alexa shows that fewer than 1% of US Internet users actually visit the Warrior Forum, and those that do tend to stay on the site less than 10 minutes on average. If we make the simplifying assumptions that these visits are distributed randomly across a three-hour evening period, the additional false-positive rate due to random visits to *other* Warrior Forum pages is no more than 6%, even if Alice is *known* to be a Warrior Forum user. Combining observations across several posts allows Bob to improve his confidence.

Over a long term, even simpler attacks may suffice: since most people's Internet usage is bursty, simply observing that Alice always actively used the Internet in some way whenever diane123 made posts can be used for deanonymization [7]. Likewise, Bob may be able to rule out Alice as a suspect if she was known to be at home (due to recent DSL activity) but her connection was idle at the times of the target posts.

Finally, Bob may be able to use the similarity between internal forum pages to his advantage. In particular, suppose that Alice publicly participates in the forum under her real identity, in addition to potentially posting under a pseudonym. Bob can use the times of Alice's posts under her real name to label traces collected from Alice's computer and create a training set. In this case, Bob does not need to simulate Alice's computing environment as the training and test environments are exactly the same—the ideal conditions we used in the DSL–DSL case. To study this attack, we collected samples from 100 different posts on the Warrior Forum site. For each sample, we attempt to match it to a fingerprint created from the other 99 posts; our process is similar to (6), except using a different threshold for each sample. From this, we estimate the false negative rate for a given target false-positive rate, calculated using (5), using the traces from 999 other websites. Figure 8 shows the results. The use of different pages to test degrades the matching performance, but it still provides sufficient detection power for deanonymization after a few posts.

5 Discussion

In this section, we discuss more on the feasibility and limitations of our work.

1. *ICMP support.* The attack scenario of our scheme relies on ICMP probe packets, hence we care about whether ICMP is enabled in real routers. In testing over 918 probable DSL hosts on the Internet in Section §4.1, we found over 25% responded to ping requests. Since we harvested these probable DSL hosts from the Internet over a period of several months, it is not clear how many that failed to respond were simply down rather than blocking our probes. Thus, we can assume that the fraction of hosts that respond to ping is even larger. Additionally, in a brief survey of consumer-grade router hardware, we found that many of them do not perform ICMP filtering, at least not in the default configuration. Moreover, even though the ping packets are blocked by firewalls on some home routers, other forms of probes may be exploited as well; for example, if the home router exposes TCP ports for file sharing or other applications, SYN packets can be used as probes with the same effectiveness.
2. *FIFO scheduling policy.* The high correlation between the user's traffic pattern and the attacker's ping RTTs comes from the fact that the router serves packets in FIFO

order. Note that most home routers today do not use QoS extensions and schedule packets on a given link in FIFO order. Thus, information leaked by these routers can be exploited with remote traffic analysis. Certainly, a fair queuing implementation [26] would reduce the impact that cross-traffic would have on the probe sequence and hence reduce the effectiveness of the side channel, but not entirely eliminate it [15].

3. *Limited Last-hop Bandwidth.* The information leaked through our side channel are the states of the queue length in the router's buffer. Hence, to have nontrivial queues built up in the buffer, the broadband link must have limited bandwidth compared to the rest of the links in the path. In our experiments, we have used speeds typical of current home broadband speeds—several Mbps, and our scheme worked well in those environments. The deployment of faster links, such as Fiber-to-the-Home (FTTH), may reduce the effectiveness of the queueing side channel, but notice that if the core network is similarly upgraded in speed, the bandwidth disparity necessary for our attack will remain.
4. *Victim's IP address.* In our attack, the attacker needs to know the user's IP address to send the probes. Although this mapping is typically only explicitly known to ISPs, many protocols, such as file sharing, instant messaging, VoIP, and email, will reveal the IP address of a user. Other forms of IP address reconnaissance may also be possible but are outside the scope of this work.
5. *Multiple users.* In the scenario of our remote traffic analysis, the attacker's probes cannot distinguish between the traffic of multiple users on the same link, so shared broadband connections present an obstacle to our attack. However, even in multi-user installations, it is still common for only one user to be using the Internet at any given point during the day. Some previous work on traffic analysis has used blind source separation to separate traffic from multiple users [35]; similar techniques may be applicable here. For example, in Figure 2, traffic follows a periodic pattern based on the RTT between Alice and the website; such periodicity might help separate the sources.
6. *Dynamic nature of websites.* Our attack relies on web sites having relatively stable fingerprints. Although the overall pattern captured by our RTT probes remains static enough within days, the website content may incur significant changes (e.g., site redesigns) over time; which in turn will result in a change of its fingerprint. Thus, for best results, the training set should be updated continuously. This limitation applies to any website fingerprinting approach even local website fingerprinting techniques which benefit from better vantage points [11,17].
7. *Content distribution networks.* Websites that use content delivery networks (CDNs) will use different servers to deliver content based on the user's location. They may present localized versions of the site to users in different countries or regions. As shown in our experimental results, this can cause fingerprints to differ significantly. If identifying these sites is a high priority for the attacker, additional work would be needed to obtain fingerprints of the right version by, for example, using proxies and other techniques to fool IP-based localization.
8. *Cache issues.* In our tests, we followed the assumption in previous work [11,17] and disabled the cache in the browser. This implies that our results demonstrates

the attacker's ability to verify that a user visits a web page for the first time. To investigate cases with cache enabled, one possible solution would be build separate fingerprints based on the time since the site was first downloaded, e.g., after 1 hour, 6 hours, 1 day, 1 week, to minimize the effect that caching would have on the attack. Note that with continuous observation of a computer, the attacker may be able to guess how long ago the last visit was.

6 Related Work

The use of network probes to infer information about traffic at a remote location has been explored in previous work in the context of anonymous communication networks. Murdoch and Danezis used a remote traffic analysis approach to expose the identity of relays participating in a circuit in the Tor [8] and MorphMix [21] anonymous communication networks [19]. Their approach was to send an on–off pattern of high-volume traffic through the anonymous tunnel and a low-volume probe to a router under test. If the waiting times of the probe showed a corresponding increase during the "on" periods, the router was assumed to be routing the flow. However, when Murdoch and Danezis evaluated their attack, the Tor network was lightly loaded and consisted only of 13 relays; to repeat their attack on today's network, with around 2 000 relays and high traffic load[14], an attacker would needs extremely large amounts of bandwidth to measure enough relays during the attack window. Evans et al. [9] strengthened Murdoch and Danezis's attack of by a bandwidth amplification attack which make their attack feasible in modern-day deployment of Tor. Hopper et al. [13,14] use a combination of Murdoch and Danezis's approach and pairwise round trip times (RTTs) between Internet nodes to correlate Tor nodes to likely clients. Chakravarty et al. [3] propose an attack for exposing Tor relays participating in a circuit of interest by modulating the bandwidth of an anonymous connection and then using available bandwidth estimation to observe this pattern as it propagates through the Tor network. Note that these techniques relied on detecting a specially-crafted coarse-grained communication pattern, whereas our attack makes use of fine-grained information obtained through remote traffic analysis.

We also survey previous work on recovering information about encrypted HTTP traffic.The fact that object sizes could be used to infer sensitive information, even after encryption, was first mentioned by Yee (as related by Wagner and Schneier [30]). A specific concern listed by Yee is that the particular page within a site accessed by the user could be revealed by considering URL and object lengths. Chen et al. [4] applied this observation to AJAX applications to recover detailed information about the internal state of the application and users' data. Cheng et al. [5] present the earliest implementation of website fingerprinting. The classification features used in their scheme are the object sizes and the HTML file sizes. Hintz [12] and Sun et al. [29] both consider website fingerprinting attacks in SSL-encrypted HTTP connections. Their classification features are object sizes and counts. While Hintz did not present implementation details and experimental results, Sun et al. use a Jaccard's coefficient based classifier and show that their attack can achieve a correct identification rate of 75%. Instead of looking at web objects, Bissias et al. [2], Liberatore et al. [17], and Herrmann et al. [11] study

[14] See `http://torstatus.blutmagie.de` (retrieved November 2010)

the statistical characteristics of individual packets in the traffic flows. Bissias et al. use packet sizes and inter-arrival timings as classification features. Their method is fragile to the changes in the network environment, as the inter-arrival timing is highly dependent on the specific routing path and varies from time to time. To address this problem, Liberatore et al. only use packet sizes and counts in classification. They implement both Jaccard coefficient and Naïve Bayes classifier, and show the efficacy of the attack in practice. Using similar scheme, Herrmann et al. further improve the classification accuracy using Multinomial Naïve Bayes classifier.

7 Conclusion

We show that traffic analysis attacks can be carried out remotely, without access to the analyzed traffic, thus greatly increasing the attack surface and lowering the barrier to entry for conducting the attack. We identify a queuing side channel that can be used to infer the queue size of a given link with good accuracy and thus monitor traffic patterns. We show how this channel can be used to carry out a remote attack to detect a remote user's browsing patterns. This highlights the importance of traffic analysis attacks in today's connected Internet.

References

1. Akella, A., Seshan, S., Shaikh, A.: An empirical evaluation of wide-area Internet bottlenecks. In: Crovella, M. (ed.) 3rd ACM SIGCOMM Conference on Internet Measurement, pp. 101–114. ACM, New York (2003), http://dl.acm.org/citation.cfm?id=948205.948219
2. Bissias, G.D., Liberatore, M., Jensen, D., Levine, B.N.: Privacy Vulnerabilities in Encrypted HTTP Streams. In: Danezis, G., Martin, D. (eds.) PET 2005. LNCS, vol. 3856, pp. 1–11. Springer, Heidelberg (2006)
3. Chakravarty, S., Stavrou, A., Keromytis, A.D.: Identifying proxy nodes in a Tor anonymization circuit. In: Dipanda, A., Chbeir, R., Yetongnon, K. (eds.) IEEE International Conference on Signal Image Technology and Internet Based Systems, pp. 633–639. IEEE Computer Society, Los Alamitos (2008)
4. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In: Evans, D., Vigna, G. (eds.) IEEE Symposium on Security and Privacy, pp. 191–206. IEEE Computer Society (May 2010), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5504714
5. Cheng, H., Avnur, R.: Traffic Analysis of SSL Encrypted Web Browsing (1998), http://www.cs.berkeley.edu/ daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps
6. Coull, S.E., Collins, M.P., Wright, C.V., Monrose, F., Reiter, M.K.: On web browsing privacy in anonymized netflows. In: Provos, N. (ed.) 16th USENIX Security Symposium. USENIX Association, Berkeley (2007), http://www.usenix.org/events/sec07/tech/coull.html
7. Danezis, G., Serjantov, A.: Statistical Disclosure or Intersection Attacks on Anonymity Systems. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 293–308. Springer, Heidelberg (2004), http://www.springerlink.com/index/TQLJB3HYBK4RUBLA.pdf

8. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Blaze, M. (ed.) USENIX Security Symposium, pp. 303–320. USENIX Association, San Diego (2004), http://portal.acm.org/citation.cfm?id=1251396
9. Evans, N.S., Dingledine, R., Grothoff, C.: A practical congestion attack on Tor using long paths. In: Monrose, F. (ed.) 18th USENIX Security Symposium, pp. 33–50. USENIX Association (August 2009), http://www.usenix.org/events/sec09/tech/full_papers/evans.pdf
10. Franklin, J., Paxson, V., Perrig, A., Savage, S.: An inquiry into the nature and causes of the wealth of Internet miscreants. In: De Capitani di Vemarcati, S., Syverson, P. (eds.) 14th ACM Conference on Computer and Communications Security, pp. 375–388. ACM, New York (2007), http://dl.acm.org/citation.cfm?id=1315245.1315292
11. Herrmann, D., Wendolsky, R., Federrath, H.: Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-Bayes classifier. In: ACM Workshop on Cloud Computing Security, pp. 31–42. ACM, Chicago (2009), http://portal.acm.org/citation.cfm?id=1655013
12. Hintz, A.: Fingerprinting Websites Using Traffic Analysis. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 171–178. Springer, Heidelberg (2003), http://www.springerlink.com/index/C4QWE6D608P2CJYV.pdf
13. Hopper, N., Vasserman, E.Y., Chan-Tin, E.: How much anonymity does network latency leak? In: De Capitani di Vimercati, S., Syverson, P. (eds.) 14th ACM Conference on Computer and Communications Security, pp. 82–91. ACM, New York (2007), http://dl.acm.org/citation.cfm?id=1315245.1315257
14. Hopper, N., Vasserman, E., Chan-Tin, E.: How much anonymity does network latency leak? ACM Transactions on Information and System Security 13(2) (2010), http://portal.acm.org/citation.cfm?id=1698753
15. Kadloor, S., Gong, X., Kiyavash, N., Tezcan, T., Borisov, N.: Low-Cost Side Channel Remote Traffic Analysis Attack in Packet Networks. In: Xiao, C., Olivier, J.C. (eds.) 2010 IEEE International Conference on Communications. IEEE (May 2010), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5501972
16. Lakshminarayanan, K., Padmanabhan, V.N.: Some findings on the network performance of broadband hosts. In: Crovella, M. (ed.) Proceedings of the 2003 ACM SIGCOMM Conference on Internet Measurement, IMC 2003, pp. 101–114. ACM Press, New York (2003), http://portal.acm.org/citation.cfm?doid=948205.948212
17. Liberatore, M., Levine, B.N.: Inferring the source of encrypted HTTP connections. In: Wright, R., De Capitani di Vemarcati, S. (eds.) 13th ACM Conference on Computer and Communications Security, pp. 255–263. ACM, New York (2006), http://portal.acm.org/citation.cfm?id=1180437
18. Lyon, G.F.: Nmap Network Scanning. Nmap Project (1999)
19. Murdoch, S., Danezis, G.: Low-Cost Traffic Analysis of Tor. In: Paxson, V., Waidner, M. (eds.) 2005 IEEE Symposium on Security and Privacy, pp. 183–195. IEEE Computer Society, Berkeley (2005), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1425067
20. Prasad, R., Davrolis, C., Murray, M., Claffy, K.: Bandwidth estimation: metrics, measurement techniques, and tools. IEEE Network 17(6), 27–35 (2003), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1248658

21. Rennhard, M., Plattner, B.: Introducing MorphMix: peer-to-peer based anonymous Internet usage with collusion detection. In: Samarati, P. (ed.) ACM Workshop on Privacy in Electronic Society, pp. 91–102. ACM Press, New York (2002), http://portal.acm.org/citation.cfm?id=644537
22. Ribeiro, V., Riedi, R., Baraniuk, R., Navratil, J., Cottrell, L.: pathchirp: Efficient available bandwidth estimation for network paths. In: Passive and Active Measurement Workshop, vol. 4. Citeseer (March 2003)
23. Rizzo, L.: Dummynet: a simple approach to the evaluation of network protocols. ACM SIGCOMM Computer Communication Review 27(1), 31–41 (1997), http://portal.acm.org/citation.cfm?doid=251007.251012
24. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Transactions on Acoustics, Speech, and Signal Processing 26(1), 43–49 (1978), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1163055
25. Saponas, T.S., Lester, J., Hartung, C., Agarwal, S., Kohno, T.: Devices that tell on you: Privacy trends in consumer ubiquitous computing. In: Provos, N. (ed.) 16th USENIX Security Symposium, pp. 55–70. USENIX Association (2007), http://portal.acm.org/citation.cfm?id=1362908
26. Shreedhar, M., Varghese, G.: Efficient fair queuing using deficit round-robin. IEEE/ACM Transactions on Networking 4(3), 375–385 (1996), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=502236
27. Song, D.X., Wagner, D., Tian, X.: Timing analysis of keystrokes and SSH timing attacks. In: Wallach, D.S. (ed.) 10th USENIX Security Symposium. USENIX Association (August 2001), http://www.usenix.org/events/sec01/song.html
28. Strauss, J., Katabi, D., Kaashoek, F.: A measurement study of available bandwidth estimation tools. In: Crovella, M. (ed.) 3rd ACM SIGCOMM Conference on Internet Measurement, pp. 39–44. ACM, New York (2003), http://portal.acm.org/citation.cfm?id=948211
29. Sun, Q., Simon, D.R., Wang, Y.M., Russell, W., Padmanabhan, V.N., Qiu, L.: Statistical identification of encrypted Web browsing traffic. In: Abadi, M., Bellovin, S.M. (eds.) IEEE Symposium on Security and Privacy, pp. 19–30. IEEE Computer Society (May 2002), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1004359
30. Wagner, D., Schneier, B.: Analysis of the SSL 3.0 Protocol. In: Tygar, D. (ed.) USENIX Workshop on Electronic Commerce. USENIX Association (November 1996), http://www.usenix.org/publications/library/proceedings/ec96/wagner.html
31. White, A.M., Matthews, A.R., Snow, K.Z., Monrose, F.: Phonotactic reconstruction of encrypted VoIP conversations: Hookt on Foniks. In: Vigna, G., Jha, S. (eds.) IEEE Symposium on Security and Privacy, pp. 3–18. IEEE Computer Society (May 2011), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5958018
32. Wright, C.V., Ballard, L., Coull, S.E., Monrose, F., Masson, G.M.: Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations. In: IEEE Symposium on Security and Privacy, pp. 35–49. IEEE Computer Society, Washington, DC (2008), http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4531143

33. Wright, C.V., Ballard, L., Coull, S.E., Monrose, F., Masson, G.M.: Uncovering Spoken Phrases in Encrypted Voice over IP Conversations. ACM Transactions on Information and System Security 13(4), 1–30 (2010), http://doi.acm.org/10.1145/1880022.1880029
34. Zhang, K., Wang, X.: Peeping Tom in the neighborhood: Keystroke eavesdropping on multi-user systems. In: Monrose, F. (ed.) 18th USENIX Security Symposium USENIX Security. USENIX Association (August 2009), http://www.usenix.org/events/sec09/tech/full_papers/zhang.pdf
35. Zhu, Y., Bettati, R.: Unmixing Mix Traffic. In: Danezis, G., Martin, D. (eds.) PET 2005. LNCS, vol. 3856, pp. 110–127. Springer, Heidelberg (2006)

k-Indistinguishable Traffic Padding in Web Applications

Wen Ming Liu[1], Lingyu Wang[1], Kui Ren[2], Pengsu Cheng[1], and Mourad Debbabi[1]

[1] Concordia Institute for Information Systems Engineering, Concordia University
[2] Department of Electrical and Computer Engineering, Illinois Institute of Technology

Abstract. While web-based applications are becoming increasingly ubiquitous, they also present new security and privacy challenges. In particular, recent research revealed that many high profile Web applications might cause private user information to leak from encrypted traffic due to side-channel attacks exploiting packet sizes and timing. Moreover, existing solutions, such as random padding and packet-size rounding, are shown to incur prohibitive cost while still not ensuring sufficient privacy protection. In this paper, we propose a novel k-indistinguishable traffic padding technique to achieve the optimal tradeoff between privacy protection and communication and computational cost. Specifically, we first present a formal model of the privacy-preserving traffic padding (PPTP). We then formulate PPTP problems under different application scenarios, analyze their complexity, and design efficient heuristic algorithms. Finally, we confirm the effectiveness and efficiency of our algorithms by comparing them to existing solutions through experiments using real-world Web applications.

1 Introduction

Web-based applications are gaining popularity. By providing software services through Web browsers, such applications demand less client-side resources and are easier to deliver and maintain than their desktop counterparts. However, they also present new security and privacy challenges partly because the untrusted Internet now becomes an integral part of the application for carrying the continuous interaction between users and service providers. Recent study showed that the encrypted traffic of many popular Web applications may actually disclose highly sensitive data, and consequently lead to serious breaches of user privacy [9]. By analyzing packets' sizes and timing, an eavesdropper can potentially identify an application's internal state transitions as well as users' inputs. Moreover, such side-channel attacks are shown to be pervasive and fundamental to Web applications due to their intrinsic characteristics , such as low entropy inputs, rich and diverse resource objects, and stateful communications.

For example, Table 1 shows the size and direction of packets observed between users and a popular real-world search engine. Observe that due to the auto-suggestion feature, with each keystroke, the browser sends a b-byte packet to the server; the server then replies with two packets of 54 bytes and s bytes, respectively; finally, the browser sends a 60-byte packet to the server. In addition, in the same input string, each subsequent keystroke increases the b value by one byte, and the s value depends not only on the current keystroke but also on all the previous ones. Clearly, an eavesdropper can pinpoint packets corresponding to an input string from observed traffic by the packets with

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 79–99, 2012.

Table 1. User Inputs and Corresponding Packet Sizes

User Input	Observed Directional Packet Sizes
a	$b_1 \rightarrow$, $\leftarrow 54$, $\leftarrow 509$, $60 \rightarrow$
00	$b_2 \rightarrow$, $\leftarrow 54$, $\leftarrow 505$, $60 \rightarrow$,
	$b_2 + 1 \rightarrow$, $\leftarrow 54$, $\leftarrow 507$, $60 \rightarrow$
	(b bytes) (s bytes)

fixed pattern in size(first, second, and last), even though the traffic has been encrypted. In this paper, we assume such a worst case scenario in which an eavesdropper can identify traffic related to a Web application (such as using de-anonymizing techniques [27]) and locate packets for user inputs using the above technique.

Moreover, the size of the third packet(s) will provide a good indicator of the input itself. Specifically, the left tabular of Table 2 shows the s value for each character entered as the first keystroke of an input string. We can see that six characters can be uniquely identified with this s value. The right tabular shows the s value for a character entered as the second keystroke. In this case, the s value for each character in the right tabular is different from that in the left, since the packet size now depends on both the current keystroke and the preceding one. Clearly, every input string can be uniquely identified by combining observations about the two consecutive keystrokes shown in both tables (for simplicity, we are only considering four characters here, whereas in reality it may take more than two keystrokes to uniquely identify an input string).

Table 2. s Value for Each Char Entered as the First or Second Keystroke (Left or Right Tabular)

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
509	504	502	516	499	504	502	509	492
j	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**
517	499	501	503	488	509	525	494	498
s	**t**	**u**	**v**	**w**	**x**	**y**	**z**	
488	494	503	522	516	491	502	501	

	Second keystroke			
First keystroke	**a**	**b**	**c**	**d**
a	487	493	501	497
b	516	488	482	481
c	501	488	473	477
d	543	478	509	499

A natural solution for preventing such a side channel attack is to pad packets such that each packet size will no longer map to a unique input. However, such a solution does not come free, since padding packets will result in additional overhead. In fact, it has been shown that a straightforward solution, such as random padding and rounding, may incur a prohibitive overhead (e.g. 21074% for a well-known online tax system [9]). Moreover, such an application-agnostic approach typically aim to maximize, but cannot guarantee, the amount of privacy protection.

In Table 3, we consider a different way for padding the packets. The first and last columns respectively show the s value and corresponding input (the second keystroke). The middle two columns give two options for padding packets (although not shown here, there certainly exist many other options). Specifically, each option first divides the six characters into three (or two) *padding groups*, as illustrated by the (absence of) horizontal lines. Packets within the same padding group are then padded in such a way that their corresponding s values are all identical to the maximum value. Thus the

Table 3. Mapping PPTP to PPDP

s **Value**	**Padding**		(1^{st} **Keystroke**) 2^{nd} **Keystroke**
	Option 1	**Option 2**	
473	477	478	$(c)c$
477	477	478	$(c)d$
478	499	478	$(d)b$
499	499	509	$(d)d$
501	509	509	$(c)a$
509	509	509	$(d)c$
Quasi-ID	**Generalization**		**Sensitive Value**

characters inside each padding group will no longer be distinguishable from each other based on their s values. The objective now is to find a padding option that can provide sufficient privacy protection and meanwhile minimize the padding cost.

Interestingly, this *privacy-preserving traffic padding (PPTP)* problem can be naturally interpreted as another well studied problem, *privacy-preserving data publishing (PPDP)* [13]. To revisit Table 3, if we regard the s value as a *quasi-identifier* (such as DoB), the input as a *sensitive value* (such as medical condition), and the padding options as different ways for generalizing the DoB into *anonymized groups* (for example, by removing the day from a DoB), then we immediately have a classic PPDP problem, that is, publishing DoBs and medical conditions while preventing adversaries from linking any published medical condition to a person through his/her DoB.

The similarity between the two problems implies we may borrow many existing efforts in the PPDP domain to address the PPTP issue. On the other hand, there also exist significant differences between them. For example, in Table 3, the second option will typically be considered as worse (than the first) in PPDP since it results in larger anonymized groups, whereas it is actually better in terms of padding cost (totally 24 bytes, in contrast to 33 by the first option). As another example, we will show later that the effect of combining two keystrokes will be equivalent to releasing multiple inter-dependent tables, which actually leads to a novel PPDP problem.

In this paper, we first briefly review the formal model of the PPTP issue based on the mapping to PPDP which introduced in our short version [18]. We then formulate several PPTP problems under different assumptions, and discuss the complexity. We show that minimizing padding cost under a given privacy requirement is generally intractable. Next, we design several heuristic algorithms for solving the PPTP problems in polynomial time with acceptable padding cost. Finally, we evaluate the effectiveness and efficiency of our algorithms by comparing them to existing solutions through experiments with real-world Web applications.

The contribution of this paper is threefold. First, the identified similarity between PPTP and PPDP establishes a bridge between the two research communities, which will not only allow for reusing many existing models and methods in the well investigated PPDP domain, but serve to attract more interest to the important PPTP issue. Second, to the best of our knowledge, our PPTP model is among the first efforts on formally addressing this issue (in contrast to our work, the formal model given by Chen *et al.* [9] lacks a clear definition of privacy requirements and only considers two application-agnostic

padding methods). Third, the proposed padding algorithms can lead to practical solutions for real world Web applications, as evidenced by our experiments.

The rest of the paper is organized as follows. Section 2 defines our PPTP model. Section 3 formulates PPTP problems and analyzes the complexity. Section 4 devises heuristic algorithms for the formulated problems. Section 5 experimentally evaluates the performance of our algorithms. Section 6 discusses the extensions and implementation of our solution. Section 7 reviews related work and Section 8 concludes the paper.

2 The Model

To be self contained, we briefly repeat here the PPTP model introduced in our short version [18], and shall delay the discussion about extending it to encompass l-diversity in Section 6. Table 4 lists main notations that will be used throughout the paper.

Table 4. The Notation Table

a, $\vec{a}$, A_i or A	Action, action-sequence, action-set
s, v, $\vec{v}$, V_i or V	Flow, flow-vector, vector-sequence, vector-set
$\vec{a}[i]$, $\vec{v}[i]$	The i^{th} element in $\vec{a}$ and $\vec{v}$
VA_i or VA	Vector-action set
$pre(a, i)$	i-Prefix
$dom(P)$	Dominant-vector
$vdis(v_1, v_2)$	Vector-distance

2.1 The Basic Model

We model the PPTP issue from two perspectives, the *interaction* between users and servers, and the *observation* made by eavesdroppers. First, Definition 1 formalizes the interaction. Our discussions about Table 2 demonstrated how one keystroke may affect another in terms of observations (packet sizes), and how an eavesdropper may combine such multiple observations for a refined inference. Such related user *actions* are modeled as an *action-sequence* in Definition 1. The concept of *action-set* models a collection of actions whose corresponding observations may be padded together. Actions inside an action-sequence are separated into different action-sets since their relationship is known from traffic patterns and thus padding them together will not work (preventing such inferences about the application's state transitions comprises a future direction).

Definition 1 (Interaction). *Given a Web application, we define*

- *an* action a *as an atomic user input that triggers traffic, such as a keystroke or a mouse click.*
- *an* action-sequence $\vec{a}$ *as a sequence of actions with known relationships, such as consecutive keystrokes entered into real-time search engine or a series of mouse clicks on hierarchical menu items. We use* $\vec{a}[i]$ *to denote the* i^{th} *action in* $\vec{a}$.
- *an* action-set A_i *as the collection of all the* i^{th} *actions in a set of action-sequences. We will simply use* A *if all action-sequences are of length one.*

Example 1. Assume "a" and "0**0**" in Table 1 to be the only possible inputs, there are two action-sequences a and 00, and two action-sets $A_1 = \{a, 0\}$ and $A_2 = \{\mathbf{0}\}$. ⊡

Definition 2 models concepts related to the observation made by an eavesdropper. Note that a *flow-vector* is intended to only model those packets that may contribute to identify an action (such as the s value in Table 1). Further, a vector-set is defined as a multiset, since it may contain duplicates (that is, packets may share the same size).

Definition 2 (Observations). *Given a web application, we define*

- *a* flow-vector v *as a sequence of* flows *where each flow* s *is an integer (a directional packet size). An action corresponds to a flow-vector based on packets it triggers.*
- *a* vector-sequence $\vec{v}$ *as a sequence of flow-vectors corresponding to an equal-length action-sequence* $\vec{a}$*, with each* $\vec{v}[i]$ *corresponding to* $\vec{a}[i]$ *(*$1 \leq i \leq \mid \vec{v} \mid$*).*
- *a* vector-set V_i *(or simply* V*) as the collection of all the* i^{th} *flow-vectors in a set of vector-sequences, which corresponds to an action-set in the straightforward way.*

Example 2. Following Example 1, we have three flow-vectors, $v_1 = 509$, $v_2 = 505$, and $v_3 = 507$ (note that we only model those packets whose sizes can help to identify an action), corresponding to actions a, 0 (as first keystroke), and 0 (as second keystroke), respectively. We have two vector-sequences, v_1 and v_2v_3, corresponding to action-sequences a and 00, respectively. We also have two vector-sets $V_1 = \{509, 505\}$ and $V_2 = \{507\}$ corresponding to the two action-sets A_1 and A_2 in Example 1. ⊡

Finally, Definition 3 models the joint information about interaction and observation, which is the collection of the pairs of the action and its corresponding flow-vector.

Definition 3 (Vector-Action Set). *Given an action-set* A_i *and its corresponding vector-set* V_i*, a* vector-action set VA_i *is the set* $\{(v, a) : v \in V_i \wedge a \in A_i\}$.

Example 3. Following above examples, given the action-set A_1 and vector-set V_1, then the vector-action set is $VA_1 = \{(509, a), (505, 0)\}$. Similarly, $VA_2 = \{(507, \mathbf{0})\}$. ⊡

2.2 Privacy and Cost Model

For simplicity, we first consider a simplified case where every action-sequence and flow-vector are of length one, namely, the Single-Vector Single-Dimension (SVSD) case. In this case, we can map a given vector-action set $VA = \{(v, a) : v \in V \wedge a \in A\}$ to a table $T(v, a)$ with two attributes, the flow-vector v (equivalent to a flow s here) as quasi-identifier and the action a as sensitive attribute. Note that we will interchangeably refer to a vector-action set and its tabular representation from now on.

Inspired by k-anonymity [24] in PPDP domain, Definition 4 quantifies the amount of privacy protection under a given vector-action set. This model follows the widely adopted approach of assuming a fixed privacy requirement while minimizing the cost.

Definition 4 (k-Indistinguishability). *Given a vector-action set* VA*, we define*

- *a* padding group *as any* $S \subseteq VA$ *satisfying that all the pairs in* S *have identical flow-vectors and no* $S' \supset S$ *can satisfy this property, and*
- *we say* VA *satisfies* k-indistinguishability *(*k *is an integer) or* VA *is* k-indistinguishable *if the cardinality of every padding group is no less than* k*.*

Discussion. One may argue that, in contrast to encryption, k-*indistinguishability* may not provide strong enough protection. However, as mentioned before, we are considering cases where encryption is already broken by side-channel attacks, so the strong confidentiality provided by encryption is already not an option. Second, in theory k could always be set to be sufficient large to provide enough confidentiality, although we believe a reasonably large k would usually satisfy users' privacy requirements for most practical applications. Finally, since most web applications are publicly accessible and consequently an eavesdropper can unavoidably learn about possible inputs, we believe focusing on protecting sensitive user input (by hiding it among other possible inputs) yields higher practical feasibility and significance than on perfect confidentiality (attempting to hide everything).

Furthermore, such mapped PPDP problems actually possess a unique characteristic. That is, the sensitive values (actions) are always unique. Thus, by satisfying k-indistinguishability, the vector-action set also satisfies l-*diversity* ($l = k$) in its simplest form [20]. We will also apply more general forms of l-diversity to address cases where not all actions should be treated equally in padding, as sketched in Section 6. Furthermore, a probabilistic approach based on *differential privacy* [12] is another possible extension to enhance our model such that the padding result will be immune to eavesdroppers' prior knowledge. Nonetheless, this simple model is sufficient to demonstrate the usefulness of mapping PPTP to PPDP.

In addition to privacy requirement, we also need a quantitative measure for the cost of padding and processing. Across the whole vector-set, Definition 5 counts the number of additional bytes after padded, while Definition 6 counts the number of flows that are involved in padding. We focus on these simple models in this paper while there certainly exist other ways for modeling such costs.

Definition 5 (Distance and Padding Cost). *Given a vector-set V, we define*

- *the* vector-distance *between two equal-length flow-vectors v_1 and v_2 as: $vdis(v_1, v_2) = \sum_{i=1}^{|v_1|}(|s_{1i} - s_{2i}|)$ where s_{1i} and s_{2i} are the i^{th} flow in v_1 and v_2, respectively.*
- *the* padding cost *of V as: $cost = \sum_{i=1}^{|V|}(vdis(v_i, v_i'))$ where v_i and v_i' denote a flow-vector in V and its counterpart after padding, respectively.*

Definition 6 (Processing Cost). *Given a vector-set V, we define the processing cost of V as the number of flows in V which corresponding packets should be padded.*

2.3 The SVMD and MVMD Cases

In the previous section, we focused on the simplified SVSD case to facilitate a focused discussion on the privacy and cost model. We now look at the more realistic cases.

First, we consider the Single-Vector Multi-Dimension (SVMD) case where each flow-vector may include more than one flows whereas each action-sequence is still composed of a single action. In this case, the vector-action set needs to be mapped to a table $T(s_1, \ldots, s_{|v|}, a)$ with multiple quasi-identifier attributes (each flow corresponds to an attribute). Thus, based on Definition 4, flow-vectors can form a padding group

only if they are identical with respect to every flow inside the vectors. Another subtlety is that the model of vector-action set requires all the flow-vectors to have the same number of flows, which is not always possible in practice. One solution is to insert dummy packets of size zero which will then be handled as usual in the process of padding.

Next, we consider the Multi-Vector Multi-Dimension (MVMD) case in which each action-sequence consists of more than one actions and each flow-vector includes multiple flows. Definition 7 expresses the relationship between actions in an action-sequence.

Definition 7 (i-prefix, adjacent-prefix). *We define*

- *the* i-prefix *of an action-sequence* $\vec{a} = (a_1, a_2, \ldots, a_t)$ *(*$i \in [1, t]$*), denoted as* $pre(\vec{a}, i)$*, as the sequence* $(a_1, a_2, \ldots, a_i)$*, and we say* a_{i-1} *is the* adjacent-prefix *(or simply prefix) of* a_i*.*
- *similarly, we define the* i-prefix *of vector-sequence* $\vec{v}$*, and the* adjacent-prefix *of* v_i*.*

In the MVMD case, due to the prefix relationship, the flow-vector for an action may provide additional information about flow-vectors that correspond to the previous actions in the same action-sequence. Such knowledge may enable the eavesdropper to refine his guesses about an action. Such a scenario is illustrated in Figure 1. Also, we slightly change the definition of a vector-action set to accommodate the added prefix action information, as shown in Definition 8. We will delay the discussion about how a padding algorithm may satisfy k-indistinguishability in this case to the next section.

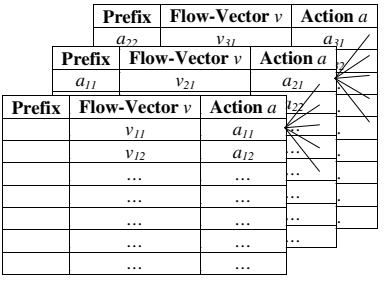


Fig. 1. The Vector-Action Set in MVMD Case

Definition 8 (Vector-Action Set (MVMD Case)). *Given* n *action-sets* $\{A_i : 1 \leq i \leq n\}$ *and its corresponding vector-sets* $\{V_i : 1 \leq i \leq n\}$*, the* vector-action set VA *is the collection of sets* $\{\{(v, a) : v \in V_i \wedge a \in A_i\} : 1 \leq i \leq n\}$.

3 PPTP Problems

The formal model introduced in the previous section enables us to formulate a series of PPTP problems and study their complexity. We first discuss the choice of our ceiling padding approach among other possibilities in Section 3.1, and then address the SVSD and SVMD cases in Section 3.2 and the MVMD case in Section 3.3.

3.1 Padding Method

In choosing a padding method, we need to address two aspects, privacy protection by satisfying the k-indistinguishability property, and minimizing padding cost. As previously mentioned, an application-agnostic approach, such as packet-size rounding and random padding, will usually incur high padding cost while not necessarily guaranteeing sufficient privacy protection [9]. We now revisit this argument by showing that a larger rounding size does not necessarily lead to more privacy. With our model, more privacy can now be clearly defined as satisfying k-indistinguishability for a larger k. Consider rounding the flows shown in the left tabular of Table 2 to a multiple of 128 (for example, 509 to $4 \times 128 = 512$). It can be shown that such rounding can achieve 5-indistinguishability (detailed calculations will be omitted due to space limitations). However, increasing the rounding size to 512 can still only satisfy 5-indistinguishability, whereas further increasing it to 520 will actually only satisfy 2-indistinguishability.

On the other hand, as demonstrated in Section 1, we can now apply the PPDP technique of generalization to addressing the PPTP problem. A generalization technique will partition the vector-action set into padding groups, and then break the linkage among actions in the same group. One unique aspect in applying generalization to PPTP is that padding can only increase each packet size but cannot decrease it, or replace it with a range of values like in normal generalization. The above considerations lead to a new padding method given in Definition 9. Basically, after partitioning a vector-action set into padding groups, we pad each flow in a padding group to be identical to the maximum size of that flow in the group.

Definition 9 (Dominance and Ceiling Padding). *Given a vector-set V, we define*

- *the* dominant-vector $dom(V)$ *as the flow-vector in which each flow is equal to the maximum of the corresponding flow among all the flow-vectors in V.*
- *a* ceiling-padded *group in V as a padding group in which every flow-vector is padded to the dominant-vector. We also say V is ceiling-padded if all the groups are ceiling-padded.*

We will focus on the ceiling padding method in the rest of the paper. When no ambiguity is possible, we will not distinguish between vector-set, vector-action set, flow-vector, and vector-sequence.

3.2 The SVSD and SVMD Cases

In the SVSD case, there is only a single flow in each flow-vector of the vector-set. Therefore, we only need to modify the vector-set by increasing the value of some flows to form padding groups. The padding problem can be formally defined as follows.

Problem 1 (SVSD Problem). Given a vector-action set VA and the corresponding vector-set V and action set A, the privacy property $k \leq |V|$, find a partition P^{VA} on VA such that the corresponding partition on V, denoted as $P^V = \{P_1, P_2, \ldots, P_m\}$, satisfies

- $\forall (i \in [1, m]), |P_i| \geq k$;
- The padding cost $\sum_{i=1}^{m} (dom(P_i) \times |P_i|)$ is minimal. ⊡

In the SVMD case, there are more than one flows in each flow-vector of the vector-set. The padding problem can be defined as follows:

Problem 2 (SVMD problem). Given a vector-action set VA and the corresponding vector-set V (in which each flow-vector includes n_p flows) and action set A, the privacy property $k \leq \mid V \mid$, find a partition P^{VA} on VA such that the corresponding partition on V, denoted as $P^V = \{P_1, P_2, \ldots, P_m\}$, satisfies

- $\forall (i \in [1, m]), |P_i| \geq k$;
- The cost $\sum_{i=1}^{m}(\sum_{j=1}^{n_p}((dom(P_i))[j]) \times |P_i|)$ is minimal. ⊡

Theorem 1 shows that the above PPTP problem is intractable. The proof is omitted due to space limitations (basically, we prove the result through a reduction to the problem of *Edge Partition Into Triangles* (EPIT) [14]). Theorem 1 indicates that Problem 2 is NP-hard even when there are only two different flow values (that is, the sizes of packets in the traffic) in the vector-set.

Theorem 1. *Problem 2 is NP-complete when $k = 3$ and the flow-vectors are from any binary alphabet $\sum$.*

Note that, at first glance, the SVMD problem may resemble the problem of *k-means clustering* [15]. However, algorithms for *k-means clustering* cannot be directly applied to our problem due to following differences between these two problems. First, *k-means clustering* needs to partition a multiset into k groups, whereas in our problem, the minimal size of each group must be at least k. Second, *k-means clustering* is to minimize the within-cluster sum of squares, while our problem is to minimize the total distance between each of the flow-vectors and the dominant-vector.

3.3 MVMD Problem

As mentioned in Section 2.3, by correlating flow-vectors in the vector-sequence, an eavesdropper may refine his guesses of the actual action-sequence. We first discuss the challenges of traffic padding in such cases by a toy example of auto-suggestion feature.

Example 4. In Table 5, the second column shows each flow corresponding to c_1, c_2, c_3, c_4 when entered as the first keystroke, respectively. Similarly, the 16 cells c_{ij} in row: $(i \in [2-5])$ and column: $(j \in [3-6])$ show the flow corresponding to the second keystroke when c_{i-1} is the first keystroke and c_{j-2} the second keystroke.

Suppose an eavesdropper has observed the flow for the second keystroke. In order to preserve 2-indistinguishability with minimal padding overhead, we will partition the 16 cells into eight groups $P_i = \{c_{jk} : \frac{c_{jk}}{10} = i\}$, such that the size of each group is not less than 2. For example, the queried strings c_1c_1 and c_3c_2 are in one group and c_1c_1 should be padded to 15. When the eavesdropper observes that the flow for the second keystroke is 15, she cannot determine whether the queried string is c_1c_1 or c_3c_2.

However, suppose that the eavesdropper also observes the flow corresponding to the first keystroke, they can determine that the first keystroke is either c_1 or c_3 when the flow is 5 or 15, respectively. Consequentially, the eavesdropper can eventually infer the queried string by combining these observations. ⊡

Table 5. Padding in the MVMD Case

		c_1	c_2	c_3	c_4
c_1	5	10	20	80	50
c_2	10	40	70	30	60
c_3	15	65	15	45	75
c_4	20	35	55	85	25

One seemingly valid solution is padding the flow-vector for each keystroke so that 2-indistinguishability is satisfied separately for each keystroke. Unfortunately, this will fail to satisfy 2-indistinguishability. To pad traffic for the first keystroke, the optimal solution is to partition $\{5, 10, 15, 20\}$ into two padding groups, $\{5, 10\}$ and $\{15, 20\}$. However, when the eavesdropper observes the flow corresponding to the first keystroke, he/she can still determine it must be either c_1 or c_3 when the size is 10 or 20, respectively, because only when the first keystroke starts with c_1 or c_3 can the flow for second keystroke be padded to 15. Therefore, the eavesdropper will eliminate c_2 and c_4 from possible guesses, which violates 2-indistinguishability.

Another seemingly viable solution is to first collect all vector-sequences for the sequence of keystrokes and then pad them such that the current input string as a whole cannot be distinguished from at least $k-1$ others. Unfortunately, such an approach cannot guarantee the privacy property, either. First, the auto-suggestion feature requires the server to immediately respond to the client upon each single keystroke. Second, when receiving a single keystroke, the server cannot predict what would be the next input and hence cannot decide which padding option is suitable. For example, suppose the flow corresponding to c_1 in c_1c_2 should be padded to 10, while in c_1c_3 to 15. When the server receives c_1, it cannot determine whether to pad c_1 to 10 or to 15.

The above challenges mainly arise due to the approach of padding each vector-set independently. We now propose a different approach. Intuitively, the partitioning of a vector-set corresponding to each action will *respect* the partitioning results of all the previous actions in the same action-sequence. More precisely, the padding of different vector-sets is correlated based on following two conditions.

- Given two t-sized vector-sequences $\vec{v}_1$ and $\vec{v}_2$, any prefix $pre(\vec{v}_1, i)$ and $pre(\vec{v}_2, i)$ $(i \in [2, t])$, can be padded together only if $\forall(j < i)$, $pre(\vec{v}_1, j)$ and $pre(\vec{v}_2, j)$ are padded together.
- For any two t-sized action-sequences $\vec{a}_1$ and $\vec{a}_2$ and corresponding vector-sequences $\vec{v}_1$ and $\vec{v}_2$, if $pre(\vec{a}_1, i) = pre(\vec{a}_2, i)(i \in [1, t])$, then $pre(\vec{v}_1, i)$ and $pre(\vec{v}_2, i)$ must be padded together.

Once a partition satisfies aforementioned conditions, no matter how an eavesdropper analyzes traffic information, either for an action alone or combining multiple observations of previous actions, the mental image about the actual action-sequence (or any of its subsets) remains the same (detailed proof is omitted due to space limitations). Due to the similarity between the conditions and a related concept in graph theory, we call a partition satisfying such conditions the *oriented-forest partition*.

Problem 3 (MVMD problem). Given a vector-action set $VA = (VA_1, VA_2, \ldots, VA_t)$ where $VA_i = (V_i, A_i)$ $(i \in [1, t])$, the privacy property $k \leq |V_t|$, find the partition

P^{VA_i} on VA_i such that the corresponding partition $P^{V_i} = \{P_1^i, P_2^i, \ldots, P_{m_i}^i\}$ on V_i satisfies

- $\forall((i \in [1, t-1]) \wedge (j \in [1, m_i]))$ $\begin{cases} |P_j^i| \geq k, & \text{if } (|V_i| \geq k), \\ |P_1^i| = |V_i|, & \text{if } (|V_i| < k); \end{cases}$
- $\forall(j \in [1, m_t]), |P_j^t| \geq k$;
- The sequence of P^{V_i} is an oriented-forest partition;
- The total padding cost of P^{V_i} $(i \in [1, t])$ is minimal. ⊡

Obviously, Problem 3 is also NP-complete when $k \geq 3$ since Problem 2 is special case of Problem 3.

4 The Algorithms

In this section, we design three algorithms for partitioning the vector-action set into padding groups to satisfy a given privacy requirement. Our intention is not to design an exhaustive list of solutions but rather to demonstrate the existence of abundant possibilities in approaching this PPTP issue.

4.1 The svsdSimple Algorithm

The main intention of presenting the svsdSimple algorithm is to show that, when applying k-indistinguishability to PPTP problems, an algorithm may sometimes be devised in a very straightforward way, and yet achieve a dramatic reduction in costs when compared to existing approaches (as shown in the next section). The svsdSimple algorithm shown in Table 6 basically attempts to minimize the cardinality of padding groups in the SVSD case. Note that when the cardinality of vector-action set is less than the privacy property k, there is no solution to satisfy the privacy property. In such cases, our algorithms will simply exit, which will not be explicitly shown in each algorithm hereafter.

Table 6. The svsdSimple Algorithm for SVSD-Problem

Algorithm svsdSimple
Input: a vector-action set VA, the privacy property k;
Output: the partition P^{VA} of VA;
Method:
1. **Let** $P^{VA} = \phi$;
2. **Let** S^{VA} be the sequence of VA in a non-decreasing order of V;
3. **Let** $N = \frac{|S^{VA}|}{k}$;
4. **For** $i = 0$ to $N - 2$
5. **Let** $P_i = \bigcup_{j=i\times k}^{(i+1)\times k-1}(S^{VA}[j])$;
6. Create partition P_i on P^{VA};
7. Create $P_{N-1} = \bigcup_{j=(N-1)\times k}^{|S^{VA}|-1}(S^{VA}[j])$ on P^{VA};
8. **Return** P^{VA};

More specifically, svsdSimple first sorts each single flow in the flow-vector into a non-decreasing order of the flows, and then selects k pairs of (flow-vector, action) each time in that order to form a padding group. This is repeated until the number of pairs is less than k. The remainder of pairs is simply appended to the last padding group.

The computational complexity is $O(nlogn)$ where $n = |VA|$, since step 2 costs $O(nlogn)$ time and each (flow-vector, action) pair is considered once for the remaining steps.

4.2 The svmdGreedy Algorithm

The svmdGreedy algorithm, which aims at both SVSD and SVMD problems, is shown in Table 7. Roughly speaking, the svmdGreedy recursively divides the padding group P_i in P^{VA}, where $|P_i| \geq 2 \times k$, into two padding groups P_{i1} and P_{i2} until the cardinality of any padding group in P^{VA} is less than $2 \times k$. When svmdGreedy splits a padding group $P_i(VA_i)$ into two, these resultant padding groups, P_{i1} and P_{i2}, must satisfy that $(P_{i1} \cup P_{i2} = P_i) \wedge (P_{i1} \cap P_{i2} = \phi) \wedge (|P_{i1}| \geq k) \wedge (|P_{i2}| \geq k)$. Obviously, there must exist many solutions of P_{i1} and P_{i2}. svmdGreedy limits the optimizing process insides a subset of possible solutions as follows. For each flow, svmdGreedy first sorts the flow-vectors in non-decreasing order of that flow, then splits P_i into P_{i1} and P_{i2} at position pos in the sorted sequence of flow-vectors where $(pos \in [k, |P_i| - k])$. There are totally $(n_p \times (|P_i| - 2 \times k))$ possible solutions for all flows in the flow-vector, where n_p is the number of flows in flow-vector. SvmdGreedy finally selects the one with minimal padding cost among this set of solutions. Clearly, this algorithm can solve SVSD-problem when n_p is set to be 1.

Table 7. The svmdGreedy Algorithm For SVMD-Problem

Algorithm svmdGreedy
Input: a vector-action set VA, the privacy property k;
Output: the partition P^{VA} of VA;
Method:
1. **If**($|VA| < 2 \times k$)
2. Create in P^{VA} the VA;
3. **Return**;
4. **Let** n_p be the number of flows in flow-vector;
5. **For** $p = 1$ to n_p
6. **Let** S_p^{VA} be the sequence of VA in the non-decreasing order of the p^{th} flow in the vector;
7. **For** $i = k$ to $|S_p^{VA}| - k$
8. **Let** $cost_{p,i}$ as the cost when S_p^V is split at position i;
9. **Let** $cost_p$ be a pair (c, i) where c is the minimal in $(cost_{p,i})$ and i is the position;
10. **Let** $cost$ be a triple (c, p, i) where c is the minimal in c of $cost_p(p \in [1, n_p])$, and p and i are the corresponding p and i;
11. Split $S_{cost.p}^{VA}$ into VA_1 and VA_2 at position $cost.i$;
12. **Return** svmdGreedy(VA_1);
13. **Return** svmdGreedy(VA_2);

The svmdGreedy algorithm has an $O(n_p \times n^2)$ time complexity in the worst case (each time, the algorithm splits P_i into k-size P_{i1} and $(|P_i| - k)$-size P_{i2}), and $O(n_p \times n \times logn)$ in average cases (each time, the algorithm halves P_i), where $n = |VA|$.

4.3 The mvmdGreedy Algorithm

Both svsdSimple and svmdGreedy algorithms tackle cases where each action-sequence consists of a single action (correspondingly, each vector-sequence consists of a single flow-vector). Our intention now in devising the mvmdGreedy is to demonstrate how the two conditions mentioned in Section 3.3 facilitate the algorithm design. In this algorithm, we extend PPDP solutions to a sequence of inter-dependent vector-action sets. The only constraint in partitioning vector-action set VA_i is to ensure all flow-vectors in a padding group should have their prefix in an identical padding group of VA_{i-1}.

The mvmdGreedy algorithm for MVMD-Problem is shown in Table 8. Roughly speaking, mvmdGreedy partitions each vector-action set in the sequence in the given order, each for the flow-vector corresponding to an action in an action-sequence. More specifically, mvmdGreedy applies svmdGreedy to partition the first vector-action set in the sequence. For each remaining vector-action set VA_i, mvmdGreedy first partitions it into $|P^{VA_{i-1}}|$ number of padding groups based on the adjacent-prefix of the flow-vectors, and then applies svmdGreedy to further partition these padding groups.

Similarly, the mvmdGreedy algorithm also has an $O(n_p \times n^2)$ time complexity in the worst case (each time, the algorithm splits VA_i into k-size VA_{i1} and $(|VA_i| - k)$-size VA_{i2}), and $O(n_p \times n \times logn)$ in average cases (each time, the algorithm halves VA), where n is the total number of flow-vectors in those vector-sets.

Table 8. The mvmdGreedy Algorithm For MVMD-Problem

Algorithm mvmdGreedy
Input: a t-size sequence D of vector-action sets, the privacy property k;
Output: the partition P^D of D;
Method:
1. **Let** $D = (VA_1, VA_2, \ldots, VA_t)$;
2. **Let** $P^1 = svmdGreedy(VA_1, k)$;
3. **For** each $(w \in [1, |P^1|])$, assign group $G_w^1 \in P^1$ a unique $gid = w$;
4. **For** $i = 2$ to t
5. Create in P^i $|P^{i-1}|$ number of empty groups $G_w^i (w \in [1, |P^{i-1}|])$;
6. **For** each v_{ia} in VA_i
7. **Let** w be the gid of the group G_w^{i-1} in P^{i-1} that the prefix of v_{ia} in VA_{i-1} belongs to;
8. Insert v_{ia} into G_w^i;
9. **For** each $(w \in [1, |P^{i-1}|])$
10. $P^i = (P^i \setminus G_w^i) \cup svmdGreedy(G_w^i, k)$;
11. **For** each $(w \in [1, |P^i|])$, assign group $G_w^i \in P^i$ a unique $gid = w$;
12. **Return** $P^D = \{P^i : 1 \leq i \leq t\}$;

5 Evaluation

In this section, we evaluate the effectiveness of our solutions and efficiency through experiments with real world Web applications. Section 5.1 first elaborates on the experimental settings. Then, Section 5.2, 5.3, and 5.4 present experimental results of the communication, computation, and processing overhead, respectively.

5.1 Experimental Setting

We collect testing vector-action sets from two real-world web applications, a popular search engine $engine^B$ (where users' searching keyword needs to be protected) and an authoritative drug information system $drug^B$ from a national institute (where users' possible health information need to be protected). Specially, for $engine^B$, we collect flow-vectors with respect to query suggestion widget for all possible combinations of four letters by crafting requests to simulate the normal AJAX connection request. For $drug^B$, we collect the vector-action set for all the drug information by mouse-selecting following the application's three-level tree-hierarchical navigation. Such data can be collected by acting as a normal user of the applications without having to know internal details of the applications. For our experiment, these data are collected using separate programs whose efficiency is not our main concern in this paper.

Information about data cardinality and action levels is shown in Table 9(a), and information about the distribution and distinct number of data sizes is shown in Table 9(b). We observe that the flows of $drug^B$ are more diverse than those of $engine^B$ evidenced by the standard deviations (σ) of the flows. The flows of $drug^B$ are also larger than those of $engine^B$ based on their means (μ). Besides, the flows of $drug^B$ are much more disparate in values than those of $engine^B$. For example, there are only 889 different flow sizes among 456976 flow-vectors in $engine^B$, while there are 1015 different among 4883 vectors in $drug^B$. Later we will show the effect of these different characteristics of flows on the costs.

Table 9. Flow Data Outline of $engine^B$ and $drug^B$

	$engine^B$	$drug^B$
Level Number	4	3
1	26,	1,
2	676,	27,
3	17.6K,	4883
4	457K	-
Vector Number in Total	475254	5091

(a).The Number of Vectors

Level	$engine^B$			$drug^B$		
	μ	σ	#	μ	σ	#
1	936	71	23	-	0	1
2	896	70	229	37833	22102	27
3	768	216	741	23411	9796	1015
4	429	173	889	-	-	-

(b). Distribution of Data Sizes

Note that the size information collected through our programs may have integrally shifted from the original one. However, we argue that such information is still sufficient and reasonable for our experimental evaluation due to following facts. First, the collected data preserve adequate characteristics of the original data with respect to the traffic-size distinction. Second, although the length of HTTP request and response may vary due to different browsers and platforms, the variance is constant for the same setting and can be determined in advance. Also, the size information may vary when adopting compression in the web objects or HTTP body. Our solutions regard such variances as different inputs.

All experiments are conducted on a PC with 2.20GHz Duo CPU and 4GB memory. We evaluate the overhead of computation, communication, and processing using execution time, padding cost ratio, and processing cost ratio, respectively. Specifically, for

each application, we first obtain the total size of all flows ttl for all possible actions before padding, and then compute the padding cost $cost$ as shown in Definition 5 after padding. The padding cost ratio of traffic padding is formulated as $\frac{cost}{ttl}$. We also count the number of flows which need to be padded, and then formulate the processing cost ratio as the percent of flows to be padded among all flows.

5.2 Communication Overhead

We first compare the communication overhead of our algorithms against an existing padding method, namely, packet-size rounding (simply rounding) [9]. We set the rounding parameter $\Delta = 512$ and $\Delta = 5120$ for $engine^B$ and $drug^B$, respectively. Note that these Δ values just lead to results satisfying 5-indistinguishability in the padded data, and are adapted only for the comparison purpose. The first set of experiments evaluate svsdSimple, svmdGreedy, and mvmdGreedy algorithms. To apply the svsdSimple algorithm, we generate two vector-action sets by synthesizing the flow-vectors for the last action of $engine^B$, $drug^B$, respectively. Note that the svmdGreedy and mvmdGreedy algorithms are equivalent with length-one action sequences.

Figure 2(a) shows padding cost of each algorithm against k. Compared to rounding [9], our algorithms have less padding cost, while svmdGreedy incurs significantly less cost than that of rounding. Table 10(a) shows the details of padding cost overhead ratio in percentage for $k = 192$. We observe that our algorithms are superior specially when the number of flow-vectors in a vector-action set is larger since our algorithms have high possibility to partition the flow-vectors with close value into padding group.

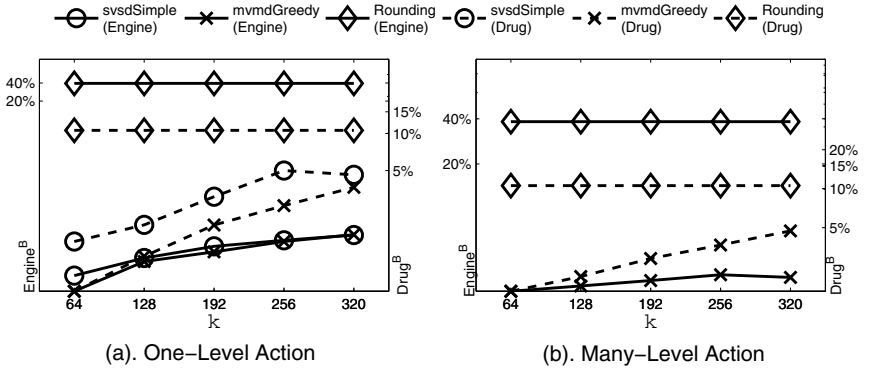


Fig. 2. Padding Cost Overhead Ratio

We then compare the mvmdGreedy with rounding algorithm in the case of action-sequences of lengths larger than one. Figure 2(b) shows padding cost of our mvmdGreedy algorithm and rounding algorithm against k. Packet-size rounding incurs larger padding cost than mvmdGreedy in all cases. Note that, rounding will get worse when Δ is set to be larger to minimize an eavesdropper's capability of inference [9]. For example, the padding ratio is 118% for $drug^B$ when applying rounding to make all drug

Table 10. Padding Cost Overhead Ratio When $k = 192$

Application	svsdSimple	mvmdGreedy	Rounding
$Engine^B$	0.0748	0.0604	39.4043
$Drug^B$	3.0743	1.8097	10.5922

(a). One-level Action

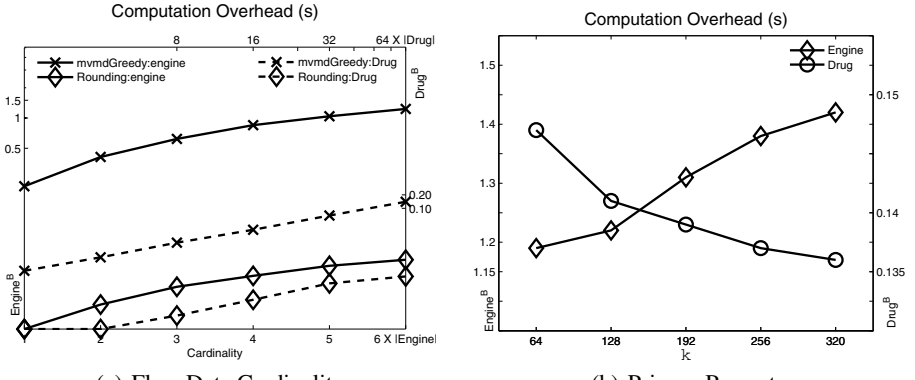
Application	mvmdGreedy	Rounding
$Engine^B$	3.3297	38.2659
$Drug^B$	2.8864	10.5672

(b). Many-level Action

information indistinguishable. Table 10(b) shows the detail of overhead ratio in percentage for $k = 192$. The reason for mvmdGreedy algorithm has more padding cost in the case of many-level action than in one-level is as follows. In many-level action, mvmdGreedy first partition each vector-action set (except VA_1) into padding groups based on the prefix of actions and regardless of the values of flow-vectors.

5.3 Computational Overhead

Figure 3(a) illustrates the computation time of mvmdGreedy algorithm and rounding algorithm against the flow data cardinality n. We generate n-sized flow data by synthesizing $\frac{n}{\sum_i(|VA_i|)}$ copies of $engine^B$, $drug^B$ respectively. We set $k = 160$ for this set of experiments, and conduct each experiment 1000 times and then take the average.



(a) Flow Data Cardinality (b) Privacy Property

Fig. 3. Execution Time

As the results show, the mvmdGreedy algorithm is practically efficient (1.2s for 2.7m flow-vectors) and the computation time increases slowly with n, although our algorithm requires slightly more overhead than rounding when it is applied to a single Δ value. However, this is partly due to the application-agnostic nature of the rounding method, which results in worse performance in terms of padding cost. Also, as shown in the previous section, such a method may require to test many Δ values for an optimal choice since larger values do not guarantee better privacy protection.

We then study computation time against privacy property k on the two synthesized vector-action sets ($6 \times engine^B$ and $64 \times drug^B$). As expected, rounding is insensitive to k since it does not have the concept of k. On the other hand, a tighter upper bound on the time required for mvmdGreedy is $O(n_p \times n \times 2k \times \lambda)$ in the worse case and $O(n_p \times n \times log(2k \times \lambda))$ in the average case, where λ is the maximal number of

actions which has the same prefix in all action-sequences. Clearly, when λ is $O(n)$, the computational complexity here is equivalent to that in Section 4.3. The reason for this tighter upper bound is that mvmdGreedy always feeds a vector-action set with maximal $2k \times \lambda$ cardinality to svmdGreedy (except VA_1 whose size is 26, a constant, for $search^B$), since:

- For each vector-action set VA_i, mvmdGreedy first partitions it into padding groups based on the prefix of each flow-vector (which has $O(|VA_i|)$ solution).
- There are at most $2k$ adjacent-prefixes in same padding group of VA_{i-1}.

Therefore, when $2k \times \lambda \ll n$, the execution time of mvmdGreedy algorithms for concrete vector-action sets is also a function of k. These two datasets in our experimental environment satisfy above condition, for example, $26(\lambda) \times 320(k) \ll 2.7m$ for $search^B$. In other words, in such case the execution time should be in the range of $[log(2k \times \lambda), 2k \times \lambda]$ times of $O(n_p \times n)$ which is the execution time of rounding algorithm. Figure 3(b) illustrates the computation time of mvmdGreedy algorithm against the privacy property k. Interestingly, the computation time increases slowly (from $1.19s$ to $1.42s$) with k for $engine^B$, and decreases slowly (from $0.147s$ to $0.136s$) for $drug^B$. Stress that the results are reasonable since both results fall within the expected range.

5.4 Processing Overhead

Our previous discussions have focused on reducing the communication overhead of padding while ensuring each flow-vector to satisfy the desired privacy property. To implement traffic padding in an existing Web application, if the HTTPS header or data is compressed, we can pad after compression, and pad to the header; if header and data are not compressed, we can pad to the data itself (e.g., spaces of required padding bytes can be appended to textual data). Clearly, the browser's TCP/IP stack is responsible for the header padding, while the original web applications regard the data padding as

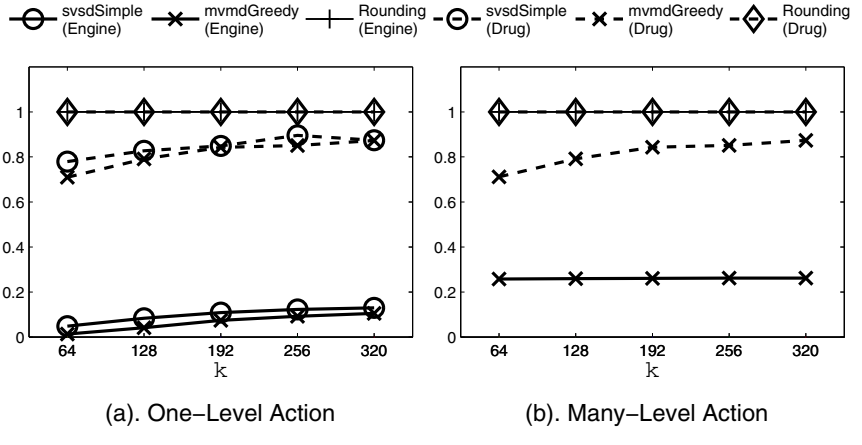


Fig. 4. Processing Cost Overhead Ratio

normal data. An application can choose to incorporate the padding at different stage of processing a request. First, an application can consult the outputs of our algorithms for each request and then pad the flow-vectors on the fly. Second, an application can modify the original data beforehand based on the outputs of our algorithms such that the privacy property is satisfied under the modifications. However, padding may incur a processing cost regardless of which approach to be taken. Therefore, we must aim to minimize the number of packets to be padded. For this purpose, we evaluate the processing cost ratio, which captures the proportion of flow-vectors to be padded among all such vectors.

Figure 4 shows the processing cost of each algorithm against k. Rounding algorithm must pad each flow-vector regardless of the k's and the applications, while our algorithms have much less cost for $engine^B$ and slightly less for $drug^B$. Note that in this paper our model and algorithm design has focused on minimizing the padding cost only. We consider refining them for reducing both the padding and processing cost as our future work.

6 Extension and Discussion

In this section, we first present an extension of our model and then discuss the implementation of our solutions.

6.1 Extension to l-Diversity

We now outline an extension of our model to further show that many existing PPDP concepts may be adapted to address PPTP issues. Specifically, we adapt l-diversity [20] to address cases that no all actions should be treated equally in padding (e.g., some statistical results regarding the likelihood of different inputs may be publicly known).

We first assign an integer *weight* to each action to represent the possibility it occurs. A larger weight indicates that the corresponding action is more likely to be performed. We also slightly change the definition of vector-action set to include the weight information. Then we apply *l-diversity* to quantify the privacy by ensuring the constraint as follows. For each padding group, the summation of weights corresponding to the actions in the group should be at least l times of the maximal weight value in that group.

With the aforementioned revisions, we reformulate the PPTP problem (MVMD problem in Section 3.3) to satisfy *l-diversity* instead. Observe that, the reformulated problem, called *diversity problem*, is simplified to Problem 3 if the weights of actions in a vector-action set are set to be identical and $l = k$. Thus, the *diversity problem* is at least as hard as Problem 3.

Although l-diversity in PPTP shares the same spirit with that in PPDP, algorithms for PPDP cannot be directly applied here, because in PPDP, many tuples may have the same sensitive values, whereas any action in an action-set is always unique, and we assign a weight for each action to distinguish its possibility to be performed from others. The detail of *diversity problem* and its algorithms are omitted due to space limitations.

6.2 Implementation Issues

In previous sections, we have presented algorithms for determining the amount of padding for each flow given the vector-action set. To incorporate our techniques into an

existing Web application requires following three steps. First, gather information about possible action-sequences and corresponding vector-sequences in the application. Second, feed the vector-action sets into our algorithms to calculate the desired amount of padding. Third, implement the padding according to the calculated sizes.

The main difference between implementing an existing method, such as rounding, and the ceiling padding method lies in the second stage. Thus, we have focused on this stage in this paper. Nonetheless, we have also briefly described how to collect the vector-action sets in Section 5.1 and how to facilitate the third stage in Section 5.4.

One may question the practicality of gathering information about possible action-sequences since the number of such sequences can be very large. However, we believe it is practical for most Web applications due to following facts. First, the aforementioned side-channel attack on web applications typically arises due to highly interactive features, such as auto-suggestion. The very existence of such features implies that the application designer has already profiled the domain of possible inputs (that is, action-sequences) for implementing the feature. Therefore, such information must already exist in certain forms and can be easily extracted at a low cost. Second, even though a Web application may take infinite number of inputs, this does not necessarily mean there would be infinite action-sequences. For example, a search engine like Google will no longer provide auto-suggestion feature once the query string exceeds a certain length. Third, all the three steps mentioned above are part of the off-line processing, and would only need to be repeated when the Web application undergoes a redesign.

We also note that implementing an existing padding method, such as packet-size rounding, will also need to go through the above three steps if only the padding cost is to be optimized. For example, without collecting and analyzing the vector-action sets, a rounding method cannot effectively select the optimal rounding parameter.

7 Related Work

The privacy preserving issue has received significant attentions in various domains, such as, data publishing and data mining [10,24], mobile and wireless network [4,5], social network [11,22], multiparty computation [21], web applications [6,9,26], and so on. In the context of privacy-preserving data publishing, since the introduction of the k-anonymity concept [24,28], much effort has been made on developing efficient privacy-preserving algorithms [1,16]. Many other models are also proposed to enhance the k-anonymity, such as l-diversity [20], t-closeness [17]. Recently, differential privacy [12] has been widely accepted as a strong privacy model for answering statistic queries.

Various side-channel leakages have been extensively studied in the literature. By measuring the amount of time taken to respond to the queries, an attacker may extract OpenSSL RSA privacy keys [7]. By differentiating the sounds produced by keys, an attacker may recognize the key pressed [3]. Ristenpart *et al.* discover cross-VM information leakage on Amazon EC2 [23]. Search histories may be reconstructed by session hijacking attack [8]. Saponas *et al.* show the transmission characteristics of encrypted video streaming may allow attackers to recognize the title of movies [25]. HTTPOS are proposed to prevent information leakages of encrypted HTTP traffic in [19]. Timing mitigator is introduced to achieve any given bound on timing channel leakage in [2].

Closest to our work, Chen *et al.* in [9] demonstrate through case studies that side-channel attacks are fundamental to web applications. Our model and solutions provide finer control over the tradeoff between privacy protection and cost. The model section of this paper has previously appeared as a short paper in [18] and now we significantly extend it with problem formulation, algorithm design, and experimental evaluations.

8 Conclusion

As Web-based applications become more popular, their security issues will also attract more attention. In this paper, we have demonstrated an interesting connection between the traffic padding issue of Web applications and the privacy-preserving data publishing. Based on this connection, we have proposed a formal model for quantifying the amount of privacy protection provided by traffic padding solutions. We have also designed three algorithms by following the proposed model. Our experiments with both real-world applications have confirmed the performance of our solutions to be superior to existing ones in terms of communication and computation overhead. For future research, we intend to investigate padding approaches for frequently updated vector-action sets, and the possibility of extrapolating the proposed model and approach to mitigate threats of other side-channel leaks.

Acknowledgment. The authors thank the anonymous reviewers for their valuable comments. Authors with Concordia University are partially supported by Natural Sciences and Engineering Research Council of Canada under Discovery Grant N01035, Strategic Project Grant STPGP/396651-2010, and Canada Graduate Scholarship, while Kui's research is supported in part by US National Science Foundation through grant CNS-1117811.

References

1. Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., Zhu, A.: Anonymizing Tables. In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 246–258. Springer, Heidelberg (2005)
2. Askarov, A., Zhang, D., Myers, A.C.: Predictive black-box mitigation of timing channels. In: CCS 2010, pp. 297–307 (2010)
3. Asonov, D., Agrawal, R.: Keyboard acoustic emanations. In: IEEE Symposium on Security and Privacy, p. 3 (2004)
4. Backes, M., Doychev, G., Dürmuth, M., Köpf, B.: Speaker Recognition in Encrypted Voice Streams. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 508–523. Springer, Heidelberg (2010)
5. Bauer, K., McCoy, D., Greenstein, B., Grunwald, D., Sicker, D.: Physical Layer Attacks on Unlinkability in Wireless LANs. In: Goldberg, I., Atallah, M.J. (eds.) PETS 2009. LNCS, vol. 5672, pp. 108–127. Springer, Heidelberg (2009)
6. Bilogrevic, I., Jadliwala, M., Kalkan, K., Hubaux, J.-P., Aad, I.: Privacy in Mobile Computing for Location-Sharing-Based Services. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 77–96. Springer, Heidelberg (2011)
7. Brumley, D., Boneh, D.: Remote timing attacks are practical. In: USENIX (2003)

8. Castelluccia, C., De Cristofaro, E., Perito, D.: Private Information Disclosure from Web Searches. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 38–55. Springer, Heidelberg (2010)
9. Chen, S., Wang, R., Wang, X., Zhang, K.: Side-channel leaks in web applications: A reality today, a challenge tomorrow. In: IEEE Symposium on Security and Privacy 2010, pp. 191–206 (2010)
10. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Samarati, P.: k-anonymous data mining: A survey. In: Privacy-Preserving Data Mining: Models and Algorithms (2008)
11. Danezis, G., Aura, T., Chen, S., Kıcıman, E.: How to Share Your Favourite Search Results while Preserving Privacy and Quality. In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 273–290. Springer, Heidelberg (2010)
12. Dwork, C.: Differential Privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part II. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
13. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey of recent developments. ACM Comput. Surv. 42, 14:1–14:53 (2010)
14. Kann, V.: Maximum bounded h-matching is max snp-complete. Inf. Process. Lett. 49, 309–318 (1994)
15. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C., Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: Analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell. 24, 881–892 (2002)
16. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient fulldomain k-anonymity. In: SIGMOD, pp. 49–60 (2005)
17. Li, N., Li, T., Venkatasubramanian, S.: t-closeness: Privacy beyond k-anonymity and l-diversity. In: ICDE 2007, pp. 106–115 (2007)
18. Liu, W.M., Wang, L., Cheng, P., Debbabi, M.: Privacy-preserving traffic padding in web-based applications. In: WPES 2011, pp. 131–136 (2011)
19. Luo, X., Zhou, P., Chan, E.W.W., Lee, W., Chang, R.K.C., Perdisci, R.: Httpos: Sealing information leaks with browser-side obfuscation of encrypted flows. In: NDSS 2011 (2011)
20. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: L-diversity: Privacy beyond k-anonymity. ACM Trans. Knowl. Discov. Data 1(1), 3 (2007)
21. Nagaraja, S., Jalaparti, V., Caesar, M., Borisov, N.: P3CA: Private Anomaly Detection Across ISP Networks. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 38–56. Springer, Heidelberg (2011)
22. Narayanan, A., Shmatikov, V.: De-anonymizing social networks. In: IEEE Symposium on Security and Privacy 2009, pp. 173–187 (2009)
23. Ristenpart, T., Tromer, E., Shacham, H., Savage, S.: Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In: CCS, pp. 199–212 (2009)
24. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. on Knowl. and Data Eng. 13(6), 1010–1027 (2001)
25. Saponas, T.S., Agarwal, S.: Devices that tell on you: Privacy trends in consumer ubiquitous computing. In: USENIX 2007, pp. 5:1–1:16 (2007)
26. Sun, J., Zhu, X., Zhang, C., Fang, Y.: Hcpp: Cryptography based secure ehr system for patient privacy and emergency healthcare. In: ICDCS 2011, pp. 373–382 (2011)
27. Sun, Q., Simon, D.R., Wang, Y.M., Russell, W., Padmanabhan, V.N., Qiu, L.: Statistical identification of encrypted web browsing traffic. In: IEEE Symposium on Security and Privacy (2002)
28. Sweeney, L.: k-anonymity: a model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems 10(5), 557–570 (2002)

Spying in the Dark: TCP and Tor Traffic Analysis

Yossi Gilad and Amir Herzberg

Department of Computer Science, Bar Ilan University, Israel
mail@yossigilad.com, amir.herzberg@gmail.com

Abstract. We show how to exploit side-channels to *identify clients without eavesdropping* on the communication to the server, and without relying on known, distinguishable traffic patterns. We present different attacks, utilizing different side-channels, for two scenarios: a fully off-path attack detecting TCP connections, and an attack detecting Tor connections by eavesdropping only on the clients.

Our attacks exploit three types of side channels: *globally-incrementing IP identifiers*, used by some operating systems, e.g., in Windows; *packet processing delays*, which depend on TCP state; and *bogus-congestion events*, causing impact on TCP's throughput (via TCP's congestion control mechanism). Our attacks can (optionally) also benefit from sequential port allocation, e.g., deployed in Windows and Linux. The attacks are practical - we present results of experiments for all attacks in different network environments and scenarios. We also present countermeasures for these attacks.

1 Introduction

Internet communication is often sensitive, and security measures must be taken to protect privacy against attackers. The exact measures depend on the exact threat; in particular, encryption protocols such as TLS [9] are necessary to protect *content* from an eavesdropping attacker.

However, encryption is insufficient to prevent *traffic analysis*, and in particular, to prevent exposure of the identities of the communicating peers. Users concerned against traffic analysis by eavesdropping attackers, use anonymizing services such as Tor. Other users may simply assume that the adversary is off-path (non-eavesdropping), and expect privacy against such (weaker) attackers.

We present three traffic-analysis attacks against these scenarios. Two attacks identify clients that communicate to a specific server directly over TCP (without anonymizing intermediaries such as Tor). Such attacks do not require eavesdropping at all, and may be launched by weak, off-path attackers, even for commercial motivations. In fact, since the attacks do not involve eavesdropping, they may even be deemed to be *legal* (not wiretapping). We believe technical measures should (and can) prevent such off-path traffic analysis.

Our third traffic analysis attack is against Tor users. It requires eavesdropping abilities only on the client side, and only spoofing abilities on the server side. We

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 100–119, 2012.

believe that this is an important scenario, since Tor clients are often concerned about attackers which can eavesdrop on their connection to the Tor relay, since the client-relay connection may be insecure (e.g., Internet cafe) or controlled by a potential snoopy organization (employer, government, etc). Our evaluation of this attack is yet incomplete; however, the preliminary results which we present in this paper provide a warning about a new attack vector on the Tor anonymity network.

Our attacks exploit different *side-channels*, providing useful information on a TCP/Tor connection to an off-path attacker (for Tor, attacker can eavesdrop, but only on the client). Side channels have been extensively used in attacks on cryptographic systems, e.g., [22], but also in attacks on privacy, e.g., [13], and more recently also applied to traffic analysis [6,26,35].

Our attacks on direct (i.e., non-anonymous) TCP connections can be viewed as instances of an *attack pattern* which we call *Query-Probe-Query*, illustrated in Figure 1; this is a generalization of the well-known *idle (stealth) scan* attack [25,34], and of the measurement method used in [5]. In the Query-Probe-Query attack pattern illustrated in Figure 1, the first query measures some 'pre-probe state'; the probe may cause some change on the state, where the change depends on the property measured, e.g., whether a specific client port is open; and the final query measures the 'post-probe state'. This attack pattern can be applied with different queries and probes, to measure different values.

In our implementations, each probe is a packet, or few packets, sent to the client $\mathcal{C}$. The probes test for some event e, such that if e holds then $\mathcal{C}$ will send some packet(s), and otherwise he will not send a packet (or send less packets). We use the pre/post probe response to infer on e: we measure the increase in the IP-ID counter, or measure time between receipt of the two response packets (1b. and 3b in Figure 1). Table 1 summarizes our results for traffic analysis on direct TCP connections.

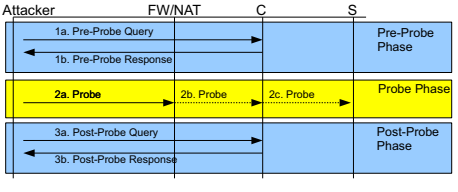


Fig. 1. *Query-Probe-Query* off-path attack pattern. Attacker uses spoofed source address for probe.

Table 1. Results for probing direct (TCP) connections. Attacker location is relative to the client (victim). Success rates can be improved by repeating the attack.

	Side Channel	Adversary Location	
		Local	Remote
Success Rate	IP-ID (Section 3)	1	0.92
	Timing (Section 4)	0.74	0.58
Attack Duration (seconds)/ Data Sent (MB)	IP-ID (Section 3)	14/0.6	38/2
	Timing (Section 4)	70/5	50/4

In the recent years, there is growth in the use of anonymity mechanisms such as *Tor* [10]. Tor is a low latency, circuit based, anonymity network that is widely used and increasing in popularity (according to [1], increase of about 70% in recent year). Tor is designed to ensure traffic privacy, even when the adversary is able to eavesdrop on either $\mathcal{C}$ or $\mathcal{S}$. However, due to its low latency, Tor cannot ensure traffic privacy against attackers eavesdropping at *both* ends ($\mathcal{C}$ and $\mathcal{S}$), see discussion on related works below.

We show how an adversary capable of eavesdropping on the client, $\mathcal{C}$, but not on the server, $\mathcal{S}$, is able to detect that $\mathcal{C}$ is communicating with $\mathcal{S}$. This attack uses a side channel as well, but does not follow the query-probe-query pattern; here the attacker causes a reduction in the rate of packets that would reach $\mathcal{C}$ in case that he is communicating with $\mathcal{S}$, and then tests whether reduction had occurred.

Our attacks on Tor are *active*, i.e., involve *sending* packets to Tor exit relay. When there is a known, distinct traffic pattern to the communication of specific server (*website fingerprint*), then alternative passive attacks may be applicable as well, e.g., [18,19,28]. It may be possible to extend our techniques to also take advantage of such site fingerprint, when available.

1.1 Related Works

IP-ID Side-Channel and Off-Path Traffic Analysis. We show how the use of globally-incrementing IP-ID field in IP headers, provides side-channel allowing effective off-path traffic analysis. The use of globally-incrementing IP-ID is recognized, in [17], as a common practice with known security implications; e.g., both globally-incrementing and per-destination incrementing IP-ID allow interception, injection and discarding of fragmented traffic [15]. Globally-incrementing IP-ID can allow estimation of the number of packets sent [32], stealth-scan for open ports (idle scan) [31] and counting hosts behind NAT [5].

The technique that we present for the case of a client that uses a globally-incrementing IP ID and is not connected via a firewall/NAT (see Figure 1) is a variant of idle-scan [25,34]. The difference is that while idle-scan probes a server for an open (i.e., 'listening') port, our attack probes a client for a connection with a server.

The only other previous work we found that performs traffic analysis by off-path attacker, using a side-channel, is [19]. Their attack is based on detecting changes in the round trip delays from the attacker to the DSL router; this is a rather crude side channel, much less efficient than both the IP-ID and the timing side channels we use. Indeed, they only present detection of whether a client is browsing or playing a video, based on the significant difference in bandwidth, and assuming no other traffic. Our results dramatically improve the impact of detection compared to theirs, provided that the attacker can communicate with the clients.

Tor Traffic Analysis. Low-latency anonymity networks are known to be vulnerable to traffic correlation attacks by an attacker that eavesdrops on both ends; this problem, and possible countermeasures, was studied in several works,

e.g., [7,24,36]; efficiency and accuracy can significantly improve if attacker can also *manipulate* traffic at the exit relay, see [30]. Indeed, Tor designers are well aware of its inability to properly protect against an attacker (eavesdropping) at both ends of the circuit.

Other attacks manipulate the traffic at the server or the last (exit) relay in the circuit, and use different techniques to detect the relay along the path based on delays [6,12,27]. These works assume that the attacker controls the server or the exit relay, but do not require client-side eavesdropping. In contrast, our attack on Tor requires client-side eavesdropping, but does not require control over the server or exit relay. An obvious challenge is to combine the results, and identify clients without controlling server or exit relay, and without eavesdropping at all. Our traffic detection attacks on TCP may be applicable.

1.2 Our Contributions

The main contribution of this paper is identification and analysis of side channels in the TCP/IP suite and their practical implications on privacy, as we verify in experiments. We provide practical countermeasures to the problems that we identify, these allow quick patching at the firewall level and require no changes to hosts or core operating system services.

This work motivates use of cryptography in lower network layers and in particular IPsec [20] as we show that higher network layer solutions such as SSL/TLS do not prevent blind traffic analysis.

1.3 Paper Organization

In Section 2 we present our attacker models and the scenarios that we consider, we also present the criteria we use to measure the effectiveness of the attacks. Sections 3, 4 present the global-ID and timing side channels; both sections provide results of empirical experiments. Section 5 presents our attack on Tor and corresponding experiments. Section 6 presents practical defenses. Finally, Section 7 presents our conclusions from this work, as well as future research directions.

2 Model

Let $\mathcal{C}$ and $\mathcal{S}$ be communicating TCP client and server (respectively). We consider two types of adversaries, depending on how $\mathcal{C}$ and $\mathcal{S}$ are connected. In Sections 3 and 4, we consider the case that $\mathcal{C}$ and $\mathcal{S}$ have a direct TCP connection. In Section 5, $\mathcal{C}$ connects to $\mathcal{S}$ through the onion routing anonymity network, Tor [10]; i.e., $\mathcal{C}$ communicates with $\mathcal{S}$ via a circuit of relays (proxies). The goal of the attacker is to identify clients who connect to a server $\mathcal{S}$. We identify $\mathcal{S}$ using its IP address and port.

We consider two types of attackers: Mallory, an *off-path adversary*, and Eve, an *eavesdropping adversary*. The attackers can send *spoofed packets*, i.e., packets with fake (spoofed) sender IP address. Due to ingress filtering [4,14,21] and other

anti-spoofing measures, IP spoofing is less commonly available than before, but still feasible, see [2,11]. Apparently, there is still a significant number of ISPs that do not perform ingress filtering for their clients (especially to multihomed customers). Furthermore, with the growing concern of cyberwarfare and cybercrime, some ISPs may intentionally support spoofing. Hence, it is still reasonable to assume spoofing ability.

We describe both adversary models in Sections 2.1 and 2.2 below. Section 2.3 presents the criteria we use to evaluate the attacks we present.

2.1 Mallory - Off-path Adversary

We assume that $\mathcal{C}$ visits a website that Mallory controls, denoted www.mallory.com. Mallory uses this (legitimate) connection, to probe whether $\mathcal{C}$ has any connections $\mathcal{S}$, see Figure 2.

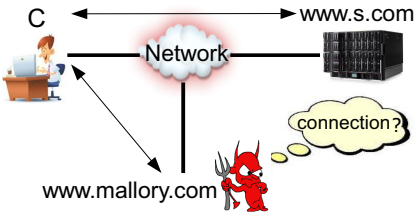


Fig. 2. $\mathcal{C}$ is surfing in both Mallory and $\mathcal{S}$'s sites, Mallory tries to detect whether there is a connection between $\mathcal{C}$ and $\mathcal{S}$

We consider three variants of Mallory, as illustrated in Figure 3: *with-*$\mathcal{C}$*, near-*$\mathcal{C}$ and *remote*. These differ with respect to Mallory's abilities to communicate with $\mathcal{C}$; the greater the distance, the more likely it is that packet loss or reordering occurs, decreasing the quality of the side channels.

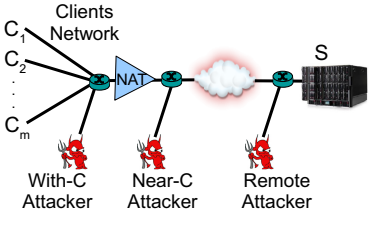


Fig. 3. Three variants of the Mallory adversary

The *with-*$\mathcal{C}$ and *near-*$\mathcal{C}$ attackers are located near the client ($\mathcal{C}$); the difference between them is that the *with-*$\mathcal{C}$ adversary directly communicates with the client, allowing Mallory to take advantage of Windows globally incrementing port allocation (if $\mathcal{C}$ runs Windows). When the adversary and $\mathcal{C}$ communicate via a NAT (near-$\mathcal{C}$ or remote), we assume that the NAT uses per destination incremental assignment of external ports (e.g., as in the widely-used IP-tables NAT/Firewall provided in Linux). See in Section 3 how we exploit different client port allocation techniques. Finally, the *remote* Mallory attacker simulates an adversary that communicates with the clients from a remote location, i.e., via a high latency, jittery and lossy channel.

2.2 Eve - Adversary for Anonymized Connections

In the attacks on Tor, we consider the adversary Eve who is able to eavesdrop on many clients that use Tor, however, Eve cannot eavesdrop on the servers

(see Figure 4). Such an adversary may include a government or an employer, spying on citizens or employees. Eve's goal is to detect which of the clients is communicating (using Tor) with a particular watched/restricted site, $\mathcal{S}$.

2.3 Attack Evaluation Criteria

In addition to measuring the success, false positive and false negative rates, we consider two additional measures. The first measure is the time that an adversary (with some reasonable constant bandwidth) needs to run the attack in order to reach a particular success probability for detecting a connection. This value also provides the minimal detectable connection time. The second measure is the average amount of data per victim that the attacker is required to send to reach a particular success rate.

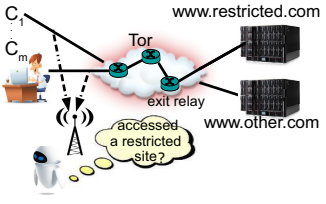


Fig. 4. Eve identifies that some of the clients she eavesdrops on are using Tor and wants to detect which of them is communicating with www.restricted.com. $\mathcal{C}$ (C_m) connects to www.restricted.com via a circuit of 3 relays.

3 Globally-Incrementing Identifier Based Traffic Analysis

This section presents a probing technique that allows an off-path (blind) adversary, Mallory, to identify a connection between a client $\mathcal{C}$ and a server $\mathcal{S}$ when $\mathcal{C}$ uses a globally incrementing IP identifier (IP-ID)[1]. This side channel is only applicable when the TCP connection is over IPv4, since in IPv6 [8] the IP-ID field is only specified in fragmented traffic and TCP packets are rarely fragmented. In the following section we introduce a general technique that does not rely on IP-ID and also applies to IPv6.

A globally-incrementing identifier is not really hidden from Mallory, who can usually learn its value simply by receiving some packet from the victim. A globally incrementing IP identifier is used in all Windows versions we tested (including XP, Vista and 7) and is also the default configuration in FreeBSD; clients running these systems are vulnerable to the attack below. The vast deployment of Windows on client machines (more than 70% according to browser user-agent based surveys, see [33]) makes IP-ID attack vector very practical.

Section 3.1 defines a *port test* that uses the leakage in the IP-ID field to detect whether $\mathcal{C}$ is communicating with $\mathcal{S}$ through a tested port. The test depends on whether $\mathcal{C}$ is connected to the network through a NAT or a stateful firewall that keeps track of existing connections; the test used when $\mathcal{C}$ is connected through a NAT/firewall device the attack is a bit simpler. We believe that this is the more common scenario, since recent versions of Windows (XP SP2 and later) ship with a

[1] $\mathcal{S}$'s IP-ID implementation does not influence the probing technique.

built in (stateful) firewall that is enabled by default, and furthermore, use of NAT devices in small local area networks connecting clients to the Internet is common. Due to space limitations we describe only this test and include the test for the complementary scenario (no firewall/NAT) in an online technical report [16].

In Section 3.2 we describe how Mallory can identify a relatively small set of client ports to test for a connection with $\mathcal{S}$; Mallory performs the port-test for all of them. Section 3.3 presents our experimental setup and empirical results.

3.1 Port-test for a Client Behind a Firewall/NAT

According to the TCP specification [29] (Section 3.9, bottom of page 69), the first check that a recipient conducts on an incoming packet, in case it belongs to an established connection, validates that the sequence number is within the congestion window. If this check finds the packet invalid, then the recipient discards the packet and sends a duplicate Ack feedback. A stateful firewall or NAT device connecting $\mathcal{C}$ to the network keeps track of existing connections and processes all incoming packets before they reach $\mathcal{C}$. We use the following observation: incoming packets that do not belong to an established connection will be discarded before reaching $\mathcal{C}$ (by firewall/NAT), whereas packets that belong to an existing connection, but specify arbitrary (probably invalid) sequence numbers will reach $\mathcal{C}$ who replies with a duplicate Ack.

The port test for the case of firewall/NAT deploying client is according to the general query-probe-query pattern. The probe specifies $\mathcal{S}$'s address and port as source (i.e., probe is spoofed) and $\mathcal{C}$'s address as destination, Mallory specifies a different destination port in each test. Figure 5 illustrates two iterations of the port test: in the first iteration, the firewall/NAT blocks the probe packet (i.e., no connection through the tested port). In the second iteration, the probe specifies existing connection parameters (IP addresses and ports) and therefore reaches $\mathcal{C}$ who processes the probe and sends a duplicate Ack to $\mathcal{S}$.

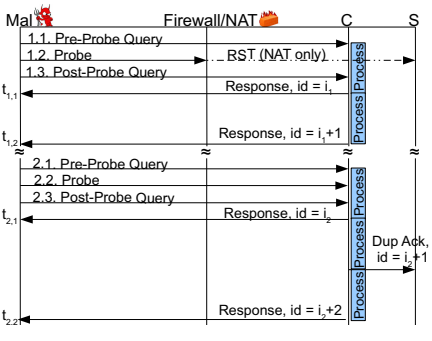


Fig. 5. Two iterations of port test

Notice that since the probe packet appears to be from $\mathcal{S}$ (in case it specifies a valid 4-tuple), it is difficult to block the probe in firewalls without blocking the legitimate connection that $\mathcal{C}$ has with $\mathcal{S}$.

When $\mathcal{C}$ uses a global identifier, the difference in the IP-ID field in $\mathcal{C}$'s responses to Mallory indicates whether $\mathcal{C}$ had sent a packet in response to the probe (duplicate Ack). If Mallory identifies that $\mathcal{C}$ had sent a packet, then it is likely that $\mathcal{C}$ is communicating with $\mathcal{S}$ via the tested port; however, the identifier may have increased since $\mathcal{C}$ had sent an independent packet to some other peer.

Repeating this test several times allows Mallory to efficiently detect whether $\mathcal{C}$ is connected to $\mathcal{S}$ and reveal $\mathcal{C}$'s, see empirical evaluation below.

We keep a 'score' for each possible port, and increment a specific port's score by 1 point for every test that seems to indicate that there is a connection through that port. We conduct $r > 1$ rounds of the attack, where each port is tested. Finally, we decide that there is a connection if there is a port with a score higher than a threshold, *TH*.

Some firewalls have an option to randomize the IP-ID; our tests would, of course, fail if the packets pass through such randomizing firewall. The attack we describe in the following section applies even in this scenario (but is less accurate).

Implementing Test Queries/Responses. Our attacks use packets that Mallory receives from $\mathcal{C}$ to learn the effect of the (spoofed) probe packet. Mallory can cause $\mathcal{C}$ to send her such packets by using the legitimate TCP connection that she has with $\mathcal{C}$: a query is some short data packet that Mallory sends to $\mathcal{C}$, the response is the $\mathcal{C}$'s acknowledgment sent back to Mallory. This allows Mallory to bypass typical firewall defenses (e.g., Windows), since all packets in the test appear to belong to legitimate connections (requests to $\mathcal{C}$-Mallory connection, probe to $\mathcal{C}$-$\mathcal{S}$ connection). See further details in the technical report [16].

3.2 Improving the Search: Client Port Allocation Algorithms

The port test that we presented allows Mallory to test whether the client has a connection to some server via a specific port. There are 2^{16} possible ports that $\mathcal{C}$ might use to communicate with $\mathcal{S}$. However, common client port allocation paradigms allow more efficient attacks.

Below we present two common paradigms and methods to reduce the number of tests for each of them.

Globally Incrementing. the client port is incremented for every new connection (initialized to a random value) Algorithm 1 in [23] describes the implementation. This approach is used in Windows and FreeBSD. If $\mathcal{C}$ uses this port allocation paradigm, then recent connections that the client forms are likely to use 'close' ports to that $\mathcal{C}$ uses in the connection with Mallory. Hence, Mallory can test only these ports.

Per-server Incrementing. the client port is incremented for every new connection with the server. Connections to different servers use different counters. This approach is used in Linux; Algorithm 3 in [23] describes the implementation. The previous 'trick' we presented does not work in this case since the port that $\mathcal{C}$ uses for the connection with Mallory does not correlate to that $\mathcal{C}$ uses to communicate with $\mathcal{S}$. However, we can still use the counter property of this paradigm: Mallory causes $\mathcal{C}$ to create x 'dummy' connections to $\mathcal{S}$ (we explain how below); since these connections all share the same counter, they are sequential. Hence, Mallory can test every port $y = 0 \pmod{x}$ and identify p, a port $\mathcal{C}$ that uses to communicate with $\mathcal{S}$. Next, Mallory checks all ports in the interval

$[p-x, p+x]$ and checks whether there are at least $x+1$ connection ports. If yes, then $\mathcal{C}$ has an 'independent' connection with $\mathcal{S}$. In this method, the attacker would test roughly $\frac{2^{16}}{x}$ different ports.

In is left to describe how Mallory causes $\mathcal{C}$ to establish multiple connections with $\mathcal{S}$. Since $\mathcal{C}$ is in Mallory's site, she can run a script (in the browser sandbox) on $\mathcal{C}$. This script, while very limited, can open connections with other servers to dynamically embed remote objects. We use it to open connections to www1.mallory.com,. . . ,wwwx.mallory.com which are domains that Mallory controls. Since Mallory controls the DNS records for these domains, she sets each of these records to point to the same IP, that of $\mathcal{S}$. Browsers open a new connection for each domain (regardless of its IP address); hence, this technique, which we verified on Internet Explorer, Firefox and Chrome, opens x new connections to $\mathcal{S}$.

The typical limitation of x is the number of connections that a browser can have simultaneously; this limitation is typically one or few dozens; e.g., 16 in Firefox. In our experiments below and in the following section, we use $x = 10$.

3.3 Empirical Evaluation

Setup. In our empirical evaluation, the client network is a class C subnet that has 5 clients running Windows 7, each of them sends on average 64 packets per second to other peers in the subnet (these packets are short, to simulate clients that usually send Ack packets or short requests). Mallory probes one of the clients in the network, $\mathcal{C}$, who connects to her (malicious) website. Mallory's bandwidth is limited to 10 mbps. We used the network topology illustrated in Figure 3, network nodes are connected through switch devices. The NAT device in the network topology is a Linux machine (kernel version 2.6.35) running IP-tables (version 1.4.4). The server machine runs Linux (kernel version 2.6.35) and uses an Apache web-server (version 2.2.14). When we evaluate the attack for the 'Remote Attacker' scenario, the adversary communicates with the clients via a traffic shaper that induces high latency (200ms), significant loss probability (0.5%) and jitter (1-10 milliseconds).

Evaluation. We first evaluated the attack in case that $\mathcal{C}$ is communicating with $\mathcal{S}$. We compared between the score of the 'connection port' (i.e., port that $\mathcal{C}$ uses for the connection) to that of the best appearing non-connection port (i.e., port with the highest score that is not the connection port) in each round (repetition of the attack, see discussion above); note that the highest scoring non-connection port may change between rounds.

Figure 6 shows results for near-$\mathcal{C}$ and remote attackers. In both environments, the score of the connection port was well above 50% of the maximal score, certainly after five or more reounds; hence, for efficiency, we can continue testing only 'high scoring' ports in advanced rounds. Namely, a port is tested in the next round only if its current score is above 50% of the maximal possible score.

We implemented an adversarial website that presents its clients a request to arbitrarily decide whether to connect ('surf') to a third-party website, $\mathcal{S}$; our website attempted to detect the clients' choice. We used an automatic client, $\mathcal{C}$, that chooses to connect to $\mathcal{S}$ with probability $\frac{1}{2}$ and implemented the port-test above.

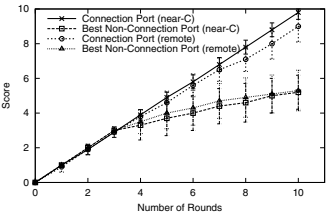


Fig. 6. Global-ID attack. Comparison of a connection port to that of the highest scoring non connection port as a function of round number. Each measurement is an average of 10 runs, error-bars mark the standard deviation values.

The choice of whether there is a connection between $\mathcal{C}$ and $\mathcal{S}$ is according to a threshold over the final score of the ports. Namely, if there exists a port with a score over this threshold, then we identify that there is a connection. Figure 6 shows that a choice of 70% of the maximal possible score as a threshold provides a good seperation between the connection port (in case it exists) and other ports. Figures 7 - 9 show the success rate in detecting whether $\mathcal{C}$ communicates with $\mathcal{S}$ for different adversary locations as a function of the duration of the attack. Figure 10 compares the average amount of data that Mallory sends (per victim) to reach different success rates and for different locations in the network.

In Figures 7 - 10, the measurements are the average of 50 runs; error-bars mark the standard deviation values (for readability, not all measurements specify the error bars). Note that the thresholds that we have used in our evaluation may not work as well in other scenarios, e.g., when the client sends much more than 64 packets per second the thresholds should be higher.

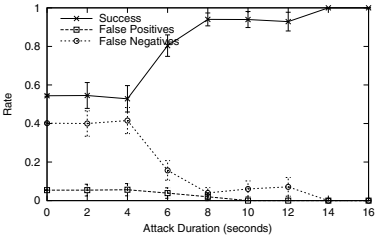


Fig. 7. Global-ID attack, with-$\mathcal{C}$ attacker

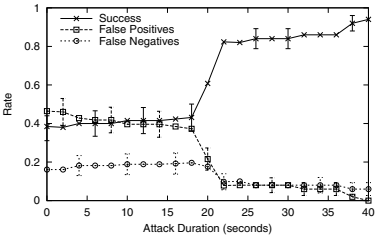


Fig. 8. Global-ID attack, near-$\mathcal{C}$ attacker

4 Time-Based Traffic Analysis

The globally incrementing IP-ID side channel, presented in Section 3, exploits an operating system flaw. In this section we explore a more generic, timing based, side channel that is applicable when $\mathcal{C}$ is behind a firewall or a NAT. We

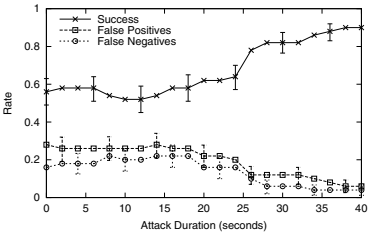


Fig. 9. Global-ID attack, remote attacker

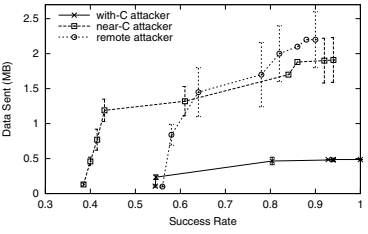


Fig. 10. Amount of data Mallory sends as a function of her success rate

define below a new port test which resembles the IP-ID based port test and is illustrated in Figure 5 as well.

The timing attack is based on the following observation: if $\mathcal{C}$ is protected by a firewall or connects through a NAT device, then in case that Mallory tests the correct port, $\mathcal{C}$ sends an additional packet to $\mathcal{S}$ (response to the probe); this delays processing of following packets, and in particular the post-probe query; see illustration in Figure 5. We use this delay to identify the connection.

4.1 Timing-Based Port Test

A significant challenge is the jitter in the network, i.e., latencies may vary while testing different ports. Thus, identifying the longest time difference between two responses and testing whether it is over a threshold is likely to produce an incorrect result. We cope with this challenge by *relatively comparing* ports: we assign each port to a small group of s arbitrary ports.

Ports in each group are tested one after the other; we assume that jitter does not vary much during the short time interval of testing a specific (small) group. After testing a group, each port is assigned with a relative rank according to the time difference between responses in the corresponding port-test; the lower the (group-relative) rank, the greater the time difference and the more likely is a connection through that port. We conduct several rounds of this attack (to reduce the probability of errors).

Similarly to the attack presented in the previous section, we keep a score for each port and after each round increase a port's score according to its rank: denote by σ_i the number of points that a port gains if it has rank i within the group, these weights are normalized; i.e., $\sum_{i=1}^{s} \sigma_i = 1$, and for every $i < j$, $\sigma_i \geq \sigma_j$. The values of s and the vector $\sigma = (\sigma_1, \cdots, \sigma_s)$ depend on the channel between Mallory and $\mathcal{C}$. We employ a machine learning approach (genetic algorithm) to learn appropriate value for the vector σ; see details of the algorithm in [16]. Let μ, μ' denote the expected scores of connection and non connection ports respectively. The target function of the learning algorithm is to maximize $\mu - \mu'$. In our empirical evaluation below we explain how Mallory obtains measurements of μ, μ' for different values of s, σ.

4.2 Empirical Evaluation

The environment we used to evaluate the timing attack is as described in Section 3.3, except that the client machines run Linux (kernel version 2.6.35) instead of Windows; hence, the attacker cannot employ the global IP-ID based attack. All Linux distributions ship with IP-tables firewall, its rule-set is empty by default; we therefore evaluated only the scenarios where Mallory is near-$\mathcal{C}$ remote (see Figure 3), i.e., Mallory communicates with $\mathcal{C}$ and $\mathcal{S}$ via a NAT device.

The first task is to obtain a good estimation of the optimal values of s, σ for the channel between $\mathcal{C}$ and Mallory (this depends on Mallory relative location to $\mathcal{C}$). The machine learning algorithm we employ uses the connection that Mallory has with $\mathcal{C}$ (see Figure 2): since for this connection Mallory knows the client's port, he is able to obtain measurements for different group sizes (s) and weights (the vector σ), see more details in [16]. We found that these values significantly differ between the two attacker locations; e.g., in our setup we found $s = 31$ to be suitable for a near-$\mathcal{C}$ attacker while $s = 4$ appeared optimal for the remote attacker. Figure 11 compares the connection-port score to that of the highest scoring non-connection port as a function of the number of rounds.

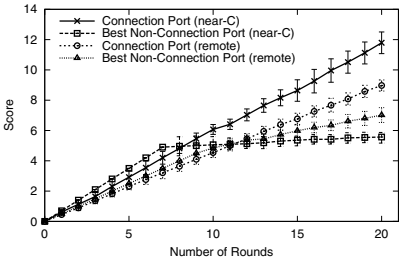


Fig. 11. Timing Attack. Comparison of a connection port to that of the highest scoring non connection port as a function of round number. Average of 10 runs, error-bars mark the standard deviation values.

Next, we derive two thresholds for promoting ports to following rounds according to their current score, this is similar to the experiments in Section 3.3. According to the training set results, a choice of 60% of the maximal score for the near-$\mathcal{C}$ attacker scenario and 40% in for the Interent attacker scenario appear to be reasonable. As in Section 3.3, these thresholds require further research for other scenarios, e.g., thresholds are effected by the victim's transmission rate.

We implemented the timing attack and conducted an experiment similar to that presented in Section 3.3. We set the threshold for deciding whether a connection exists between $\mathcal{C}$ and $\mathcal{S}$ according to the difference between the expected scores of a connection port (μ) and a non connection port (μ') as derived from our training measurements. See analysis in [16]; in this experiment we set the threshold to $0.2\mu' + 0.8\mu$. We measured our success rate in probing whether $\mathcal{C}$ is communicating with a (third-party) website, $\mathcal{S}$. Results are in Figures 12 - 14.

Comparing these results to those of the ID based attack, more time is required to obtain similar success rates, and the maximal success rates reached are lower. However, the results show that the timing attack does provide information on the connection (since success ratio is greater than 0.5); but its hint is often

misleading (since success ratio is significantly less than 1). Attacker can repeat the attack several times and select by the majority.

Figure 15 illustrates the average amount of data that Mallory needs to send in order to reach a particular success rate for different locations in the network and number of probes in each test.

In Figures 12 - 15, the measurements are the average of 50 runs; error-bars mark the standard deviation values.

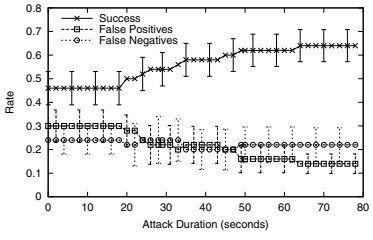


Fig. 12. Timing attack, near-$\mathcal{C}$ attacker. Mallory sends 2 probes in each test.

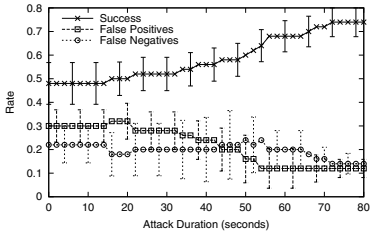


Fig. 13. Timing attack, near-$\mathcal{C}$ attacker. Mallory sends 5 probes in each test.

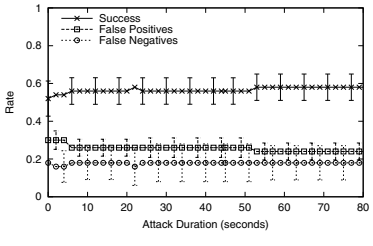


Fig. 14. Timing attack, remote attacker. Mallory sends 5 probes in each test.

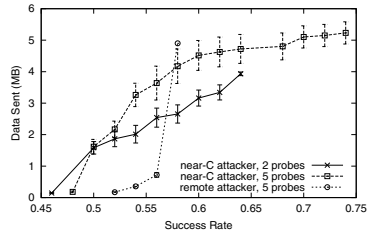


Fig. 15. Amount of data Mallory sends as a function of her success rate.

5 Traffic Analysis for Tor Clients

In this section we consider the second scenario presented in Section 2, where $\mathcal{C}$ uses an onion routing infrastructure to connect to $\mathcal{S}$. We focus on the popular Tor network, but similar attacks may apply to other low latency anonymity networks. In this section we assume that the attacker, Eve, is able to eavesdrop on $\mathcal{C}$ (but not on $\mathcal{S}$).

Here, Eve actively interferes in the possible connection between $\mathcal{C}$ and $\mathcal{S}$ and then tests whether a change in the rate of packets that $\mathcal{C}$ receives occurred. If the result is positive, then it is an indication that $\mathcal{C}$ communicates with $\mathcal{S}$. As of writing this version of the paper, we only did preliminary testing of this attack; more work is required to evaluate the practicality of this attack.

A Tor client connects to a remote server via a chain of relays (proxies). The last relay in the chain, i.e., the exit relay, has a direct TCP connection with the server. The number of possible Tor exit relays is important for our attacks (since a direct connection exists between the exit relay and the server); the Tor network comprises of few thousand relays, about one thousand of which can perform as exit relays (see [1]). However the number of different exit relays that a client is *likely* to use is significantly lower: first, a client can only use online relays; second, Tor clients typically choose the exit relays according to various parameters such as stability and bandwidth. We have formed Tor circuits from two clients in different geographic locations and kept track on the exit relay that was used. The measurements show that 20% of the 2000 circuits which we created (using Tor client version 0.2.2.35) had one of 7 specific exit relays. Figure 16 illustrates our measurements.

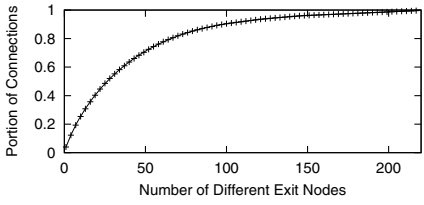


Fig. 16. The portion of 2000 circuits we created using the Tor client as a function of the number of different exit relays used

5.1 The Indirect Rate Reduction Attack

In this section we present an attack that uses the following observation: if Eve influences the rate of communication between $\mathcal{S}$ and the exit relay, then this, in turn, will change the rate of the connection between $\mathcal{C}$ and the first (entrance) relay. Eve sees the latter connection and is able to detect the change.

Since Eve can only observe the aggregated rate of data that $\mathcal{C}$ receives from the entrance relay (since communication is encapsulated), this attack vector weakens when $\mathcal{C}$ communicates with several other servers via one Tor circuit and $\mathcal{C}$'s connection rate with $\mathcal{S}$ is relatively small to that of the other servers.

The following attack uses TCP congestion control mechanisms to fake congestion events; hence, reducing the communication rate. This attack is based on the insight previously noted in Section 3.1: by sending a (spoofed) packet to an exit relay, Eve would cause that relay to immediately send a duplicate acknowledgment (Ack) in response to $\mathcal{S}$, as long as Eve's packet appears from an existing connection between the exit relay and $\mathcal{S}$. The duplicate Ack that the exit relay sends to $\mathcal{S}$ in response, has a valid sequence number and $\mathcal{S}$ will accept it. A sequence of three duplicate Acks is interpreted by TCP as a congestion event, see [3]; when it occurs, $\mathcal{S}$' congestion window shrinks. The exact effect depends on the TCP implementation that the server runs. Until recently, TCP Reno variant was default in Linux (the common operating system of server machines); for this variant each congestion event halves the size of the congestion window. Recent Linux kernels use the TCP Cubic variant, where the TCP window size is multiplied by a constant of 0.8 for each congestion event.

The congestion window size directly effects the sender's ($\mathcal{S}$) transmission rate: $\mathcal{S}$ only sends as much as the congestion window allows. Thus, by causing the exit relay to send a sequence of 3 duplicate Acks to the server, Eve causes the latter to significantly reduce its 'sending' rate. This attack is illustrated in Figure 17, which shows the effect when Eve sends the spoofed packets to an exit relay and port through which there is a connection with $\mathcal{S}$.

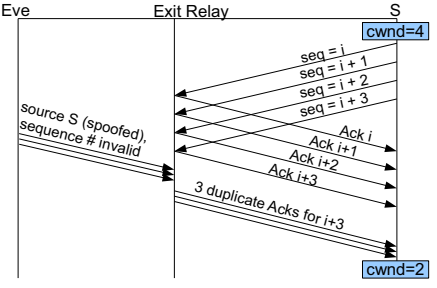


Fig. 17. Eve causes the exit relay to send 3 duplicate Acks to $\mathcal{S}$. $\mathcal{S}$' congestion window is halved as a result.

Attack Process. We use the asymmetry in the distribution of client choice for exit relays to reduce the number of packets that the attacker needs to send to perform indirect rate reduction. Namely, while there are many exit relays available, there are only few 'likely' exit relays that a client might use (see discussion above and Figure 16). For every server IP address s and likely exit relay x, Eve can optionally employ one of the attacks in the Sections 3 and 4 to identify those exit relays that communicate with the server. This optional step will reduce the effort in the following steps of the attack. The techniques in Sections 3 and 4 do not only identify the existence of a connection between two peers, but also identify the client port – if a connection exists, then this is the port with the highest score; see details on how Eve employs these techniques in [16]. Next, for each of the 'suspected' connections, she performs the indirect rate reduction attack described above and checks which of the clients had experienced 'rate reduction'. This process repeats several times for statistical coherency; after each iteration the attack is suspended to allow $\mathcal{S}$'s congestion window to recover.

An important property of this attack is that the spoofed packets that Eve sends to the exit relay in order to reduce the server's rate, are not client specific. Hence, in case that Eve eavesdrops on multiple clients (e.g., a government spying on its citizens) this attack would simultaneously check which of these possible clients has a connection with $\mathcal{S}$.

Characteristics of Vulnerable Connections. Since the attack repeats for several iterations with intermediate suspensions, this attack requires connections lasting several minutes (see evaluation below). Furthermore, the connection must be 'active', i.e., the server should send data to the client while the attack takes place; this allows Eve to detect rate reductions and allows the congestion window to recover when the attack is suspended. These type of connections include, for example, file transfers (over FTP or HTTP).

5.2 Analysis

Our analysis in this subsection assumes that Eve does not try to detect a direct connection between the exit relays and the server $\mathcal{S}$ (the optional step). Instead,

she performs the indirect rate reduction attack on every likely exit relay and all possible ports.

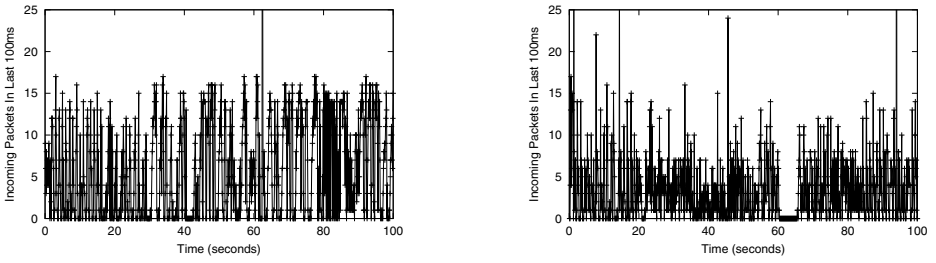
When using Tor, clients connect to $\mathcal{S}$ via proxies; therefore, clients' geographic location does not hint Eve on the server IP address that they will connect to (in case $\mathcal{S}$ has multiple physical servers, e.g., for load balancing). As a result, Eve must enumerate all the IP addresses of $\mathcal{S}$ during the attack.

For each of the n_s server addresses and for every exit relay that Eve tries, she performs 2^{16} iterations, trying a different port in each iteration; for each port she sends three packets that would cause the exit relay to send three duplicate Acks to the server, if a connection exists through that port. These packets can be short, with only one byte of data, i.e., 41 bytes long. Hence, the overall data that Eve sends to a particular exit relay, using a particular source IP of $\mathcal{S}$ in a single attack is $2^{16} \cdot 3 \cdot 41 < 7.7$MB. As shown in Figure 16, a small set of exit relays allows a good 'hit' rate. If Eve enumerates on all n_s possible server addresses and the most likely seven exit relays, then by our measurements the attack results in a 'hit' rate of about $\frac{1}{5}$ (see Figure 16); in this attack, she sends $53 \cdot n_s$ MB in each round. As noted at the end of the previous subsection, Eve's effort is divided on the number of clients (victims) that she probes.

5.3 Empirical Evaluation

Setup. We used the Tor network to evaluate the indirect rate reduction attack. To simplify the experiment and limit the effect on other Tor users, we performed the following measures: the restricted web-site server, a Linux machine (kernel version 2.6.35) which runs an Apache web-server (version 2.2.14), had only one IP address; furthermore, when running the attack, Eve was aware of the exit relay that is used and its port used for the connection. Given these three parameters, Eve only sends 3 packets of 41 bytes, i.e., 123 bytes, to carry out a single rate reduction iteration. Below, we describe the frequency of iterations and show that we send about 0.5 KB per second; we believe that this did not load the exit relay or caused damage for other Tor users. The client machine in our experiments runs Windows 7 and uses Tor (version 0.2.1.30) to connect to web-servers. While running our evaluation, we created Tor circuits using 12 different exit relays.

Evaluation. First, we observed the effect of the rate reduction attack (three duplicate Ack technique). To measure this effect, $\mathcal{C}$ connects to one of our servers through Tor; our server reports to Eve the IP address and port of the exit relay. Eve sends her packets only to the reported exit relay and only to the specific port used in the connection with our server. Eve performs three iterations of rate reduction every second, aiming to fake three congestion events and decrease the congestion window to about half of its size (in case of cubic variant). This implies that in every second, Eve sends 369 bytes to the exit relay. In Figure 18 we compare between the rate of packets that the client receives (as observed by Eve) on normal conditions and when Eve attacks the exit relay; our attack reduces the average rate.



(a) Normal Conditions (b) Under Attack

Fig. 18. Comparison between a rate of a TCP connection (via Tor) in normal conditions and when under rate reduction attack

We next tested the scenario considered in Section 2, i.e., of a client that connects through Tor to one of two sites. Eve uses rate reduction to test whether the client is communicating with the restricted site. We conducted the experiment as follows: the victim $\mathcal{C}$ connects to one of two servers in each time, each server is chosen with probability $\frac{1}{2}$. Regardless of the choice that the client makes, the 'restricted' server sends Eve an IP address and port, allegedly describing the exit relay connected to it. In case that the client does not connect to the restricted server, these values specify an arbitrary exit relay and port. Eve then employs the attack above, performing three rate reductions per second and sending a total of 369 Bps to the specified exit relay.

If client rate decreased by at least 20% during the last 30 seconds, then the client's score is incremented. The 20% threshold is motivated by the results in Figure 18, but may change in other scenarios, e.g., for a different server. This process is repeated; between iterations there is a 30 seconds suspension that allows the TCP connection between the server and exit relay to recover to its normal rate (in case the connection exists) and allows Eve to obtain a recent measurement of the average rate in $\mathcal{C}$'s connection. Eventually, Eve decides that $\mathcal{C}$ is communicating with the restricted site if $\mathcal{C}$ has more than half of the possible points. Figure 19 shows Eve's success rate as a function of the duration of the attack. In these experiments, the servers run TCP Cubic variant; an improvement in success rate is observed when server runs TCP Reno.

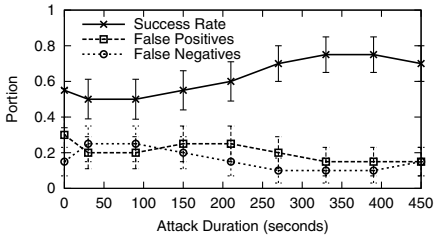


Fig. 19. Eve's success rate in detecting client access to a restricted site via Tor. Each measurement is the average of 20 runs. Server runs TCP cubic variant.

6 Defense Mechanisms

The countermeasures that we propose in this section do not completely eliminate the related side channel threat, however, they make it more difficult to exploit. These defenses are suitable for deployment on firewalls to ease deployment.

The globally incrementing IP identifier side channel, as mentioned in Section 3, is only relevant while still using IPv4. One way to avoid it is to use random IP-ID values; however, this can result in collisions and loss for fragmented traffic. The attack in Section 3 can be prevented by simply moving from globally-incrementing IP-ID to per-destination IP-ID; this would preferably be done by hosts, but until hosts do so[2], a firewall can implement this by adding (pseudo)random per-destination offset to the IP-ID. See analysis and better ways to fix the IP-ID in [15,17].

It is more challenging to block or reduce the timing side channels and cope with the rate reduction attack presented in Section 5. The flaw that we identify is that a blind adversary is able to cause a TCP recipient an involuntary reaction by sending arbitrary (spoofed) packets. We propose keeping a small window of acceptable sequence numbers that may be processed. This window resembles the receiver's congestion window, but is more aggressive: while packets outside the congestion window cause a duplicate acknowledgment (which we use in the attacks described in Sections 3-5), packets that specify sequence numbers outside the acceptable-window are silently discarded. The acceptable-window is larger than the host's congestion window and includes it. A congestion window is usually up to 2^{16} bytes, an acceptable-window that is twice as large, i.e., 2^{17} bytes, will significantly degrade the attacker's ability to conduct all the attacks in this paper. Since the sequence number is 32 bits long, the attacker is required to send $\frac{2^{32}}{2^{17}} = 2^{15}$ times the number of packets to conduct similar attacks. However, this technique requires that the firewall will inspect the sequence numbers in incoming TCP packets, which increases the packet processing overhead.

7 Conclusions and Future Work

Our primary conclusion is that TCP implementations leak information that allows attackers to study the existence of connections via side channels as we demonstrated in three attacks.

We leave several research directions for future work. Specifically, a more extensive empirical study is required to complete the evaluation of the Indirect Rate Reduction attack on the Tor network. Furthermore, it would be desirable to provide an analytic analysis for the attacks presented in this paper.

An important question is, can we perform a more efficient and more accurate attack on Tor anonymity by combining the indirect rate reduction attack presented in this paper with other existing attacks on Tor anonymity which exploit other attack vectors, e.g., [12,18,19,28].

[2] We informed Microsoft to the IP-ID issues, but we are not aware of intention to fix the IP-ID in Windows.

Acknowledgements. Thanks to Moti Geva, Amit Klein, Roger Dingledine and the anonymous referees for their comments and suggestions.

References

1. Tor Metrics Portal. Network and Usage Graphs (November 2011), http://metrics.torproject.org/graphs.html
2. Advanced Network Architecture Group. ANA Spoofer Project (2012), http://spoofer.csail.mit.edu/summary.php
3. Allman, M., Paxson, V., Blanton, E.: TCP Congestion Control. RFC 5681 (Draft Standard) (September 2009)
4. Baker, F., Savola, P.: Ingress Filtering for Multihomed Networks. RFC 3704 (Best Current Practice) (March 2004)
5. Bellovin, S.M.: A Technique for Counting Natted Hosts. In: Internet Measurement Workshop, pp. 267–272. ACM (2002)
6. Chakravarty, S., Stavrou, A., Keromytis, A.D.: Traffic Analysis against Low-Latency Anonymity Networks Using Available Bandwidth Estimation. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 249–267. Springer, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-15497-3
7. Danezis, G.: The Traffic Analysis of Continuous-Time Mixes. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 35–50. Springer, Heidelberg (2005)
8. Deering, S., Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Updated by RFCs 5095, 5722, 5871, 6437 (December 1998)
9. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Updated by RFCs 5746, 5878, 6176 (2008)
10. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The Second-Generation Onion Router. In: USENIX Security Symposium, pp. 303–320. USENIX (2004)
11. Ehrenkranz, T., Li, J.: On the State of IP Spoofing Defense. ACM Transactions on Internet Technology (TOIT) 9(2) (2009)
12. Evans, N.S., Dingledine, R., Grothoff, C.: A Practical Congestion Attack on Tor Using Long Paths. In: USENIX Security Symposium, pp. 33–50. USENIX Association (2009)
13. Felten, E.W., Schneider, M.A.: Timing Attacks on Web Privacy. In: Jajodia, S. (ed.) Proceedings of the 7th ACM Conference on Computer and Communications Security, Greece, pp. 25–32. ACM Press (November 2000)
14. Ferguson, P., Senie, D.: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827 (Best Current Practice), Updated by RFC 3704 (May 2000)
15. Gilad, Y., Herzberg, A.: Fragmentation Considered Vulnerable: Blindly Intercepting and Discarding Fragments. In: Proceedings of USENIX Workshop on Offensive Technologies (August 2011)
16. Gilad, Y., Herzberg, A.: Spying in the Dark: TCP and Tor Traffic Analysis - Technical Report (April 2012), http://u.cs.biu.ac.il/~herzbea/security/TR/TR12_02
17. Gont, F.: Security Assessment of the Internet Protocol Version 4. RFC 6274 (Informational) (July 2011)

18. Hintz, A.: Fingerprinting Websites Using Traffic Analysis. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 171–178. Springer, Heidelberg (2003)
19. Kadloor, S., Gong, X., Kiyavash, N., Tezcan, T., Borisov, N.: Low-Cost Side Channel Remote Traffic Analysis Attack in Packet Networks. In: ICC, pp. 1–5. IEEE (2010)
20. Kent, S., Seo, K.: Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard) (December 2005)
21. Killalea, T.: Recommended Internet Service Provider Security Services and Procedures. RFC 3013 (Best Current Practice) (November 2000)
22. Kocher, P.C.: Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
23. Larsen, M., Gont, F.: Recommendations for Transport-Protocol Port Randomization. RFC 6056 (Best Current Practice) (January 2011)
24. Levine, B.N., Reiter, M.K., Wang, C.-X., Wright, M.: Timing Attacks in Low-Latency Mix Systems. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 251–265. Springer, Heidelberg (2004)
25. Lyon, G.: Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning (2009), http://nmap.org/book/
26. Mittal, P., Khurshid, A., Juen, J., Caesar, M., Borisov, N.: Stealthy Traffic Analysis of Low-Latency Anonymous Communication Using Throughput Fingerprinting. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM Conference on Computer and Communications Security, pp. 215–226. ACM (2011)
27. Murdoch, S.J., Danezis, G.: Low-Cost Traffic Analysis of Tor. In: IEEE Symposium on Security and Privacy, pp. 183–195. IEEE Computer Society (2005)
28. Panchenko, A., Niessen, L., Zinnen, A., Engel, T.: Website Fingerprinting in Onion Routing Based Anonymization Networks. In: Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, WPES 2011, pp. 103–114. ACM, New York (2011)
29. Postel, J.: Transmission Control Protocol. RFC 793 (Standard), Updated by RFCs 1122, 3168, 6093, 6528 (September 1981)
30. Pries, R., Yu, W., Fu, X., Zhao, W.: A New Replay Attack Against Anonymous Communication Networks. In: IEEE International Conference on Communications (ICC), pp. 1578–1582 (2008)
31. Sanfilippo, S.: A New TCP Scan Method (1998), http://seclists.org/bugtraq/1998/Dec/79
32. Sanfilippo, S.: About the IP Header ID (December 1998), http://www.kyuzz.org/antirez/papers/ipid.html
33. Wikipedia. Usage Share of Operating Systems (2011), http://en.wikipedia.org/wiki/Usage_share_of_operating_systems
34. Zalewski, M.: Silence on the wire: a field guide to passive reconnaissance and indirect attacks. No Starch Press (2005)
35. Zander, S., Murdoch, S.J.: An Improved Clock-Skew Measurement Technique for Revealing Hidden Services. In: van Oorschot, P.C. (ed.) USENIX Security Symposium, pp. 211–226. USENIX Association (2008)
36. Zhu, Y., Fu, X., Graham, B., Bettati, R., Zhao, W.: On Flow Correlation Attacks and Countermeasures in Mix Networks. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 207–225. Springer, Heidelberg (2005)

Secure Distributed Framework for Achieving ϵ-Differential Privacy

Dima Alhadidi, Noman Mohammed, Benjamin C.M. Fung, and Mourad Debbabi

Concordia Institute for Information Systems Engineering, Concordia University, Montreal, Quebec, Canada
{dm_alhad,no_moham,fung,debbabi}@encs.concordia.ca

Abstract. Privacy-preserving data publishing addresses the problem of disclosing sensitive data when mining for useful information. Among the existing privacy models, ϵ-*differential privacy* provides one of the strongest privacy guarantees. In this paper, we address the problem of private data publishing where data is horizontally divided among two parties over the same set of attributes. In particular, we present the first generalization-based algorithm for differentially private data release for horizontally-partitioned data between two parties in the semi-honest adversary model. The generalization algorithm correctly releases differentially-private data and protects the privacy of each party according to the definition of secure multi-party computation. To achieve this, we first present a two-party protocol for the exponential mechanism. This protocol can be used as a subprotocol by any other algorithm that requires exponential mechanism in a distributed setting. Experimental results on real-life data suggest that the proposed algorithm can effectively preserve information for a data mining task.

1 Introduction

Data can be horizontally-partitioned among different parties over the same set of attributes.These distributed data can be integrated for making better decisions and providing high-quality services. However, data integration should be conducted in a way that no more information than necessary should be revealed between the participating entities. At the same time, new knowledge that results from the integration process should not be misused by adversaries to reveal sensitive information that has not been available before the data integration. In this paper, we propose an algorithm to securely integrate sensitive data , which is horizontally divided among two parties over the same set of attributes, whereby the integrated data still retains the essential information for supporting data mining tasks. The following scenario further motivates the problem.

Consider a blood bank collects and examines the blood provided from donors and then distributes the blood to different hospitals. Periodically, hospitals are required to submit the blood transfusion information, together with the patient surgery data, to the blood bank for classification analysis [1]. Due to privacy concerns and privacy regulations, hospitals cannot provide any information about

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 120–139, 2012.

Table 1. Data Set D_1

ID	Class	Job	Sex	Age	Surgery
1	N	Janitor	M	34	Transgender
2	Y	Lawyer	F	58	Plastic
3	Y	Mover	M	58	Urology
4	N	Lawyer	M	24	Vascular
5	Y	Mover	M	34	Transgender
6	Y	Janitor	M	44	Plastic
7	Y	Doctor	F	44	Vascular

Table 2. Data Set D_2

ID	Class	Job	Sex	Age	Surgery
8	N	Doctor	M	58	Plastic
9	Y	Doctor	M	24	Urology
10	Y	Janitor	F	63	Vascular
11	Y	Mover	F	63	Plastic

individual medical records to the blood bank. Accordingly, there is a desideratum for an approach that allows anonymizing horizontally-partitioned data from different providers for data release. The resulted anonymizing data should not contain individually identifiable information and at the same time the data providers should not reveal their private data or the ownership of the data to each other.

Example 1. Suppose the first hospital P_1 and the second hospital P_2 own the data sets D_1 and D_2 as shown in Table 1 and Table 2, respectively. Each hospital has records for different individuals. The attribute *Class* contains the label Y or N, representing whether or not the patient has received blood transfusion. Both parties want to integrate their data and use the integrated data to build a classifier on the *Class* attribute. After the integration, the sensitive data of the patient #5 can be uniquely identified since he is the only 34-year mover in the data set. Moreover, we can infer that a 34-year male has performed a transgender surgery since both patients in the integrated data set has performed it.

In this context, Jurczyk and Xiong [2] have proposed an algorithm to securely integrate horizontally-partitioned data from multiple data owners. Mohammed *et al.* [1] have proposed a distributed algorithm to integrate horizontally-partitioned high-dimensional health care data. Their methods [1,2] adopt k-anonymity [3,4] or its extensions [5,6] as the underlying privacy principle. Recently, Wong *et al.* [7] and Zhang *et al.* [8] have shown that algorithms, which satisfy k-anonymity [3,4] or its extensions [5,6], are vulnerable to minimality attack and do not provide the claimed privacy guarantee. Although several fixes against minimality attack have been proposed [9], new attacks such as composition attack [10] and deFinetti attack [11] have emerged against algorithms that adopt k-anonymity or its extensions.

In this respect, differential privacy [12], which is a recently proposed privacy model, provides provable privacy guarantee and it is, by definition, immune against all these attacks. A differentially-private mechanism ensures that the probability of any output (released data) is equally likely from all nearly identical input data sets and thus guarantees that all outputs are insensitive to any individual's data. In other words, an individual's privacy is not at risk because of the participation in the data set. In this paper, we present the first generalization-based algorithm for differentially-private data release for horizontally-partitioned data between two parties in the semi-honest adversary model. We take the single-party algorithm

Table 3. Related Work - Summary

Algorithms	Data Owner			Privacy Model	
	Single	Multi		Differential Privacy	Partition-based Privacy
		Horizontally	Vertically		
LeFevre *et al.* [15], Fung *et al.* [16], etc	✓				✓
Xiao *et al.* [17], Mohammed *et al.* [13], etc.	✓			✓	
Jurczyk and Xiong [2], Mohammed *et al.* [1]		✓			✓
Jiang and Clifton [18], Mohammed *et al.* [19]			✓		✓
Our proposal		✓		✓	

for differential privacy that has been recently proposed by Mohammed *et al.* [13] as a basis and extend it to the two-party setting. The main contribution of our paper can be summarized as follows:

- We present a two-party protocol for the exponential mechanism for horizontally-partitioned data. We use this protocol as a subprotocol of our main algorithm.
- We present the first non-interactive two-party data publishing algorithm for horizontally-partitioned data which achieves differential privacy and satisfies the security definition of secure multiparty computation (SMC). In a non-interactive framework, a database owner first anonymizes the raw data and then releases the anonymized version for data analysis. This approach is also known as privacy-preserving data publishing (PPDP) [14].
- We experimentally show that the proposed algorithm can preserve information for classification analysis..

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 overviews privacy and security models adopted in this paper.The two-party data publishing algorithm for horizontally-partitioned data is presented in Section 4. In Section 5, we describe the two-party protocol for the exponential mechanism. We discuss in Section 6 the correctness, the security and the efficiency of the two-party data publishing algorithm. Section 7 presents the experimental results, and estimates the computation and communication cost of the algorithm for a real data set. Section 8 answers some frequently raised questions. Finally, concluding remarks as well as a discussion of future work are presented in Section 9.

2 Related Work

The primary goal of our study in this paper is to share data. In contrast, privacy preserving distributed data mining (PPDDM) [20] allows sharing of the computed result (e.g., a classifier), but completely prohibits sharing data. In PPDDM, multiple data owners want to compute a function based on their inputs without sharing their data with others. This function can be as simple as a count query or as complex as a data mining task such as classification, clustering, etc. However, compared to data mining result sharing, data sharing gives greater flexibility because recipients can perform their required analysis and data exploration, and apply different modeling methods and parameters.

Our approach allows anonymizing data from different sources for data release without exposing the sensitive information. Jiang and Clifton [18] have proposed Distributed k-Anonymity (DkA) framework to securely integrate two data tables while satisfying k-anonymity requirement. Mohammed *et al.* [19] have proposed an efficient anonymization algorithm to integrate data from multiple data owners. Unlike the distributed anonymization problem for horizontally-partitioned data studied in this paper, these methods [18,19] propose algorithms for vertically-partitioned data. Jurczyk and Xiong [2] have proposed an algorithm to securely integrate horizontally-partitioned data from multiple data owners. Mohammed *et al.* [1] have proposed a distributed algorithm to integrate horizontally-partitioned high-dimensional health care data. To the best of our knowledge, these are the only two methods [1,2] that generate an anonymous table for horizontally-partitioned data. However, both the methods adopt k-anonymity [3,4] or its extensions [5,6] as the underlying privacy principle; therefore, are vulnerable to the recently discovered privacy attacks [7,10,11].

Differential privacy [12] has received considerable attention recently as a substitute for partition-based privacy models for PPDP . However, most of the research on differential privacy so far concentrates on the interactive [12,21] and non-interactive [13,17] setting for the single-party scenario. Therefore, these techniques do not address the problem of privacy-preserving data sharing for classification analysis; the primary theme of this paper. Finally, Dwork *et al.* [22] have proposed a distributed interactive protocol for computing a function while guaranteeing differential privacy. Given a function, each party first computes the function on its own data and then perturbs the result appropriately such that the summation of all the perturbed results from all the parties generates a differentially private output. As mentioned already, interactive approach does not allow data sharing and therefore does not address the problem studied in this paper.

3 Background

In this section, we first present an overview of differential privacy. Then, we briefly discuss the security definition in the semi-honest adversary model. Additionally, we overview the required cryptographic primitives for the proposed algorithm.

3.1 Privacy Model

Differential privacy is a recent privacy definition that provides a strong privacy guarantee. It guarantees that an adversary learns nothing more about an individual, regardless of whether her record is present or absent in the data.

Definition 1. (ϵ-Differential Privacy) *[12] A randomized algorithm Ag is differentially private if for all data sets D and D' where their symmetric difference contains at most one record (i.e., $|D \triangle D'| \leq 1$), and for all possible anonymized data sets $\hat{D}$,*

$$Pr[Ag(D) = \hat{D}] \leq e^{\epsilon} \times Pr[Ag(D') = \hat{D}], \tag{1}$$

where the probabilities are over the randomness of the Ag.

A standard mechanism to achieve differential privacy is to add random noise to the true output of a function. The noise is calibrated according to the *sensitivity* of the function. The sensitivity of a function is the maximum difference of its outputs from two data sets that differ only in one record.

Definition 2. (Sensitivity) *[12] For any function $f : D \rightarrow \mathbb{R}^d$, the sensitivity of f is*

$$\Delta f = \max_{D,D'} ||f(D) - f(D')||_1 \tag{2}$$

for all D, D' differing in at most one record.

For example, let f be the count function. The Δf is 1 because $f(D)$ can differ at most by 1 due to the addition or to the removal of a single record.

Dwork *et al.* [12] have proposed the Laplace mechanism. The mechanism takes a data set D, a function f, and the parameter λ that determines the magnitude of noise as inputs. It first computes the true output $f(D)$, and then perturbs the output by adding noise. The noise is generated according to a Laplace distribution with probability density function $\Pr(x|\lambda) = \frac{1}{2\lambda}\texttt{exp}(-|x|/\lambda)$; its variance is $2\lambda^2$ and its mean is 0. Laplace mechanism guarantees that perturbed output $f(\hat{D}) = f(D) + \texttt{Lap}(\Delta f/\epsilon)$ satisfies ϵ-differential privacy, where $\texttt{Lap}(\Delta f/\epsilon)$ is a random variable sampled from the Laplace distribution.

McSherry and Talwar [23] have proposed the exponential mechanism to achieve differential privacy whenever it makes no sense to add noise to outputs. The exponential mechanism can choose an output $t \in \mathcal{T}$ that is close to the optimum with respect to a utility function while preserving differential privacy. It takes as inputs a data set D, an output range $\mathcal{T}$, a privacy parameter ϵ, and a utility function $u : (D \times \mathcal{T}) \rightarrow \mathbb{R}$ that assigns a real valued score to every output $t \in \mathcal{T}$, where a higher score means better utility. The mechanism induces a probability distribution over the range $\mathcal{T}$ and then samples an output t. Let $\Delta u = \max_{\forall t,D,D'} |u(D,t) - u(D',t)|$ be the sensitivity of the utility function. The probability associated with each output is proportional to $\texttt{exp}(\frac{\epsilon u(D,t)}{2\Delta u})$; that is, the output with a higher score is exponentially more likely to be chosen.

3.2 Security Model

In this subsection, we briefly present the security definition in the semi-honest adversary model. Moreover, we overview the required cryptographic primitives.

Secure Multiparty Computation. In the semi-honest model, adversaries follow the protocol but may try to deduce additional information from the received messages. A protocol is private in a semi-honest environment if the view of each party during the execution of the protocol can be effectively simulated by a probabilistic polynomial-time algorithm knowing only the input and the output of that party [24]. Many of the protocols, as it is the case with the proposed algorithm in this paper, involve the composition of privacy-preserving subprotocols in which all intermediate outputs from one subprotocol are inputs to the

next subprotocol. These intermediate outputs are either simulated given the final output and the local input for each party or computed as random shares. Using the composition theorem [24], it can be shown that if each subprotocol is privacy-preserving, then the resulting composition is also privacy-preserving.

Cryptographic Primitives. The required cryptographic primitives utilized in this paper are:

- Yao's Protocol [26]. It is a constant-round protocol for secure computation of any probabilistic polynomial-time function in the semi-honest model. More specifically, assume that we have two parties P_1 and P_2 with their inputs x and y, respectively. Both parties want to compute the value of the function $f(x, y)$. Then, P_1 needs to send P_2 an encrypted circuit computing $f(x, .)$. The received circuit is encrypted and accordingly P_2 learns nothing from this step. Afterwards, P_2 computes the output $f(x, y)$ by decrypting the circuit. This can be achieved by having P_2 obtaining a series of keys corresponding to its input y from P_1 such that the function $f(x, y)$ can be computed given these keys and the encrypted circuit. However, P_2 must obtain these keys from P_1 without revealing any information about y. This is done by using oblivious transfer protocol [24].
- Random Value Protocol (RVP) [27]. It describes how two parties can share a value $R \in \mathbb{Z}_Q$ where R has been chosen uniformly at random and $Q \in \mathbb{Z}_N$ is not known by either party, but is shared between them. More specifically, P_1 has $R_1 \in \mathbb{Z}_N$ and P_2 has $R_2 \in \mathbb{Z}_N$ such that $R = R_1 + R_2 \ mod \ N \in [0, Q - 1]$ where N is the public key for the an additive homomorphic scheme.
- Oblivious Polynomial Evaluation (OPE) [28]. It is a protocol involving two parties, a sender whose input is a polynomial P, and a receiver whose input is a value α. At the end of the protocol, the receiver learns $P(\alpha)$ and the sender learns nothing.

4 Two-Party Differentially Private Data Release

In this section, we present our two-party algorithm for differentially-private data release for horizontally-partitioned data. To facilitate understanding the algorithm, we first present the notation that is used along this paper.

4.1 Notation and Preliminaries

Suppose two parties P_1 and P_2 own data table D_1 and D_2, respectively. Both the parties want to release an integrated anonymous data table $\hat{D}(A_1^{pr}, \ldots, A_d^{pr}, A^{cls})$ to the public for classification analysis. The attributes in D_1 and D_2 are classified into three categories: (1) An explicit identifier attribute A^i that explicitly identifies an individual, such as *SSN* and *Name*. These attributes are removed before releasing the data. (2) A class attribute A^{cls} that contains the class value, and the goal of the data miner is to build a classifier to accurately predict the value of this attribute. (3) A set of predictor attributes

Algorithm 1. Two-Party Algorithm

Input: Raw data set D_1, privacy budget ϵ, and number of specializations h
Output: Anonymized data set $\hat{D}$

Initialize D_g with one record containing top most values;
Initialize Cut_i to include the topmost value;
$\epsilon' \leftarrow \frac{\epsilon}{2(|A_n^{pr}|+2h)}$;
for $l = 1$ to h **do**
    Determine winner candidate w by $\texttt{DEM}(D_1, D_2, \cup Cut_i, \epsilon')$;
    Specialize w on D_g;
    Replace w with $child(w)$ in $\cup Cut_i$;
end for
for each leaf node of D_g **do**
    Compute the share C_1 of the true count C;
    Compute $X_1 = C_1 + \texttt{Lap}(2/\epsilon)$;
    Exchange X_1 with P_2 to compute $(C + 2 \times \texttt{Lap}(2/\epsilon))$;
end for
return Each leaf node with count $(C + 2 \times \texttt{Lap}(2/\epsilon))$

$\mathcal{A}^{pr} = \{A_1^{pr}, \ldots, A_d^{pr}\}$, whose values are used to predict the class attribute. Given a table D_1 owned by P_1, a table D_2 owned by P_2 and a privacy parameter ϵ, our objective is to generate an integrated anonymized data table $\hat{D}$ such that (1) $\hat{D}$ satisfies ϵ-differential privacy and (2) the algorithm to generate $\hat{D}$ satisfies the security definition of the semi-honest adversary model.

We require the class attribute to be categorical. However, the values of the predictor attribute can be either numerical v_n or categorical v_c. Further, we require that for each predictor attribute A^{pr}, which is either numerical or categorial, a taxonomy tree is provided. We assume that there is no trusted third party who computes the output table $\hat{D}$ and the parties are semi-honest. Moreover, we assume that the two data sets include disjoint tuples and are defined on exactly the same schema.

4.2 Anonymization Algorithm

In this section, we present our distributed differentially-private anonymization algorithm based on generalization for two parties as shown in Algorithm 1. Algorithm 1 is executed by the party P_1 (same for the party P_2). The algorithm first generalizes the raw data and then adds noise to achieve ϵ-differential privacy.

Generalizing the Raw Data. The general idea is to anonymize the raw data by a sequence of specializations starting from the topmost general state. A specialization, written $v \rightarrow child(v)$ replaces the parent value v with its set of child values $child(v)$. The specialization process can be viewed as pushing the "cut" of each taxonomy tree downwards. A *cut* of the taxonomy tree for an attribute A_i^{pr}, denoted by Cut_i, contains exactly one value on each root-to-leaf path. Each party keeps a copy of the current $\cup Cut_i$ and a generalized table D_g, in addition to the private table D_1 or D_2. Initially, all values in $\mathcal{A}^{pr}$ are generalized to the

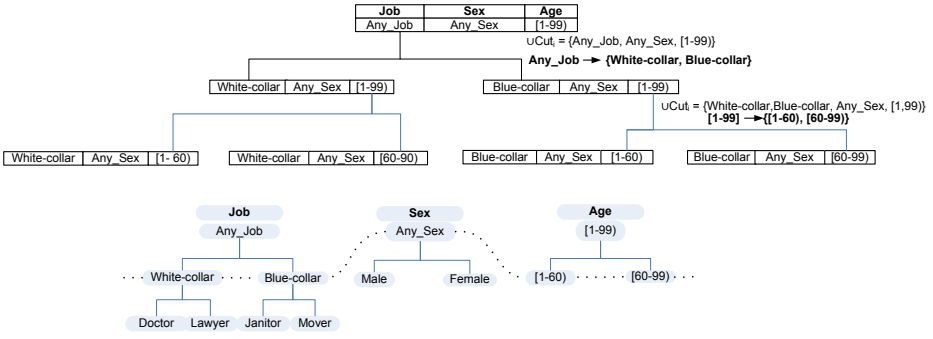


Fig. 1. Generalized Data Table (D_g)

topmost value in their taxonomy trees, and Cut_i contains the topmost value for each attribute A_i^{pr}. At each iteration, Algorithm 1 uses the distributed exponential mechanism to select a candidate for specialization (Line 5) depending on its score. This can be achieved by calling Algorithm 2 detailed in Section 5. Once a candidate is determined, both the parties specialize the winner candidate w on D_g (Line 6) by splitting their records into child partitions according to the provided taxonomy trees. Then, the parties update their local copy of $\cup Cut_i$ (Line 7). This process is repeated according to the number of specializations h.

Example 2. Consider Table 1 and Table 2 and the taxonomy trees presented at the bottom of Fig. 1. We do not show the class and the surgery attributes in Fig. 1 due to space limitation. Initially, D_g contains one root node representing all the records that are generalized to $\langle Any\_Job, Any\_Sex, [1\text{-}99)\rangle$. $\cup Cut_i$ is represented as $\{Any\_Job, Any\_Sex, [1\text{-}99)\}$ and includes the initial candidates. To find the winner candidate, both parties run DEM. Suppose that w is $Any\_Job \rightarrow \{White\text{-}collar, Blue\text{-}collar\}$. Both parties create two child nodes under the root node as shown in Fig. 1 and updates $\cup Cut_i$ to $\{White\text{-}collar, Blue\text{-}collar, Any\_Sex, [1\text{-}99)\}$. Suppose that the next winning candidate is $[1\text{-}99) \rightarrow \{[1\text{-}60), [60\text{-}99)\}$. Similarly, the two parties create further specialized partitions resulting the generalized table in Fig. 1.

Adding Noisy Count. Each party computes the number of its records under each leaf node (Line 10). To have an exchange between the parties that is differentially-private, each party adds a Laplace noise to its count (Line 11) and sends the result to the other party (Line 12). The protocol ends up with two Laplace noises added to the count of each leaf (Line 14).

5 Two-Party Protocol for Exponential Mechanism

Exponential mechanism chooses a candidate that is close to optimum with respect to a utility function while preserving differential privacy. In the

Table 4. `MAX` score calculation for the candidate *Any_Job*

Max	Class		Job	Data Set
	Y	N		
5	3	1	Blue-collar	D_1
	2	1	White-collar	
3	2	0	Blue-collar	D_2
	1	1	White-collar	
8	5	1	Blue-collar	Integrated D_1 and D_2
	3	2	White-collar	

distributed setting, the same candidates are owned by two parties while records are horizontally-partitioned among them. Consequently, we need a private mechanism to compute the same output while ensuring that no extra information is leaked to any party. In this section, we present a two-party protocol for exponential mechanism in a distributed setting. We adopt the max utility function to compute the scores. For this reason, we illustrate first how this function is computed. Other utility functions can be adopted as discussed in Section 8.

5.1 Max Utility Function

To compute the score of each candidate, we adopt the max utility function [13].

$$\texttt{Max}(D, v) = \sum_{a \in child(v)} \max_{c}(|T_D(a, c)|) \tag{3}$$

where the notation T denotes the set of transactions (records) and $|T_D(a, c)|$ denotes the number of records in D having the generalized value a and the class value c. Thus, $\texttt{Max}(D, v)$ is the summation of the highest class frequencies over all child values. The sensitivity $\Delta\texttt{u}$ of the `Max` function is 1 because the value $\texttt{Max}(D, v)$ can vary at most by 1 due to a record change. The following example clarifies how to evaluate the max utility function.

Example 3. The maximum utility function of the candidate *Any_Job* of Table 1 is 5. Table 4 demonstrates how the value 5 is computed. For each possible child value of the candidate *Any_Job*, we compute the number of records having the class value Y and the class value N. Afterwards, we pick the maximum class frequency for each child value and sum them. In the same vein, the maximum utility function of the the candidate *Any_Job* of Table 2 is 3. If we integrate the two tables, the maximum utility function of the candidate *Any_Job* is 8. Note that, the maximum utility function of an integrated table is not the sum of the values of maximum utility function of each source data set.

5.2 Distributed Exponential Mechanism

The *distributed exponential mechanism (DEM)* presented in Algorithm 2 takes the followings as inputs:(1) Two raw data sets D_1 and D_2 owned by P_1 and P_2, respectively, (2) set of candidates $\{v_1, \ldots, v_k\}$, and (3) privacy budget ϵ. The

Algorithm 2. Distributed Exponential Mechanism (DEM)

Input: Raw data set D_1 owned by P_1, raw data set D_2 owned by P_2, a set of candidates $\{v_1, \ldots, v_k\}$ and privacy budget ϵ

Output: Winner w

1: **for** each candidate v_x where $x = 1$ to k **do**
2: **for** (each possible value of a_j of v_x where $j = 1$ to m) **do**
3: **for** (each class value c_i where $i = 1$ to l) **do**
4: P_1 computes $|T_{D_1}(a_j, c_i)|$;
5: P_1 computes $|T_{D_2}(a_j, c_i)|$;
6: **end for**
7: **end for**
8: P_2 generates a random share α_2;
9: $(P_1 \leftarrow \alpha_1, P_2 \leftarrow \perp) \leftarrow \texttt{MAX}(|T_{D_1}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}, |T_{D_2}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}, \alpha_2)$;
10: P_1 chooses a random share β_x and defines the following polynomial $Q(z) = lcm(2!, \ldots, w!)\ .10^{sw}.\ \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta u})_s .10^s .(\alpha_1 + z))^i}{10^{s(i-1)} .i!} - \beta_x$;
11: P_1 and P_2 execute a private polynomial with P_1 inputting $Q(.)$ and P_2 inputting α_2, in which P_2 obtains $\beta'_x = Q(\alpha_2)$.
12: **end for**
13: $(P_1 \leftarrow \gamma_1, P_2 \leftarrow \perp) \leftarrow \texttt{SUM}(\beta_{x,x=1\ to\ k}, \beta'_{x,x=1\ to\ k}, \gamma_2)$;
14: P_1 and P_2 execute RVP to compute random shares R_1 and R_2, where $(R_1 + R_2) \in \mathbb{Z}_{(\gamma_1+\gamma_2)}$;
15: P_1 and P_2 evaluates $x \leftarrow \texttt{COMPARISON}(R_1, R_2, \beta_{x,x=1\ to\ k}, \beta'_{x,x=1\ to\ k})$;
16: **return** v_x;

protocol outputs a winner candidate depending on its score using the exponential mechanism. The scores of the candidates can be calculated using different utility functions [13]. In this paper, we adopt the max utility function described previously to calculate the scores. Given the scores of all the candidates, exponential mechanism selects the candidate v having score u with the following probability where Δu is the sensitivity of the chosen utility function.

$$\frac{\texttt{exp}(\frac{\epsilon u}{2\Delta \texttt{u}})}{\sum_{n=1}^{k} \texttt{exp}(\frac{\epsilon u_n}{2\Delta \texttt{u}})} \tag{4}$$

Next, we detail the steps of the distributed exponential mechanism (DEM).

Computing Max Utility Function. To compute the max utility function for each candidate v_x, the parties P_1 and P_2 compute $|T_{D_1}(a_j, c_i)|$ and $|T_{D_1}(a_j, c_i)|$, respectively for every possible value a_j of v_x and for every possible value c_i of the class attribute (Lines 2 to 7). After that, the two parties engage in a secure circuit evaluation process using Yao's Protocol (Line 9). The values $|T_{D_1}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$, $|T_{D_2}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$ and α_2 are passed to the `MAX` circuit where α_2 is randomly generated by P_2. For each child value a_j of the candidate v_x, the circuit `MAX`, as shown in Algorithm 3, adds the corresponding values $|T_{D_1}(a_j, c_i)|$ and $|T_{D_2}(a_j, c_i)|$ for every possible value c_i of the

Algorithm 3. `MAX` Circuit

Input: $|T_{D_1}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$, $|T_{D_2}(a_j, c_i)|_{i=1\ to\ l, j=1\ to\ m}$ and α_2
Output: α_1 to P_1,$\perp$ to P_2

```
1: sum = 0;
2: for j = 1 to m do
3:    max = 0;
4:    for i = 1 to l do
5:       ss= |T_D1(a_j, c_i)| + |T_D2(a_j, c_i)|;
6:       if (ss > max) then
7:          max = ss;
8:       end if
9:    end for
10:   sum = sum + max;
11: end for
12: α1 = sum - α2;
13: return α1,⊥;
```

Algorithm 4. `COMPARISON` Circuit

Input: Random shares R_1 and R_2, $\beta_{x,x=1\ to\ k}$, and $\beta'_{x,x=1\ to\ k}$
Output: Index x to P_1 and P_2

```
1: L = 0;
2: R = R1 + R2;
3: for x = 1 to k do
4:    β = β_x + β'_x;
5:    L = L + β;
6:    if (R ≤ L) then
7:       return x;
8:    end if
9: end for
```

class attribute. It then computes the maximum value of the results. After that, the maximum values associated with each child value a_j should be summed to get the max utility function for the candidate v_x. To produce random shares of the max utility function, the circuit subtracts α_2, which is randomly generated by P_2, from the resulted score and outputs the result α_1 to P_1.

Computing Equation 4. The exponential function, $exp(x)$ can be defined using the following Taylor series:

$$1 + \frac{x}{1} + \frac{x^2}{2!} + \cdots + \frac{x^i}{i!} + \ldots \tag{5}$$

To evaluate the nominator of Equation 4 for each v_x, we need to evaluate the expression $\mathtt{exp}(\frac{\epsilon u}{2\Delta \mathtt{u}})$ which is equal to $\mathtt{exp}(\frac{\epsilon(\alpha_1+\alpha_2)}{2\Delta \mathtt{u}})$. Given the aforementioned

Taylor series:

$$\texttt{exp}(\frac{\epsilon(\alpha_1+\alpha_2)}{2\Delta \texttt{u}}) = \sum_{i=0}^{w} \frac{(\frac{\epsilon(\alpha_1+\alpha_2)}{2\Delta \texttt{u}})^i}{i!} \tag{6}$$

Hence, the next step involves computing shares of the Taylor series approximation. In fact, it computes shares of:

$$lcm(2!,\ldots,w!)\ .10^{s(w+1)}.\ \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta \texttt{u}})_s.(\alpha_1+\alpha_2))^i}{i!}$$

where:

- $lcm(2!,\ldots,w!)$ is the lowest common multiple of {2!,. . . ,w!} and we multiply by it to ensure that there are no fractions.
- $(\frac{\epsilon}{2\Delta \texttt{u}})_s$ refers to approximating the value of $\frac{\epsilon}{2\Delta \texttt{u}}$ up to a predetermined number s after the decimal point. For example, if we assume $s = 4$ and $\epsilon = ln2$ then $(\frac{ln2}{2\times 1})_4 = (0.3465)$. Note that, this approximation does not effect privacy guarantee since we are using less privacy budget. Also, the impact on the utility is insignificant. In Section 7, we experimentally show the accuracy for different privacy budgets.
- $10^{sw}.10^s$ is multiplied by the series to ensure that we end up with an integer result such that:

$$\begin{aligned} & lcm(2!,\ldots,w!)\ .10^{sw}.10^s.\ \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta \texttt{u}})_s.(\alpha_1+\alpha_2))^i}{i!} \\ &= lcm(2!,\ldots,w!)\ .10^{sw}.\ \sum_{i=0}^{w} 10^s.\frac{((\frac{\epsilon}{2\Delta \texttt{u}})_s.(\alpha_1+\alpha_2))^i}{i!} \\ &= lcm(2!,\ldots,w!)\ .10^{sw}.\ \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta \texttt{u}})_s.10^s.(\alpha_1+\alpha_2))^i}{10^{s(i-1)}.i!} \end{aligned}$$

Since s and w are known to both parties, the additional multiplicative factor $lcm(2!,\ldots,w!).10^{sw}.10^s$ is public and can be removed at the end (if desired). This equation is accurate up to an approximation error which depends on the value of w. Therefore, scaling is needed and consequently the accuracy of the exponential mechanism could be affected. However, if the scaling factor is very large, the total cost in terms of bits will increase. We experimentally measure the impact of scaling in Section 7 and show that the scaling has very little impact for the max utility function. The parties should agree on the number of the considered digits s after the decimal point. The higher accuracy (in terms of the number of the considered digits after the decimal point) we demand, the higher cost we pay (in terms of bits). That is for s decimal points, we need $log_2 10^s$ extra bits. These extra bits result additional computation and communication cost. More details are provided in Section 7. Note that restricting the values of

$\texttt{exp}(\frac{\epsilon u}{2\Delta \texttt{u}})$ to a finite range is completely natural as calculations performed on computers are handled in this manner due to memory constraints.

To evaluate the nominator of Equation 4 for each v_x in Algorithm 2, P_1 chooses a random share β_x and defines the following polynomial where s is a constant number (Line 10):

$$Q(z) = lcm(2!, \ldots, w!) \ .10^{sw}. \ \sum_{i=0}^{w} \frac{((\frac{\epsilon}{2\Delta \texttt{u}})_s.10^s.(\alpha_1 + z))^i}{10^{s(i-1)}.i!} - \beta_x$$

Afterwards, P_1 and P_2 execute a private polynomial with P_1 inputting $Q(.)$ and P_2 inputting α_2, in which P_2 obtains $\beta'_x = Q(\alpha_2)$ (Line 11). To evaluate the denominator of Equation 4, the two parties execute the circuit SUM which takes as input the random shares β_x and β'_x for each candidate v_x and a random number γ_2 generated by P_2 (Line 13). The circuit computes the total sum of the results that come out because of adding the random shares β_x and β'_x for each candidate v_x. It then subtracts γ_2, which is randomly generated by P_2, from the value of the total sum and outputs the share γ_1 to P_1.

Once we compute the denominator and numerator of Equation 4, we can implement the exponential mechanism by first partitioning the interval [0,1] into segments according to the corresponding probability mass of each candidate. Next, we sample a random number uniformly in the range [0,1] and the partition in which the random number falls determines the winner candidate. However, this method involves computing a secure division (Equation 4). Unfortunately, we are not aware of any secure division scheme that fits our scenario where the nominator value is less than the denominator value. Alternatively, we solve this problem without a secure division protocol. We first partition the interval $[0, \sum_{x=1}^{k} \texttt{exp}(\frac{\epsilon u_x}{2\Delta \texttt{u}})]$ into k segments where $\sum_{x=1}^{k} \texttt{exp}(\frac{\epsilon u_x}{2\Delta \texttt{u}}) \approx \gamma_1 + \gamma_2$ and each segment corresponds to a candidate v_x has a subinterval of length equal to $\beta_x + \beta'_x$. We then sample a random number uniformly in the range $[0, \gamma_1 + \gamma_2]$ and the segment in which the random number falls determines the winner candidate.

Picking a Random Number. The parties P_1 and P_2 need to pick a random number uniformly in the range $[0, \gamma_1 + \gamma_2]$, where $\gamma_1 + \gamma_2 \approx \sum_{x=1}^{k} \texttt{exp}(\frac{\epsilon u_x}{2\Delta \texttt{u}})$. This can be achieved by using the Random Value Protocol (RVP) [27] (Line 14). RVP takes γ_1 and γ_2 from the parties as input and outputs the random value shares R_1 and R_2 to the respective parties, where $R = R_1 + R_2$.

Example 4. Suppose the values of the expression $\texttt{exp}(\frac{\epsilon u}{2\Delta \texttt{u}})$ is approximated to 60, 150 and 90 for three candidates as shares. Both parties then pick a random number in the range $[0, 300]$ using the RVP where $300 = 50 + 150 + 90$.

Picking a Winner. The two parties engage again in a simple secure circuit evaluation process using Yao's Protocol [26] (Line 15). The circuit COMPARISON compares their random number R with the sum L. The winner v_x is the first candidate such that $R \leq L$ where $L = \sum_{r=1}^{x} (\beta_x + \beta'_x)$.

6 Analysis

We discuss in this section the correctness, security and efficiency of Algorithm 1.

Proposition 1. (Correctness) *Assuming both parties are semi-honest, Algorithm 1 releases ϵ-differentially private data when data records are divided horizontally among two parties over the same set of attributes.*

Proof. Algorithm 1 performs the same function as the single-party algorithm DiffGen [13] but in a distributed setting. DiffGen is ϵ-differentially private. Therefore, we prove the correctness of Algorithm 1 by just proving the steps that are different from DiffGen:

- Candidate selection. Algorithm 1 selects a candidate for specialization (Line 5) using Algorithm 2. Algorithm 2 selects a candidate v_w with probability $\propto \texttt{exp}(\frac{\epsilon u_w}{2\Delta \texttt{u}})$. The two parties compute cooperatively $\texttt{exp}(\frac{\epsilon u}{2\Delta \texttt{u}})$ for the candidates. Then the parties build an interval in the range $[0,\sum_{x=1}^{k} \texttt{exp}(\frac{\epsilon u_x}{2\Delta \texttt{u}})]$ and partition it among the candidates where each subinterval has a length equal to $\texttt{exp}(\frac{\epsilon u}{2\Delta \texttt{u}})$. Since, the random value lies uniformly between $[0,\sum_{x=1}^{k} \texttt{exp}(\frac{\epsilon u_x}{2\Delta \texttt{u}})]$ and a candidate is chosen according to this value, the probability of choosing any candidate is $\frac{\texttt{exp}(\frac{\epsilon u}{2\Delta \texttt{u}})}{\sum_{x=1}^{k} \texttt{exp}(\frac{\epsilon u_x}{2\Delta \texttt{u}})}$. Therefore, Algorithm 2 correctly implements exponential mechanism.
- Updating the tree D_g and $\cup Cut_i$. Each party has its own copy of D_g and $\cup Cut_i$. Each party updates these items exactly like DiffGen (Lines 6-7).
- Computing the noisy count. Algorithm 1 also outputs the noisy count of each leaf node (Line 14), where the noise is equal to $2 \times Lap(2/\epsilon)$. Thus, it guarantees $\frac{\epsilon}{2}$-differential privacy.

Since Algorithm 1 performs exactly the same sequence of operations as the single-party algorithm in a distributed setting where D_1 and D_2 are kept locally, it is also ϵ-differentially private. □

Proposition 2. (Security) *Algorithm 1 is secure under the semi-honest adversary model.*

Proof. The security of Algorithm 1 depends on the following steps where the parties exchange information:

- Algorithm *DEM* (Line 5): The privacy proof of DEM is as follows:
  - Circuit `MAX`: It can be evaluated securely [24]. Parties input their local counts $|T(a_j, c_i)|$ and receive the random share of the MAX value.
  - Oblivious Polynomial Evaluation: It has been proven to be secure [28].
  - Random Value Protocol (RVP): It has proven to be secure [27].
  - Circuits `SUM` and `COMPARISON`: Similarly, these circuits can be evaluated securely [24].

  Since, all the above protocols produce random shares and proved to be secure, DEM is also secure due to the composition theorem [24].

– *Exchanging noisy counts (Line 12)*: Each party initially adds Laplace noise to its local count and then exchange the noisy count with the other party. This does not violate differential privacy because the noisy count is already private according to Laplace mechanism [12].

Therefore, due to Composition Theorem [24], Algorithm 1 is secure. □

Proposition 3. (Complexity) *The encryption and the communication costs of Algorithm 1 are bounded by* $O(hk \log R)$ *and* $O(hk \log RK)$*, respectively.*

Proof. Distributed exponential mechanism (Algorithm 2) dominates the overall complexity of Algorithm 1. The complexity of DEM is computed as follows:

– Circuit `MAX`: This circuit is composed of simple *add* and *compare* operations and thus can be implemented by the number of gates linear to the input size of the circuit. The input includes $m \times l$ local counts $|T(a_j, c_i)|$ and these values are of size at most $\log |D|$. Hence, the encryption and the communication complexity of `MAX` are bounded by $O(ml \log |D|)$ and $O(ml \log |D|K)$, respectively, where K is the length of the key for a pseudorandom function [29]. The `MAX` protocol is called at most k times. Therefore, the encryption and the communication costs are $O(kml \log |D|)$ and $O(kml \log |D|K)$, respectively.
– Oblivious Polynomial Evaluation: This protocol involves the private evaluation of a polynomial of degree w. Thus, the encryption and the communication complexity are bounded by $O(w)$ and $O(we)$, where e is the length of an encrypted element [30]. This protocol is also called k times. Therefore, the encryption and the communication cost are $O(kw)$ and $O(kwe)$, respectively.
– Random Value Protocol (RVP): The costs of RVP are negligible and therefore they are ignored.
– Circuit `SUM` and `COMPARISON`: The analysis is similar to `MAX` circuit. The encryption and the communication complexity of both the circuits are bounded by $O(k \log R)$ and $O(k \log RK)$, where $R = \left\lfloor \exp(\frac{\epsilon' u_x}{2\Delta u}) \times 10^s \right\rfloor$.

Both the parties execute DEM (Algorithm 2) h times to select the winner candidates. Note that Lines 1-12 of Algorithm 2 are not executed in every iteration. Rather, these lines are only invoked once for each candidate. Hence, the overall encryption and communication costs are $O(\max\{kml \log |D|, kw, hk \log R\})$ and $O(\max\{kml \log |D|K, kwe, hk \log RK\})$, respectively. Since the value of R is usually very large, the encryption and communication costs can be defined as $O(hk \log R)$ and $O(hk \log RK)$, respectively. □

7 Performance Analysis

In this section, we evaluate the scaling impact on the data quality in terms of classification accuracy. Moreover, we estimate the computation and the communication costs of Algorithm 1. We employ the publicly available data set *Adult*; a real-life census data set that has been used for testing many anonymization

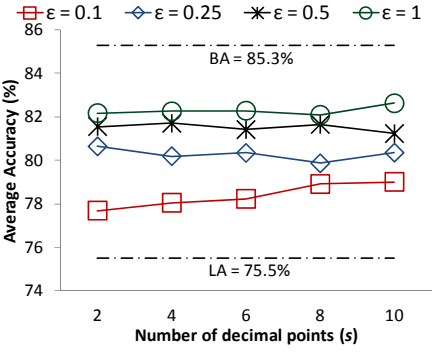


Fig. 2. Classification Accuracy for Different Scaling

algorithms [18,5,6]. It has 45, 222 census records with 6 numerical attributes, 8 categorical attributes, and a binary class column representing two income levels, $\leq$50K or >50K. All experiments are conducted on an Intel Core $i7$ 2.7GHz PC with 12GB RAM.

7.1 Experiments

To evaluate the impact on classification quality, we divide the data into training and testing sets. First, we apply our algorithm to anonymize the training set and to determine the $\cup Cut_i$. Then, the same $\cup Cut_i$ is applied to the testing set to produce a generalized testing set. Next, we build a classifier on the anonymized training set and measure the Classification Accuracy (CA) on the generalized records of the testing set. For classification models, we use the well-known C4.5 classifier [31]. To better visualize the cost and benefit of our approach, we provide additional measures: (1) Baseline Accuracy (BA) is the classification accuracy measured on the raw data without anonymization; (2) BA - CA represents the cost in terms of classification quality for achieving a given ϵ-differential privacy requirement; (3) Lower bound Accuracy (LA) is the accuracy on the raw data with all attributes (except for the *class* attribute) removed and (4) CA - LA represents the benefit of our method over the naive non-disclosure approach.

Fig. 2 depicts the classification accuracy CA for the utility function `Max` where the privacy budget $\epsilon \in \{0.1, 0.25, 0.5, 1\}$ and the number of considered digits after the decimal point $2 \leq s \leq 10$ (i.e., scaling as described in Section 5). The BA and LA are 85.3% and 75.5%, respectively, as shown in the figure by the dotted lines. We use 2/3 of the records (i.e., 30,162) to build the classifier and measure the accuracy on the remaining 1/3 of the records (i.e., 15060). For each experiment, we execute 10 runs and average the results over the runs. The number of specializations h is 10 for all the experiments. For $\epsilon = 1$ and $s = 10$, BA - CA is around 2.6% whereas CA - LA is 7.1%. For $\epsilon = 0.5$, BA - CA spans from 3.6% to 4%, whereas CA - LA spans from 5.7% to 6.2%. However, as ϵ decreases to 0.1, CA quickly decreases to about 79% (highest point), the

cost increases to about 6.5%, and the benefit decreases to about 3.3%. The experimental result demonstrates that the classification accuracy is insensitive to the scaling (the number of considered digits after the decimal points) for the `Max` function. This is because the value of $\texttt{exp}(\frac{\epsilon'}{2\Delta \texttt{u}}\texttt{u})$ is large due to the score of the `Max` function which is usually a large integer. Therefore, scaling has hardly any impact on the data utility.

7.2 Cost Estimates

Most of the computation and the communication of Algorithm 1 take place during the execution of the DEM (Line 5). The runtime of the other steps is less than 30 seconds for *Adult* dataset. Hence, we only elaborate the runtime of the DEM. As discussed in Section 6, the computation and the communication complexity of the distributed exponential mechanism are dominated by the cost of the `SUM` (Line 13) and `COMPARISON` (Line 15) circuits. In the following, we provide an estimate for the computation and the communication costs of evaluating the `SUM` and `COMPARISON` circuit. Here, we assume that P_1 encodes and P_2 evaluates the encrypted circuit. The roles of P_1 and P_2 can be swapped.

Computation. The cost of an encryption is denoted by C_m which is 0.02 second for 1024-bit numbers on a Pentium III processor [28]. For both the circuits, P_2 needs to execute a 1-out-of-2 oblivious transfer protocol to get the corresponding encryption key for its input bits. This is the major computational overhead of the distributed exponential mechanism. The computation cost of an oblivious transfer protocol is roughly equal to the cost of a modular exponentiation, which is C_m. Therefore, the computation overhead is equal to the number of input bits of P_2 times C_m. Each input of the circuit is bounded by $\lceil \log_2 R \rceil$ bits, where $R = \left\lfloor \texttt{exp}(\frac{\epsilon'}{2\Delta \texttt{u}} u(D, v_i)) \times 10^s \right\rfloor$.

$$\begin{aligned}
\lceil \log_2 R \rceil &= \left\lceil \log_2 \left(\left\lfloor \texttt{exp}(\frac{\epsilon'}{2\Delta \texttt{u}}) \times 10^s \right\rfloor \right) \right\rceil \\
&= \left\lceil \frac{\frac{\epsilon'}{2\Delta \texttt{u}}}{\ln 2} + \log_2 10^s \right\rceil \\
&= \left\lceil \frac{\frac{\epsilon'}{2\Delta \texttt{u}}}{\ln 2} + (3.3219 \times s) \right\rceil
\end{aligned}$$

Here, $\Delta \texttt{u} = 1$, $\epsilon' = \frac{1}{2(6+2\times 10)} = 0.02$, $u(D, v_i)$ is bounded by the number of the records $|D| = 30162$ for `Max` function, and $s = 10$ suffices the desired accuracy. Hence, we have $\lceil \log_2 R \rceil = 469$ bits. The input size of P_2 is $O(k \log R)$ bits, where the constant is fairly small. Here, k is the total number of candidates which is 24 at most for *Adult* data set. Thus, the cost is $k \times \lceil \log_2 R \rceil \times C_m$

$= 24 \times 469 \times 0.02s \approx 225$ seconds. As mentioned in Section 6, there are at most h invocations of these circuits. Here, h is the number of specializations which is set to 10. Hence, the total computational cost is $h \times 225 \approx 37.5$ mins .

Communication. P_1 needs to send a table of size $4K$ for each gate of the `SUM` and `COMPARISON` circuit, where we assume the key size K is 128 bits. This is the major communication overhead of the distributed exponential mechanism. Since these circuits only use addition and comparison operations, the total number of gates needed to implement these circuits are $O(k \log R)$. Thus, the number of gates, $T_g \approx 24 \times 469 = 11256$. Therefore, the communication cost of sending the tables is $h \times 4K \times T_g \approx 5.76 \times 10^7$ bits, which takes approximately 37.3 seconds using a T1 line with 1.544 Mbits/second bandwidth.

Remark. Our estimation ignores the computational cost of evaluating the circuit and the communication cost of the oblivious transfer protocol. The evaluation of the circuit involves decrypting a constant number of ciphertexts (symmetric encryption) for every gate which is very efficient compared to oblivious transfer (modular exponentiations) since the number of gates of the circuit is linear to the number of input bits. Also, the communication cost of the obvious transfer protocol is negligible compared to the cost of sending the tables.

8 Discussion

Is differential privacy good enough? What changes are required if there are more than two parties? Can the algorithm be easily adapted to accommodate a different utility function? In this section, we provide answers to these questions.

Differential privacy. Differential privacy is a strong privacy definition. However, Kifer and Machanavajjhala [32] have shown that if the records are not independent or an adversary has access to aggregate level background knowledge about the data, then privacy attack is possible. In our application scenario, each record is independent of each other and we assume that no deterministic statistics of the raw database have ever been released. Hence, differential privacy is appropriate for our problem.

More than two parties. The proposed algorithm is only applicable for the two-party scenario because the distributed exponential algorithm, and the other primitives (e.g., random value protocol) are limited to two-party scenario. The proposed algorithm can be extended for more than two parties by modifying all the subprotocols while keeping the general top-down structure of the algorithm as it is.

Other Utility functions. For each new utility function, we only need to devise an algorithm to calculate the utility function. Hence, we only have to change Algorithm 3 to adapt our approach for other utility function.

9 Conclusion

In this paper, we have presented a two-party differentially-private data release algorithm for horizontally-partitioned data for the non-interactive setting. We have shown that the proposed algorithm is differentially private and secure under the security definition of semi-honest adversary model. Moreover, we have experimentally evaluated the data utility of the algorithm. An intersecting research direction, as a future work, is devising different heuristics for different data mining tasks.

Acknowledgments. We sincerely thank the reviewers for their valuable comments. The research described in this paper is part of the project in cloud computing security and privacy with Ericsson Canada and Alcatel Lucent, funded by an NSERC Strategic Grant and NSERC Canada Graduate Scholarships.

References

1. Mohammed, N., Fung, B.C.M., Hung, P.C.K., Lee, C.: Centralized and distributed anonymization for high-dimensional healthcare data. ACM Transactions on Knowledge Discovery from Data (TKDD) 4(4), 18:1–18:33 (2010)
2. Jurczyk, P., Xiong, L.: Distributed anonymization: Achieving privacy for both data subjects and data providers. In: Proceedings of the Annual IFIP WG 11.3 Working Conference on Data and Applications Security, DBSec (2009)
3. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Transaction on Knowledge and Data Engineering (TKDE) (2001)
4. Sweeney, L.: k-anonymity: A model for protecting privacy. International Journal on Uncertainty, Fuzziness and Knowledge-based Systems (2002)
5. Machanavajjhala, A., Kifer, D., Gehrke, J., Venkitasubramaniam, M.: ℓ-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD) (2007)
6. Wang, K., Fung, B.C.M., Yu, P.S.: Handicapping attacker's confidence: An alternative to k-anonymization. Knowledge and Information Systems (KAIS) 11(3), 345–368 (2007)
7. Wong, R.C.W., Fu, A.W.C., Wang, K., Pei, J.: Minimality attack in privacy preserving data publishing. In: Proceedings of the International Conference on Very Large Data Bases (VLDB) (2007)
8. Zhang, L., Jajodia, S., Brodsky, A.: Information disclosure under realistic assumptions: Privacy versus optimality. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS) (2007)
9. Cormode, G., Srivastava, D., Li, N., Li, T.: Minimizing minimality and maximizing utility: Analyzing methodbased attacks on anonymized data. In: Proceedings of the International Conference on Very Large Data Bases (VLDB) (2010)
10. Ganta, S.R., Kasiviswanathan, S., Smith, A.: Composition attacks and auxiliary information in data privacy. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) (2008)
11. Kifer, D.: Attacks on privacy and de finetti's theorem. In: Proceedings of the ACM Conference on Management of Data (SIGMOD) (2009)

12. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
13. Mohammed, N., Chen, R., Fung, B.C.M., Yu, P.S.: Differentially private data release for data mining. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) (2011)
14. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey of recent developments. ACM Computing Surveys 42(4), 1–53 (2010)
15. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Mondrian multidimensional k-anonymity. In: Proceedings of the IEEE International Conference on Data Engineering (ICDE) (2006)
16. Fung, B.C.M., Wang, K., Yu, P.S.: Anonymizing classification data for privacy preservation. IEEE Transactions on Knowledge and Data Engineering (TKDE) 19(5), 711–725 (2007)
17. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. In: Proceedings of the International Conference on Data Engineering (ICDE) (March 2010)
18. Jiang, W., Clifton, C.: A secure distributed framework for achieving k-anonymity. Very Large Data Bases Journal (VLDBJ) 15(4), 316–333 (2006)
19. Mohammed, N., Fung, B.C.M., Debbabi, M.: Anonymity meets game theory: secure data integration with malicious participants. Very Large Data Bases Journal (VLDBJ) 20(4), 567–588 (2011)
20. Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.Y.: Tools for privacy preserving distributed data mining. ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) Explorations Newsletter 4(2), 28–34 (2002)
21. Roth, A., Roughgarden, T.: Interactive privacy via the median mechanism. In: Proceedings of the ACM Symposium on Theory of Computing (STOC) (2010)
22. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our Data, Ourselves: Privacy Via Distributed Noise Generation. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 486–503. Springer, Heidelberg (2006)
23. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: Proceedings of the IEEE Symposium on Foundations of Computer Science (2007)
24. Goldreich, O.: Foundations of Cryptography, vol. 2. Cambridge University Press (2001)
25. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
26. Yao, A.C.: Protocols for secure computations. In: Proc. of the IEEE Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS) (1982)
27. Bunn, P., Ostrovsky, R.: Secure two-party k-means clustering. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), 486–497 (2007)
28. Naor, M., Pinkas, B.: Efficient oblivious transfer protocol. In: Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (2001)
29. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game - a completeness theorem for protocols with honest majority. In: Proceedings of the ACM Symposium on the Theory of Computing (STOC) (1987)
30. Lindell, Y., Pinkas, B.: Privacy preserving data mining. Journal of Cryptology 15(3), 177–206 (2002)
31. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
32. Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: Proceedings of the ACM Conference on Management of Data (SIGMOD) (2011)

Differentially Private Continual Monitoring of Heavy Hitters from Distributed Streams

T.-H. Hubert Chan[1], Mingfei Li[1], Elaine Shi[2,⋆], and Wenchang Xu[3]

[1] The University of Hong Kong
[2] UC Berkeley
[3] Tsinghua University

Abstract. We consider applications scenarios where an untrusted aggregator wishes to continually monitor the heavy-hitters across a set of distributed streams. Since each stream can contain sensitive data, such as the purchase history of customers, we wish to guarantee the privacy of each stream, while allowing the untrusted aggregator to accurately detect the heavy hitters and their approximate frequencies. Our protocols are scalable in settings where the volume of streaming data is large, since we guarantee low memory usage and processing overhead by each data source, and low communication overhead between the data sources and the aggregator.

1 Introduction

Consider k data streams at k data sources, where items from some set $\mathcal{U}$ arrive at each stream. An *untrusted* aggregator wishes to continually monitor the most recent *heavy hitters* (i.e. the frequent items) over a sliding window – however, the data sources do not trust the aggregator, and wish to guarantee the privacy of their data streams. For example, a public health provider would like to monitor the potential outbursts of new epidemics (where the heavy hitters are the most common symptoms or diseases) by studying hospital visit records from k hospitals. Since medical records contain highly sensitive information, the hospitals may be legally obliged to protect their patients' privacy from the third-party public health provider. In Figure 1, we show another example where each stream represents a store, and the aggregator wishes to track the most popular items in the past week.

1.1 Results and Contributions

In this paper, we propose novel protocols that allow an untrusted aggregator to continually monitor the heavy hitters over a sliding window of duration W, while protecting the privacy of each data source. Since the aggregator is untrusted and there is no single trusted entity, standard privacy frameworks like PINQ [17] cannot be used directly

⋆ This material is based upon work partially supported by the Air Force Office of Scientific Research under MURI Grant No. 22178970-4170 and No. FA9550-08-1-0352. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 140–159, 2012.

Fig. 1. Problem setup. In this example, k stores wishes to continually monitor the popular items over the past week. The aggregator is assumed to be untrusted; and the stores may be concerned about protecting their secret business information such as sales revenue, and protecting the privacy of their customers.

in our distributed setting. In our protocols, each data source periodically sends sanitized (and potentially also encrypted) updates to the aggregator in order to notify the aggregator of latest trends as observed by the data source. The aggregator is then able to reconstruct the most recent popular items and their respective frequencies through these sanitized updates.

We conduct experiments using the Netflix Contest Dataset, and demonstrate that our algorithms can achieve low communication bandwidth and good utility in realistic application scenarios. We next explain the privacy guarantees and desirable features that our constructions achieve.

Two Levels of Privacy Protection. We propose protocols that achieve the following two different aspects of privacy.

Event-Level Differential Privacy. Our first construction, referred to as the PDCH-LU algorithm (which stands for *Private Distributed Continual Heavy-hitter - Lazy Update*), achieves *event-level differential privacy*. Roughly speaking, the sanitized updates released should be insensitive to the occurrence or non-occurrence of a single event. Intuitively, event-level differential privacy allows a store to guard the privacy of its customers, by concealing whether a certain purchase has taken place.

In our constructions, we achieve event-level differential privacy through the addition of appropriate noises before the release of any statistics. Note also that ϵ event-level differential privacy immediately implies $m\epsilon$ user-level differential privacy, where m is the maximum number of items for each user.

Aggregator Obliviousness. Through the use of bloom filters and special encryption schemes, our second protocol, referred to as PDCH-BF (which stands for *Private Distributed Continual Heavy-hitter - Bloom Filter*), achieves even stronger privacy guarantees: specifically, it achieves *aggregator obliviousness* in addition to *event-level differential privacy*.

On a high level, aggregator obliviousness advocates the *need-to-know* principle, i.e., the aggregator should ideally learn the *least amount of information* necessary to perform the heavy-hitter monitoring task. Specifically, in our second construction PDCH-BF,

apart from the approximate frequencies of a subset of the relatively more popular items across all streams, the aggregator learns nothing else.

To achieve aggregator obliviousness, our second protocol PDCH-BF employs bloom filters, as well as special encryption schemes [15; 20; 21] which support secure aggregation and controlled decryption of selective statistics.

Notice that aggregator obliviousness immediately implies the following: 1) in the example in Figure 1 the aggregator can learn the approximate frequencies of (a subset of) the items, but it cannot learn the transaction volume of each individual store which may be considered secret business information; and 2) although the aggregator can learn which the heavy hitters are and their approximate frequencies across all streams, the aggregator fails to learn which streams are contributing to these heavy-hitters, and how much each stream is contributing to these heavy hitters.

A more detailed discussion of our privacy notions and their nuances can be found in Section 2.3.

Low Computational and Communication Overhead. Our protocols require only a small amount of computation and memory from each data source. To process an item from the stream, a data source needs to update only a small number of counters; and in each time step, it needs to sample only a small number of random variables. This is desirable in numerous application settings – for example, in sensor network applications, where each node has low computational resources; or network intrusion detection scenarios, where routers cannot afford expensive real-time computation due to the large bandwidth throughput.

Our protocols also requires low communication costs between the data sources and the aggregator. Moreover, all communications are uni-directional from the data sources to the aggregator, and the data sources need not have interactions among themselves. In contrast, generic secure multi-party computation construction [13] requires expensive interactive communication between the data sources.

1.2 Related Work

Our work builds on several well-known lines of research. We describe some of the works that are the most related to ours and refer the readers to the cited references for more extensive review on the relevant research areas.

Differential Privacy in Continual Setting. Ever since Dwork [7] has introduced differential privacy, this notion has gained popularity in both the theory and security communities (see [8] for a quick review of the latest development). The idea of introducing randomness to perturb the outputs of algorithms allows a clean and formal way to analyze the tradeoff between preserving input privacy and achieving output accuracy. Recently, privacy has been studied in the continual setting [5; 9; 10]. Specifically, a change in the input in the current time step would not only affect the output in the current time step, but also might have a long lasting effect in the future. Useful continual differentially private algorithms would need to mask this long term effect without sacrificing too much on accuracy. Chan *et al.* [5] gave a differentially private mechanism to continually report the number of 1's seen so far in a bit stream with additive error

that is polylogarithmic in the number of time steps. In the streaming model, Dwork *et al.* [10] also distinguish between *event-level privacy* and *user-level privacy*: event-level privacy hides the occurrence of a particular event, while user-level privacy prevents adversaries from determining whether the stream contains any of a particular user's activities at all. In this paper, we use event-level differential privacy as our privacy notion. Mir *et al.* [18] also considered the problem of private streaming algorithms to return the counts of heavy hitters, not the heavy hitters themselves. However, they consider a more general setting in which in each update, the counter of an item can be increased or decreased arbitrarily, as long as the counter remains non-negative. Moreover, their notion of privacy hides the following change in the stream: any subset of occurrences of an item can be replaced with another item, and remaining updates can be arbitrarily reordered; on the other hand, the total count of all items is public knowledge in their setting.

Untrusted Aggregator. There have been works on studying the case when the aggregator is untrusted [15; 20; 21], where cryptographic techniques are employed. We first consider protocols in which each node will desensitize its data first so that cryptography will not be necessary; in order to achieve a stronger notion of privacy and security, we augment our protocols by employing cryptographic techniques.

Streaming Algorithms for Heavy Hitters. The first algorithm to output frequent items was given by Misra and Gries [19] (MG algorithm). They designed a deterministic algorithm that reads a stream of W items and at the end gives the count of every item in the stream with relative error λ (i.e., additive error at most λW); the algorithm only uses $O(\frac{1}{\lambda})$ words of memory. The MG algorithm was rediscovered several times [6; 14].

Using the MG algorithm concurrently on overlapping blocks of different sizes, Arasu and Manku [1] gave a deterministic algorithm that continually estimate the count of every item with relative error λ with respect to the current window; the query and the update time is $O(\frac{1}{\lambda}\log\frac{1}{\lambda})$, while $O(\frac{1}{\lambda}\log^2\frac{1}{\lambda})$ words of memory is required. They also gave a randomized version, where both the time and the memory is $O(\frac{1}{\lambda}\log\frac{1}{\delta\lambda})$, where δ is the failure probability.

Lee and Ting [16] augmented the counters in the MG algorithm to include approximate positions of where items occur, and consequently they improved Arasu and Manku's algorithm performance to $O(\frac{1}{\lambda})$, for both running time and memory requirement.

Distributed Streaming Protocols with Low Communication Cost. In the distributed streaming model, each of k nodes receives its own stream, and the nodes communicate with the aggregator, who wishes to estimate the number of times each item appears in all the streams in the current window with relative error λ. Yi and Zhang [22] considered the special case with infinite window size and gave a 2-way communication (between nodes and aggregator) protocol with total communication cost of $O(\frac{k}{\lambda}\log N)$ words, where N is the total number of items arriving at all streams. Chan *et al.* [3] considered the case of a sliding window, and gave a one-way communication (from nodes to aggregator) protocol. Under the special case of exactly one item arriving in each time step

for each stream, the communication cost for their protocol in L consecutive time steps is $O(\frac{k}{\lambda} \cdot \lceil \frac{L}{W} \rceil \log W)$ words.

Related Notion of Privacy. In Gantal *et al.* [11], it is mentioned that Dwork and McSherry proposed *semantic privacy* which measures how the posterior distribution on the database changes after the transcript is observed. In particular, it is shown that ϵ-differential privacy implies $(e^{\epsilon} - 1)$-semantic privacy.

2 Preliminaries

2.1 Notations and Conventions

Given a positive integer m, we let $[m] := \{1, 2, \ldots, m\}$, and use $\mathbb{N} := \{1, 2, 3, \ldots\}$ to index time steps. Let $\mathcal{U}$ be a set of n items. We use the standard notation $\tilde{O}(\cdot)$ to suppress poly-logarithmic factors.

We assume that an integer can be represented by $O(1)$ words. Although later on we use random distribution on unbounded integers, we show that with high probability the magnitudes of the sampled integers are small. Hence, we can use modulo arithmetic over some large enough integer. We do not explicit explain how each data source obtains its source of randomness, but we assume that it takes $O(1)$ operations to sample a random variable that is independent of the input data stream.

2.2 Problem Setup

We assume a set of $k \in \mathbb{N}$ *streams*, originating at k *data sources* (also referred to as *nodes*) respectively. We assume that each data source only has limited memory and computational power. Each stream $\sigma^{(i)} \in \mathcal{U}^{\mathbb{N}}$ where $i \in [k]$ is a sequence of items from $\mathcal{U}$, where $\sigma^{(i)}(t) \in \mathcal{U}$ is the item appearing at time step t in the i-th stream.

We consider an *untrusted aggregator* who wishes to continually monitor the heavy hitters over a sliding window of size $W \in \mathbb{Z}$. Formally, the window at time step t over stream σ is the multiset $\mathcal{W}_t(\sigma) := \sigma([t - W + 1, t])$ containing all items coming to stream σ between time $t - W + 1$ and t. Given k streams $\{\sigma^{(i)} : i \in [k]\}$, we write $\mathcal{W}_t^{(i)} := \mathcal{W}_t(\sigma^{(i)})$ and denote $\mathcal{W}_t^{[k]} := \uplus_{i \in [k]} \mathcal{W}_t(\sigma^{(i)})$ as the multiset containing all items from the k streams in the window at time t.

The notion of *heavy hitters* is formally defined as below. Given a multiset $\mathcal{W}$, we use $|\mathcal{W}|$ to denote the number of items it contains and for $x \in \mathcal{U}$, $\mathsf{count}_x(\mathcal{W})$ is the number of times item x appears in $\mathcal{W}$. Given $0 < \theta < 1$, we say an item $x \in \mathcal{U}$ is a θ-heavy hitter in the multiset $\mathcal{W}$ if $\mathsf{count}_x(\mathcal{W}) \geq \theta \cdot |\mathcal{W}|$.

Definition 1 (Approximate Heavy Hitters). *Given $0 < \lambda, \theta < 1$ and a multiset $\mathcal{W}$, a set $S \subseteq \mathcal{U}$ is a λ-approximation for θ-heavy hitters in $\mathcal{W}$ if*

1. *the set S contains all θ-heavy hitters in $\mathcal{W}$; and*
2. *if $x \in S$, then x is a $(\theta - \lambda)$-heavy hitter in $\mathcal{W}$.*

Definition 2 (λ-approximate Count). *Given a multi-set $\mathcal{W}$ on items in $\mathcal{U}$, and an item $x \in \mathcal{U}$, an estimate $\widehat{c}(x)$ is called a λ-approximate count for x with respect to $\mathcal{W}$, if $|\widehat{c}(x) - \mathsf{count}_x(\mathcal{W})| \leq \lambda |\mathcal{W}|$.*

Observe that if we have a λ-approximate count for every item $x \in \mathcal{U}$ with respect to $\mathcal{W}$, then we can compute a 2λ-approximation for heavy hitters in $\mathcal{W}$.

Communication Protocol. Consider a node receiving some stream σ. At every time step t, upon receiving the item $\sigma(t)$, the node might send messages to the aggregator to update some counters. In the protocols that we consider, each message contains items of the form $c \in \mathbb{Z}$ or $\langle x, c\rangle \in \mathcal{U} \times \mathbb{Z}$, each of which we assume can be expressed in $O(1)$ words. Given a (randomized) protocol Π, we denote by $\Pi(\sigma)$ the (randomized) transcript which consists of the messages sent at every time step by the node that applies the protocol on the stream σ. We wish to reduce the amount of communication, say the average number of words sent per time step.

Remark 1. In order to show that each item count has small relative error, we need a lower bound on the total number items in the finite stream to absorb the noise error. Moreover, the way in which we use PMG in combination with Arasu and Manku's algorithm [1] for fix-sized windows requires the assumption that exactly one item comes in the stream at every time step.

2.3 Defining Privacy

As mentioned earlier, we define our privacy notions based on the following principles: 1) We advocate a need-to-know principle, the aggregator should ideally learn the least amount of information necessary to perform the heavy hitter monitoring task. 2) While the amount of information revealed to the aggregator is kept at a minimum, the information eventually revealed to the aggregator should satisfy event-level differential privacy. In other words, any statistics revealed should be insentive to the occurrence or non-occurrence of a single event. Intuitively, this helps to conceal whether some event of interest has happened, e.g., whether a customer Alice has purchased a specific item.

Below, we formally define differential privacy and aggregator obliviousness.

Differential Privacy. In our setting, each node regards the contents on its stream as private data. In particular, from the transcript of a node, the aggregator should not be able to distinguish between input streams that are close to each other. Formally, two different streams $\sigma_1, \sigma_2 \in \mathcal{U}^{\mathbb{N}}$ are *adjacent* or *neighbors* (denoted as $\sigma_1 \sim \sigma_2$) if they differ at exactly one time step. We use the notion of event-level differential privacy for protocols.

Definition 3 (Differential Privacy for Protocols). *Given $\epsilon > 0$, a (randomized) protocol Π is ϵ-differentially private if for any adjacent streams σ_1 and σ_2, any subset $\mathcal{O}$ of possible output transcripts, $\Pr[\Pi(\sigma_1) \in \mathcal{O}] \leq \exp(\epsilon) \cdot \Pr[\Pi(\sigma_2) \in \mathcal{O}]$, where the randomness comes from the protocol.*

Aggregator Obliviousness. As we shall see, as an intermediate step in our protocols, each node has some private number and the goal is for the aggregator to learn the sum of all the nodes' numbers, but nothing more. Formally, each node has some data in $\mathcal{D}$ and we use $\mathbf{x} \in \mathcal{D}^n$ to denote a configuration of all the nodes' data. Intuitively, a protocol Π is *aggregator oblivious* with respect to some function $f : \mathcal{D}^n \to \mathcal{O}$ if for all $\mathbf{x}$ and $\mathbf{y}$ such that $f(\mathbf{x}) = f(\mathbf{y})$, no polynomial-time adversary can distinguish between the transcripts $\Pi(\mathbf{x})$ and $\Pi(\mathbf{x})$ with non-negligible probability.

Definition 4 (Aggregator Obliviousness). *Let $\kappa \in \mathbb{N}$ be a security parameter. A protocol ensemble $\{\Pi_\kappa\}_{\kappa\in\mathbb{N}}$ is aggregator obliviousness with respect to the function $f : \mathcal{D}^n \to \mathcal{O}$ if there exists a negligible function $\eta : \mathbb{N} \to \mathbb{R}^+$ such that for all $\mathbf{x}$ and $\mathbf{y}$ such that $f(\mathbf{x}) = f(\mathbf{y})$, for all decisional probabilistic polynomial-time Turing machines $\mathcal{A}$,*

$$|\Pr[\mathcal{A}(\Pi_\kappa(\mathbf{x})) = 1] - \Pr[\mathcal{A}(\Pi_\kappa(\mathbf{y})) = 1]| \leq \eta(\kappa),$$

where the probability is over the randomness of the protocol Π_κ and the Turing machine $\mathcal{A}$.

2.4 Defining Utility

Recall that for each $i \in [k]$, each node i receives some stream $\sigma^{(i)}$ and follows some (randomized) protocol to send messages to the aggregator in every time step. Based on the messages received up to time t, the aggregator computes for each $x \in \mathcal{U}$ some number $\mathcal{A}(t, x)$, which is an estimate for $\mathsf{count}_x(\mathcal{W}_t^{[k]})$. Observe that $\mathcal{A}(t, x)$ is a random variable, whose randomness comes from the randomized protocols. We use the following notion to measure the usefulness of $\mathcal{A}(t, \cdot)$ with respect to $\mathcal{W}_t^{[k]}$ for each t.

Definition 5 ((ξ, δ)-Usefulness). *Suppose $\mathcal{W}$ is a multiset containing items in $\mathcal{U}$, and $\mathcal{A} \in \mathbb{R}^{\mathcal{U}}$ is a collection of random variables indexed by $\mathcal{U}$. Then, $\mathcal{A}$ is (ξ, δ)-useful with respect to $\mathcal{W}$, if with probability at least $1 - \delta$, for every item $x \in \mathcal{U}$, $|\mathcal{A}(x) - \mathsf{count}_x(\mathcal{W})| \leq \xi$; in particular, if $\xi = \lambda|\mathcal{W}|$, then $\mathcal{A}(x)$ is a λ-approximate count for x with respect to $\mathcal{W}$.*

Definition 6 ((ξ, δ)-Simultaneous Usefulness.). *Let $\mathcal{T}$ be an index set. Suppose for any $t \in \mathcal{T}$, $\mathcal{W}_t$ is a multiset containing items in $\mathcal{U}$, and $\mathcal{A}_t$ is a collection of random variables indexed by $\mathcal{U}$. Then, $\{\mathcal{A}_t\}_{t\in\mathcal{T}}$ is (simultaneously) (ξ, δ)-useful with respect to $\{\mathcal{W}_t\}_{t\in\mathcal{T}}$, if with probability at least $1 - \delta$, for every $t \in \mathcal{T}$ and every item $x \in \mathcal{U}$, $|\mathcal{A}_t(x) - \mathsf{count}_x(\mathcal{W}_t)| \leq \xi$.*

3 Achieving Differential Privacy

3.1 Roadmap

This section describes a protocol between the data sources and an aggregator, allowing the aggregator to continually monitor the heavy hitters over a sliding window. We will show how to achieve event-level differential privacy in this section. Later in Section 4, we show how to achieve aggregator obliviousness.

We proceed with the following three-step recipe:

1. **The PMG algorithm outputs the heavy hitters in a single stream.** In Section 3.2, we describe a private streaming algorithm, which allows a single data source to output the approximate heavy hitters in a stream, after a one-pass scan of the entire stream. Since this algorithm builds on top of the Misra-Gries streaming algorithm [19], we refer to it as the Private Misra-Gries (PMG) algorithm. The MG Algorithm maintains an approximate vector of item counts by storing only non-zero counts explicitly for only a small number of items. The main technical challenge to privatize the MG Algorithm is to show that this approximate vector has small sensitivity.

2. **The PCC algorithm continually monitors the recent heavy hitters in a single stream.** In Section 3.3, we extend the aforementioned PMG algorithm to derive a Private Continual Heavy-hitter (PCC) algorithm, which supports the continual monitoring of heavy hitters over a sliding window in a single stream. The main technique in this step is the use of a *binary interval tree* which allows us to achieve small memory when the window size is large.
3. **The PDCH-LU protocol continually monitors the recent heavy hitters across multiple streams.** Finally, in Section 3.4, we extend the above PCC algorithm, which works for a single stream, to the distributed setting. In the resulting protocol PDCH-LU (*Private Distributed Continual Heavy-hitter - Lazy Update*), in order to save communication cost, each data source sends sanitized updates to the aggregator whenever necessary (hence lazy updates), to inform the aggregator of latest trends in its observed stream. The aggregator can in turn continually output the approximate heavy hitters across all streams over a sliding window.

3.2 Private Misra-Gries Algorithm

In this section, we consider a sub-problem, which will be a useful building block to construct the private streaming protocol at a node. Given a stream of length T and an error parameter $0 < \lambda < 1$, the goal is to estimate the number of times each item in $\mathcal{U}$ appears in the stream with additive error λT.

Our approach is based on the (non-private) MG Algorithm [19], which keeps explicit counters for only $O(\frac{1}{\lambda})$ items. Observe that since we accept λT additive error, if an item $x \in \mathcal{U}$ appears for less than λT times in the stream, then we do not need to keep a counter explicitly for $x \in U$ and can give an estimate count of zero. Because at most $O(\frac{1}{\lambda})$ items can appear for at least λT times in a stream of length T, intuitively it is sufficient to keep $O(\frac{1}{\lambda})$ explicit counters. The MG Algorithm makes sure that at any point in time at most $O(\frac{1}{\lambda})$ items have explicit non-zero counts. If an item arrives and we need to create an extra counter, all existing non-zero counters are decremented by 1; this step keeps the number of non-zero counters small. On the other hand, whenever a non-zero counter of some item x decreases by 1, there are $\Theta(\frac{1}{\lambda})$ other items that also have their counts decreased by 1. Since this can happen for at most λT times, the final count for each item has additive error at most λT.

At the end of the MG Algorithm, the output corresponds to a count vector $f \in \mathbb{Z}^{\mathcal{U}}$, which has at most $O(\frac{1}{\lambda})$ non-zero coordinates. Lemma 2 states that this vector has sensitivity at most $O(\frac{1}{\lambda})$, and hence we can apply the techniques of geometric noise to achieve differential privacy. The properties of the private version of the MG Algorithm are given in the following lemma, whose proof is given in the full version [4].

Lemma 1 (Private MG Algorithm). *Given a privacy parameter $\epsilon > 0$ and an approximation parameter $0 < \lambda < 1$, there is a (randomized) mechanism $\mathcal{M}$, denoted as $\mathsf{PMG}(\epsilon, \lambda)$ (Private Misra-Gries), that takes any finite stream σ and after one pass outputs a vector $\widehat{f} \in \mathbb{Z}^{\mathcal{U}}$ such that the following properties hold.*

1. *ϵ-Differential Privacy: for any adjacent streams σ_1 and σ_2, any subset $\mathcal{O} \subseteq \mathbb{Z}^{\mathcal{U}}$, $\Pr[\mathcal{M}(\sigma_1) \in \mathcal{O}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(\sigma_2) \in \mathcal{O}]$, where the probability is over the randomness from the mechanism.*

2. *Utility: Suppose* $0 < \delta < 1$, *and the length* T *of the stream* σ *satisfies* $T \geq \frac{32}{\lambda^2 \epsilon} \log \frac{n}{\delta}$. *Then, the vector* $\widehat{f}$ *is* $(\lambda T, \delta)$*-useful with respect to the multiset* $\sigma([T])$ *of items in the stream.*
3. *The mechanism uses only* $O(\frac{1}{\lambda})$ *words of memory. In particular, at most* $O(\frac{1}{\lambda})$ *coordinates of* $\widehat{f}$ *are non-zero. Moreover, it takes* $O(\frac{1}{\lambda})$ *operations to process each item, and samples* $O(\frac{1}{\lambda})$ *random variables in total.*

We modify the Misra-Gries Algorithm [16; 19] to get a counting mechanism, which we call the Private Misra-Gries Algorithm (PMG) and is given in Algorithm 1. We outline the main ideas of the algorithm. In the literature, it is common to achieve differential privacy by adding geometric noise. However, since we want to be careful about memory usage, we want the output of the mechanisms to be integral. Therefore, we add noises sampled from symmetric geometric distributions [12].

Definition 7 (Geometric Distribution). *Let* $\alpha > 1$. *We denote by* $\mathsf{Geom}(\alpha)$ *the symmetric geometric distribution that takes integer values such that the probability mass function at* l *is* $\frac{\alpha-1}{\alpha+1} \cdot \alpha^{-|l|}$.

The following property of symmetric geometric distribution is useful for designing differentially private counting mechanisms.

Fact 1. *Let* $u, v \in \mathbb{Z}^n$ *be two vectors such that* $||u - v||_1 \leq \Delta$, *where* $||u - v||_1 = \sum_{i=1}^{n} |u_i - v_i|$ *is the* ℓ_1*-norm of* $u - v$. *Let* $r \in \mathbb{Z}^n$ *be a random vector whose coordinates are independent random variables sampled from symmetric geometry distribution* $\mathsf{Geom}(\exp(\frac{\epsilon}{\Delta}))$. *Then, for any vector* $p \in \mathbb{Z}^n$, $\Pr[u + r = p] \leq \exp(\epsilon) \cdot \Pr[v + r = p]$.

Definition 8 (Sensitivity). *Let* $f : \mathcal{U}^T \to \mathbb{Z}^n$ *be a function that takes a stream of length* T *as input. The sensitivity of* f, *denoted by* $\Delta(f)$, *is* $\max_{\sigma_1 \sim \sigma_2} ||f(\sigma_1) - f(\sigma_2)||_1$.

By Fact 1, for any function $f : \mathcal{U}^T \to \mathbb{Z}^n$ such that $\Delta(f) \leq \Delta$, we can make its output ϵ-differentially private by adding independent random noise sampled from $\mathsf{Geom}(\exp(\frac{\epsilon}{\Delta}))$ to coordinates of f. The following lemma explains what parameter one should use for the geometric distribution in the PMG Algorithm to achieve differential privacy. We give its proof in the full version [4].

Lemma 2. *In Algorithm 1 after Line 10, the function* $f : \mathcal{U}^T \to \mathbb{Z}^{\mathcal{U}}$ *has sensitivity at most* $\beta + 1$, *where* $\beta := \lceil \frac{2}{\lambda} \rceil$.

3.3 Private Continual Heavy-Hitter Monitoring over a Sliding Window

In this section, we use PMG to construct a differentially private mechanism named Private Continual Heavy-hitter (PCC), that uses small memory and maintains some efficient data structure at every time step.

As intuitively illustrated as in Figure 2, we build a binary interval tree, where each leaf node represents $\lceil \frac{\lambda W}{4} \rceil$ (= 1 in the figure) time steps, and each non-leaf node represents a range of time steps. (Note that Figure 2 only depicts the bottom few levels of this binary interval tree due to reasons stated below). We refer to each node as a *block*,

Input: A privacy parameter ϵ, an approximation parameter λ, and a finite stream $\sigma \in \mathcal{U}^T$ of length T.
Output: A vector $\widehat{f} \in \mathbb{Z}^{\mathcal{U}}$, where $\widehat{f}(x)$ is a λ-approximate count of x with respect to $\sigma([T])$ with high probability.

For each $x \in \mathcal{U}$, $f(x)$ and $\widehat{f}(x)$ are (implicitly) initialized to 0;
$\beta \leftarrow \lceil \frac{2}{\lambda} \rceil$;
for $t \leftarrow 1$ **to** T **do**
  $f(\sigma(t)) \leftarrow f(\sigma(t)) + 1$;
  if *the number of items* x *such that* $f(x) > 0$ *exceeds the threshold* β **then**
    `// the decreasing step`
    for $x \in \mathcal{U}$ *such that* $f(x) > 0$ **do**
      $f(x) \leftarrow f(x) - 1$;
    end
  end
end
`// The sensitivity` $\Delta(f) \leq \beta + 1$
for *each* $x \in \mathcal{U}$ **do**
  `// In the full version, we give a faster procedure that samples only` $O(\beta)$ `random variables and achieves the same output distribution.`
  Sample a fresh independent noise $r_x \sim \mathsf{Geom}(\exp(\frac{\epsilon}{\beta+1}))$;
  $\widehat{f}(x) \leftarrow \max\{f(x) + r_x, 0\}$;
  `// we only keep the top` β `non-zero` $\widehat{f}(x)$`'s`
  if *there are more than* β *items* x *with* $\widehat{f}(x) \neq 0$ **then**
    Let y be the item with smallest non-zero $\widehat{f}(y)$ (resolving ties arbitrarily);
    Set $\widehat{f}(y) \leftarrow 0$;
  end
end
Output $\widehat{f}$;

Algorithm 1. Private Misra-Gries Algorithm $\mathsf{PMG}(\epsilon, \lambda)$

and run the subroutine PMG algorithm for each block, with appropriate privacy and approximation parameters. To output the heavy hitters for any time range, it suffices to "sum up" a logarithmic number of blocks in the binary interval tree – since any range can be represented by the union of a logarithmic number of disjoint blocks. In Figure 2, since we consider a window size of $W = 7$, we only need the bottom 3 levels of the binary tree. The main purpose of the binary interval tree technique is to save memory and allow faster computation.

Small Memory. The amount of memory necessary is $\tilde{O}(\frac{1}{\lambda})$ independent of W. To achieve this, we use a *garbage collection* technique, where a data source saves only the blocks which will later be needed, and discard all "expired" blocks which will no longer be needed. As shown in Figure 2, at any point of time, a block can be one of the following four types: 1) *expired*, i.e., will no longer be needed in the future; 2) *active*, i.e., the block has completed construction, and will be needed now or in the future; 3) *under-construction*, i.e., the heavy hitters for this block are currently being

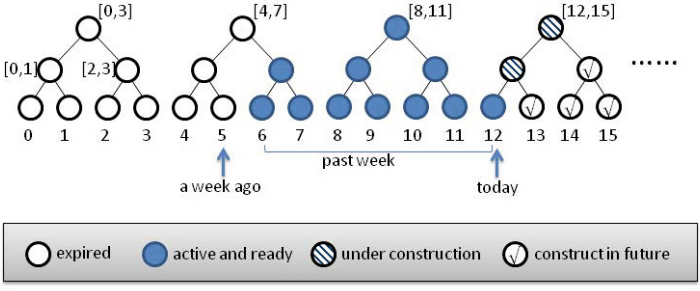


Fig. 2. Continual counting over a sliding window of the past week

constructed; or 4) *future*, i.e., construction for this block will start at some point in the future. Specifically, in the PCC algorithm, a data source saves only the blocks that are either active or under construction – and the number of such nodes is $O(\frac{1}{\lambda})$ at any point in time. Observe that the number of counters kept for binary nodes at different levels are different, and we later give a calculation to show that the total number of counters kept at any time is $O(\frac{1}{\lambda}\log^2\frac{1}{\lambda})$.

Faster Computation. Observe that we could achieve even smaller memory if we only store the leaf nodes of the binary construction. However, in order to produce an estimate count over a window, we would need to look at $\Omega(\frac{1}{\lambda})$ leaf nodes. Using the binary construction, we only need to look at $O(\log\frac{1}{\lambda})$ binary tree nodes for count estimation at each step.

Low Communication Bandwidth. Notice we could achieve even smaller memory if only one leaf node (corresponding to $W_0 = \lceil\frac{\lambda W}{4}\rceil$ time steps) in the binary tree construction is stored at any time; however, updates need to be sent to the aggregator every W_0 time steps. As we shall see in Section 3.4, if information about each stream in the current window is kept at each node, then the communication bandwidth to the aggregator can be greatly reduced.

Note that a similar binary tree technique was also used in [1; 5].

We now give a formal description of the PCC algorithm, as well as its theoretic guarantees. At every time step t, PCC maintains a dictionary $\mathcal{P}_t$, which is a collection of at most $O(\frac{1}{\lambda}\log\frac{1}{\lambda})$ pairs $(x, c_x) \in \mathcal{U}\times\mathbb{Z}$ where for every item $x \in \mathcal{U}$, item x appears in at most one pair in $\mathcal{P}_t$. Hence, $\mathcal{P}_t$ can also be interpreted as a vector $\mathbb{Z}^{\mathcal{U}}$ or a function from $\mathcal{U}$ to $\mathbb{Z}$ in the natural way: $\mathcal{P}_t(x) := c_x$ if $(x, c_x) \in \mathcal{P}_t$, and $\mathcal{P}_t(x) := 0$ otherwise. Observe that only non-zero counts need to be stored in the dictionary , and we denote by $|\mathcal{P}_t|$ the number of items x having non-zero counts $\mathcal{P}_t(x)$. The following lemma is the main result of the section.

Lemma 3 (Private Continual Heavy Hitter Monitoring). *Given a privacy parameter $\epsilon > 0$, and an approximation parameter $0 < \lambda < 1$, there exists a continual counting mechanism $\mathcal{M}$, denoted as* PCC(ϵ, λ) *(Private Continual Heavy-hitter), that takes an infinite data stream $\sigma \in \mathcal{U}^{\mathbb{N}}$, and maintains at every time step t a dictionary $\mathcal{P}_t \in \mathbb{Z}^{\mathcal{U}}$. We write $\mathcal{M}(\sigma) := (\mathcal{P}_t)_{t\in\mathbb{N}} \in \mathbb{Z}^{\mathbb{N}\times\mathcal{U}}$. The following properties hold.*

1. *ϵ-Differential Privacy: for any adjacent streams σ_1 and σ_2, any subset $\mathcal{O} \subseteq \mathbb{Z}^{\mathbb{N}\times\mathcal{U}}$, $\Pr[\mathcal{M}(\sigma_1) \in \mathcal{O}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{M}(\sigma_2) \in \mathcal{O}]$, where the probability is over the randomness from the mechanism.*
2. *Utility: Suppose $0 < \delta < 1$ and $L > 0$. If $W \geq \Theta(\frac{1}{\lambda^2\epsilon}(\log^2 \frac{1}{\lambda}) \cdot \log(\frac{n}{\delta} \log \frac{1}{\lambda}))$. Then, at every time step $t \in \mathbb{N}$, $\mathcal{P}_t$ is $(\lambda W, \delta)$-useful with respect to $\mathcal{W}_t$. If $W \geq \Theta(\frac{1}{\lambda^2\epsilon}(\log^2 \frac{1}{\lambda}) \cdot \log(\frac{L+W}{\lambda W} \cdot \frac{n}{\delta}))$, then for any $T \geq L+1$, $\{\mathcal{P}_t\}_{t\in[T-L,T]}$ is simultaneously $(\lambda W, \delta)$-useful with respect to $\{\mathcal{W}_t\}_{t\in[T-L,T]}$.*
3. *The mechanism uses only $O(\frac{1}{\lambda}\log^2 \frac{1}{\lambda})$ words of memory. Moreover, it takes $O(\frac{1}{\lambda}\log \frac{1}{\lambda})$ operations to process each item in the stream, and samples at most $O(\frac{1}{\lambda}\log \frac{1}{\lambda})$ random variables in each time step.*
4. *For every time step t, $|\mathcal{P}_t| = O(\frac{1}{\lambda}\log \frac{1}{\lambda})$.*

Scheme Description $\mathsf{PCC}(\epsilon, \lambda)$**.** We assume $\frac{1}{\lambda}$ is a power of 2; otherwise, let $\lambda' := \max\{\frac{1}{2^k} : \frac{1}{2^k} < \lambda\}$ and run $\mathsf{PCC}(\epsilon, \lambda')$.

Let $W_0 := \lceil \frac{\lambda W}{4} \rceil > 0$, and let $\ell := \log \frac{4}{\lambda}$. Note that we have $\frac{W}{2} \leq W_0 \cdot 2^\ell \leq W$. We divide the time steps into binary hierarchical blocks with $\ell + 1$ levels, where all blocks at the same level have the same size, are disjoint and cover all time steps. In particular, for $0 \leq i \leq \ell$ and $j \geq 1$, the jth block at level i is the interval $[(j-1)W_i + 1, jW_i]$ of $W_i := 2^i \cdot W_0$ time steps; we use block $\mathcal{B}_j^i$ to denote the multiset of items from stream σ contained in this interval, i.e., $\mathcal{B}_j^i = \sigma([(j-1)W_i + 1, jW_i])$. At any time t, each block $\mathcal{B}_j^i$ is in one of the following four states.

1. **future:** None of $\mathcal{B}_j^i$'s items has come into $\mathcal{W}_t(\sigma)$, i.e., $(j-1)W_i + 1 > t$;
2. **under-construction:** Some of $\mathcal{B}_j^i$'s items are in $\mathcal{W}_t(\sigma)$, and the remaining items have not come into $\mathcal{W}_t$, i.e., $(j-1)W_i + 1 \leq t$ and $jW_i > t$;
3. **active:** $\mathcal{B}_j^i$ is totally within $\mathcal{W}_t(\sigma)$, i.e., $t - W + 1 \leq (j-1)W_i + 1 \leq jW_i \leq t$;
4. **expired:** At least one of $\mathcal{B}_j^i$'s items has expired, i.e., $(j-1)W_i + 1 < t - W + 1$.

For each block $\mathcal{B}_j^i$, right before the time step when its state becomes under-construction, i.e., $t = (j-1)W_i + 1$, $\mathsf{PCC}(\epsilon, \lambda)$ initiates an instance of $\mathsf{PMG}(\epsilon_i, \lambda_i)$ on $\mathcal{B}_j^i$, where $\epsilon_i := \frac{\epsilon}{2^{\ell-i+1}}$ and $\lambda_i := \frac{1}{2^i(\ell+1)}$. When $\mathcal{B}_j^i$ becomes active, $\mathsf{PMG}(\epsilon_i, \lambda_i)$ produces a vector $\widehat{f}_{\mathcal{B}_j^i} \in \mathbb{Z}^{\mathcal{U}}$, which has $O(\frac{1}{\lambda_i})$ non-zero coordinates. Then, $\mathsf{PCC}(\epsilon, \lambda)$ uses $O(\frac{1}{\lambda_i})$ words of memory to maintain this vector until $\mathcal{B}_j^i$ expires.

Cover by Disjoint Active Blocks. Observe that at any time t, there exists a collection $\mathcal{C}_t$ of disjoint active blocks, such that $\mathcal{C}_t$ contains at most two blocks from each level, and that the union of blocks in $\mathcal{C}_t$ is the union of all active level-0 blocks. At any time, $\mathsf{PCC}(\epsilon, \lambda)$ maintains the dictionary $\mathcal{P}_t$, where $\mathcal{P}_t = \{(x, \sum_{\mathcal{B}\in\mathcal{C}_t} \widehat{f}_{\mathcal{B}}(x)) : x \in \mathcal{U} \text{ and } \sum_{\mathcal{B}\in\mathcal{C}_t} \widehat{f}_{\mathcal{B}}(x) > 0\}$.

We prove that PCC maintains differential privacy. The analysis of the remaining properties is similar to that in [1, Section 5], and we include the proofs in the full version [4].

Privacy Guarantee. Observe that the output of PCC is a deterministic function of PMG's outputs on all blocks. Hence, we need only to show ϵ-differential privacy is maintained with respect to $(\widehat{f}_{\mathcal{B}})_{\forall\mathcal{B}}$. Consider an item arriving at time step t. We analyze which of the blocks would be affected if $\sigma(t)$ is replaced with a different item. It is not

hard to see that the item $\sigma(t)$ can be in at most one block at each level. Observe that $\widehat{f}_{\mathcal{B}}$ for a level-i block $\mathcal{B}$ maintains ϵ_i-differential privacy, where $\epsilon_i = \frac{\epsilon}{2^{\ell-i+1}}$ and observe that $\sum_{i=0}^{l} \epsilon_i \leq \epsilon$. Hence, we conclude that $\mathsf{PCC}(\epsilon, \lambda)$ preserves ϵ-differential privacy.

3.4 Privately and Continually Monitoring Heavy Hitters Across Distributed Streams

In this section, we use PCC to design a protocol between k data sources and an aggregator, allowing the aggregator to continually monitor the global heavy hitters. The resulting protocol, called PDCH-LU (which stands for *Private Distributed Continual Heavy-hitter - Lazy Update*) has low communication cost; moreover, the messages sent by each data source is differentially private against the aggregator.

Observe that each node could send the privatized updates to the aggregator at every time step in order for the aggregator to compute the approximate heavy hitters. However, to save communication bandwidth, we use a *lazy update* approach: updates are only required when the count of an item changes by a huge amount. Chan et al. [3] gave a distributed algorithm (called Approximate Counting (AC)) based on this idea and proved that it achieves small error. Since the AC Algorithm in [3] only needs an approximate count in the current window for each item in each stream at any time, our PCC algorithm is sufficient for this purpose. We give the main result and the construction of the protocol; the detailed analysis is given in the full version [4].

Theorem 1. *Suppose $\epsilon > 0$ is a privacy parameter, $0 < \lambda < 1$ is an approximation parameter, W is the window size and L is some positive integer. Given k streams each received by a node, every node can run an ϵ-differentially private communication protocol with the same time and space performance as $\mathsf{PCC}(\epsilon, \frac{\lambda}{11})$ in Lemma 3 to send messages to the aggregator such that if $W \geq \Theta(\frac{1}{\lambda^2\epsilon}(\log^2 \frac{1}{\lambda}) \log(\frac{L+W}{\lambda W} \cdot \frac{kn}{\delta}))$, then for every time interval $\mathcal{T} = [T+1, T+L]$ (where $T \geq W$), with probability at least $1-\delta$, at every time $t \in \mathcal{T}$, the aggregator can maintain a λ-approximate count for every item with respect to the current window in all streams, and the total communication cost by all nodes in the period $\mathcal{T}$ is $O(\frac{k}{\lambda} \cdot \lceil \frac{L}{W} \rceil \log W)$ words.*

Algorithm for the Aggregator. The aggregator maintains a counter $c_i(x)$ (initially 0) for each stream $i \in [k]$ and each item $x \in \mathcal{U}$. Upon receiving a message $\langle x, c \rangle$ from node i, the aggregator updates the counter $c_i(x) := c$. In each time step t, the aggregator calculates a count $c(x) = \sum_{i \in [k]} c_i(x)$; to produce a 2λ-approximate set of θ-heavy hitters, the aggregator releases the set of items x such that $c(x) \geq (\theta - \lambda)kW$.

Protocol for Each Data Source. We use Algorithm AC (Approximate Counting) in [3, Section 2.2] to get a protocol (shown in Algorithm 2), denoted as $\mathsf{PDCH\text{-}LU}(\epsilon, \lambda)$ (Private Distributed Heavy-hitter - Lazy Update). Each node i runs an instance of $\mathsf{PDCH\text{-}LU}(\epsilon, \lambda)$ on the stream $\sigma^{(i)}$ it receives.

4 Achieving Aggregator Obliviousness

The main construction described earlier in Section 3 is a protocol for k nodes to communicate with an aggregator, whose task is to keep track of heavy hitters over a sliding

Input: A privacy parameter ϵ, an approximation parameter λ, and a stream $\sigma \in \mathcal{U}^{\mathbb{N}}$
Run an instance of $\mathsf{PCC}(\epsilon, \frac{\lambda}{11})$ and (implicitly) initialize $\mathsf{Last}(x) := 0$ for each $x \in \mathcal{U}$;
// We only store non zero $\mathsf{Last}(x)$'s.
for $t \leftarrow 1$ **to** ∞ **do**
  for *each* x *such that* $\mathcal{P}_t(x) > 0$ *or* $\mathsf{Last}(x) > 0$ **do**
    Up: if $\mathcal{P}_t(x) > \mathsf{Last}(x) + \frac{9}{11} \cdot \lambda W$, send $\langle x, \mathcal{P}_t(x) \rangle$ and set $\mathsf{Last}(x) \leftarrow \mathcal{P}_t(x)$;
    Off: if $\mathsf{Last}(x) > 0$ and $\mathcal{P}_t(x) < \frac{3}{11} \cdot \lambda W$, send $\langle x, 0 \rangle$ and set $\mathsf{Last}(x) \leftarrow 0$;
    Down: if $\mathcal{P}_t(x) < \mathsf{Last}(x) - \frac{9}{11} \cdot \lambda W$, send $\langle x, \mathcal{P}_t(x) \rangle$ and set $\mathsf{Last}(x) \leftarrow \mathcal{P}_t(x)$;
  end
end

Algorithm 2. $\mathsf{PDCH\text{-}LU}(\epsilon, \lambda)$

window. This protocol guarantees differential privacy at the event level, i.e., the statistics released to the aggregator is not affected by the change of one event.

In this section, we describe a protocol which achieves a stronger level of privacy protection, i.e., we additionally achieve aggregator obliviousness on top of event-level differential privacy. Specifically, we wish to reveal the minimum amount of information possible to the aggregator, for it to successfully perform the heavy hitter monitoring task.

The main techniques we use to achieve aggregator obliviousness include Bloom filters [2] as well as special encryption schemes [15; 20; 21] that support the controlled decryption of selected statistics. Using these techniques to augment the PMG protocol described earlier, we achieve aggregator obliviousness in the sense that the aggregator learns only the approximate counts of each item, but nothing else. In particular, the aggregator does not learn which data sources are contributing to the heavy hitters and how much their contributions are.

In our protocol to be described later in this section, each node communicates the update with the aggregator every $W_0 = \lceil \frac{\lambda W}{4} \rceil$ time steps. We then employ Bloom filters to effectively reduce the bandwidth overhead. Observe that without the Bloom filters, we would need to perform n secure additions, one for each item, for each update. We shall see that using Bloom filters, we can reduce the dependence of the number of additions per update on n to $O(\log n)$.

4.1 Background on Special Encryption Scheme

As a building block for achieving aggregator obliviousness, we employ a special encryption scheme which supports the conditional decryption of selected statistics. In particular, we can use either the encryption scheme proposed by Shi *et al.* [21], Rastogi *et al.* [20], or Kursawe *et al.* [15] In comparison, the scheme by Shi *et al.* [21] requires uni-directional communication from the data sources to the aggregator, but the decryption algorithm is more expensive; whereas the scheme by Rastogi *et al.* [20] requires bi-directional communication between the data sources and the aggregator, but has smaller decryption overhead. The scheme by Kursawe *et al.* [15] only needs uni-directional communication and has low overhead, but each node needs to store $\Theta(k)$ keys corresponding to all other nodes. We now give a high-level overview of these

special encryption schemes. The special encryption schemes we employ typically involve the following algorithms or phases:

Setup. In a one-time setup phase, cryptographic keying materials are distributed to all data sources and the aggregator. In particular, each data source receives an encryption key, and the aggregator receives a cryptographic capability which will allow it to later decrypt the sum of all data sources in each aggregation time step. The setup phase can either be performed by an offline authority (which will no longer be needed after the setup phase); or through an interactive multi-party protocol amongst the data sources and the aggregator.

Periodic Encryption and Aggregation. In each time step, each data source $i \in [k]$ encrypts a value x_i using the encryption key established in the setup phase, and sends the ciphertext to the aggregator. After receiving ciphertexts from all data sources, the aggregator can use its cryptographic capability to decrypt the value $\sum_{i=1}^{k} x_i$, but learn nothing else. In the construction by Shi *et al.* [21] and Kursawe *et al.* [15], this decryption is done solely by the aggregator, whereas in the scheme by Rastogi *et al.* [20], the aggregator needs to communicate with the data sources to perform decryption.

In the remainder of this section, we will use this special encryption scheme as a blackbox – for more algebraic details on how these schemes are constructed, we refer the readers to [15; 20; 21].

4.2 Augmenting PMG Algorithm with Secure Bloom Filters

Straightforward Solution using Cryptography. We first describe a straightforward construction using one of the special encryption schemes [15; 20; 21]. A brief background on these encryption schemes was given in Section 4.1. We show that one drawback of this straightforward construction is its high bandwidth overhead. Later, we shall employ Bloom filters to reduce the bandwidth consumption.

The basic idea is to apply the special encryption scheme all items in the universe.

Suppose each of the k nodes is running $\mathsf{PMG}(\epsilon, \lambda)$ on its finite stream as described in Section 3.2. Each node $v \in [k]$ produces some $\widehat{f}_v : \mathcal{U} \to \mathbb{Z}$, which is represented by at most $\beta = O(\frac{1}{\lambda})$ non-zero counters. In every time step, for each $x \in \mathcal{U}$, each data source encrypts its observed frequency $\widehat{f}_v(x)$, and sends the ciphertext to the aggregator. The aggregator may then use its cryptographic capability to decrypt the total frequency $\sum_{v \in [k]} \widehat{f}_v(x)$ for each $x \in \mathcal{U}$ – and meanwhile, the security of these encryption schemes guarantee that the aggregator learns nothing else beyond the total frequency of each item.

It is not hard to see that each node needs to send $\Omega(n)$ words (proportional to the size of $\mathcal{U}$) to the aggregator. Even though each node only has β non-zero counters, it still has to participate in every addition such that the aggregator does not know where the non-zero values come from. We would like to decrease the communication cost through the use of Bloom filters.

Construction with Bloom Filters. Let $0 < \delta < 1$ be the desired failure probability, i.e., with probability at least $1 - \delta$, the aggregator can retrieve $\sum_{v \in [k]} \widehat{f}_v(x)$ for all $x \in \mathcal{U}$. Let $P := \lceil \ln \frac{n}{\delta} \rceil$ and $Q := \lceil ek\beta \rceil$, where e is the natural number and β is the maximum

number of non-zero counters for each node. We assume there is a public family $\{\mathcal{H}_p : \mathcal{U} \to [Q]\}_{p\in[P]}$ of random hash functions that satisfy the following properties.

1. The functions $\mathcal{H}_p$ are totally independent over different $p \in [P]$.
2. For each $p \in [P]$, $\mathcal{H}_p$ is pairwise independent over $\mathcal{U}$, i.e., for $x \neq y$, $\mathcal{H}_p(x)$ and $\mathcal{H}_p(y)$ are independent; moreover, for each $x \in \mathcal{U}$ and each $q \in [Q]$, $\Pr[\mathcal{H}_p(x) = q] = \frac{1}{Q}$.

Bloom Filters for Each Node. Each node v constructs a table A_v of size $P \times Q$ that is constructed in the following way.

1. Initially, every entry $A_v[p][q] := 0$.
2. For each $x \in \mathcal{U}$ such that $\widehat{f}_v(x) \neq 0$ (note that there are at most β such x's), for each $p \in [Q]$, increment $A[p][\mathcal{H}_p(x)]$ by $\widehat{f}_v(x)$.

Secure Addition of Bloom Filters. Using one of the secure periodic schemes [15; 20; 21] described above, the aggregator learns for each $p \in [P]$ and $q \in [Q]$, $A[p][q] := \sum_{v\in[k]} A_v[p][q]$. Observe that each node sends $O(PQ) = O(\frac{k}{\lambda} \log \frac{n}{\delta})$ words to the aggregator.

Retrieving Sum of Counts for Each Item. For an item x, the aggregator computes $\min_{p\in[P]} A[p][\mathcal{H}_p(x)]$.

Theorem 2. *With probability at least $1 - \delta$, the aggregator retrieves $\sum_{v\in[k]} \widehat{f}_v(x)$ accurately for all $x \in \mathcal{U}$.*

Proof. Fix some $x \in \mathcal{U}$. Observe that the aggregator makes a mistake for item x *iff* for all $p \in [P]$, there exists some $y \neq x$ such that $\mathcal{H}_p(x) = \mathcal{H}_p(y)$ and there exists $v \in [k]$ such that $\widehat{f}_v(y) \neq 0$.

Observe that each node v can only have at most β non-zero counts. For fixed $p \in \mathcal{P}$, by pairwise independence of $\mathcal{H}_p$ over $\mathcal{U}$ and uniformity of $\mathcal{H}_p(y)$ over $[Q]$, the probability that there exists some $y \neq x$ such that some node has non-zero count for y and $\mathcal{H}_p(x) = \mathcal{H}_p(y)$ is at most $\frac{k\beta}{Q} \leq \frac{1}{e}$.

By the total independence of $\mathcal{H}_p$ over p's, it follows that the probability that the aggregator makes a mistake for item x is at most $\frac{1}{e^P} \leq \frac{\delta}{n}$. Using the union bound over all $x \in \mathcal{U}$, it follows that the probability that the aggregator makes a mistake for some item is at most δ, as required.

4.3 Distributed Protocol Achieving Aggregator Obliviousness

We next describe how the PMG Algorithm augmented with Bloom Filters can be used to design a distributed protocol that achieves aggregator obliviousness – the resulting protocol is referred to as PDCH-BF (Private Distributed Continual Heavy-hitter - Bloom Filter).

Each node performs the binary construction as in the PCC algorithm described in Section 3.3. In particular, for a block at level i with size $W_i = 2^i \cdot W_0 = 2^i \cdot \frac{\lambda W}{4}$, $\mathsf{PMG}(\epsilon_i, \lambda_i)$ is run with $\epsilon_i := \frac{\epsilon}{2^{\ell-i+1}}$ and $\lambda_i := \frac{1}{2^i(\ell+1)}$. As soon as PMG is completed for a block, a Bloom filter is constructed for that block as described in Section 4.2. The Bloom filter for that block is encrypted and sent to the aggregator.

For a particular block, after the aggregator has received the ciphertexts of the Bloom filters from all the k nodes, it can decrypt the sum of the Bloom filters and reconstruct the counts for all items in that block.

Observe that to estimate the counts in a window, the aggregator just needs the counts from at most $2 \log \frac{1}{\lambda}$ blocks. Hence, in order to achieve failure probability of δ due to the Bloom filters for each window, it suffices to set the Bloom filter failure probability for each block to be $\frac{\delta}{2 \log \frac{1}{\lambda}}$. There is also failure probability δ due to the randomness introduced to achieve differential privacy. Hence, Lemma 3 implies the aggregator can compute λ-approximate heavy hitters within a window with probability $1 - 2\delta$.

Communication Cost. For each node, the communication cost for a block at level i is $O(\frac{k}{\lambda_i} \log \frac{n}{\delta_0})$ words, where $\lambda_i = \frac{1}{2^i \cdot (l+1)}$, $l = \log \frac{4}{\lambda}$ and $\delta_0 = \frac{\delta}{2(l+1)}$. Moreover, a block at level i is constructed every $W_i := 2^i \cdot W_0 = 2^i \cdot \frac{\lambda W}{4}$ time steps. Hence, it follows that the average number of words of communication per time step is $O(\frac{k}{\lambda W} \log^2 \frac{1}{\lambda} \log \frac{n \log \frac{1}{\lambda}}{\delta})$.

5 Experiments

5.1 Experimental Setup

The data used in our experiment was constructed from the Netflix Contest Dataset, which contains $n = 17770$ movies with 480189 users' ratings from 1999-11-11 to 2005-12-31. We divided the users randomly into 100 groups to construct 100 streams. We selected roughly 200 days' data of each stream from 2002-09-23 to 2003-04-30 to conduct our experiments, where we continually monitored the moving average over a window size of 90 days. We plot the result for the last 10 days by when the system should have become stable. We use the following parameters in our experiments: heavy-hitter fraction $\theta = 0.004$ and error $\lambda = 0.001$. In our experiments, we consider differential privacy parameter $\epsilon = 1, 2, 5, 10$. Note that there is no consensus on what privacy parameters are acceptable in practice, and even for $\epsilon = 1$, the scheme still offers some guarantee on privacy.

For the Bloom filters, we choose the number of hash functions to be $P = 8$, and the array size for each hash function to be $Q = \lceil e \cdot 3600 \rceil$ (we explain the choice of Q below).

Practical Optimization. Several aspects of our protocols can be further optimized in this application.

1. *Empirical Sensitivity of* PMG *Count Vector and Bloom Filter Size.* Since the value of λ we choose is small, in practice, the number of distinct movies observed by each node in each day never exceeds $O(\frac{1}{\lambda})$, the number of counters in PMG. If we assume that the daily number of distinct movies never exceeds the number of counters in PMG, the real sensitivity is $O(1)$, and so we can use $\mathsf{Geom}(e^{\epsilon})$ noise for PMG, instead of $\mathsf{Geom}(e^{\Theta(\epsilon\lambda)})$. Moreover, we observe that the number of distinct movies observed by all nodes in each day never exceeds 3600 and hence we can choose $Q = \lceil e \cdot 3600 \rceil$ to be the size of the Bloom filter array for each hash.

2. *Unnecessary Internal Blocks in Binary Construction.* Since the width W of our window is small, $W_0 = \lceil \frac{\lambda W}{4} \rceil = 1$. Hence, each leaf node in the binary tree corresponds to one time step, and therefore, the binary construction does not help to save memory in this case. Moreover, since W is small, the cost of count reconstruction from W leaf nodes is still better than the overhead of keeping track of the binary construction.
3. *Relaxing Assumption on Item Arrival.* Since we no longer need the binary construction, we can further relax the assumption that exactly one item arrives at each stream per time step. In our dataset, at least one item arrives at each stream at each time step, and we do not need any upper bound on the number of arriving items. Within each time step, PMG for each stream serializes the arriving items and process them in any arbitrary order.
4. *Estimating the Total Number of Arriving Items.* With the assumption that only one item arrives at each time step for each stream, it is trivial to compute the total number items within a window; this quantity is required for estimating the fraction an item appears and deciding when to do lazy updates (this is the W in the conditions for **Up**, **Off** and **Down** in PDCH-LU). However, now each stream also needs to report to the aggregator the number of items that arrive each day, which has sensitivity 1 for neighboring streams. Hence, in order to achieve ϵ-differential privacy for the whole protocol, we can assign $\epsilon_1 := \frac{9\epsilon}{10}$ and $\epsilon_2 := \frac{\epsilon}{10}$ such that each stream runs PMG with ϵ_1-differential privacy (using $\mathsf{Geom}(e^{\epsilon_1})$ noise) and estimates the number of items each day with ϵ_2-differential privacy (using $\mathsf{Geom}(e^{\epsilon_2})$ noise).[1]

Performance Metric. We consider the following performance measures. For each performance measure, we plot its mean with an error bar of 2 standard deviations (hence each plot could go negative).

1. *Error in estimated fraction.* For each day, suppose H is the set of items whose true fraction in the current window is at least θ and $\widehat{H}$ is the set of items whose estimated fraction in the current window is at least $\theta - \lambda$. For each item $x \in H \cup \widehat{H}$, we calculate the error $\mathcal{E}(x)$ which is the absolute difference between the true and the estimated fractions of item x in the current window. We compute the mean and the standard deviation of $\mathcal{E}(x)$ over x in $H \cup \widehat{H}$. We evaluate these statistics over time for different protocols with various parameters.
2. *Communication Cost.* We measure the number of words each node sends to the aggregator each day. For each day, we compute the mean and the standard deviation of the communication cost over different nodes.

5.2 Results

Utility. In Figures 3, 4, and 5, we observe that the error in each case is well below the theoretic guarantee $\lambda = 0.001$, and we interpret each figure as follows.

[1] We give more privacy budget to PMG as it is more complicated, and less privacy budget to the estimation of number of items each day as it is relatively simpler. It does not really affect the asymptotic error as long as each part get a constant fraction of the privacy budget, but experiment suggests that these parameters work well.

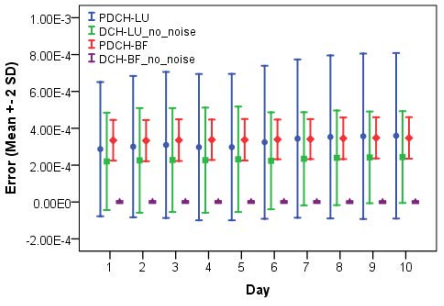


Fig. 3. Error under different protocols

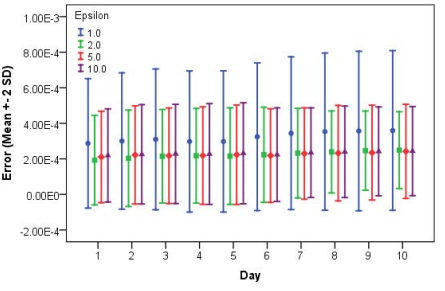


Fig. 4. Error under different privacy parameters for the PDCH-LU algorithm

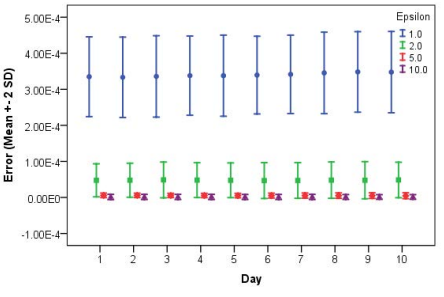


Fig. 5. Error under different privacy parameters for the PDCH-BF algorithm

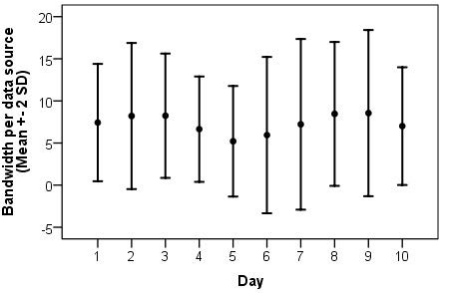


Fig. 6. The number of words sent per data source in the PDCH-LU protocol

Figure 3 compares the errors of the PDCH-LU and PDCH-BF protocols, and also demonstrates a breakdown of the error, i.e., how much error is introduced by compressing the bandwidth, and how much due to the noise necessary for differential privacy. With our choice of parameters, the Bloom filter should introduce almost no error and hence DCH-BF(no noise) forms the baseline for comparison. The plot for protocol PDCH-BF essentially reflects the error introduced when we wish to preserve differential privacy. The plot for DCH-LU(no noise) reflects the error introduced by lazy update in order to save communication bandwidth. The interesting unexpected result is that for PDCH-LU when we use lazy update together with noise to ensure differential privacy, the extra noise does not seem to increase the error by much. In fact, the effect of lazy updates seem to smooth out some of the error introduced by the added random noise.

Figures 4 and 5 plot the utility of PDCH-BF and PDCH-LU under different differential privacy parameter ϵ. As we decreases ϵ, the magnitude of noise increase and we can see in Figure 5 that as expected the error for PDCH-BF is increased as well. However, we can see that in Figure 4, that reducing ϵ has only a small effect on the performance of PDCH-LU.

Communication cost. Figure 6 shows that in the PDCH-LU protocol, the number of item updates sent by each node per day is around 5 and almost never above 20. Typically, each item update is under 10 bytes of data.

In comparison, we need to pay higher (but still reasonable) communication cost if we wish to ensure aggregator obliviousness. In our PDCH-BF experiment, given 8 hashes each having a filter of size $Q = \lceil e \cdot 3600 \rceil$, and assuming that each Diffie-Hellman ciphertext is 1024 bits, then the each sends about $10MB$ data per day to the aggregator.

References

[1] Arasu, A., Manku, G.S.: Approximate counts and quantiles over sliding windows. In: PODS (2004)

[2] Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)

[3] Chan, H.-L., Lam, T.-W., Lee, L.-K., Ting, H.-F.: Continuous monitoring of distributed data streams over a time-based sliding window. In: STACS (2010)

[4] Chan, T.-H.H., Li, M., Shi, E., Xu, W.: Differentially private continual monitoring of heavy hitters from distributed streams. In: Cryptology ePrint Archive (2012)

[5] Hubert Chan, T.-H., Shi, E., Song, D.: Private and Continual Release of Statistics. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part II. LNCS, vol. 6199, pp. 405–417. Springer, Heidelberg (2010)

[6] Demaine, E.D., López-Ortiz, A., Munro, J.I.J.: Frequency Estimation of Internet Packet Streams with Limited Space. In: Möhring, R.H., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 348–360. Springer, Heidelberg (2002)

[7] Dwork, C.: Differential Privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part II. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)

[8] Dwork, C.: A firm foundation for private data analysis. Commun. ACM 54(1), 86–95 (2011)

[9] Dwork, C., Naor, M., Pitassi, T., Rothblum, G.N.: Differential privacy under continual observation. In: STOC, pp. 715–724 (2010)

[10] Dwork, C., Naor, M., Pitassi, T., Rothblum, G.N., Yekhanin, S.: Pan-private streaming algorithms. In: ICS, pp. 66–80 (2010)

[11] Ganta, S.R., Kasiviswanathan, S.P., Smith, A.: Composition attacks and auxiliary information in data privacy. In: KDD, pp. 265–273 (2008)

[12] Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. In: STOC (2009)

[13] Goldreich, O.: The Foundations of Cryptography - Volume 2, Basic Applications. Cambridge University Press (2004)

[14] Karp, R.M., Shenker, S., Papadimitriou, C.H.: A simple algorithm for finding frequent elements in streams and bags. ACM Trans. Database Syst. 28, 51–55 (2003)

[15] Kursawe, K., Danezis, G., Kohlweiss, M.: Privacy-Friendly Aggregation for the Smart-Grid. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 175–191. Springer, Heidelberg (2011)

[16] Lee, L.K., Ting, H.F.: A simpler and more efficient deterministic scheme for finding frequent items over sliding windows. In: PODS (2006)

[17] McSherry, F.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In: SIGMOD Conference, pp. 19–30 (2009)

[18] Mir, D.J., Muthukrishnan, S., Nikolov, A., Wright, R.N.: Pan-private algorithms via statistics on sketches. In: PODS, pp. 37–48 (2011)

[19] Misra, J., Gries, D.: Finding repeated elements. Sci. Comput. Program. 2(2), 143–152 (1982)

[20] Rastogi, V., Nath, S.: Differentially private aggregation of distributed time-series with transformation and encryption. In: SIGMOD 2010, pp. 735–746 (2010)

[21] Shi, E., Chan, H., Rieffel, E., Chow, R., Song, D.: Privacy-preserving aggregation of time-series data. In: NDSS (2011)

[22] Yi, K., Zhang, Q.: Optimal tracking of distributed heavy hitters and quantiles. In: PODS (2009)

Adaptive Differentially Private Histogram of Low-Dimensional Data

Chengfang Fang and Ee-Chien Chang*

School of Computing
National University of Singapore
{c.fang,changec}@comp.nus.edu.sg

Abstract. We want to publish low-dimensional points, for example 2D spatial points, in a differentially private manner. Most existing mechanisms publish noisy frequency counts of points in a fixed predefined partition. Arguably, histograms with adaptive partition, for example V-optimal and equi-depth histograms, which have smaller bin-widths in denser regions, would provide more statistical information. However, as the adaptive partitions leak significant information about the dataset, it is not clear how differentially private partitions can be published accurately. In this paper, we propose a simple method based on the observation that the sensitivity of publishing the *sorted* sequence of a dataset is independent of the size of dataset. Together with isotonic regression, the dataset can be reconstructed with high accuracy. One advantage of the proposed method is its simplicity, in the sense that there are only a few parameters to be determined. Furthermore, the parameters can be estimated solely from the privacy requirement ϵ and the total number of points, and hence do not leak information about the data. Although the parameters are chosen to minimize the earth mover's distance between the published data and original data, empirical studies show that the proposed method also achieves high accuracy w.r.t. to some other measurements, for example range query and order statistics.

1 Introduction

The popularity of personal devices equipped with location sensors leads to a large amount of location data being gathered. Such data contain rich information and would be valuable if they can be shared and published. As the data may reveal location of an identified individual, it is important to anonymize the data before publishing. The recently developed notion of differential privacy [5] provides a strong form of privacy assurance regardless of the background information held by the adversaries. Such assurance is important, as many case studies and past events have shown that a seemingly anonymized dataset together with additional knowledge held by the adversary could reveal information on individuals.

Most studies on differential privacy focus on publishing statistical values, for instance, k-means [3], private coreset [7], and median of the database [20]. Publishing specific statistics or data-mining results is meaningful if the publisher

* Chang is partially supported by grant TDSI/09-003/1A.

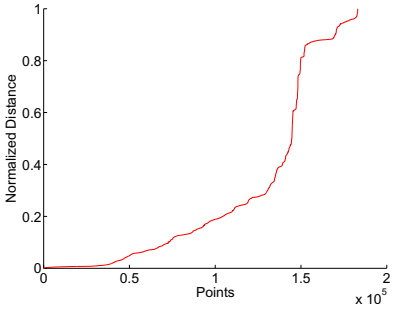
S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 160–179, 2012.

knows what the public specifically wants. However, there are situations where the publishers want to give the public greater flexibility in analyzing and exploring the data, for example, using different visualization techniques. In such scenarios, it is desired to *"publish data, not the data mining result"* [8].

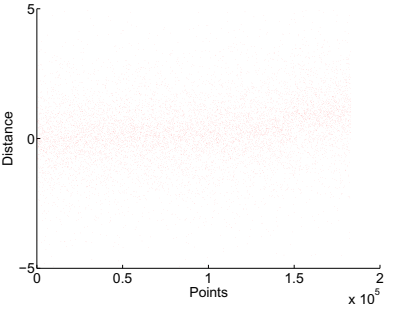
The histogram of a dataset contains rich information that can be harvested by subsequent analysis. In the context of different privacy, *parallel composition* can be exploited to treat non-overlapping bins independently and thus achieving high accuracy. There are a number of research efforts [14,2] investigating the dependencies of frequencies counts of fixed overlapping bins, where parallel composition can not be directly applied. Such overlapping bins are interesting as different domain partition could lead to different accuracy and utility. For instance, Xiao et al. [28] proposes publishing wavelet coefficients of an equi-width histogram, which can be viewed as publishing a series of equi-width histograms with different bin-widths, and is able to provide higher accuracy in answering range queries compare to a single equi-width histogram.

It is generally well accepted that equi-depth histogram and V-optimal histogram provide more useful statistical information compare to equi-width histogram [21,22], especially for multidimensional data. These histograms are adaptive in the sense that the domain partitions are derived from the data such that denser regions will have smaller bin-widths and the sparser regions will have larger bin-widths, as illustrated in Fig. 7(b). Since the bin-widths are derived from the dataset, they leak information about the original dataset. There are relatively few works that consider adaptive histogram in the context of differential privacy. One exception is the work by Xiao et al. [29]. Their method consists of two steps where firstly synthetic data are generated from the differentially private equi-width histogram. After that, a k-d tree (which can be viewed as an adaptive histogram) is generated from the synthetic data, and the noisy counts are then released with the partition. Machanavajjhala et al. [16] proposed a mechanism that publishes 2D histograms with varying bin-widths, where the bin-widths are determined from a previously released similar data. The histograms generated are not adaptive in the sense that the partitions do not depend on the data to be published.

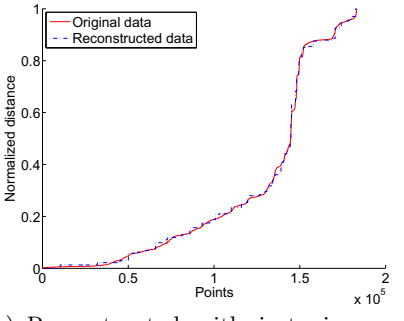
In this paper, instead of publishing the noisy frequency counts in equi-width bins, we propose a method that directly publishes the noisy data, which in turn leads to an adaptive histogram. To illustrate, let us first consider a dataset consisting of a set of real numbers from the unit interval, for example, the normalized distance of Twitter users' locations [1] to New York City (Fig. 1(a)). We observe that sorting, as a function that takes in a set of real numbers from the unit interval and outputs the sorted sequence, interestingly has sensitivity one (Theorem 1). Hence, the mechanism that first sorts, and then adds independent Laplace noise of $\mathrm{LAP}(1/\epsilon)$ to each element achieves ϵ-differential privacy. Fig. 1(b) shows the noisy output data after the Laplace noise has been added to the sorted sequence. Although seemingly noisy, there are dependencies to be exploited because the original sequence is sorted. By using isotonic regression, the noise can be significantly reduced (Fig. 1(c)). To further reduce noise, before adding the Laplace noise, consecutive elements in the sorted data can be



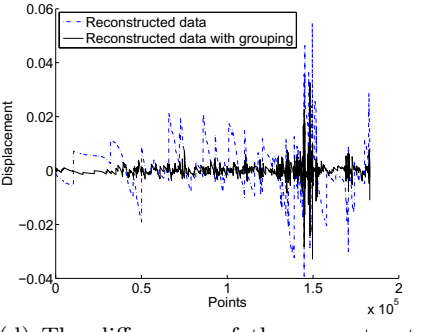
(a) Sorted 1D points.



(b) The sorted points with Laplace noise added. To avoid clogging, only 10% of the points (randomly chosen) are plotted.



(c) Reconstructed with isotonic regression.



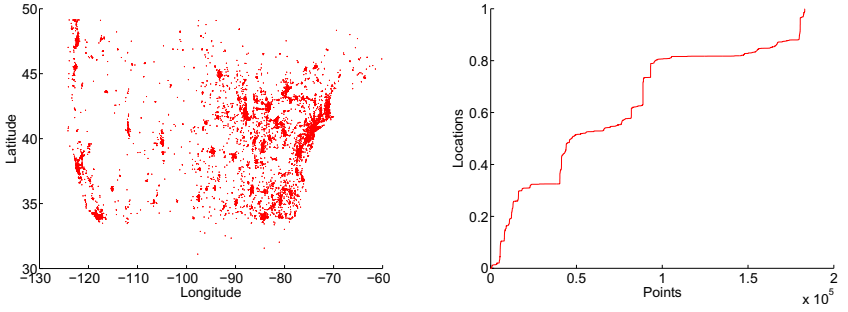
(d) The differences of the reconstructed points from the original.

Fig. 1. Overview of proposed method

grouped and each point is replaced by the average of its group. Fig. 1(d) shows the difference of the original and the reconstructed points with and without grouping.

To extend the proposed method to higher dimension data, for example, location data of 183,072 Twitter users in North America as shown in Fig. 2(a), we employ *locality-preserving mapping* to map the multidimensional data to one-dimension (Fig. 2(b)), such that any two close points in the one-dimension domain are mapped from two close multidimensional points. After that, the publisher can apply the proposed method on the 1D points, and publish the reverse mapped multidimensional points.

One desired feature of our scheme is its simplicity: there is only one parameter, the group size, to be determined. The group size affects the accuracy in three ways: (1) its effect on the *generalization error*, which is introduced due to averaging; (2) its effect on the level of Laplace noise to be added by the differentially private mechanism; and (3) its effect on the number of constraints in the isotonic regression. Based on our error model, the optimal parameter can be estimated without knowledge of the dataset distribution. In contrast, many



(a) Locations of Twitter users. To avoid clogging, only 10% of the points (randomly chosen) are plotted.

(b) Sorted 1D images of the data.

Fig. 2. Twitter location data and their 1D images of a locality-preserving mapping

existing methods have many parameters whose optimal values are difficult to be determined differentially privately. For instance, although the equi-width histogram has only one parameter, i.e. the bin-width, its value significantly affects the accuracy, and it is not clear how to differentially privately obtain a good choice of the bin width.

Our error model utilizes the earth mover's distance (EMD) to measure the accuracy of the published data. Some existing works measure the accuracy of a histogram by its distance, such as L_2 distance or KL divergence, to a reference equi-width histogram. One limitation of this measurement is that the reference histogram can be arbitrary and thus arguably ill-defined. If the reference bin-width is too small, each bin will contain either one or no point, which leads to significantly large distance from a seemingly accurate histogram. On the other hand, if its bin-width is too large, the reference histogram would be over generalized. In contrast, EMD measures the distance of the published data and original points, where the "reference" is the original points and thus well-defined. We conduct empirical studies to compare against a few related known methods: equi-width histogram, wavelet-based method [28] and smooth sensitivity based median-finding [20]. Although our method is designed to minimize the EMD, it also attains high accuracy w.r.t. other measurements. Empirical studies shows that our method outperforms the wavelet-based method w.r.t. accuracy of range-query, even for ranges with large sizes. It is also comparable to the smooth sensitivity based method in publishing median.

Organization: We first describe some background materials in the next section. In Section 3 we present our main ideas and mechanism, and show that the proposed mechanism achieves differential privacy in Section 4. Next, in Section 5, we formulate and analyze how the group size affects the accuracy and derive a strategy to choose the group size based on this model. In Section 6, we compare our mechanism with three known mechanisms: (1) equi-width histogram, (2) wavelet-based method, and (3) smooth sensitivity based median-finding. In

Section 7, we discuss the extensions and limitations of our method. Lastly, we describe related works in Section 8 and conclude in Section 9.

2 Background

2.1 Differential Privacy and Laplace Noise

We treat a database as a multi-set (i.e. a set with possibly repeating elements), and consider two datasets D_1 and D_2 of size n to be neighbors when D_2 can be obtained from D_1 by replacing one element, i.e. $D_1 = \{x\} \cup D_2 \setminus \{y\}$ for some x and y. Differential privacy with this definition of neighborhood is known as the *bounded differential privacy* [6,13].

A randomized algorithm (also known as *mechanism*) $\mathcal{A}$ achieves ϵ differential privacy if,

$$Pr[\mathcal{A}(D_1) \in S] \leq exp(\epsilon) \times Pr[\mathcal{A}(D_2) \in S]$$

for all $S \subseteq Range(\mathcal{A})$, where $Range(\mathcal{A})$ denotes the output range of the algorithm $\mathcal{A}$, and for any pair of neighboring datasets D_1 and D_2.

For a function $f : D \to \mathbb{R}^k$, the *sensitivity* [5] of f is defined as

$$\Delta(f) := \max \|f(D_1) - f(D_2)\|_1,$$

where the maximum is taken over all pairs of neighboring D_1 and D_2. It is shown [6] that the mechanism $\mathcal{A}$

$$\mathcal{A}(D) = f(D) + (Lap(\Delta(f)/\epsilon))^k$$

achieves ϵ-differential privacy, where $(Lap(\Delta(f)/\epsilon))^k$ is a vector of k independently and randomly chosen values from the Laplace distribution with standard deviation $2\Delta(f)/\epsilon$.

2.2 Isotonic Regression

Given a sequence of n real numbers $a_1, \ldots, a_n$, the problem of finding the least-square fit $x_1, \ldots, x_n$ subjected to the constraints $x_i \leq x_j$ for all $i < j \leq n$ is known as the isotonic regression. Formally, we want to find the $x_1, \ldots, x_n$ that minimizes

$$\sum_{i=1}^{n} (x_i - a_i)^2, \quad \text{subjected to } x_i \leq x_j \text{ for all } 1 \leq i < j \leq n.$$

The unique solution can be efficiently found using pool-adjacent-violators algorithms in $O(n)$ time [10]. When minimizing w.r.t. ℓ-1 norm, there is also an efficient $O(n \log n)$ algorithm [25]. There are many variants of isotonic regression, for example, variants with a smoothness component in the objective function [27,17].

2.3 Locality-Preserving Mapping

A locality-preserving mapping $T : [0,1]^d \to [0,1]$ maps d-dimensional points to the unit interval, while preserving locality. In this paper, we seek a mapping that, if the mapped points $T(x)$, $T(y)$ are "close", then x and y are "close" in the d-dimensional space. More specifically, there is some constant c s.t. for any x, y in the domain of the mapping T,

$$\|x - y\|_2 \leq c \cdot (\|T(x) - T(y)\|)^{1/d}. \tag{1}$$

The well-known Hilbert curve [9] is a locality-preserving mapping. It is shown that for any 2D points x, y in the domain of T, $\|x - y\|_2 \leq 3\sqrt{|T(x) - T(y)|}$. Niedermeier et al. [19] showed that with careful construction, the bound can be improved to $2\sqrt{|T(x) - T(y)|}$ for 2D points and $3.25\sqrt[3]{\|T(x) - T(y)\|}$ for 3D points. In our construction, for simplicity, we use Hilbert curve in our experiments.

Note that it is challenging in preserving locality "in the other direction", that is, any two "close" points in the d-dimensional domain are mapped to "close" points in the one-dimensional range [18]. Fortunately, in our problem, such property is not required.

2.4 Datasets

We conduct experiments on two datasets: locations of Twitter users [1] (herein called the Twitter location dataset) and the dataset collected by Kaluža et al. [12] (herein called Kaluža's dataset). The Twitter location dataset contains over 1 million Twitter users' data from the period of March 2006 to March 2010, among which around 200,000 tuples are labeled with location (represented in latitude and longitude) and most of the tuples are in the North American continent, concentrating in regions around the state of New York and California. Fig. 2(a) shows the cropped region covering most of the North American continent. The cropped region contains 183,072 tuples. The Kaluža's dataset contains 164,860 tuples collected from tags that continuously record the location information of 5 individuals. While some of the tuples consist of many attributes, in our experiments, only the 2D location data are being used.

3 Proposed Approach

Before receiving the data, the publisher has to make a few design choices. The publisher need to decide on a locality-preserving mapping T, and the strategy (which is represented as a lookup table) of determining the group size from the privacy requirement ϵ and the size of dataset n. Now, given the dataset D of size n, and the privacy requirement ϵ, the publisher carries out the following:

A1. The publisher maps each point in D to a real number in the unit interval $[0,1]$ using T, and lookups the group size based on n and ϵ. Let $T(D)$ be the set of transformed points. For clarity in exposition, let us assume that k divides n.

A2. The publisher sorts the mapped points, divides the sorted sequence into groups of k consecutive elements, and then for each group, determines its average over the k elements. Let the averages be $S = \langle s_1, \ldots, s_{n/k} \rangle$.
A3. The publisher releases $\widetilde{S} = S + (\mathrm{Lap}(\epsilon^{-1})/k)^{(n/k)}$ and the group size k.

A public user may extract information from the published data as follow:

B1. The user performs isotonic regression on $\widetilde{S}$ and obtains $\mathrm{IR}(\widetilde{S})$, and then replaces each element $\widetilde{s}_i$ in $\mathrm{IR}(\widetilde{S})$ with k points of value $\widetilde{s}_i$. Let P be the set of resulting points.
B2. The user maps the data point back to the original domain, that is, computes $\widetilde{D} = T^{-1}(P)$. Let us call $\widetilde{D}$ the reconstructed data.

Note that the public user is not confined to performing step B1 and B2. The user may, for example, incorporates some background knowledge to enhance accuracy. To relieve the public from computing step B1 and B2, the regression and the inverse mapping can be carried out by the publisher on behalf of the users. Nevertheless, the raw data $\widetilde{S}$ should be (although it is not necessary) published alongside the reconstructed data for further statistical analysis.

4 Security Analysis

In this section, we show that the proposed mechanism (Step A1 to A3) achieves differential privacy. The following theorem shows that sorting, as a function, interestingly has sensitivity 1. Note that a straightforward analysis that treats each element independently could lead to a bound of n, which is too large to be useful.

Theorem 1. *Let $S_n(D)$ be a function that on input D, which is a multi-set containing n real numbers from the unit interval $[0, 1]$, outputs the sorted sequence of elements in D. The sensitivity of S_n w.r.t. the bounded differential privacy is 1.*

Proof. Let D_1 and D_2 be any two neighboring datasets. Let $\langle x_1, x_2 \ldots x_i \ldots x_n \rangle$ be $S_n(D_1)$, i.e. the sorted sequence of D_1. WLOG, let us assume that an element x_i is replaced by a larger value A to give D_2, for some $1 \leq i \leq n-1$ and $x_i < A$. Let j to be largest index s.t. $x_j < A \leq 1$. Hence, the sorted sequence of D_2 is:

$$x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_j, A, x_{j+1}, \ldots, x_n.$$

The L_1 difference due to the replacement is,

$$\begin{aligned} & \|S_n(D_1) - S_n(D_2)\|_1 \\ = & |x_{i+1} - x_i| + |x_{i+2} - x_{i+1}| + |x_j - x_{j-1}| + |A - x_j| \\ = & (x_{i+1} - x_i) + (x_{i+2} - x_{i+1}) + (x_j - x_{j-1}) + (A - x_j) \\ = & A - x_i \leq 1. \end{aligned}$$

We can easily find an instance of D_1 and D_2 where the difference $A - x_i = 1$. Hence, the sensitivity is 1. □

Since the sensitivity is 1, the mechanism $S_n(D) + Lap(1/\epsilon)^n$ enjoys ϵ-differential privacy. Also note that the value of n is fixed. Hence, the size of D is not a secret and is made known to the public.

The following corollary shows (proof omitted) that grouping (in Step A2) has no effect on the sensitivity.

Corollary 1. *Consider a partition $H = \{h_1, h_2 \dots h_m\}$ of the indices $\{1, 2, \dots, n\}$. Let $S_H(D)$ be the function that, on input D, which is a multi-set containing n real numbers from the unit interval $[0, 1]$, outputs a sequence of m numbers:*

$$y_i = \sum_{j \in h_i} x_j,$$

for $1 \leq i \leq m$ where $\langle x_1, x_2, \dots, x_n \rangle$ is the sorted sequence of D. The sensitivity of S_H is 1.

Note that the grouping in step A2 is a special partition with equal-sized h_i's, whereas Corollary 1 gives a more general result where H can be any partition. From Corollary 1, the proposed mechanism achieves ϵ-differential privacy.

5 Analysis and Parameter Determination

The main goal of this section is to analyze the effect of the privacy requirement ϵ, dataset size n and the group size k on the error in the reconstructed data, which in turn provides a strategy in choosing the parameter k given n and ϵ.

Intuitively, when n and ϵ are fixed, the choice of parameter k affects the accuracy in following three ways: (1) a larger k decreases the number of constraints in isotonic regression, which leads to lower noise reduction; (2) a larger k reduces the effect of the Laplace noise; and (3) a larger k introduces higher generalization error due to averaging.

Our analysis consists of the following parts. We first describe our utility function in Section 5.1. In Section 5.2, we consider the case where $k = 1$ and empirically show that the expected error of a typical dataset can be well approximated by the expected error on a synthetic equally-spaced dataset. Let us call this error $Err_{n,\epsilon}$. Next in Section 5.3, we investigate and estimate the generalization error due to the averaging and show that with a reasonable assumption on the dataset distribution, the expected error can be approximated by $\frac{k}{4n}$. Let us call this error $Gen_{n,k}$. Finally, in Section 5.4, we consider the general case of $k \geq 1$ and give an approximation of the expected error in terms of $Err_{n,\epsilon}$ and $Gen_{n,k}$.

5.1 Error Function

We use an error function based on the earth mover's distance(EMD) [24] to quantify the utility of the published data. The EMD between two pointsets of equal size is defined to be the minimum cost of bipartite matching between the

two sets, where the cost of an edge linking two points is the cost of moving one point to the other. Hence, EMD can be viewed as the minimum cost of transforming one pointset to the other. Different variants of EMD differ on how the cost is defined. In this paper, we adopt the typical definition that defines the cost as the Euclidean distant between the two points.

In one-dimensional space, the EMD between two sets D and $\widetilde{D}$ is simply the L_1 norm of the differences between the two respective sorted sequences, i.e. $\|S_n(D) - S_n(\widetilde{D})\|_1$, which can be efficiently computed. Recall that $S_n(D)$ outputs the sorted sequence of elements in D. In other words,

$$\mathrm{EMD}(D, \widetilde{D}) = \sum_{i=1}^{n} |p_i - \widetilde{p}_i|, \tag{2}$$

where p_i's and $\widetilde{p}_i$'s are the sorted sequence of D and $\widetilde{D}$ respectively. Note that this definition assumes D and $\widetilde{D}$ have the same number of points, which is ensured by step B1 of our scheme.

Given a dataset D and the published dataset $\widetilde{D}$ of a mechanism $\mathcal{M}$ where $|D| = |\widetilde{D}| = n$, let us define the *normalized error* as $\frac{1}{n}\mathrm{EMD}(D, \widetilde{D})$ and denote $Err_{\mathcal{M},\mathcal{D}}$ the expected normalized error,

$$Err_{\mathcal{M},D} = Exp\left[\ \frac{1}{n}\ \mathrm{EMD}(D, \widetilde{D})\ \right], \tag{3}$$

where the expectation is taken over the randomness in the mechanism.

Our mechanism publishes $\widetilde{D}$ based on two parameters: the privacy requirement ϵ and the group size k. Therefore, let us write $Err_{\epsilon,k,D}$ for the expected normalized error of the dataset published in Step B2.

5.2 Effects on Isotonic Regression

Let us consider the expected normalized error when $k = 1$, in other words, we first consider the mechanism without grouping. In such case, the reconstructed dataset is $\mathrm{IR}(S_n(D) + \mathrm{Lap}(\epsilon^{-1})^n)$. Thus, the expected normalized error is

$$Err_{\epsilon,1,D} = Exp\left[\frac{1}{n}\ \mathrm{EMD}(D, \mathrm{IR}(S_n(D) + \mathrm{Lap}(\epsilon^{-1}))^n)\ \right].$$

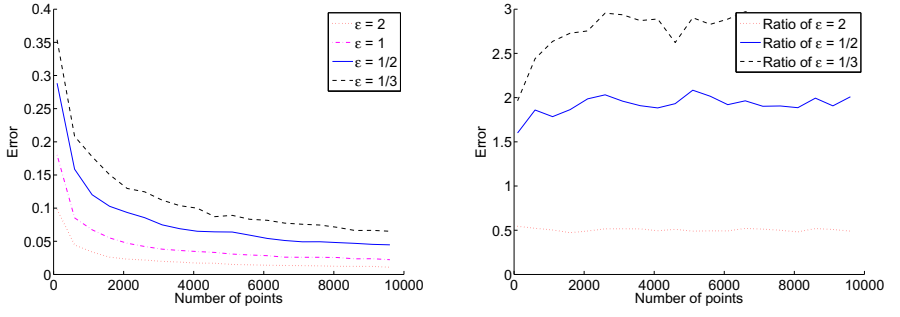
To estimate the above expected error, we compute the expected normalized error on a few datasets of varying size n: (1) Multi-sets containing elements with the same value 0.5 (herein called repeating single-value dataset), (2) sets containing equally-spaced numbers $(i/(n-1))$ for $i = 0, \ldots, n-1$ (herein call equally-spaced dataset), (3) sets containing n randomly chosen elements from the Twitter location data [1], and (4) sets containing n randomly chosen elements from the Kaluža's data [12].

Fig. 4(a) shows the expected error $Err_{1,1,D}$ for the four datasets with different n. Each sample in the graph is the average over 500 runs. Observe that the error

on equally-spaced data well approximates the errors on the two real-life dataset (Twitter location dataset and Kaluža's dataset). Hence, we take the error on the equally-spaced dataset as an approximation of the errors on other datasets. For abbreviation, let $Err_{\epsilon,n}$ denote the expected error $Err_{\epsilon,1,D}$ where D is the equally-spaced dataset with n points. Based on experiences on other datasets, we suspect that the expected error depends on the difference of the minimum and the maximum element in D, and the repeating single-value dataset is the extreme case whose error could be served as a lower bound as shown in Fig. 4(a).

Fig. 3(a) shows the expected error $Err_{\epsilon,1,D}$ for dataset on equally-spaced points for different ϵ and n, and Fig. 3(b) shows the ratios of error for different ϵ to $Err_{1,n}$. The results agree with the intuition that when ϵ is increased by a factor of c, the error would approximately decrease by factor of c, that is,

$$Err_{\epsilon,1,D} \approx \frac{1}{c} Err_{c\epsilon,1,D}. \tag{4}$$



(a) The normalized error $Err_{\epsilon,n}$. (b) The ratio of $Err_{\epsilon,n}$ to $Err_{1,n}$.

Fig. 3. The normalized error for different security parameter ϵ on equally-spaced dataset, each sample is the average of 500 runs

5.3 Effect on Generalization Noise

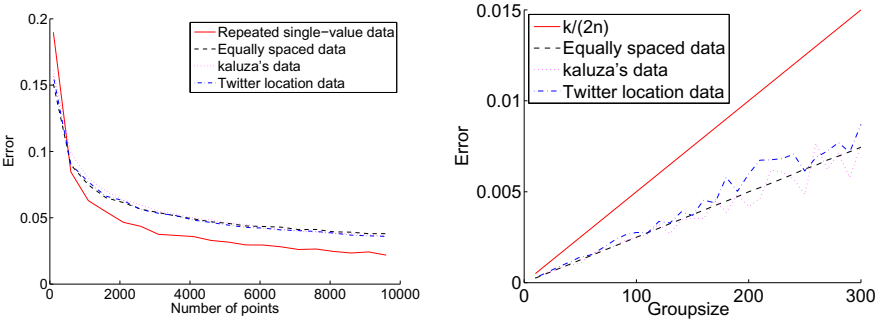
When $k > 1$, the grouping introduces a *generalization error*, which is incurred when all elements in a group are represented by their mean. Before giving formal description of generalization error, let us introduce some notations.

Given a sequence $D = \langle x_1, \ldots, x_n \rangle$ of n numbers, and a parameter k, where k divides n, let us call the following function *downsampling*:

$$\downarrow_k (D) = \langle s_1, \ldots, s_{(n/k)} \rangle,$$

where each s_i is the average of $x_{k(i-1)+1}, \ldots, x_{ik}$. Given a sequence $D' = \langle s'_1, \ldots, s'_m \rangle$ and k, let us call the following function *upsampling*,

$$\uparrow_k (D') = \langle x'_1, \ldots, x'_{mk} \rangle,$$



(a) Expected normalized error $Err_{1,1,D}$. (b) Normalized generalization error $Gen_{D,k}$.

Fig. 4. The expected normalized error and normalized generalization error with $\epsilon = 1$ on different dataset D

where $x'_i = s'_{\lfloor (i-1)/k \rfloor + 1}$ for each i.

The *normalized generalization error* is defined as,

$$Gen_{D,k} = \frac{1}{n} \| D - \uparrow_k (\downarrow_k (D)) \|_1.$$

It is easy to see that, for any k and D of size n, the normalized generalization error is at most $k/(2n)$. However, this bound is often an overestimate. Fig. 4(b) shows the generalization error of different group size a dataset containing $10,000$ equally-spaced values, a dataset containing $10,000$ numbers randomly drawn from the transformed Kaluža's dataset, and a dataset of $10,000$ numbers randomly drawn from the transformed Twitter location data.

Observe that, empirically, the generalization error can be well approximated by $\frac{k}{4n}$. To see that such approximation holds for a typical dataset, consider the following partition of the unit interval: $0 = p_0 < p_1 < p_2, \ldots, p_{(n/k)-1} < p_{n/k} = 1$. Let us consider a sorted sequence S of elements in dataset D, where the $jk+1, jk+2, \ldots (j+1)k$-th elements in S are uniformly independent and identically distributed over $[p_j, p_{j+1})$ for $j = 0, 1, \ldots, (n/k) - 1$. We can verify that the expected generalization error $Gen_{D,k} \approx \frac{k}{4n}$ with simulations. Hence, we approximate the generalization error by $\frac{k}{4n}$ and denote it as $Gen_{n,k}$.

5.4 Determining the Group Size *k*

Now, let us combine the components and build an error model of how k affects the accuracy. First, grouping reduces the number of constraints by a factor of k. As suggested by Fig. 4(a), when the number of constraints decreases, the error reduction from isotonic regression decreases. On the other hand, recall that the regression is performed on the published values divided by k (see the role of k in Step A3). This essentially reduces the level of Laplace noise by a factor of k. Hence, the accuracy attained by grouping k elements is "equivalent" to the

accuracy attained without grouping but with the privacy parameter ϵ increased by a factor of k. These two components can be estimated in terms of $Err_{\epsilon,n}$ as follow:

$$Err_{\epsilon,k,D} \approx \frac{1}{k} Err_{\epsilon,n/k}.$$

For general k, the reconstructed dataset is

$$\widetilde{D} = \uparrow_k (\mathrm{IR}(\widetilde{S})),$$

where $\widetilde{S}$ is an instance of $\downarrow_k (S_n(D)) + \mathrm{Lap}(1)^{n/k}$. Now, we have,

$$\begin{aligned}
&\mathrm{EMD}\ (D, \widetilde{D}) = \|S_n(D) - \uparrow_k (\mathrm{IR}(\widetilde{S}))\|_1 \\
&= \|S_n(D) - \uparrow_k (\downarrow_k (S_n(D)) + \uparrow_k (\downarrow (S_n(D))) - \uparrow_k (\mathrm{IR}(\widetilde{S}))\|_1 \\
&\leq n \cdot Gen_{D,k} + \| \uparrow_k (\downarrow_k (S_n(D))) - \uparrow_k (\mathrm{IR}(\widetilde{S}))\|_1 \\
&= n \cdot Gen_{D,k} + k \cdot \| \downarrow_k (S_n(D)) - \mathrm{IR}(\widetilde{S})\|_1 \\
&= n \cdot Gen_{D,k} + k \cdot \mathrm{EMD}(\downarrow_k (S_n(D)), \mathrm{IR}(\widetilde{S})). \qquad (5)
\end{aligned}$$

Note that the first term $n \cdot Gen_{D,k}$ is a constant independent of the random choices made by the mechanism. Also note that the second term is the EMD between the down-sampled dataset and its reconstructed copy obtained using group size 1. Thus, by taking expectation over randomness of the mechanism, we have

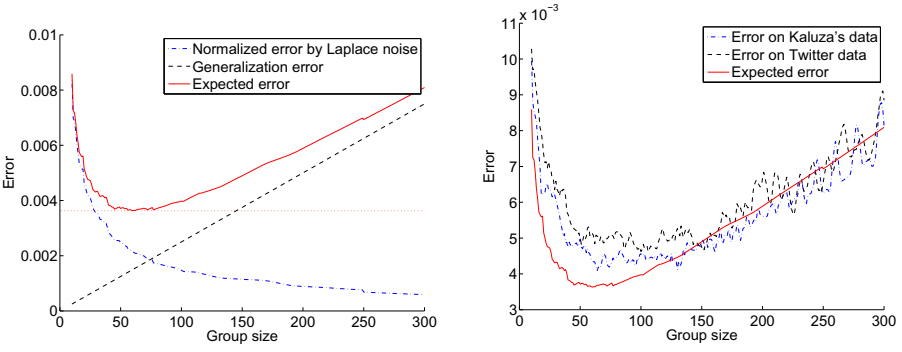
$$Err_{\epsilon,k,D} \leq Gen_{D,k} + \frac{1}{k} Err_{\epsilon,1,\downarrow_k(D)}. \qquad (6)$$

In other words, the expected normalized error is bounded by the sum of normalized generalization error, and the normalized error incurred by the Laplace noise. Fig. 5(a) shows the three values versus different group size k for equally-spaced data of size 10,000. The minimum of the expected normalized error suggests the optimal group size k.

Fig. 5(b) illustrates the expected errors for different k on the Twitter location data with 10,000 points. The red dotted line is $Err_{\epsilon,k,D}$ whereas the blue solid line is the sum in the right-hand-side of the inequality (6). Note that the differences between the two graphs are small. We have conducted experiments on other datasets and observed similar small differences. Hence, we take the sum as an approximation to the expected normalized error,

$$Err_{\epsilon,k,D} \approx Gen_{n,k} + \frac{1}{k} Err_{\epsilon,n/k}. \qquad (7)$$

Now, we are ready to find the optimal k given ϵ and n. From Fig. 4(a) and Fig. 4(b) and the approximation given in equation (7), we can determine the best group size k when given the size of the database n and the security requirement ϵ. From the parameter ϵ, we can obtain the value $\frac{1}{k} Err_{n/k,e}$ for different k. From the database's size n, we can determine $Gen_{n,k}$ which is $\frac{k}{4n}$. Thus, we can approximate the normalized error $Err_{k,D}$ with equation (7) as illustrated in Fig. 5(a). Using the same approach, the best group size given different n and ϵ can be calculated and is presented in table 1.



(a) The expected error. (b) Comparison with the actual error.

Fig. 5. The expected error derived based on the equally-spaced dataset and the comparison with actual error on the Kaluža's dataset with $\epsilon = 1$

Table 1. The best group size k given n and ϵ

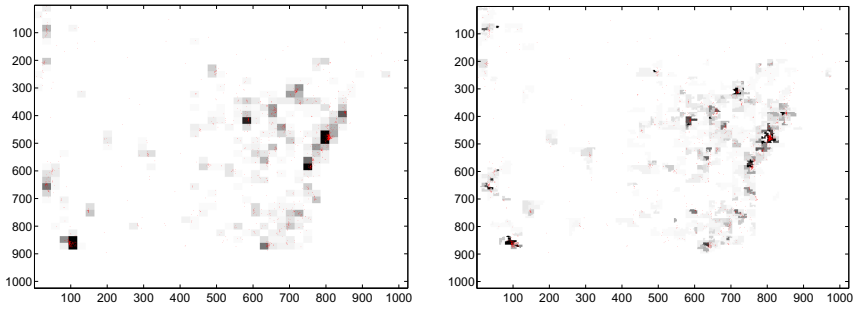
	$\epsilon = 0.5$	$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$
n= 2,000	44	29	20	12
n= 5,000	59	37	27	18
n= 10,000	79	51	36	27
n= 20,000	121	83	61	41
n= 100,000	234	150	98	73
n= 180,000	300	177	110	94

6 Comparisons

In this section, we compare the performance of the proposed mechanism with three known mechanisms w.r.t. different utility functions. We first compare the mechanism that outputs equi-width histograms. Next, we investigate the wavelet-based mechanism proposed by Xiao et al. [28] and measure accuracy of range queries. Lastly, we consider the problem of estimating median, and compare with a mechanism based on smooth sensitivity proposed by Nissim et al. [20]. We do not conduct experiments to compare with the k-d tree method [29] because it is designed for high dimensional data and it is not clear how to apply it to low dimension effectively. For comparison purposes, we empirically choose the best parameters for the known mechanisms, although this apriori information is not available to the publisher. We remark that the parameter k of our proposed mechanism is chosen from Table 1.

6.1 Equi-width Histogram

We want to compare the performance of our method with the equi-width histogram method. Fig. 6(a) shows a differentially private equi-width histogram. To



(a) Equi-width method. (b) Proposed method.

Fig. 6. Visualization of the density functions, where the darker region corresponds to higher value. The superposing red dots are randomly selected from original data points for comparison purposes.

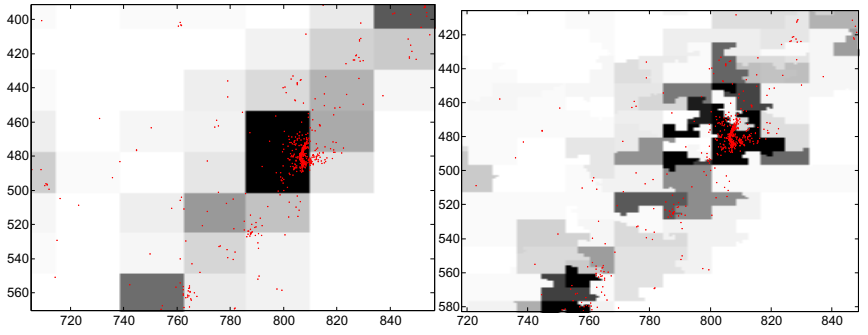
visualize the reconstructed points of our method as a histogram, we construct the bins in the following way: let B be the set of *distinct*-points in D, and we construct the Voronoi diagram of B. The cells in the Voronoi diagram are taken to be the bins of a histogram as depicted in Fig. 6(b).

To facilitate comparison, we treat the histograms as estimations of the underlying probability density function f, and use the statistical distance between density functions as a measure of utility. The value of $f(x)$ can be estimated by the ratio of the number of samples, over the width of the bin where x belongs to, with some normalizing constant factor.

In this section, we qualify the mechanism's utility by the distance between the two density functions: one that is derived from the original dataset, and the other that is derived from the mechanism's output.

Fig. 6(a) and 6(b) show the estimated density function from the Twitter's location dataset, by equi-width histogram mechanism and by our mechanism. For comparisons, 1% of the original points are plotted on top of the two reconstructed density functions. Fig. 7(a) and 7(b) show the zoom-in view of the dense region around New York City. Observe that the density function produced by our mechanism has "variable-sized" cells and thus is able to adaptively capture the fine details.

The statistical difference, measured with ℓ_1-norm and ℓ_2-norm, between the two estimated density functions derived from the original and the mechanism's output are shown in Table 2. We remark that it is not easy to determine the optimal bin-width for the equi-width histogram prior to publishing. Fig. 8 shows that the optimal bin-width differs significantly for three different datasets. For comparison purposes, we empirically choose the best parameters to the advantage of the compared algorithms, although such parameters could be dependent on the dataset.



(a) Zoom in view of Fig. 6(a). (b) Zoom in view of Fig. 6(b).

Fig. 7. A more detailed view of the density functions

6.2 Range Query

We consider the scenario where a dataset is to be published, and subsequently used to answer a series of range queries, where each range query asks for the total number of points within a query range. Publishing an equi-width histogram would not attain high accuracy if the size of the query ranges varies drastically. Intuitively, wavelet-based techniques [28] are natural solutions to address such multi-scales queries. However, there are many parameters, including the bin-widths at various scales and the amounts of privacy budget they consume, to be determined prior to publishing.

To apply the proposed method in this scenario, given a query, we obtain the number of points within the range from the estimated density function (as described in Section 6.1) by accumulating the probability over the query region and then multiplying by the total number of points.

We compare the range query results of the wavelet-based mechanism, the equi-width histogram mechanism and our mechanism on the 1D Twitter data, and on the 2D Twitter location dataset. To incorporate the knowledge of the database's size n, the total number of points is adjusted to n for the histogram mechanism and the DC component of the wavelet transform is set to be exactly n for the wavelet mechanism. For each range query, the absolute difference between the the true answer and the answer derived from the mechanism's output is taken as the error. We compare the results over different query range sizes and for each query range. For each range size s, 1,000 randomly chosen queries of size s are asked, and the corresponding errors are recorded. More precisely, the center of a 1D query range of size s is chosen uniformly at random in the continuous interval $[\frac{s}{2}, 1 - \frac{s}{2}]$, whereas the center of a 2D query range of size s is chosen uniformly at random in the region $[\frac{s}{2}, 1 - \frac{s}{2}] \times [\frac{s}{2}, 1 - \frac{s}{2}]$.

To determine the parameters for the two compared mechanisms, we conduct experiments on a few selected values and choose the values to the advantage of

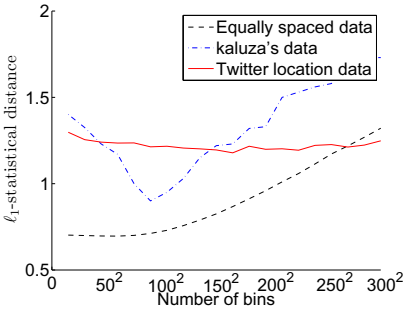
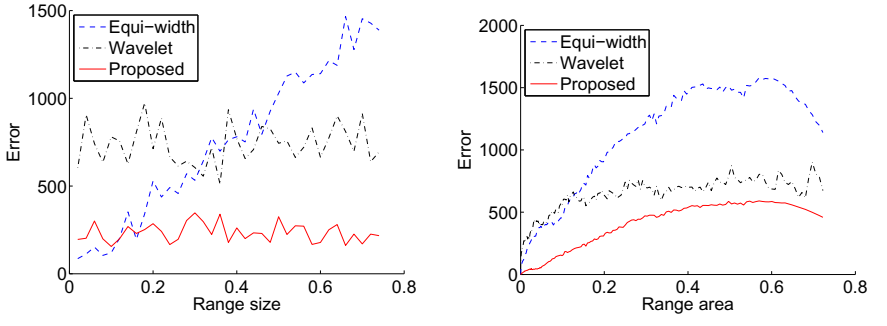


Fig. 8. The statistical differences versus bin-widths for different dataset D with $\epsilon = 1$ and $n = 60,000$

Table 2. The statistical differences of the two methods

	equi-width	proposed method
ℓ_1-norm	1.23	1.13
ℓ_2-norm	0.25	0.20



(a) 1D range query. (b) 2D range query.

Fig. 9. Comparison of range query performance

the compared mechanisms. For the equi-width histogram, the only parameter is the number of bins (n_1). For the wavelet-based mechanism, the parameter we considered is the number of bins (n_2) of the histogram whereby wavelet transformation is performed on, that is, the number of bins in the "finest" histogram. From our experiments, we choose $n_1 = 1000$ and $n_2 = 1024$ for the 1D data, and $n_1 = 40 \times 40$ and $n_2 = 512 \times 512$ for the 2D data. The parameter k for our mechanism is looked up from Table 1. The choice of group size k according to Table 1 is 177 ($n = 180,000, \epsilon = 1$). The average errors of the range query is shown in Fig. 9(a) and 9(b).

Observe that our proposed method is less sensitive to the query range in the 1D case as expected because the accuracy of our range query results depend only on the boundary points, as opposed to the equi-width histogram method where errors are induced by each bins within the range. The wavelet-based mechanism outperforms the equi-width histogram mechanism in larger size range queries, but performs badly for small range due to the accumulation of noise.

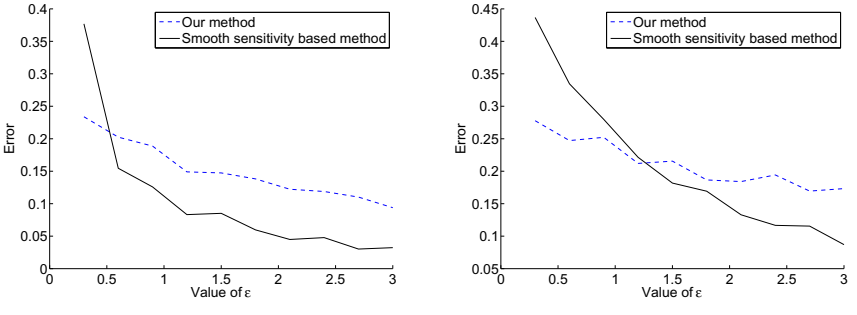
6.3 Median

Finding the median accurately in a differentially private manner is challenging due to the high "global sensitivity": there are two datasets that differ by one element but having a completely different median. Nevertheless, for many instances, their "local sensitivity" are small. Nissim et al. [20] showed that in general, by adding noise proportional to the "smooth sensitivity" of the database instance, instead of the global sensitivity, can also ensure differential privacy. They also gave an $\Theta(n^2)$ algorithm that find the smooth sensitivity w.r.t. median.

Our mechanism outputs the sorted sequence differentially privately, and thus naturally gives the median. Compare to the smooth sensitivity-based mechanism, our mechanism provides more information in the sense that it outputs the whole sorted sequence. Furthermore, our mechanism can be efficiently carried out in $O(n \log n)$ time.

We conduct experiments on synthetic datasets of size 129 to compare the accuracy of both mechanisms. The experiments are conducted for different local sensitivity and different ϵ values. To construct a dataset with a particular local sensitivity, 66 random numbers are generated with the exponential distribution and then scaled to the unit interval. The dataset contains the 66 random numbers and 63 ones. Fig. 10(a) and 10(b) shows the noise level with different ϵ on datasets that has a local sensitivity of 0.1 and 0.3.

When the local sensitivity of the median is high, our mechanism tends to provide a better result. In addition, our mechanism performs well under higher requirement of security: when the ϵ is smaller, the accuracy of our mechanism decreases slower than the smooth sensitivity-based method.



(a) Local sensitivity of 0.1. (b) Local sensitivity of 0.3.

Fig. 10. The error of median versus different ϵ from two datasets

7 Discussion and Future Work

7.1 Hybrid Method

The proposed mechanism can be viewed as the publishing of a "equi-depth" histogram, where the "depth" is the group size. Potentially, our proposed method

and equi-width histogram could complement each other, by alternatively publishing one after another. For example, we can first apply an equi-width histogram to get a coarse distribution of the data, followed by our method for a "zoom-in" view. Alternatively, we can apply equi-width histogram after our method to "break" the stepping effect of isotonic regression.

7.2 Effect of Dimension

We rely on a locality-preserving mapping T to extend the mechanism to higher dimension. Although it is shown that the distant between two d dimensional points x, y is preserved and bounded by $c(\|T(x) - T(y)\|)^{1/d}$ (see Section 2.3), the curse-of-dimensionality is still in play in higher dimension. Firstly, to our best knowledge, there is no known efficient constructions for dimensions higher than 3. Secondly, the exponential factor $1/d$ amplifies the error: for example, Fig. 5 shows that our scheme can reduce the error of $\|T(x) - T(y)\|$ to 0.005, where y is the reconstructed point for x. When d is 2, we can have $\|x - y\|_2$ bounded by $0.07c$; when d is 3, the bound is increased to $0.17c$. We are unable to verify the performance in higher dimension due to the lack of efficient construction, and leave the accurate extension to higher dimensional data as future work.

8 Related Work

There are extensive works on privacy-preserving data publishing. The recent survey by Fung et al. [8] gives a comprehensive overview on various notions, for example, k-anonymity [26], ℓ-diversity [15], and the recently proposed concept of differential privacy [5].

Xiao et al. [28] proposed a mechanism of adding Laplace noise to the coefficients of a wavelet transformation of an equi-width histogram. The noisy wavelet coefficients are then published, from which range queries can be answered. Essentially, what being published is a series of equi-width histograms for different bin-widths where the noise added to the histograms of larger bin-width are smaller. A range query can then be decomposed and answered from the histograms series different scales.

Isotonic regression has been used to improve a differentially private query result. Hay et al. [11] proposed a method that employs isotonic regression to boost accuracy, but in a way different from our mechanism. They consider publishing *unattributed histogram*, which is the (unordered) multi-set of the frequencies of a histogram. As the frequencies are unattributed (i.e. order of appearance is irrelevant), they proposed publishing the sorted frequencies and later employing isotonic regression to improve accuracy.

Machanavajjhala et al. [16] proposed a 2D dataset publishing method that can handle the sparse data in 2D equi-width histogram. To mitigate the sparse data, their method shrinks the sparse blocks by examining publicly available data such as a previously release of similar data. They demonstrate this idea on the commuting patterns of the population of the United States, which is a

real-life sparse 2D map in large domain. As their method partitions the space based on a previously released data, we consider the partition as pre-determined partition and is not adaptive to the publishing dataset.

The median is an important statistic, and a differentially private median finding process can be useful in many constructions, such as in pointset spatial decomposition [4,23]. However, finding the median differentially privately is not easy due to the large global sensitivity. Nissim et al. [20] introduced the notion of smooth sensitivity and proposed an accurate mechanism with $\Theta(n^2)$ running time.

9 Conclusion

Our mechanism is very simple from the publisher's point of view. The publisher just has to sort the points, group consecutive values, add Laplace noise and publish the noisy data. There is also minimal tuning to be carried out by the publisher. The main design decision is the choice of the group size k, which can be determined using our proposed noise models, and the locality-preserving mapping for which the classic Hilbert curve suffices to attain high accuracy. Through empirical studies, we have shown that the published raw data contain rich information for the public to harvest, and provide high accuracy even for usages like median-finding and range-searching that our mechanism is not initially designed for.

References

1. Twitter census: Twitter users by location, http://www.infochimps.com/datasets/twitter-census-twitter-users-by-location
2. Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., Talwar, K.: Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In: Symposium on Principles of Database Systems, pp. 273–282 (2007)
3. Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: the sulq framework, pp. 128–138 (2005)
4. Cormode, G., Procopiuc, M., Shen, E., Srivastava, D., Yu, T.: Differentially private spatial decompositions. To be appeared in ICDE (2012)
5. Dwork, C.: Differential privacy. Automata, languages and programming, p. 1 (2006)
6. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
7. Feldman, D., Fiat, A., Kaplan, H., Nissim, K.: Private coresets, p. 361 (2009)
8. Fung, B., Wang, K., Chen, R., Yu, P.: Privacy-preserving data publishing: A survey of recent developments. ACM Computing Surveys, 14 (2010)
9. Gotsman, C., Lindenbaum, M.: On the metric properties of discrete space-filling curves. IEEE Transactions on Image Processing, 794–797 (1996)
10. Grotzinger, S., Witzgall, C.: Projections onto order simplexes. Applied Mathematics and Optimization, 247–270 (1984)
11. Hay, M., Rastogi, V., Miklau, G., Suciu, D.: Boosting the accuracy of differentially private histograms through consistency. VLDB Endowment, 1021 (2010)

12. Kaluža, B., Mirchevska, V., Dovgan, E., Luštrek, M., Gams, M.: An Agent-Based Approach to Care in Independent Living. In: de Ruyter, B., Wichert, R., Keyson, D.V., Markopoulos, P., Streitz, N., Divitini, M., Georgantas, N., Mana Gomez, A. (eds.) AmI 2010. LNCS, vol. 6439, pp. 177–186. Springer, Heidelberg (2010)
13. Kifer, D., Machanavajjhala, A.: No free lunch in data privacy. In: Management of Data, pp. 193–204 (2011)
14. Li, C., Hay, M., Rastogi, V., Miklau, G., McGregor, A.: Optimizing linear counting queries under differential privacy, pp. 123–134 (2010)
15. Machanavajjhala, A., Gehrke, J., Kifer, D., Venkitasubramaniam, M.: ℓ-diversity: Privacy beyond k-anonymity. In: International Conference on Data Engineering, pp. 24–24 (2006)
16. Machanavajjhala, A., Kifer, D., Abowd, J., Gehrke, J., Vilhuber, L.: Privacy: Theory meets practice on the map. In: International Conference on Data Engineering, pp. 277–286 (2008)
17. Meyer, M.C.: Inference using shape-restricted regression splines. Annals of Applied Statistics, 1013–1033 (2008)
18. Mitchison, G., Durbin, R.: Optimal numberings of an n x n array. Algebraic Discrete Methods, 571–582 (1986)
19. Niedermeier, R., Reinhardt, K., Sanders, P.: Towards optimal locality in mesh-indexings, pp. 364–375 (1997)
20. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis, pp. 75–84 (2007)
21. Piatetsky-Shapiro, G., Connell, C.: Accurate estimation of the number of tuples satisfying a condition, pp. 256–276 (1984)
22. Poosala, V., Haas, P., Ioannidis, Y., Shekita, E.: Improved histograms for selectivity estimation of range predicates. In: ACM SIGMOD Record, p. 294 (1996)
23. Qardaji, W., Li, N.: Recursive partitioning and summarization: a practical framework for differentially private data publishing. To be appeared in ASIACCS (2012)
24. Rubner, Y., Guibas, L., Tomasi, C.: The earth movers distance, multi-dimensional scaling, and color-based image retrieval, pp. 661–668 (1997)
25. Stout, Q.F.: Optimal algorithms for unimodal regression. Computer Science and Statistics, 109–122 (2000)
26. Sweeney, L.: k-anonymity: a model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based System, 557–570 (2002)
27. Wang, X., Li, F.: Isotonic smoothing spline regression. Journal Computational and Graphical Statistics, 21–37 (2008)
28. Xiao, X., Wang, G., Gehrke, J.: Differential privacy via wavelet transforms. IEEE Transactions on Knowledge and Data Engineering, 1200–1214 (2010)
29. Xiao, Y., Xiong, L., Yuan, C.: Differentially Private Data Release through Multidimensional Partitioning. In: Jonker, W., Petković, M. (eds.) SDM 2010. LNCS, vol. 6358, pp. 150–168. Springer, Heidelberg (2010)

PRISM – Privacy-Preserving Search in MapReduce

Erik-Oliver Blass[1], Roberto Di Pietro[2], Refik Molva[3], and Melek Önen[3]

[1] Northeastern University, Boston, MA
[2] Università di Roma Tre, Rome, Italy
[3] EURECOM, Sophia Antipolis, France

Abstract. We present PRISM, a privacy-preserving scheme for word search in cloud computing. In the face of a curious cloud provider, the main challenge is to design a scheme that achieves privacy while preserving the efficiency of cloud computing. Solutions from related research, like encrypted keyword search or Private Information Retrieval (PIR), fall short of meeting real-world cloud requirements and are impractical. PRISM's idea is to transform the problem of word search into a set of parallel instances of PIR on small datasets. Each PIR instance on a small dataset is efficiently solved by a node in the cloud during the "Map" phase of MapReduce. Outcomes of map computations are then aggregated during the "Reduce" phase. Due to the linearity of PRISM, the simple aggregation of map results yields the final output of the word search operation. We have implemented PRISM on Hadoop MapReduce and evaluated its efficiency using real-world DNS logs. PRISM's overhead over non-private search is only 11%. Thus, PRISM offers privacy-preserving search that meets cloud computing efficiency requirements. Moreover, PRISM is compatible with standard MapReduce, not requiring any change to the interface or infrastructure.

1 Introduction

Today, users take advantage of public clouds operated by large companies like Google or Amazon. Instead of setting up and maintaining their own data centers, users reduce their costs by outsourcing both storage and processing to a cloud. One prominent example allowing cloud-based storage and processing is Hadoop MapReduce [3], a variant of Google's MapReduce system [17]. Hadoop MapReduce is widely used, and public MapReduce clouds are offered by companies such as Amazon [2, 25].

The advantages of cloud computing unfortunately come with a high cost in terms of new security and privacy exposures. Apart from classical security challenges of shared services, outsourcing of data storage and processing raises new challenges in the face of potentially malicious cloud providers. *Privacy* of outsourced data appears to be a major requirement in this context. Some regulations are already provisioned as to the privacy protection of outsourced governmental documents [11, 12, 18]: these regulations usually aim at assuring privacy against *curious clouds* or against clouds with data centers located in "rogue" countries or with insufficient security guarantees; they also are defined to avoid data leakage in case of operational failures in the cloud. Along these lines, there is also a raising corporate concern about the privacy of sensitive business data stored in the cloud [14]. Although cloud providers thrive to meet the increased privacy

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 180–200, 2012.

demand by certifying their services [24], malicious insiders have still been identified as one of the top threats in cloud computing [15].

While encryption of outsourced data by the users seems to be a viable protection against most privacy problems, traditional data encryption does not suit the requirements of cloud computing: the cloud not only serves as high capacity memory, but is also involved in data processing such as statistical data analysis, log analysis, indexing, data mining, and searching [25]. However, data processing performed by the cloud would not be feasible or would be *inefficient* with encrypted data.

Among data processing primitives, word search, i.e., verifying, whether a certain word is part of a dataset, is not only one of the most fundamental operations, but surprisingly also one of the most demanded applications in, e.g., MapReduce cloud computing [25]. **Related work** on search in encrypted data, e.g., Boneh et al. [6], Ogata and Kurosawa [31], falls short of meeting cloud computing privacy and performance requirements. These techniques are *impractical* as they are designed for centralized execution models that are incompatible with today's highly parallel cloud architectures.

In this paper, we present PRISM, a new scheme for *privacy-preserving and efficient word search* for MapReduce clouds. PRISM pursues two specific objectives: 1.) privacy against potentially malicious cloud providers and 2.) high efficiency through the integration of security mechanisms with the operations performed in the cloud.

In order to achieve efficiency, PRISM takes advantage of the inherent parallelization akin to cloud computing: the word search problem on a very large encrypted dataset is partitioned into several instances of word search in small datasets that are executed in parallel ("Map" phase). The individual word search operations performed in the cloud yield a result amenable to straightforward *aggregation* in the ultimate phase ("Reduce" phase) of the word search operation. The word search operation builds on a Private Information Retrieval (PIR) [29] technique which is extended in order to generate intermediate search results that are still encrypted and that can be *combined* through linear operations to yield the global result of the word search over the entire dataset.

Summarizing our contributions, PRISM:

• **is suited to cloud computing:** PRISM is the first privacy-preserving search scheme suited to cloud computing; it brings together storage and search privacy with high performance by leveraging the efficiency of the MapReduce paradigm. PRISM is parallelizable and also allows efficient combination of individual results. Its efficiency has been evaluated through searching in DNS logs provided by an Internet Service Provider. Although PRISM's overhead within the core Map function is large compared to non-privacy-preserving search (factor of 9), the total system overhead is only 11%.

• **preserves privacy in the face of potentially malicious cloud providers:** PRISM allows carrying out these critical operations in the cloud without trusting the cloud.

• **is compatible to standard MapReduce:** PRISM only requires a standard MapReduce interface without modifications in the underlying system. PRISM can thus be integrated on any cloud that provides a standard MapReduce interface such as Amazon.

• **provides flexible search:** In contrast to traditional encrypted keyword search techniques, PRISM is not limited to searching for a fixed set of *predetermined* keywords to be known in advance, but offers flexible search for any words.

2 Problem Statement and Adversary Model

Throughout this paper, we will use an application example to motivate our work. Inspired by recent events [30], we envision a *data retention* scenario. Due to regulatory matters, a small, residential Internet Service Provider (ISP) must retain logs of client accesses. Due to the sheer amount of data to retain, the ISP outsources logfiles to the cloud. Files are encrypted as they contain sensitive data. Still, e.g., law enforcement authorities will contact the ISP to search for words (strings, text, ...) in outsourced files.

More concretely, assume service provider $\mathcal{U}$ (the cloud "user") providing DNS services to clients. $\mathcal{U}$ logs each client's access, i.e., $\mathcal{U}$ logs the tuple (timestamp, client ID, hostname queried). Due to the large amount of log data and cost reasons, $\mathcal{U}$ outsources its logfiles into a cloud. Regularly, say each day i, $\mathcal{U}$ creates a new logfile L_i. At the end of a longer period, $\mathcal{U}$ wants to (or is forced to) find out, whether there was an interest in a suspicious host w. So, $\mathcal{U}$ checks, at which day, i.e., in which logfiles L_i, word w occurs. $\mathcal{U}$ queries the cloud for w, and the cloud responses with an answer R telling $\mathcal{U}$ which of the L_i contain(s) w.

Note that $\mathcal{U}$ does not know in advance which word w it has to search for. This automatically disqualifies protocols for *predefined* keyword search, such as PEKS [6] and derivatives. Also, data retention regulations require outsourced data to be fully recoverable; storing only digests of data in the cloud, e.g., hash values, is insufficient.

The cloud is assumed to be untrusted, more precisely *semi-honest* ("honest-but-curious"). Regulatory matters imply that the cloud must not learn any information about the content it hosts and search queries performed. This implies both, the encryption of data by $\mathcal{U}$ before outsourcing it to the cloud and "obliviously" processing queries on encrypted data by the cloud.

Before we formalize our privacy requirements, we first define the main components for a cloud word search scheme.

Definition 1 (Cloud Word Search). *Let $\mathcal{L}$ denote a sequence of files $\mathcal{L} := \{L_1, \ldots, L_m\}$ and Σ the set of possible words. Each file L_i consists of a sequence of words $L_i := \{w_{(i,1)}, w_{(i,2)}, \ldots, w_{(i,|L_i|)}\}, w_{(i,j)} \in \Sigma$.*

A cloud word search scheme comprises the following algorithms:

1. *$KeyGen(s)$: using a security parameter s, this algorithm outputs a secret $\mathcal{S}$.*
2. *$Encrypt(\mathcal{S}, \mathcal{L})$: uses the secret $\mathcal{S}$ to encrypt the content of files $L_i \in \mathcal{L}$ and outputs the set of resulting encryptions $\mathcal{E} := \{E_{L_1}, \ldots, E_{L_m}\}$. Here, E_{L_i} denotes the encryption of file L_i.*
3. *$Upload(\mathcal{E})$: uploads $\mathcal{E}$ to the cloud.*
4. *$PrepareQuery(\mathcal{S}, w)$: takes $\mathcal{S}$, the word w to search for, and produces a query Q.*
5. *$Process(\mathcal{E}, Q)$: with encryptions $\mathcal{E}$ and query Q, this algorithm produces result R.*
6. *$Decode(\mathcal{S}, R, w)$: taking result R, secret $\mathcal{S}$, and word w, this algorithm outputs the set of indices $\mathcal{I} := \{i_1, \ldots, i_r\}$ such that $\forall i \in \mathcal{I} : L_i \in \mathcal{L} \wedge w \in L_i$, if $R = Process(\mathcal{E}, Q)$ with $Q = PrepareQuery(\mathcal{S}, w)$, and $\mathcal{E} = Encrypt(\mathcal{S}, \mathcal{L})$.*

The basic interaction between user $\mathcal{U}$ and the cloud can be summarized as follows: first, user $\mathcal{U}$ encrypts and uploads files. Then, $\mathcal{U}$ prepares a search query Q for word w and sends Q to the cloud. The cloud processes this query using algorithm $Process$

and produces output R. This output is sent back to $\mathcal{U}$. Using output R and another algorithm $Decode$, $\mathcal{U}$ can compute the list of files containing w. While describing PRISM's details later in Section 4, we will show how they map to these algorithms.

Note that, in a cloud setting, $\mathcal{U}$ executes $KeyGen$, $Encrypt$, $Upload$, $PrepareQuery$, and $Decode$, while the cloud executes $Process$. The idea is that algorithms $KeyGen$, $Encrypt$, $Upload$, $PrepareQuery$, and $Decode$ are computationally very "lightweight" for $\mathcal{U}$ compared to $Process$. The main computational burden lies on the cloud side.

Privacy Requirements

Intuitively, our application demands for two main types of privacy. The cloud (now called "adversary $\mathcal{A}$") must neither be able to infer any details about stored files nor learn details about $\mathcal{U}$'s queries and results delivered back to $\mathcal{U}$. This implies not only the secrecy or confidentiality of the content, but also the inability to compute statistics on the content. Informally, in our setting:

- given $\mathcal{E}$, $\mathcal{A}$ must not learn the content of $\mathcal{L}$ and must not discover whether files L_i contain a word w, e.g., multiple times;
- given a set of queries $\{Q_i\}$, $\mathcal{A}$ must not learn the words $\{w_i\}$ $\mathcal{U}$ is looking for and must not discover whether the same word is queried multiple times;
- given the result R_i of a query Q_i, $\mathcal{U}$ must not learn which file(s) contain the word corresponding to this specific query W_i.

Instead, the adversary should only learn "trivial" properties, such as the total number of files, the file size, and the total number of queries. Along the same lines as traditional indistinguishability [23], $\mathcal{A}$ should not be able to infer any additional information from encrypted files, queries, and results. We formally define privacy for a cloud word search scheme using a game between adversary $\mathcal{A}$ (the cloud) and a challenger (user $\mathcal{U}$).

Definition 2 (Privacy). *Let $\mathcal{W}$ denote a sequence of words $\mathcal{W} := \{w_1, \ldots, w_n\}$. The game* GAME *is played as follows.*

1. *The challenger executes $KeyGen(s)$ to derive secret $\mathcal{S}$.*
2. *$\mathcal{A}$ selects a distinct pair of sequences of files and words $(\mathcal{L}_0, \mathcal{W}_0)$ and $(\mathcal{L}_1, \mathcal{W}_1)$, where $|\mathcal{L}_0| = |\mathcal{L}_1|$, $\forall(L_i^0 \in \mathcal{L}_0, L_i^1 \in \mathcal{L}_1) : |L_i^0| = |L_i^1|$, and $|\mathcal{W}_0| = |\mathcal{W}_1|$.*
   $\mathcal{A}$ sends $(\mathcal{L}_0, \mathcal{W}_0)$ and $(\mathcal{L}_1, \mathcal{W}_1)$ to the challenger.
3. *The challenger randomly selects $b \in \{0, 1\}$ and*
   - *executes $Encrypt(\mathcal{S}, \mathcal{L}_b)$, i.e., the challenger computes encrypted files $\mathcal{E}_b = \{E(\mathcal{S}, L_i) | L_i \in \mathcal{L}_b\}$.*
   - *executes $Upload(\mathcal{E}_b)$ to send encrypted files back to $\mathcal{A}$.*
   - *executes $PrepareQuery(\mathcal{S}, w)$ for each w in $\mathcal{W}_b$. This results in the sequence of queries $\mathcal{Q}_b := \{Q_1, \ldots, Q_{|\mathcal{W}_b|}\}$ that the challenger also sends to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs $b' \in \{0, 1\}$. The outcome of* GAME *is "1" iff $b' = b$.*

A cloud word search scheme is called privacy-preserving iff

$$Pr(\text{GAME}(\mathcal{A}) = 1) \leq \frac{1}{2} + \epsilon(s)$$

for all probabilistic polynomial-time adversaries $\mathcal{A}$. Here, $\epsilon(s)$ is a negligible function, $\epsilon(s) < \frac{1}{P(s)}$ for every polynomial P with sufficiently large security parameter s.

The specification of the $(\mathcal{L}_i, \mathcal{W}_i)$ in step 2. of Definition 2 reflects the fact that $\mathcal{A}$ can learn the total number of files, the size of each file, and the number of queries.

Limitations: We consider semi-honest clouds. Fully malicious clouds might perform DoS-attacks or deviate from protocol execution. Similar to "reaction attacks" [26], the cloud might return garbage to $\mathcal{U}$, to observe $\mathcal{U}$'s reaction (e.g., sending the same query). Although realistic, we leave such attacks for future work. Also, our privacy definition does not capture trivial privacy properties, e.g., the size of outsourced files. Mitigation strategies (e.g., padding files) might be contradictory to cloud efficiency. We conjecture that, for many applications, losing "trivial" privacy properties is acceptable.

3 Background

3.1 MapReduce

We target a system suited for the MapReduce [3] paradigm. We will now give a condensed overview of MapReduce, focusing on aspects necessary to understand PRISM.

Upload. A MapReduce cloud comprises a set of "slave" node computers and a "master" computer. While $\mathcal{U}$ uploads files into the MapReduce cloud, each file is automatically split into blocks called *InputSplits*. InputSplits have a fixed size $S_{\text{InputSplit}}$ which is a pre-configured system parameter. If S_{File} denotes the size of an uploaded file, the number of InputSplits c computes to $c = \frac{S_{\text{File}}}{S_{\text{InputSplit}}}$. For each InputSplit, a workload sharing algorithm selects a slave node and places the InputSplit on it.

In addition to data, the MapReduce also allows $\mathcal{U}$ to upload "operations", i.e., compiled Java classes. These classes represent the implementation of three functions.
1.) $Scan(\text{INPUTSPLIT}) \rightarrow [(k, v)])$, a functions that takes an InputSplit as an input, parses it, i.e., scans it and generates a set of key-value pairs $[(k, v)]$ out of it.
2.) $Map(k, v) \rightarrow [(k', v')]$, a function that takes as an input a single key-value pair (k, v) and outputs a set of "intermediate" key-value pairs $[(k', v')]$.
3.) $Reduce([(k', v')]) \rightarrow \text{FILE}$, a function that takes as an input a set of intermediate key-value pairs $[(k', v')]$ and writes arbitrary output into a file.

Uploaded Java classes are sent to all slave nodes storing an InputSplit.

Map Phase. After data and implementations have been uploaded, $\mathcal{U}$ specifies one uploaded file and triggers MapReduce operations on that file. The first phase of operation is the "Map" phase. Each slave node becomes a "mapper" node. Each mapper executes $\mathcal{U}$'s $Scan$ function on the InputSplit it stores locally. This generates a set of key-value pairs on each mapper. Furthermore, the mapper node executes $\mathcal{U}$'s Map function on this generated key-value pairs to produce a set of intermediate key-value pairs.

Reduce Phase. MapReduce starts the "Reduce" phase. Slave nodes are scheduled to become "reducers". For each of the intermediate pairs (k', v'), MapReduce selects a reducer and sends (k', v') to this reducer. MapReduce selects the *same* reducer for all pairs (k', v') having the same key. Each reducer executes $\mathcal{U}$'s reduce function on its set of intermediate key-value pairs and writes the output to a file. This file is sent to $\mathcal{U}$.

3.2 Trapdoor Group Private Information Retrieval

PIR allows a user to retrieve data from a server without revealing which data is retrieved. For PRISM, we make use of a simple and efficient PIR mechanism as previously suggested by Trostle and Parrish [37]. As this mechanism is just a building block for PRISM, we will only give a summary of its mode of operation and rationale.

Overview: Matrix $\mathcal{M}$ is a $t \times t$ matrix of elements in $\mathbb{Z}_N$ stored at a server. For example, $N = 2$ for a binary matrix. User $\mathcal{U}$ is only interested in receiving elements of the k^{th} row in $\mathcal{M}$, but the server must not learn k. The idea is now that $\mathcal{U}$ sends two "types" of values to the server. For each row that $\mathcal{U}$ is not interested in, he sends a value of the "first" type. For the one row that $\mathcal{U}$ is interested in, he sends a single value of the "second" type. To prevent the server from distinguishing between the two types of values, $\mathcal{U}$ blinds each value with a blinding factor b. This blinding factor can later be removed by $\mathcal{U}$. The server now performs simple additions with received values and elements stored in $\mathcal{M}$. The result is sent back to $\mathcal{U}$ who removes the blinding factor and determines the values of the row of his interest.

Preparation: Assume $\mathcal{U}$ is interested in row k. $\mathcal{U}$ chooses a group $\mathbb{Z}_p$, with a prime p of m bits. $\mathcal{U}$ also chooses a random $b \in \mathbb{Z}_p$ and t random values $a_i \in \mathbb{Z}_p$. Therewith, $\mathcal{U}$ computes t values $e_i < \frac{p}{t \cdot (N-1)}$ such that: $e_k := 1 + a_k \cdot N$ **and** $\forall i \neq k : e_i := a_i \cdot N$.

Finally, $\mathcal{U}$ computes $\alpha_i := b \cdot e_i \mod p$ and sends the α_i to the server. Other values $(p, m, b, \{e_i\}, \{a_i\})$ remain secret. The server treats α_i as large integers and performs the following integer operations, i.e., without any modulo.

Server computation: Let $\boldsymbol{u}$ be the vector $\boldsymbol{u} := (\alpha_1, \ldots, \alpha_t)$. The server computes the matrix product $\boldsymbol{v}$ and sends it back to $\mathcal{U}$,

$$\boldsymbol{v} := (\beta_1, \ldots, \beta_t) = \boldsymbol{u} \cdot \mathcal{M} = (\sum_{i:=1}^{t} \alpha_i \cdot \mathcal{M}_{i,1}, \ldots, \sum_{i:=1}^{t} \alpha_i \cdot \mathcal{M}_{i,t}).$$

Result analysis: Upon receipt, in order to "un-blind" values, $\mathcal{U}$ computes the t inverse values $z_i := \beta_i \cdot b^{-1} \mod p$. Now, U can conclude that $z_i \mod N$ equals the i^{th} element of the k^{th} row in $\mathcal{M}$. Therewith, $\mathcal{U}$ has retrieved the t elements of the k^{th} row of $\mathcal{M}$ in a privacy-preserving fashion. Note the linearity for two β_i and β_i' received during different PIR runs: $\beta_i \cdot b^{-1} + \beta_i' \cdot b^{-1} = (\beta_i + \beta_i') \cdot b^{-1} \mod p$. That is, the sum of two individually un-blinded vectors equals un-blinding the sum of two received vectors $\boldsymbol{v}, \boldsymbol{v'}$. We will later use this linearity during PRISM's reduce phase.

Security Rationale: Security and privacy of this protocol are based on the trapdoor group assumption. With only knowledge of α_i, but not secret trapdoor p, it is computationally hard for the server to infer any information about low order bits, i.e., the modulo of z or e_i, cf., Trostle and Parrish [37].

Discussion: Again, we stress that this particular PIR scheme is an exchangeable building block. In general, any of the "traditional" PIR techniques based on group homomorphic encryption [29, 33] is suited for use within PRISM. We have chosen Trostle and Parrish [37] only due to the straightforward way to implement it (Section 6). Other PIR schemes might reduce the (already small) overhead, but this is out of scope here.

4 PRISM Protocol

PRISM comprises three parts: *upload* of data into the cloud, the MapReduce *search*, and the *result analysis* where the user decides whether the word has been found. We will briefly give an overview about each part.

1.) Upload. During upload, $\mathcal{U}$ encrypts each word (of a logfile) using symmetric encryption. Ciphertexts are stored in a file, and this file is sent to the MapReduce cloud. The cloud automatically splits large files and distributes splits (*InputSplits*) among *mapper* nodes. We use a standard blockcipher (AES) to perform ciphering of words. However, to ensure privacy as of Definition 2, plaintext is modified before encryption using a "stateful cipher" construction. Therewith, $\mathcal{U}$ can still search for some word w, but the cloud cannot compute statistics about ciphertexts.
2.) Search. Eventually, $\mathcal{U}$ wants to search his encrypted files for some word w. Therefore, $\mathcal{U}$ sends implementations of "algorithms" for the map and reduce phases to the MapReduce cloud, and the cloud executes these on uploaded data. For example, $\mathcal{U}$ sends Java ".class" files for the mappers and Java ".class" files for *reducer* nodes. MapReduce distributes these implementations to each mapper and reducer, respectively. PRISM's rationale is to *transform* the word search problem into a set of small PIR instances. To do so, each mapper, scanning through its locally stored InputSplit, creates a binary matrix. Ciphertexts in the InputSplit are assigned to individual elements in that matrix. If a ciphertext is present in an InputSplit, its corresponding element in the matrix is set to either "0" or "1". Using private information retrieval techniques, PRISM can extract the value of a single element in the matrix with the mapper being totally oblivious to which element is extracted. Consequently, $\mathcal{U}$ can specify which element to extract in a privacy-preserving way. All mappers send their obliviously extracted elements as key-value pairs to reducers. Reducers simply sum up received values and return sums to $\mathcal{U}$. Therewith, neither mappers nor reducers can learn any information about which ciphertext $\mathcal{U}$ was interested in.
3.) Result analysis. Finally, $\mathcal{U}$ receives an encrypted sum for each of the originally uploaded files from reducers. $\mathcal{U}$ can decrypt them and decide which of the files contain w. However, due to the probability of "collisions" in matrices, i.e., two different ciphertexts can be assigned to the same element, and due to ambiguities of received sums, $\mathcal{U}$'s decision whether w is inside some file might be wrong. Therefore, PRISM repeats the above process in a total of q so called "rounds". In each of the rounds, a new matrix is generated, elements are set to "1" or "0" depending on the round number, and results are returned as described. This reduces the probability of $\mathcal{U}$ making incorrect decisions.

Initialization: Before the actual uploading, initially, and only once, $\mathcal{U}$ has to execute $KeyGen$. In PRISM, $KeyGen$ outputs secret $\mathcal{S} := \{K, N, p\}$, where $|K|, |N|, |p|$ are

Input: words w_i
Output: ciphertexts C_i uploaded to cloud

```
1 Initialize all γ to 0;
2 foreach word w_i do
3 |   γ_{w_i} := get (w_i);  //from hash table
4 |   γ_{w_i} := γ_{w_i} + 1;
5 |   insert (w_i, γ_{w_i});  //into hash table
6 |   C_i := E_{K_d}(w_i, γ_{w_i});
7 |   upload C_i ;
8 end
```

Algorithm 1. "Stateful Cipher" example and upload to MapReduce

specified by security parameter s. K is a symmetric key, and N and p are Trapdoor Group PIR parameters as presented in Section 3.2.

4.1 Upload

Overview. In our scenario, cloud user $\mathcal{U}$ continuously logs customer access and sends logfiles to the cloud. Each day, $\mathcal{U}$ starts using a new logfile. For simplicity, we assume that entries logged by $\mathcal{U}$ are simple words. Each logfile is encrypted word by word using a "stateful cipher" E_K, and resulting ciphertexts are written to a file, respectively. The encrypted files are sent to the cloud.

Definition 3 (Stateful Cipher). *Given standard symmetric encryption E_K with key K, e.g., AES, we extend E to a* stateful cipher *by adding "counters" γ_{w_i} that count the history of inputs w_i. Each time E encrypts w_i, counter γ_{w_i} is increased by one.*

In conclusion, a stateful cipher is a cipher that knows how often it has encrypted a specific plaintext. The following presents one trivial stateful cipher construction used in PRISM to encrypt before uploading.

Stateful Cipher Example (see Algorithm 1): For simplicity, user $\mathcal{U}$ uses a secret key K to derive a different key for each day d, e.g., $K_d := \mathrm{HMAC}_K(d)$.

For each day, $\mathcal{U}$ maintains a hash table containing the list of counters γ_{w_i} in $\mathcal{U}$'s local storage. At the beginning of each day, $\mathcal{U}$ initializes all counters to 0, i.e., $\gamma_{w_i} = 0$. Now, for each logentry w_i that should be stored in the cloud, $\mathcal{U}$ computes γ_{w_i} and increases γ_{w_i} by 1. Then, $\mathcal{U}$ computes ciphertext $C_i := E_{K_d}(w_i, \gamma_{w_i})$. User $\mathcal{U}$ sends ciphertext C_i to the cloud that stores it in this day's file. For the (AES) encryption $E_{K_d}(w_i, \gamma_{w_i})$, "," denotes an unambiguous pairing of inputs. We discuss the reason for using a "stateful cipher" over using, e.g., a CBC mode of encryption in Section 5.1.

Summarizing, with respect to Definition 1, *Encrypt* in PRISM takes K to derive a separate key K_d for each file to be encrypted. Actual encryption of each file is performed word by word using the stateful cipher and key K_d, so $\mathcal{E} := \{E_{L_1}, \ldots, E_{L_m}\}$ where $E_{L_i} := \{E_{K_i}(w_1, \gamma_{w_1}), \ldots, E_{K_i}(w_{|L_i|}, \gamma_{w_{|L_i|}})\}$. *Upload* in PRISM can be regarded as simply sending the encrypted files $\mathcal{E}$ to MapReduce.

4.2 Search

User $\mathcal{U}$ wants to search a set of files for word w within a period of time. For ease of understanding, we will restrict our description below to PRISM working on a single file

specified by the user, i.e., the file of day d. With multiple files, all files will be separately (but in parallel) processed with PRISM exactly like with a single file.

$\mathcal{U}$ sends map and reduce implementations of PRISM to MapReduce, and the map phase starts. In the following, we describe the PRISM algorithms for, first, the mappers and in Section 14 the reducers. We would like to stress that the PRISM algorithms, e.g., Java ".class" files, are not encrypted and not specially protected against a curious cloud. Even though mappers and reducers know what operations they perform, they cannot deduce any private information about stored data or details about the search.

Overview. Before scanning through its local InputSplit, a mapper node creates a matrix with all elements initialized to "0". PRISM's main idea is that while the mapper scans the ciphertexts in its InputSplit, each ciphertext is assigned to one position, a certain element in the matrix by computing a hash of the ciphertext. Additionally, for each ciphertext, the mapper computes a single bit hash, and if the hash output bit is "1", the mapper puts a "1" in the matrix at the assigned position. The idea is that user $\mathcal{U}$ can also compute the position in the matrix and the one bit hash output for a word w he is looking for. Roughly speaking, $\mathcal{U}$ now queries the mapper for the value of that bit in the matrix using private information retrieval. If the bit retrieved from the mapper differs from the bit computed by $\mathcal{U}$, then $\mathcal{U}$ can decide, e.g., that w is not in this InputSplit.

Problem is that due to the limited size of the matrix and the properties of the hash function, there might be collisions in the assignment process. That is, by chance there can be two different ciphertexts being assigned with the same position in the matrix. By chance, the bit retrieved by $\mathcal{U}$ can therefore be unrelated to w. This problem is amplified by the fact that $\mathcal{U}$ does not only receive a single bit for a single InputSplit, but a combination (the *sum*) of all bits from all mappers working on InputSplits. To mitigate this problem, PRISM repeats generation and filling of matrices a total of q rounds. Also, setting an element in a matrix to "1" depends on the round number. After q rounds, the probability that the information $\mathcal{U}$ retrieved from this mapper is unrelated to w therefore decreases, and $\mathcal{U}$ can finally decide whether w is inside this file.

Definition 4 (PIR Matrix). *A binary $t \times t$ matrix $\mathcal{M}$ with $t = 2^i, i \in \mathbb{N}$ is called a PIR matrix. The mapper uses $\mathcal{M}$ to implicitly perform the privacy-preserving word search.*

Definition 5 (Candidate Position). *For each ciphertext C_i in an InputSplit, the candidate position $(\mathcal{X}_i, \mathcal{Y}_i)$ of C_i in $\mathcal{M}$ is computed by $(\mathcal{X}_i || \mathcal{Y}_i) := \lfloor C_i \rfloor_{2 \cdot \log_2(t)}$. Here, $\lfloor \ldots \rfloor_{2 \cdot \log_2(t)}$ denotes truncation after $2 \cdot \log_2(t)$ bits. So, the first $\log_2 t$ bits of C_i determine $\mathcal{X}_i$, and the second $\log_2 t$ bits determine $\mathcal{Y}_i$.*

Definition 6 (PIR Input). *If $\mathcal{U}$ is interested in a specific element $(\mathcal{X}, \mathcal{Y})$ in $\mathcal{M}$, he computes PIR input $\{\alpha_1, \alpha_2, \ldots, \alpha_t\}$, where $\alpha_{\mathcal{X}} := b \cdot (1 + a_{\mathcal{X}} \cdot N) \mod p$, and $\forall i \neq \mathcal{X}, \alpha_i := b \cdot (a_i \cdot N) \mod p$. Random values b and a_i are chosen as for the Trapdoor Group PIR scheme presented in Section 3.2.*

Definition 7 (Column Sum). *The column sum σ_i of the i^{th} column of PIR matrix $\mathcal{M}$ is defined as*

$$\sigma_i := \sum_{\mathcal{M}_{1 \leq j \leq t, i} = 1} \alpha_j,$$

where $\mathcal{M}_{1 \leq j \leq t, i} = 1$ denotes the entries in the i^{th} column of $\mathcal{M}$ that are set to 1.

Note that additions in this definition are integer additions.

The above computation of column sums is simply a digest of the PIR technique by Trostle and Parrish [37]. In short, if a mapper computes such a column sum on a given PIR matrix $\mathcal{M}$ and given PIR inputs α_i, it is impossible for the mapper to derive $(\mathcal{X}, \mathcal{Y})$. $\mathcal{U}$, however, can compute whether $\mathcal{M}_{\mathcal{X},\mathcal{Y}} = 1$, because $\mathcal{M}_{\mathcal{X},\mathcal{Y}} = 1$ **iff** $(\sigma_{\mathcal{Y}} \cdot b^{-1} \mod p) \mod 2 = 1$ holds.

It is important to point out that not only a mapper can compute a candidate position for some ciphertext in its InputSplit, but also $\mathcal{U}$ can compute candidate positions. More precisely, as $\mathcal{U}$ is looking for w, he can compute $E(w, 1)$ and candidate position $(\mathcal{X}||\mathcal{Y}) := \lfloor E(w, 1) \rfloor_{2 \cdot \log_2(t)}$. If w has been uploaded into a particular InputSplit at least once, then this InputSplit contains at least $E(w, 1)$ (maybe also $E(w, 2)$, $E(w, 3)$, ...). Therefore, it is sufficient for $\mathcal{U}$ to search for $E(w, 1)$. We will now give detailed descriptions of $PRISM$'s Map and Reduce algorithms.

Query preparation – User. To start, $\mathcal{U}$ chooses parameters $t, q \in \mathbb{N}$, where t determines the size of the PIR matrix and q the number of rounds. For day d that $\mathcal{U}$ wants to search for w, he determines key $K_d := \mathrm{HMAC}_K(d)$ and the target candidate position $(\hat{\mathcal{X}}||\hat{\mathcal{Y}}) := \lfloor E_{K_d}(w, 1) \rfloor_{2 \cdot \log_2(2)}$. To prepare PIR, $\mathcal{U}$ computes t PIR Inputs $\{\alpha_1, \alpha_2, \ldots, \alpha_t\}$ as described above. $\mathcal{U}$ sends all α as part of the following map algorithm implementation to the cloud.

The above preparation of PIR Input depending on w represents $PrepareQuery$ of Definition 1 in PRISM. The algorithm's output Q is the PIR Input. PRISM's implementation of $Process$, i.e., the cloud's operation on the encrypted file using a query Q comprises the following cloud-side Map as well as the whole cloud-side reduce below.

Map Details – Cloud. On the cloud side, all mappers process PRISM in parallel, each of them on its own, locally stored InputSplit of the current file. More precisely, a mapper executes Algorithm 2. Initially, the mapper generates q PIR matrices M_l, where each element is initially set to 0. We will now write $\mathcal{M}_{l,\mathcal{X},\mathcal{Y}}$ to denote an element $(\mathcal{X}, \mathcal{Y})$ in matrix $\mathcal{M}_l$.

The mapper node *scans* its local InputSplit consisting of ciphertexts $\{C_1, \ldots, C_n\}$. For each ciphertext C_i, the mapper creates a key-value pair (i, C_i). Then, the mapper fills matrices $\mathcal{M}_l, 1 \leq l \leq q$. For pair (i, C_i),

- the mapper computes candidate position $(\mathcal{X}_i||\mathcal{Y}_i) := \lfloor C_i \rfloor_{2 \cdot \log_2(t)}$.
- the mapper puts in PIR matrix $\mathcal{M}_j$, in element $\mathcal{M}_{j,\mathcal{X}_i,\mathcal{Y}_i}$, a "1", if the bit $bit_j := \lfloor h(C_i, j) \rfloor_1 = 1$. Here, h denotes a cryptographic hash function and "," again an unambiguous pairing of inputs. If $bit_j = 0$, element $\mathcal{M}_{j,\mathcal{X}_i,\mathcal{Y}_i}$ remains untouched. This means that entries in $\mathcal{M}_j$ can flip from 0 to 1, but never from 1 back to 0.

After all q PIR matrices are filled, the mapper computes for each matrix the t *column sums* $\sigma_{1 \leq j \leq t, 1 \leq l \leq q}$ based on $\mathcal{U}$'s input $\{\alpha_1, \ldots, \alpha_t\}$: values α_k with corresponding element $\mathrm{M}_{l,k,j}$ set to "1" are simply added. Finally, the mapper outputs intermediate key-value pairs (k, v). The *key* comprises the name of the file of the InputSplit this mapper was working on, e.g., the file name could be day d, and the number of the column sum of $\mathcal{M}_l$. The *value* consists of a list of the q column sums. These intermediate key-value pairs will now be input for the reducers during the Reduce phase.

Input: pairs (i, C_i), values $\{\alpha_1, \ldots, \alpha_t\}$
Output: intermediate key-value pairs (k, v)

```
1  for l := 1 to q do
2  |  INITIALIZE M_l;
3  end
4  SCANTHROUGHINPUTSPLIT;
5  foreach pair (i, C_i) do  //Fill
   matrices
6  |  (X_i || Y_i) := ⌊C_i⌋_{2·log_2(t)};
7  |  for j := 1 to q do
8  |  |  bit_j := ⌊h(C_i, j)⌋_1;
9  |  |  if bit_j = 1 then
10 |  |  |  M_{j,x_i,y_i} := 1;
11 |  |  end
12 |  end
13 end
14 for l := 1 to q do  //q rounds
15 |  for j := 1 to t do  //Compute
   |  column sums
16 |  |  σ_{j,l} := Σ_{M_{l,1≤k≤t,j}=1} α_k;
17 |  end
18 end
19 for j := 1 to t do  //Intermediate
   (k,v) pairs
20 |  (k, v) :=({FILE, j},{σ_{j,1}, ..., σ_{j,q}});
21 |  OUTPUT (k, v);
22 end
```

Algorithm 2. Computation of matrices $\mathcal{M}$

Input: reducers' files FILE
Output: decision whether $w \in$ FILE

```
1  foreach file FILE do
2  |  for i := 1 to q do
3  |  |  if ⌊h(C, i)⌋_1 = 1 then
4  |  |  |  U reads s_{FILE,Y,i};
5  |  |  |  s_i := (s_{FILE,Y,i} · b^{-1}
   |  |  |  mod p)  mod N
6  |  |  |  //s_i = bit_j, see
   |  |  |  Alg. 2
7  |  |  |  if s_i = 0 then
8  |  |  |  |  OUTPUT w ∉ FILE;
   |  |  |  |  //Contradiction
9  |  |  |  |  break;
10 |  |  |  end
11 |  |  end
12 |  end
13 |  OUTPUT w ∈ FILE;
14 end
```

Algorithm 3. $\mathcal{U}$ decides $w \in$ FILE

Reduce Phase – Overview. Recall that there are c InputSplits and therefore c mappers. A single reducer receives from all the c mappers working on the same file all their q column sums for the *same* column. The reducer simply adds these received sums and writes the result into a file which is sent back to $\mathcal{U}$.

Reduce Phase – Details. For all key-value pairs $[(\{\text{FILE}, i\},\{\sigma_{i,1}, \ldots, \sigma_{i,q}\})]$ using the same $\{\text{FILE}, i\}$ as key, the MapReduce framework designates the same reducer. This reducer receives from all c different mappers working on the same file all intermediate key-value pairs with the same key. That is, a reducer receives c pairs which we rewrite as $(\{\text{FILE}, i\}, \{\sigma_{i,1,1}, \ldots, \sigma_{i,q,1}\}), \ldots, (\{\text{FILE}, i\}, \{\sigma_{i,1,c}, \ldots, \sigma_{i,q,c}\})$.

Here, for a given $\sigma_{i,j,k}$, i, $1 \leq i \leq t$, denotes the column, j, $1 \leq j \leq q$, denotes the round, and k, $1 \leq k \leq c$, the InputSplit.

Using integer addition, reducer computes q "final PIR sums" $s_{\text{FILE},i,j} := \sum_{k=1}^{c} \sigma_{i,j,k}$, $1 \leq j \leq q$, and stores values $\{s_{\text{FILE},i,1}, \ldots, s_{\text{FILE},i,q}\}$ into an output file R. To summarize, $s_{\text{FILE},i,j}$ represents the sum of column sums of all the mappers of one particular column i in PIR matrix j. This concludes the cloud's $Process$ algorithm in PRISM. The output file R is downloaded by $\mathcal{U}$.

4.3 Result Analysis

The only piece left is the $Decode$ algorithm of Definition 1 which we will describe in the following. For each outsourced file (day d), user $\mathcal{U}$ retrieves an output file generated

by reducers. Now, $\mathcal{U}$ analyzes retrieved files' content to finally conclude which of the outsourced files contain w (using $\mathcal{S}$). Again for ease of understanding, we restrict our description to the analysis of the result generated from PRISM on a single outsourced file called FILE. $\mathcal{U}$ repeats this process with all other results from the other files accordingly.

Definition 8 (Collision). *Assume $\mathcal{U}$ is looking for w, so $C := E_{K_d}(w, 1)$. Similar to hash functions, a* collision *in PIR matrix $\mathcal{M}$ denotes the case of an event where the candidate position $(\mathcal{X}', \mathcal{Y}')$ of another ciphertext $C' \neq C$ matches the candidate position $(\hat{\mathcal{X}}||\hat{\mathcal{Y}}) = \lfloor E(w, 1) \rfloor_{2 \cdot \log_2(t)}$ of w in $\mathcal{M}$. That is, $\lfloor C \rfloor_{2 \cdot \log_2(t)} = \lfloor C' \rfloor_{2 \cdot \log_2(t)}$.*

Definition 9 (One-Collision). *A* one-collision *is the event where in an InputSplit a ciphertext $C' \neq E_{K_d}(w, 1)$ puts a 1 into the same candidate position in $\mathcal{M}$ as $E_{K_d}(w, 1)$.*

Overview: The rationale for the result analysis protocol of PRISM is to observe the candidate position of C over q rounds to mitigate the effect of one-collisions. Of particular interest will be rounds where $\lfloor h(C, i) \rfloor_1 = 1$.

First, $\mathcal{U}$ un-blinds all values received from reducers. Based on the result, $\mathcal{U}$ distinguishes two cases.

Case 1.) If a reducer, reducing for a specific file FILE, has returned the value 0 for C's candidate position, then U knows for sure that all mappers have output 0 for this candidate position. Consequently, the candidate position in matrix $\mathcal{M}$ of each mapper is 0. Therefore, C has not been in any of the InputSplits of FILE, and $\mathcal{U}$ reasons $w \notin$ FILE. If C would have been in one InputSplit, then at least the mapper working on this InputSplit would have returned a 1 in this round.

Definition 10 (Contradiction). *Let w be the word $\mathcal{U}$ is looking for, and C its ciphertext. If in some round i, $\lfloor h(C, i) \rfloor_1 = 1$ holds, and the reducer for file* FILE *sends $\mathcal{U}$ a value of* 0 *then this is called a* contradiction.

In case of such a contradiction, $\mathcal{U}$ for sure knows that w is not in file FILE.

Case 2.) If, however, this reducer returns a value > 0, then w was in at least one InputSplit *or* a one-collision has occurred in at least one InputSplit. User $\mathcal{U}$ can neither decide $w \notin$ FILE nor $w \in$ FILE with absolute certainty.

U's strategy is to keep the probability for one-collisions low and run multiple rounds q, such that eventually a contradiction occurs ($\Rightarrow \mathcal{U}$ decides $w \notin$ FILE), or, if no contradiction occurs, $\mathcal{U}$ decides $w \in$ FILE with only a small error probability P_{err}.

Details: $\mathcal{U}$ executes Algorithm 3. For each file, $\mathcal{U}$ is only interested in row $\hat{\mathcal{Y}}$ of matrices $\mathcal{M}$, as they can refer to candidate position $(\hat{\mathcal{X}}, \hat{\mathcal{Y}})$, only. Therefore, $\mathcal{U}$ keeps values $\{s_{\text{FILE},\hat{\mathcal{Y}},1}, \ldots, s_{\text{FILE},\hat{\mathcal{Y}},q}\}$ only and discards the rest. In each round where $\lfloor h(C, i) \rfloor_1 = 1$, un-blinds $s_{\text{FILE},\hat{\mathcal{X}},i}$ to get value $s_i := \sum_{j=1}^{c} bit_j$. If $s_i = 0$, then we have a contradiction, and $\mathcal{U}$ can infer $w \notin$ FILE. If none of the s_i values has been 0 after all the q rounds, then $\mathcal{U}$ will decide $w \in$ FILE. $\mathcal{U}$ will be wrong with P_{err}.

Note that, although PIR matrices are binary matrices, $\mathcal{U}$ sets $N > c$ to cope with the larger possible values that sums might take due to collisions.

In conclusion, $\mathcal{U}$'s strategy can be summarized by: output $w \notin$ FILE, if $\exists i, s_i = 0$ **or** output $w \in$ FILE, if $\forall i, s_i \neq 0$. We will compute $\mathcal{U}$'s error probability P_{err} for the latter case and dependencies between P_{err} and values t and q in Section 5.2.

Saving Computation: To save some computation in PRISM, we can modify the hash-based mechanism that determines whether to put a "1" or a "0" in a certain element in $\mathcal{M}$. Recall that the first $2 \cdot log_2(t)$ bit of a ciphertext C are used to determine its position (element) in $\mathcal{M}$. However, instead of computing an expensive hash function $\lfloor h(C_i, j) \rfloor_1$ to get a single bit in round j, we can simply replace the hash and take C's bit on position $(2 \cdot log_2(t) + j)$. Assuming that cipher E has good security properties (each bit of C is "1" with probability $\frac{1}{2}$), this results in the same property as using the hash: eventually two different ciphertexts that collide in $\mathcal{M}$ will differ and lead to a contradiction. We use this computation reduction in our evaluation in Section 6.

5 PRISM Analysis

5.1 Privacy

We will now show why PRISM is privacy-preserving. The main *rationale* behind our proof is to show that pairs of output generated by both our stateful cipher construction $E_K(w_i, \gamma_{w_i})$ (Section 4.1) and the PIR-based search mechanism (Section 4.2) are computationally indistinguishable for $\mathcal{A}$. Below, we assume a sufficiently large security parameter s and probabilistic polynomial time adversaries $\mathcal{A}$.

Theorem 1. *PRISM is a privacy-preserving cloud word search scheme assuming pseudorandom properties for E and the trapdoor group property of the PIR scheme.*

Proof (Sketch). Assume there would be an adversary $\mathcal{A}$ with $Pr(\text{GAME}(\mathcal{A}) = 1) > \frac{1}{2} + \epsilon(s)$, i.e., $\mathcal{A}$ has non-negligible advantage over guessing. As PRISM generates $\mathcal{E}_b$ and $\mathcal{Q}_b$ independently from each other, this would indicate that $\mathcal{A}$ has non-negligible advantage over guessing in determining b from either $\mathcal{E}_b$ or $\mathcal{Q}_b$ (or both).

We will now show with the following two lemmas that this is impossible.

Lemma 1. *In PRISM, any pair of sequences of ciphertexts (files E_L and $E_{L'}$) generated by a pair of sequences of words (files L and L') is computationally indistinguishable for $\mathcal{A}$, assuming E is a pseudorandom permutation and "," an unambiguous pairing of inputs.*

Proof (Sketch). First, note that our stateful-cipher uses a different random key for each file. In a learning phase, $\mathcal{A}$ makes a number of queries to two stateful-cipher oracles encrypting with two different keys K_0, K_1. Then, $\mathcal{A}$ prepares word w, submits to a challenge oracle and gets back $E_{K_b}(w, \gamma_{b,w})$, $b \in \{0, 1\}$. $\mathcal{A}$ has to output b correctly with only negligible advantage over guessing.

However, we now show by using the hybrid argument [28] that the distributions generated by $E_{K_i}(w, \gamma_{i,w})$ are computationally indistinguishable for $\mathcal{A}$. That is, pairs $E_{K_0}(w, \gamma_{0,w})$, $E_{K_1}(w, \gamma_{1,w})$ are distinguishable with only negligible advantage over guessing). As "," is an unambiguous pairing, we now write w'_i instead of $(w, \gamma_{i,w})$.

Our hybrid distributions are: (1.) $PRP_{K_0}(w'_0)$, (2.) $RP_{K_0}(w'_0)$, (3.) $RF_{K_0}(w'_0)$, (4.) $RF_{K_1}(w'_1)$, (5.) $RP_{K_1}(w'_1)$, and (6.) $PRP_{K_1}(w'_1)$. "PRP" means pseudorandom permutation, "RP" random permutation, and "RF" random function.

(1.) - (2.) and (5.) - (6.): by definition of pseudorandom permutation, the probability to distinguish $PRP_K(w'_i)$ from $RP_K(w'_i)$ is negligible.

(2.) - (3.) and (4.) - (5.): the probability to distinguish a random permutation from a random function is negligible, cf., Section 3.6.3 in Katz and Lindell [28].

(3.) - (4.): If $\mathcal{A}$ observes $RF_{K_0}(w'_0) = RF_{K_1}(w'_1)$ for a pair w'_0, w'_1, then this only indicates a collision in RF_{K_0} and RF_{K_1}. Even if $\mathcal{A}$ queries the same w multiple times, output $RF_{K_0}(w, \gamma_{0,w})$ or $RF_{K_1}(w, \gamma_{1,w})$ will always be different as counters increase. If RF_{K_0} (or RF_{K_1}) outputs the same value twice (unlikely), this only indicates a collision in RF_{K_0} (or RF_{K_1}). The advantage over guessing in distinguishing $RF_{K_0}(w'_0)$ from $RF_{K_1}(w'_1)$ is zero.

$\mathcal{A}$'s advantage over guessing in distinguishing pairs $E_L, E_{L'}$ is negligible. □

Lemma 2. *Based on the trapdoor group assumption ("*TGA*"), PRISM's PIR-search produces computationally indistinguishable pairs of queries Q_i.*

Proof (Sketch). Assume $\mathcal{A}$ submits two words w_1, w_2 to an oracle. The oracle picks $\hat{b} \in \{0, 1\}$ and returns $Q_{\hat{b}} := \{\alpha_{\hat{b},1} := b \cdot e_{\hat{b},1} \mod p, \ldots, \alpha_{\hat{b},t} := b \cdot e_{\hat{b},t} \mod p\}$, with $e_{\hat{b},\mathcal{X}_{\hat{b}}} := 1 + a_{\mathcal{X}_{\hat{b}}} \cdot N, \forall i \neq \mathcal{X}_{\hat{b}} : e_{\hat{b},i} := a_i \cdot N$. Here, $\mathcal{X}_{\hat{b}} := \lfloor E_K(w_{\hat{b}}, 1) \rfloor_{\log_2(t)}$, and a_i are chosen randomly. $\mathcal{A}$ has to output $\hat{b}$ with non-negligible advantage over guessing.

However, we will now show that any pair of sequences of α values is computationally indistinguishable for $\mathcal{A}$. The proof is a direct implication of the security of the PIR protocol, based on *TGA*: for all adversaries $\mathcal{A}$, $Pr[\mathcal{A}(b \cdot e_1, \ldots, b \cdot e_t) = LSB(e_1, \ldots, e_t)] = \epsilon(s)$. That is, given $b \cdot e_i$, the probability that $\mathcal{A}$ computes low order bits of $e_i \mod N$ ("LSB") is negligible [37].

Assume that $\mathcal{A}$ can distinguish sequences $Q_0 = \{\alpha_{0,i}\}$ and $Q_1 = \{\alpha_{1,i}\}$ with non-negligible advantage. This would violate *TGA* as follows. First, note that besides $\alpha_{0,\mathcal{X}_0}, \alpha_{1,\mathcal{X}_1}$ all elements in both sequences $\{\alpha_{0,i}\}$ and $\{\alpha_{1,i}\}$ are created in the same way (multiplication of b with a random number). Therefore, besides $\alpha_{0,\mathcal{X}_0}, \alpha_{1,\mathcal{X}_1}$, any pair $(\alpha, \alpha') \in \{\alpha_{0,i}\} \cup \{\alpha_{1,i}\}$ is computationally indistinguishable for $\mathcal{A}$. If $\mathcal{A}$ can still distinguish between sequences $\{\alpha_{0,i}\}$ and $\{\alpha_{1,i}\}$, then $\mathcal{A}$ can determine with non-negligible probability $\mathcal{X}_0$ or $\mathcal{X}_1$ and thus value i with $e_i = 1 \mod N$, violating *TGA*. □

5.2 Statistical Analysis

We now discuss how $\mathcal{U}$ chooses parameters t and q to get a certain error probability P_{err}. This probability describes the chance that, despite $w \notin \text{FILE}$, $\mathcal{U}$ wrongly outputs $w \in \text{FILE}$ after q rounds without a contradiction, cf., Algorithm 3. Let n be the number of ciphertexts in one InputSplit, $n := \frac{S_{\text{InputSplit}}}{\text{CipherBlockSize}}$. The total number of ciphertexts stored in the cloud is $(c \cdot n)$. We consider for simplicity only rounds where $\lfloor h(C, i) \rfloor_1 = 1$, cf., Algorithm 3. With h a cryptographic hash, $\lfloor h(C, i) \rfloor_1 = 1$ in $q' \approx \frac{q}{2}$ rounds.

While inserting any ciphertext, the collision probability is $P_{\text{collision}} := \frac{1}{t^2}$. The probability for a one-collision is $P_{\text{one-collision}} := \frac{P_{\text{collision}}}{2}$. If w is *not* inside an InputSplit,

the probability that, after inserting the n ciphertexts of that InputSplit into $\mathcal{M}$, the candidate position is *not* set to 1 is $P_{\text{InputSplit,no-one-collision}} := (1 - P_{\text{one-collision}})^n$.

If $w \notin$ FILE, i.e., in *none* of the InputSplits, the probability that the candidate position is not set to 1 in *any* InputSplit is $P_{\text{contradiction}} := (P_{\text{InputSplit,no-one-collision}})^c$. This is the probability that a contradiction occurs in a single round. If $w \notin$ FILE, the probability that a contradiction occurs in *at least one* round is $P_{\text{contradiction,q-rounds}} := 1 - (1 - P_{\text{contradiction}})^q$.

After q rounds without a contradiction, $\mathcal{U}$ automatically decides that w is in FILE. In case that $w \notin$ FILE, and *no* contradiction occurs in q rounds, $\mathcal{U}$ is therefore wrong with $P_{\text{err}} := 1 - P_{\text{contradiction,q-rounds}} = (1 - (1 - \frac{1}{2 \cdot t^2})^{cn})^q$.

Given a certain file size, the size of InputSplits, and the blocksize of the symmetric cipher, $\mathcal{U}$ computes c and n. Therewith, $\mathcal{U}$ can target a false-positive probability by appropriately selecting t and q. We evaluate this using a real-world scenario in Section 6.

6 Evaluation

To show its real-world feasibility, we have implemented and evaluated PRISM with the scenario described in the introduction. The source code is available for public download [1]. We received 16 days of log data from May 2010 from a small local Internet provider. This provider logs and retains all customers' DNS resolve requests for possible forensic analysis and intrusion detection. Log data is split into files on a daily basis. Each file contains one day of logged 3-tuples: timestamp, customer IP (anonymized by provider for regulatory matters), hostname. The scenario for our evaluation is to use PRISM to upload this data encrypted to MapReduce and perform a search for specific hostnames in a privacy-preserving manner. This is useful for, e.g., "passive DNS analysis" to determine at which day certain command-and-control centers of botnets have been accessed by customer machines, cf., Bilge et al. [5]. The goal of our experiments was to analyze the computational overhead induced by PRISM's privacy mechanism, i.e., the additional time consumed by PRISM over non-privacy-preserving MapReduce.

6.1 Setup

For the 16 days, the log data contains $\approx 3 \cdot 10^8$ log entries, i.e., $\approx 2 \cdot 10^7$ per file/day. The total space required by all files uploaded into MapReduce using PRISM is 27 GByte, on average 1.7 GByte per file.

Our experiments have been performed on a small "cloud" comprising 1 master computer and 9 slaves. Computers featured a 2.5 GHz Pentium Dual Core and 4 GByte of RAM, running a standard desktop installation of Fedora 11. With this hardware configuration, a total of 18 CPUs were available for maps and reduces. We installed Hadoop version 0.20.2 on our cloud. Being aware that tailoring MapReduce's configuration parameters can have a huge impact on performance, we use the standard, out-of-the-box configuration of Hadoop 0.20.2 without any configuration tweaks. Performance tuning is out of scope of this paper. Similarly, as the InputSplit size is recommended to be between 64 MByte and 128 MByte, we chose $S_{\text{InputSplit}} = 96$ MByte (InputSplits must be dividable by $3 \cdot 32$ Byte, since log entries are 3-tuples).

Table 1. Parameters t, q to achieve $P_{\text{err}} < 0.01$

	File size (GByte)															
	0.45	1.21	1.32	1.36	1.38	1.45	1.52	1.67	1.78	1.93	2.00	2.08	2.09	2.14	2.21	2.25
t	2^{10}	2^{11}							2^{12}							
q	100	60					80		20							

In addition to the evaluation with 96 MByte InputSplits, we also performed a second measurement with larger InputSplits of 120 MByte. We expected a slightly improved performance of PRISM due to the fact that for the larger files the total number of InputSplits c reduced to less than our 18 available CPUs. Therewith, no costly (re-)scheduling takes places, and mappers do not have to process 2 InputSplits sequentially.

Finally, to put timing results into perspective, we implemented and measured a trivial, non-privacy-preserving MapReduce search called *Baseline*. Baseline search consists of an empty map phase, where mappers simply scan over InputSplits and compare each word of the InputSplit with a predetermined one, but do not generate any key-value pairs. Only at the end of the map phase, a single intermediate key-value pair per mapper (e.g., "found") is sent to reducers. Reducers discard this key-value pair and write empty files to disk. This trivial baseline only serves in deducing the overhead implied by PRISM, not taking MapReduce specific delays due to rescheduling, speculative execution of backup tasks etc. [17, 34] into account. Note that linear scanning through the entire InputSplit is mandatory, as we assume our data to be unordered and unsorted.

For the private information retrieval algorithm, we set $m = 400$ as suggested by Trostle and Parrish [37] for good security. Our Java implementation is a naive, straightforward implementation using Java's BigInteger without any performance optimizations. As symmetric encryption cipher, we used AES with 256 Bit blocksize from the GNU Crypto Library V2.0.1 [20]. As individual DNS entries occurred way less than 2^{16} times per day, we reserved $|\gamma| = 2$ Bytes and truncated entries longer than 30 Byte down to the last 30 Byte. Because the size of input $|w_i|+|\gamma_{w_i}|$ is less than E's blocksize (using standard padding for w_i), *concatenation* provides an unambiguous pairing.

Simulating $\mathcal{U}$, we computed n and c using blocksize, InputSplit size $S_{\text{InputSplit}}$, and individual file size S_{File}. Assuming that $\mathcal{U}$ targets an error probability of $P_{\text{err}} < 0.01$, we derived t and q. Table 1 summarizes parameters (t, q) computed for each file individually. Compared to q, we observed that parameter t has a much higher impact on P_{err}, but a comparatively lower impact on computations. Therefore, we increased preferably t than q. Higher values for (t, q) will achieve even smaller values for P_{err}, but Table 1 shows the computationally "cheapest" combination of (t, q).

6.2 Results

Computational overhead at the cloud is low as indicated by Figure 1 (PRISM's timing results). We have sorted the 16 files based on their size in an increasing order, i.e, the size of the smallest log file we received from the Internet provider was 0.45 GByte, the largest one was 2.25 GByte. PRISM's execution time was clocked on each file 6 times, respectively, and Fig. 1 shows the average. For each file, Fig. 1 shows two stacked boxes, respectively: the first one for 96 MByte and the second one for 120 MByte

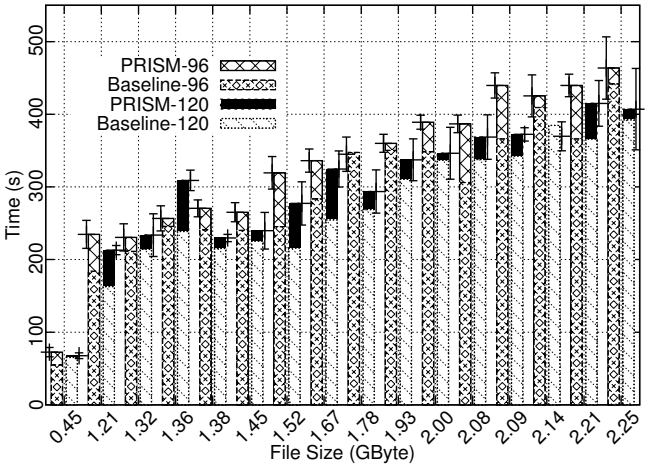


Fig. 1. Wall clock timings for PRISM and Baseline, with $S_{\text{InputSplit}} = 96$ and 120 MByte

InputSplit size. Each of the stacked boxes comprises, first, the baseline timing and, second, the additional time required to run PRISM. To give trust into the evaluation, Fig. 1 also shows 95% confidence intervals drawn right next to each box.

Timings shown in Fig. 1 are "wall clock" timings. This captures the *complete* time elapsed from submitting the PRISM map and reduce classes and starting the job until the end of the reduce phase. In the real-world, wall clock time reflects the time a cloud, e.g., Amazon [2], would charge a user $\mathcal{U}$. In conclusion, the additional overhead over the trivial Baseline MapReduce jobs was on average 11% with a 95% confidence interval of ± 3. The largest overhead seen was 24% over Baseline. This overhead is mostly computational overhead, as there is no difference in disk access between Baseline and PRISM and network volume increases only little by sending slightly larger values during "Reduce". These results do not only show the feasibility of PRISM in practice, but also demonstrate the low overhead implied by PRISM over the non-privacy preserving MapReduce job. We claim that a performance optimized (not based on Java BigInteger) implementation improves performance significantly and furthermore reduces overhead.

The simple increase from 96 MByte InputSplit size to 120 MByte InputSplit size has reduced wall clock times for MapReduce jobs by 9% on average (95% confidence interval of ± 4). Files of size smaller than 2 GByte are split into ≤ 18 InputSplits, and both jobs, PRISM and Baseline, are processed completely in parallel. This indicates that a careful configuration of Hadoop MapReduce's many system parameters, handcrafted and specific to the scenario and jobs to be executed, will lead to substantial performance improvements. This also indicates that in a cloud with more CPUs than in our small setup, the increased number of CPUs will enable to configure way *smaller* InputSplits being processed in parallel. Substantially smaller InputSplits will be beneficial for the overall performance of PRISM or any MapReduce job. However, increasing the number of InputSplits also implies a performance penalty due to (re-)scheduling and coordination activities of the central job tracker, cf., Pavlo et al. [34], so a trade-off has to be found. MapReduce configuration optimizations are, however, out of scope.

To better understand the cloud's computational overhead, we also measured the **computation time for a PRISM mapper**. On a single CPU, execution of an isolated PRISM map function on a single InputSplit is ca. 9 times slower than Baseline (9.3 for 96 MByte and 9.1 for 120 MByte, 95% confidence interval of ± 0.1). While this seems to be a lot, we remark that 1.) this map overhead is constant for an InputSplit and does not depend on or scale with the total size of the data, 2.) there is a lot of potential to improve our map implementation, 3.) this overhead is obviously amortized by other MapReduce aspects such as the Reduce phase and also disk latency, network overhead etc., and 4.) a user is charged for the total system time, i.e., the wall clock time.

Computational overhead at the User is also low in PRISM: per file, the preparation of, e.g., 2^{12} α values for the underlying PIR scheme is barely measurable (≈ 200 ms) on a PC with 2.5 GHz CPU. During result analysis, $\mathcal{U}$ automatically discards all received values that he is not interested in, i.e., all besides $s_{\text{FILE},\mathcal{Y},1\leq i\leq q}$. For these q values, a total of q Java BigInteger multiplications with modulo have to be performed. For our examples with $q \leq 100$, this was not measurable at less than 1 ms.

Memory consumption for $\mathcal{U}$ is, on the one hand, constant; $\mathcal{U}$ only stores the 256 bit AES key K. On the other hand however, the cloud user $\mathcal{U}$'s memory consumption scales linearly with $O(\Sigma)$, i.e., the number of *different* words. This is due to the construction of our stateful cipher that stores counters γ in a hashtable. In our straightforward implementation with Java's standard Hashtable, memory consumption of this hashtable was 548 MByte for the largest log file. While this is certainly a lot of RAM, we conjecture this to be available on PC hardware – moreover, as there is a large potential for performance tuning with such data structures.

Communication overhead for PRISM is dominated by the underlying PIR scheme. $\mathcal{U}$ sends, besides .class files once, only the t α values per file to the cloud. For example, with $t = 2^{12}$ and $m = 400$ Bit, this computes to 200 KByte per file. The response from the cloud is, for each round, t values of size m. The most expensive configuration in terms of communication in our experiments has been $t = 2^{10}, q = 100$; this results in ≈ 5 MByte communication overhead. Note that communication complexity in the underlying PIR scheme by Trostle and Parrish [37] is linear in the square root of the total table size, i.e., $O(t)$. This can be further reduced by using recursive PIR queries to $O(t^{\epsilon})$, for any $\epsilon > 0$ [29]. Those optimizations as well as amortization techniques discussed by Ostrovsky and Skeith [33] are out of scope.

In conclusion, PRISM is very lightweight for a user using standard PC hardware.

Discussion: On a larger cluster in a more professional environment (hundreds or even thousands of CPUs [25]), all files will be processed in parallel. As shown in Fig. 1, total time for the 2 GByte file is ≈ 350 s. However, already ≈ 340 s are required by MapReduce just to "scan" through the various InputSplits, see Baseline. Such inefficiency with non-optimal configurations has been observed before, and our results are along the lines of Pavlo et al. [34]. Here, a "grep"-like MapReduce job on 1 TByte of data took $\approx 1,500$ s on 50 CPUs which would be ≈ 20 times faster than our Baseline. However, Pavlo et al. [34] use a slightly tuned configuration and moreover a more efficient scanning through InputSplits (100 Byte text values instead of 32 Byte binary values in our case) which is known to lead to significant performance increases [27].

7 Related Work

Private Information Retrieval: Private Information Retrieval (and similarly oblivious transfer and oblivious RAM) has received a lot of attention [9, 13, 19, 22, 29, 32, 33, 35]. In PIR, a user retrieves a specific data from a database. The only "privacy" goal in PIR is access privacy whereby the server should not discover which data a user is interested in. Note that PIR does not ensure privacy of data in the database. PRISM, however, focuses on *searching* for a word and uses PIR only as a tool.

Searchable Encryption: With searching on encrypted data techniques [6], user privacy is guaranteed thanks to the encryption of the queries and the stored data. However, PRISM offers *higher* privacy guarantees since in existing searchable encryption solutions [4, 6–8, 10, 16, 21, 31, 36], the result ("found" or "not found") originating from a query is known to the adversary; therefore as opposed to PRISM, standard searchable encryption techniques do not ensure *query privacy*. Moreover, existing mechanisms *cannot* be easily extended to leverage from a parallelized cloud setup: while in theory the search on encrypted data itself could be run in parallel on subsets of data, today's solution do not support the *combination* (*aggregation*) of results (as in a reduce phase). To conclude, PRISM not only ensures both *storage privacy* and *query privacy*, but also enables the aggregation of results originating from intermediate parallelized operations.

8 Conclusion

PRISM is the first privacy-preserving search scheme suited for cloud computing. That is, PRISM provides storage and query privacy while introducing only limited overhead. PRISM is specifically designed to leverage parallelism and efficiency of the MapReduce paradigm. Moreover, PRISM is compatible with any standard MapReduce-based cloud infrastructure (such as Amazon's), and does not require modifications to the underlying system. Thanks to this compatibility, PRISM has been efficiently implemented on an experiemental cloud computing environment using Hadoop MapReduce. Besides a throughout analysis, performance of PRISM has been evaluated on that environment through search operations in DNS logs provided by an ISP. PRISM's overhead over non-privacy-preserving search is only 11% on average, acertaining its efficiency.

References

[1] PRISM source code (2012), http://www.ccs.neu.edu/~blass/prism.tgz
[2] Amazon. Elastic mapreduce (2010), http://aws.amazon.com/elasticmapreduce/
[3] Apache. Hadoop (2010), http://hadoop.apache.org/
[4] Bellovin, S.M., Cheswick, W.R.: Privacy-enhanced searches using encrypted Bloom filters (2007), http://mice.cs.columbia.edu/getTechreport.php?techreportID=483
[5] Bilge, L., Kirda, E., Krügel, C., Balduzzi, M.: Exposure: Finding malicious domains using passive dns analysis. In: Proceedings of 18th Annual Network and Distributed System Security Symposium, San Diego, USA, pp. 195–211 (2011) ISBN 1891562320

[6] Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
[7] Boneh, D., Kushilevitz, E., Ostrovsky, R., Skeith III, W.E.: Public Key Encryption That Allows PIR Queries. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 50–67. Springer, Heidelberg (2007)
[8] Brassard, G., Crépeau, C., Robert, J.M.: All-or-Nothing Disclosure of Secrets. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 234–238. Springer, Heidelberg (1987)
[9] Cachin, C., Micali, S., Stadler, M.A.: Computationally Private Information Retrieval with Polylogarithmic Communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–412. Springer, Heidelberg (1999)
[10] Chang, Y.-C., Mitzenmacher, M.: Privacy Preserving Keyword Searches on Remote Encrypted Data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
[11] Chief Information Officer's Council. Proposed security assessment & authorization for U.S. government cloud computing (2010), http://www.digitalgovernment.com/media/Knowledge-Centers/asset_upload_file652_2491.pdf
[12] Chief Information Officer's Council. Privacy recommendations for the use of cloud computing by federal departments and agencies (2010), http://www.cio.gov/
[13] Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: Proceedings of Symposium on Foundations of Computer Science, Milwaukee, USA, pp. 41–51 (1995)
[14] Cloud Security Alliance. Security guidance for critical areas of focus in cloud computing (2009), https://cloudsecurityalliance.org/guidance/csaguide.v2.1.pdf
[15] Cloud Security Alliance. Top cloud computing threats (2010), https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf
[16] Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of Conference on Computer and Communications Security, CCS, Alexandria, USA, pp. 79–88 (2006)
[17] Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: Proceedings of OSDI, San Francisco, USA, pp. 137–150 (2004)
[18] EU, Eu information management instruments (2010), http://europa.eu/
[19] Gertner, Y., Ishai, Y., Kushilevitz, E.: Protecting data privacy in private information retrieval. In: Proceedings of Symposium on Theory of Computing, Dallas, USA, pp. 151–160 (1998) ISBN 0-89791-962-9
[20] GNU, The gnu crypto project (2011), http://www.gnu.org/software/
[21] Goh, E.-J.: Secure indexes. Cryptology ePrint Archive Report 2003/216 (2003), http://eprint.iacr.org/2003/216
[22] Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious ram. Journal of the ACM 45, 431–473 (1996) ISSN 0004-5411
[23] Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984) ISSN 0022-0000
[24] Google. Google apps for government (2010), http://googleenterprise.blogspot.com/2010/07/google-apps-for-government.html
[25] Hadoop. Powered by hadoop, list of applications using hadoop mapreduce (2011), http://wiki.apache.org/hadoop/PoweredBy

[26] Hall, C., Goldberg, I., Schneier, B.: Reaction Attacks against Several Public-Key Cryptosystem. In: Varadharajan, V., Mu, Y. (eds.) ICICS 1999. LNCS, vol. 1726, pp. 2–12. Springer, Heidelberg (1999)
[27] Jian, D., Ooi, B.C., Shi, L., Wu, S.: The performance of mapreduce: An in-depth study. Proceedings of the VLDB Endowment 3(1), 472–483 (2010)
[28] Katz, J., Lindell, Y.: Introduction to modern cryptography. Chapman & Hall/CRC (2008) ISBN 978-1-58488-551-1
[29] Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: Proceedings of Symposium on Foundations of Computer Science, Miami Beach, USA, pp. 364–373 (1997)
[30] McCullagh, D.: Fbi wants records kept of web sites visited (2010), http://news.cnet.com/8301-13578_3-10448060-38.html
[31] Ogata, W., Kurosawa, K.: Oblivious keyword search. Journal of Complexity – Special Issue on Coding and Cryptography 20, 356–371 (2004) ISSN 0885-064X
[32] Ostrovsky, R., Skeith III, W.E.: Private Searching on Streaming Data. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 223–240. Springer, Heidelberg (2005)
[33] Ostrovsky, R., Skeith III, W.E.: A Survey of Single-Database Private Information Retrieval: Techniques and Applications. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 393–411. Springer, Heidelberg (2007)
[34] Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., DeWitt, D.J., Madden, S., Stonebraker, M.: A comparison of approaches to large-scale data analysis. In: Proceedings of International Conference on Management of Data, Rhode Island, USA, pp. 165–178 (2009)
[35] Sion, R., Carbunar, B.: On the computational practicality of private information retrieval. In: Proceedings of Network and Distributed Systems Security Symposium, San Diego, USA, pp. 1–10 (2007)
[36] Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of Symposium on Security and Privacy, Berkeley, USA, pp. 44–55 (2000)
[37] Trostle, J., Parrish, A.: Efficient Computationally Private Information Retrieval from Anonymity or Trapdoor Groups. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 114–128. Springer, Heidelberg (2011)

Practical Privacy Preserving Cloud Resource-Payment for Constrained Clients

Martin Pirker[1], Daniel Slamanig[2], and Johannes Winter[1]

[1] Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology (TUG), Inffeldgasse 16a, 8010 Graz, Austria
{martin.pirker,johannes.winter}@iaik.tugraz.at
[2] Department of Engineering and IT, Carinthia University of Applied Sciences, Primoschgasse 10, 9020 Klagenfurt, Austria
d.slamanig@cuas.at

Abstract. The continuing advancements in microprocessor technologies are putting more and more computing power into small devices. Today smartphones are especially popular. Nevertheless, for resource intensive tasks such devices are still too constrained. However, the simultaneous trend of providing computing resources as a commodity on a pay-as-you-go basis (cloud computing) combined with such mobile devices facilitates interesting applications: Mobile clients can simply outsource resource intensive tasks to the cloud. Since clients have to pay a cloud provider (CP) for consumed resources, e.g. instance hours of virtual machines, clients may consider it as privacy intrusive that the CP is able to record the activity pattern of users, i.e. how often and how much resources are consumed by a specific client. In this paper we present a solution to this dilemma which allows clients to anonymously consume resources of a CP such that the CP is not able to track users' activity patterns. We present a scenario which integrates up-to-date security enhanced platforms as processing nodes and a recent cloud payment scheme together with a concrete implementation supporting the practicality of the proposed approach.

1 Introduction

The sustained advancements in microprocessor technologies put more and more computing power into smaller and smaller devices. Today's *smartphones* essentially contain the computing "brainpower" of a desktop PC of not many years ago. This development, along with improved wireless connectivity, naturally fuels a shift in device usage patterns. It is no longer necessary to use a full-size desktop PC or laptop for certain tasks. Today, a smartphone is very well capable to perform many of the tasks desired by common end-users, such as browsing the web, or participation in social network services. Due to the small form factor, however, a smartphone's resources are limited in terms of battery power, storage, and consequently on available processing capacity.

A parallel trend is the rapid advancement of virtualisation technologies in conjuction with cheap storage and fast (wireless) Internet connections. This has created a market for providing computing resources as a commodity – so called *cloud*

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 201–220, 2012.

computing. Large data centers can take advantage of the economics of scale and dynamically lease computing power or storage capacities to clients on demand. This trend promises to use IT resources more efficiently and to reduce costs.

Consequently, in the future the desktop PC may continue to loose importance and the *split processing model* may rise to dominance. In the split processing model small tasks are executed directly on end-user devices such as smartphones, tablets or other gadgets, while more complex and demanding jobs are delegated to remote cloud computing services where resources are leased on demand.

Contribution. In this paper, we address the privacy concerns of the split processing model. We introduce a scenario where a low-resource client obtains cloud processing credits from a reseller and then uses them to pay for high-powered services at a remote cloud provider. Our main motivation for a "privacy-preserving by design" approach is that such cloud providers will be able to link data and information about resource consumption behaviour of their consumers (clients), allowing them to build dossiers. For many customers such transparency can be too intrusive. We want clients to be able to hide this information from cloud providers. As, for instance, argued in [9], activity patterns may constitute confidential business information and if divulged could lead to reverse-engineering of customer base, revenue size, and the like.

More precisely, we consider a setting where clients should be able to purchase a set of prepaid resources in form of cloud credits (CCs) from a reseller, e.g. represented as a scratch-off card. This card contains information which allows the client to obtain a single compact software token from a cloud provider that includes how many CCs a client is allowed to consume from this cloud provider. Then, clients should be able to consume their CCs from the cloud provider in an anonymous and unlinkable yet authorized fashion. For instance, if a client wants to consume n credits, he has to convince the CP that he possesses a valid token and he is still allowed to consume n credits. If this holds, the anonymous client is allowed to consume the resources and obtains an updated token corresponding to the remaining CCs in a privacy-preserving manner. Although the CP should be unable to track clients and determine how much credits a client has already consumed, it must be guaranteed that the client only consumes as much resources as CCs available in his token.

The novelty of our approach presented in this paper is that we use a recent *anonymous yet authorized and bounded cloud resource scheme* from [20] in a scenario which integrates *up-to-date security enhanced platforms as processing nodes*. We assume ARM *TrustZone* architecture enabled smartphone clients and *Trusted Execution Technology* attestable servers in the cloud. This scenario resembles a realistic use-case for a cloud computing environment and could be realised with today's mass-market hardware in the near future. We also provide experimental results from a prototypical implementation of our scheme on various current platforms.

Outline. The remainder of the paper is structured into the following major sections. In Section 2 we present our scenario, in which we introduce all

entities, their motivations as well as their privacy requirements. We then continue in Section 3 with preliminaries, including a brief background summary of the underlying *anonymous yet authorized and bounded cloud resource scheme* and the capabilities of trusted platform security technologies – in particular the *TrustZone* and *TXT* architectures. Section 4 introduces our concrete scheme and presents a description of all operations between the entities. We report practical implementation results in Section 5. We consider supplemental security, privacy implications, and trade-offs in Section 6. In Section 7 we present a brief overview of related work. Finally, Section 8 concludes the paper.

2 Scenario and High Level Description

In this section we present a practical scenario for deployment of our privacy preserving resource payment scheme. First we identify the core participants interacting and then describe the scenario. Then, we provide a high level description of the operations between the entities. We also discuss privacy issues for all involved entities.

2.1 Entities

In the scenario of this paper a small, resource limited *client* (C) wants to outsource computations to a powerful cloud datacenter.

The *cloud provider* (CP) professionally runs this large datacenter which takes advantage of the economics of scale. He rents computing power to anyone who can pay for the resources consumed. We make no assumption about what kind of resources are leased, we just define them to be accounted in discrete units of *cloud credits* (CC). Naturally, the client and the cloud provider must consent to a protocol so that the client can prove to the CP that he is eligible to consume a certain amount of resources. Obviously, client and cloud provider must interact directly when the client uses a CP service. However, before this step the client first has to obtain a certain amount of cloud credits.

As significant feature we thus introduce a third entity, the *credit reseller* (CR). The reseller obtains cloud credit units from the cloud provider in bulk and subsequently distributes and resells them through (small) distribution branches. This enables a wide range of use cases, such as cloud credits "gift-cards", and explicitly time-delayed, asynchronous interaction between the entities.

2.2 Scenario

We assume the client to be a state-of-the-art smartphone, based on an ARM platform with TrustZone capabilities. The smartphone is connected to the Internet, and thereby the cloud provider's servers, through a wireless connection to a mobile network. Figure 1 shows the major building blocks of our scenario and subsequently we reflect on the role of each:

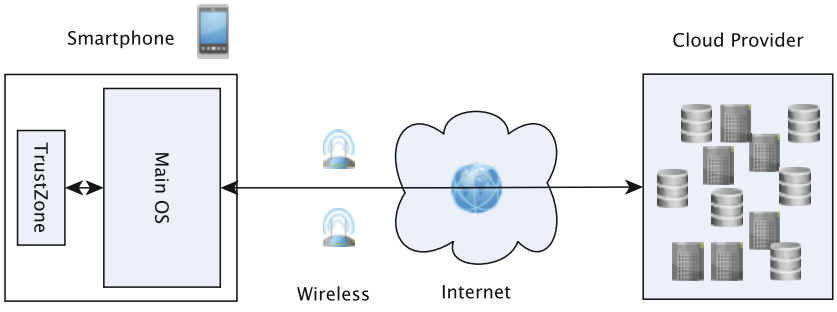


Fig. 1. Scenario in which a smartphone client connects to a cloud datacenter

Cloud provider: Computing resources provided by the cloud provider are servers which were booted with a Trusted Execution Technology measured process (cf. Section 3.2). This enables the client to verify that he actually receives what he paid for.

Network: In our scenario the client is a smartphone in a mobile network, and thus the client is uniquely identifiable and his activities can be easily monitored by the mobile network provider. The network connection continues via the global Internet to the cloud provider. As privacy feature we assume the use of an anonymizing network like Tor [13] to hide the effective network address of the client from the cloud provider. Consequently, also the mobile provider is unable to log where the smartphone is connecting to[1].

Smartphone: The client device is essentially split into two processing worlds: The normal-world which hosts the mobile's OS, e.g. Android as is common today, and end user applications. There is also a small isolated execution area – the secure-world – for sensitive data and operations. The link between the two worlds is strictly monitored and only well-defined calls are allowed[2].

The client's cloud credits accounting and private data storage is done in the secure-world. Once the initial credit data structures (representing the token) are imported, they never leave again (only in the case when they are transferred to another client).

2.3 High Level Description of the Operations

We model the flows of cloud credits in our scenario as essentially a circle between our three entities. The cloud provider is the central entity. He is responsible for providing resources and requires clients to pay for the consumed resources by providing the necessary number of CCs to him. Clients who want to consume resources from a cloud provider need to acquire a certain number of CCs from a credit reseller beforehand. This transaction is carried out without involving the

[1] We ignore sophisticated global network surveillance scenarios and defer discussion of these security edge cases to the Tor community.

[2] See e.g. [24] for a practical realization of this approach.

cloud provider and thus can be carried out offline. After acquiring the credits from the reseller, clients can activate the credits at the cloud provider, maybe at some later point in time. Clients may not only consume acquired credits by themselves, but may also give or sell them to other clients. Furthermore, if clients pay too many credits at the cloud provider for a certain task, e.g. because they do not know how many resources the task actually requires beforehand, then a cloud provider can issue vouchers for unused credits. A graphical sketch of our high-level operations for the client, cloud provider, and credit reseller setting is given in Figure 2. Subsequently we present a high level description of the operations.

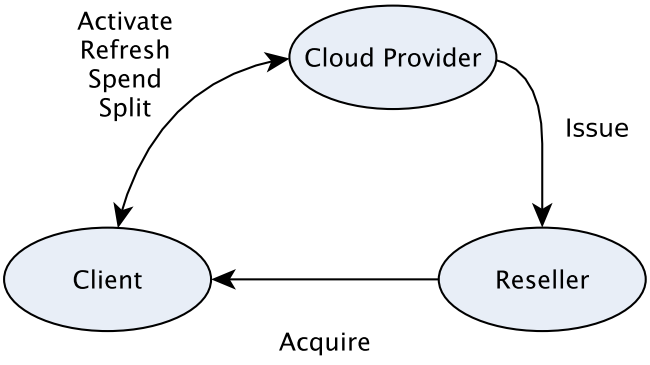


Fig. 2. Core Payment Scheme Operations

Issue: This transaction is an initial bulk transfer of cloud credits (CC) from the cloud provider (CP) to the credit reseller (CR). It is a personal business relationship operation and assumed to happen in a secure and reliable way.

Acquire: This transaction is carried out between a client (C) and a CR. At the end of this transaction C holds an amount of CCs, which cannot be used before they are activated.

Activate: This transaction is carried out between C and the CP. Essentially, C ends up holding an amount of activated CCs (in form of one compact token), which may have a limited validity.

Spend: This transaction is carried out between a client C and CP, where C pays a certain number of CCs and consumes an equivalent amount of resources from CP. The client ends up with the result of the task performed at the CP, and the remaining number of activated CCs.

SplitCredits: This is a transaction between C and CP in which C ends up with (at most) two valid tokens. Let us assume that C wants to split off n CCs from his collection of m CCs (if $m > n$)[3]. Then the first token contains $m - n$ CCs and the second token n CCs.

[3] Otherwise, if $m = n$ holds the first token will not be valid any more and if $m < n$ holds, the second token will only contain m CCs. Nevertheless, how such an operation ends is application dependent and one may also wish to terminate the operation if $m < n$.

Refresh: This transaction is carried out between C and CP. It is required when CCs have a limited validity. When the validity period ends, C carries out this transaction and gets issued the same amount of activated credits he already holds, but valid within the new period.

Transfer (not shown in Figure)**:** This transaction is carried out between two clients C_1 and C_2, whereas C_1 transfers a number of activated CCs to C_2 and C_1's CCs are deleted, i.e. we explicitly want to support *transferability.*

2.4 Privacy Issues

With the entities known we now discuss the privacy requirements which are effective in our scenario. We assume that two entities can always connect to each other using a private, i.e., confidential and authenticated, channel[4]. Note that this does not mean that the entities need to mutually authenticate to each other using their identities. Essentially, for our scenario we will realize privacy by anonymity and unlinkability of conducted transactions. This presents a good compromise in which clients have privacy guarantees while at the same time the other entities can be sure that only authorized actions can be conducted.

Cloud Provider. From a privacy perspective the cloud provider is in the position to be honest, but curios. Thus, he has the means to sniff, log and track the cloud's activities and is subsequently able to figure out who his clients are and what they are doing. He must be able to demonstrate a certain level of security of his services-for-rent. Without such a proof, customers will eschew doing business with him as they cannot be sure what is happening with their data.

Credit Reseller. The relationship between the cloud provider and the credit reseller is a business one. By definition, the cloud provider wants a reliable and clearly identifiable reseller, and the reseller wants to ensure that he is dealing with the correct provider contact point. From this follows the requirement for secure communications and transactions between the two when they trade credits for money – and privacy is not important in this context.

The reseller is the entity who comes into real-life contact with the end-user. Consequently, the privacy of the client very much depends on the method of payment for the prepaid cloud credits offered and how the actual handover is organized. Similar to the cloud provider, the reseller has no motivation to invest extra effort to identify his clients without additional external pressure.

Client. The client is the entity motivated to ensure that privacy is enforced at all steps in the transactions. First he has to obtain credits from the reseller. A straightforward solution is to pay with money in cash, an intuitive way to ensure privacy of payment. By definition the client already knows which cloud provider he wants to use, thus he must be able to verify that the offered credits from the reseller are genuine. Second, for establishment of an untraceable, i.e. anonymous and unlinkable, connection to the cloud provider the client uses an anonymous communication network, e.g. Tor.

[4] E.g., using Transport Layer Security (TLS) [11] over a Tor [13] connection.

The process of exchanging cloud credits for resources represents the most challenging issue. *How to consume them at the cloud provider without the cloud provider being able to identify and track clients?* The most straightforward non-privacy friendly solution for client payment is to require clients to register with the cloud provider and to have the cloud provider maintain an account for every client. The provider then bills all consumed resources to this account. In this scenario clients could pay for their consumed resources on a regular basis. However, this does not satisfy our privacy demands and we want to achieve the following privacy requirements for the payment process:

Anonymity. Clients do not want the cloud provider to be able to identify them. Assuming that we have an anonymous channel between the client and the cloud provider, this means that none of the operations Activate, Spend, SplitCredits and Refresh must involve any information that can be used by the cloud provider to identify a client. When we present our approach in Section 4 it will be clear that anonymity solely depends on the underlying scheme for which it is shown in [20] that anonymity is given.

Unlinkability. Clients do not want the cloud provider to be able to link different consumptions of resources (spending of credits), nor to discover how many credits a client still possesses. This gives the client the guarantee that he is unobservable from the cloud provider's perspective. As in case of anonymity, the operations Activate, Spend, SplitCredits and Refresh must be unlinkable and this follows from the underlying scheme as shown in [20].

3 Background and Preliminaries

This section is devoted to building blocks and technologies that are employed by our concrete scheme and deployment scenario. First we provide a compact description of the underlying cryptographic scheme and present some abstractions used throughout the paper. We then provide a brief overview of trusted platform security technologies, how they allow attestation of a platform's state, and how they support isolated, secured processing areas.

3.1 Anonymous Cloud Resource Scheme

In the following we represent all cloud credits (CCs) of a client as a single token, i.e. one may think of a wallet of CCs, and a consumption of CCs from this token can be thought of as spending some CCs out of the wallet.

We start by reviewing the *anonymous yet authorized and bounded cloud resource scheme* (AABCRS) introduced in [20], which is the most important building block for the work presented in this paper. The main idea behind such schemes is that clients are able to purchase a contingent of credits for resources, such as virtual machine instance hours, from a cloud provider (CP). While clients

spend their credits for resources at the CP, the CP does not learn anything about the resource consumption behaviour of users. In particular, users can consume an arbitrary number of their credits as long as there are still enough credits from their purchased amount. Thereby, in any interaction with a client, the CP is convinced whether a client is allowed to consume resources, but cannot *identify* the client nor *link* any of the client's actions and thus cannot figure out their consumption patterns.

Below we present the definition of [20] and slightly modify it by discarding one of the protocols (the `Reclaim` protocol). This modification, however, has no impact on the concrete scheme. Hence, an AABCRS is a tuple (`ProviderSetup`, `ObtainCredits`, `Consume`) of polynomial time algorithms or protocols between clients C and a cloud provider CP and works as follows:

- `ProviderSetup`. On input a security parameter k, this algorithms outputs a private key sk and a public key pk of a suitable signature scheme and an empty blacklist BL which is required for double-spending detection.
- `ObtainCredits`. In this protocol a client c wants to obtain a token t for L credits (the credit limit) from the CP. The client's output is a token t with corresponding signature σ_t issued by CP. The token contains the credits L and the up to now consumed credits s. These values may be represented by a single value $L' := L - s$ within the token. Clearly, initially no credits have been spent. The output of CP is a transcript T_{OL} of the protocol.
- `Consume`. In this protocol a client c wants to consume n credits from the credits still available in his token. The client shows a value id (a unique token identifier) of a token t and convinces the CP that he holds a valid signature σ_t for token t and that t contains at least n credits. If the token was not already shown in a previous run of a `Consume` protocol, i.e. $t.id \notin BL$, the signature is valid and there are still enough credits available in the token, i.e. $s + n \leq L$ (or $L' - n \geq 0$ if both values are represented as a single one), then c's output is `accept` and an updated token t' (with new id) for credit limit L and up to now consumed credits $s + n$ (or $L' - n$ if both values are represented as a single one) with an updated signature $\sigma_{t'}$ from CP. Finally, CP includes id into BL. Otherwise the user's output is `reject`. The output of CP is a transcript T_C.

In [20] the author presents two variants of an AABCRS scheme based on the pairing based Camenisch-Lysyanskaya (CL) signature scheme [7]. In both variants, a token t is represented as an ordered sequence of values, whereas the number of elements depends on the variant. In the first Variant (V1), a token is of the form $t = (C(id), C(s), L)$, whereas $C(x)$ denotes an unconditionally hiding commitment to value x. The value id represents a unique identifier of the token, s represents the number of CC's that have been consumed up to now and L represents the credit limit. The drawback of V1 is that L is included in plain in the token and thus is always visible to the CP. Hence, in the worst case, i.e. L is issued to exactly *one* client, the cloud provider can link actions of this client. However, if the set of clients associated to the same value L is reasonable

large, unlinkability is no longer a problem in a practical setting[5]. In the second version (V2) even the value L is hidden from the cloud provider and a token is of the form $t = (C(id), C(s))$. Here, s represents the number of CCs that are still available from this token.

Intuition Behind V1. [20] We briefly sketch the idea of V1 for simplicity, whereas the modifications for V2 are straightforward (see [20]): The main intuition of the construction is to let a client solely prove in each `Consume` protocol that enough cloud credits are still available in a token. Therefore, the cloud provider generates a key-pair (sk, pk) for the CL signature scheme, publishes pk and initializes an empty blacklist BL. Then, a client obtains a CL signature σ_t for a token $t = (C(id), C(s), L)$, whereas initially no CCs s are consumed. Let us assume that the client holds a token $t = (C(id), C(s), L)$ and corresponding signature σ_t. It is important to note, that id (a random token identifier) and s were signed as commitments and thus the CP is not aware of these values. If a user wants to consume n CCs from his token, he computes a commitment $C(id')$ representing the updated token identifier for the updated token, randomizes the signature σ_t to σ'_t (σ_t and σ'_t are then unlinkable) and proves in zero-knowledge that σ'_t is a valid signature for id and L. This includes showing the values id and L to the CP. Additionally, the client proves that the token includes an unknown value s, which satisfies $(s + n) \in [0, L]$ or equivalently $s \in [0, L - n]$. This proves convinces CP that at least n CCs are available from this token. If id is not contained in BL and all proofs succeed, then the client is eligible to consume n CCs. Consequently, the signature will be updated to a signature for $C(id + id')$, $C(s + n)$ and L in an interactive manner between the client and CP. Subsequently, the CP adds id to BL and the client obtains an updated signature for an updated token $t' = (C(id + id'), C(s + n), L)$. This signature can be randomized by the client (which makes t and t' unlinkable) and used for the next consumption of resources. Otherwise, CP will reject the client's request to consume n resources.

In the following we abstract from the details of the scheme, and use the following high level parameters and state information respectively:

- The public parameters $\mathsf{cpparams}_{pub}$ represent the public key pk of the CL signature scheme and are public knowledge.
- The private parameters $\mathsf{cpparams}_{priv}$ represent the private key sk of the CL signature scheme and the blacklist BL and are solely known to the cloud provider.
- The state cstate_{pub} represents the actual token-signature pair of the client.
- The state cstate_{priv} represents the values id, s, (L in V1) as well as the randomizers for the commitments and the randomization factors of the CL signature of the client's token. The state information is updated during every `Consume` operation, since only the current values are required.

[5] Note that cloud resellers may sell cloud credits only in specific well-known amounts, similar to cards for prepaid mobile phones, e.g. 5$, 10$, etc.

We note that it is not necessary to keep the token-signature pair $\mathsf{cpparams}_{pub}$ secret, since without knowing cstate_{priv} no one will be able to convince CP that the signature σ_t is a valid signature for token t.

3.2 Trusted Platforms

In the last few years, mass-market computer platforms and devices have been enhanced with functions dedicated to support advanced security. In the following we give a short introduction to the features available in industry standard PCs as well as mobile platforms.

Trusted Platform Modules. The concept of Trusted Computing as promoted by the Trusted Computing Group (TCG) extends the industry standard PC architecture with a specialised hardware component, the Trusted Platform Module (TPM) [22]. A TPM features cryptographic primitives similar to a smartcard, but is physically bound to its host platform.

An important concept of Trusted Computing is the measurement logging and reporting of the platform state. Upon platform hardware reset a special set of platform configuration registers (PCRs) in the TPM are reset to a well defined start value. PCRs cannot be directly written to, rather, a PCR with index i, $i \geq 0$, in state n is extended with input x by setting $PCR_i^{n+1} = \text{SHA-1}(PCR_i^n||x)$. This enables the construction of a chain-of-trust. From the BIOS onwards, every block of code is measured into a PCR before execution control is passed to it. Thus, the current values in the set of PCRs represent a log of what happend since system reboot, up to the current state of the system. The current state may then be TPM signed with the TPM *Quote* operation and reported in a so-called *remote attestation* protocol.

Intel Trusted Execution Technology. Recent platforms from Intel[6] extend the basic TCG model of a *static* chain-of-trust from hardware reboot and trust rooted in early BIOS. They provide the option of a *dynamic* switch to a well-defined, measured system state [16], meaning at any point of execution after platform reboot. Consequently, this capability significantly cuts down the complexity of the chain-of-trust measurements to assess the platform state by excluding the early, messy bootup operations, leading to a simpler and practical implementation.

ARM TrustZone. Many computing devices, especially in the embedded and mobile domain, and more recently also in high-density data centers, do not use x86 microprocessors, but are rather powered by a processor of the ARM family. The ARM architecture follows a building-blocks approach, where the main processor design is developed and controlled by one company. Individual vendors select and license the intellectual property of the core and desired support components as needed, and then enhance them with their own functional units according to the needs of their customers.

[6] We restrict our discussion to Intel's Trusted Execution Technology (TXT) as this is currently the dominant technology provider – comparable features are also available on e.g. AMD platforms.

The *TrustZone* architecture extension for ARM CPUs is an instruction set extension for security critical scenarios. Basically, it provides a separation of the memory resources and the CPU of a device, thereby creating two virtual domains which are the so-called *secure-world* (SW) and *normal-world* (NW) [3]. This approach is an improvement to the basic concept of privileged/unprivileged mode-split which can be found on many conventional architectures, including earlier ARM cores.

The normal-world is the containment for user programs or any kind of untrusted applications. Security critical code is executed in the secure-world. The isolation mechanisms of the TrustZone prohibit normal-world applications from accessing the secure-world. Consequently, the data flow between both worlds is controlled by a secure monitor entity, which is under control of the secure-world.

The total memory available for software inside the TrustZone is vendor dependent and ranges from 64 kBytes up to 256 kBytes on typical systems. This size enables the running of a small – hopefully evaluated and certified – core in the secure-world along with trusted executables.

Due to the ARM building-block approach there is no standardized way to report the genuinity of the TrustZone implementation of a certain vendor. However, typically TrustZone implementors provide a symmetric device-key for device identification and a asymmetric key for the purpose of secure boot. This allows construction of hardware authentication similar to that implemented by a TPM device and enforcement of a measured boot chain like with Intel TXT.

4 Practical Anonymous Payment

Building on the scenario, definitions and restrictions outlined in Section 2 we are now ready to present our resource payment scheme in more detail. When we write $\mathsf{A}(C(a_1,\ldots,a_n), CP(b_1,\ldots,b_m))$ we mean that operation A is run between entity C with private inputs $a_1,\ldots,a_n$ and entity CP with private inputs $b_1,\ldots,b_m$. All operations are conducted by the entities client (C), cloud provider (CP) and credit reseller (CR). Furthermore, we will use the algorithms of an AABCRS as defined in section 3.1 as subroutines.

We note, that the Acquire operation can be conducted by several means. One suitable and also privacy friendly scenario is to require the cloud provider to hand over scratch-off cards to the credit reseller. These scratch-off cards are of different monetary denominations representing some equivalent of cloud credits (CCs). If a user buys, e.g. with cash, such a card at a CR, he can scratch off the opaque covering and a QR-Code is revealed. This QR-Code contains information about the denomination, a serial number and potentially a validity period along with a digital signature for those values. A client can scan this QR-Code with the built-in camera of his smartphone[7] and then holds all information necessary to conduct an Activate operation with the CP. We denote the information which is necessary

[7] The use of a mobile phone camera to provide a trusted import path for cryptographic data was demonstrated viable in the Seeing-is-Believing effort [17].

for the activation as params_{act} subsequently. Note, that the aforementioned approach is advantageous from a privacy perspective, since CR does not learn the serial number of the card and thus cannot link (in cooperation with the cloud provider) the Acquire operation to the respective Activate operation.

Now, we present the remaining operations in a more formal manner, whereas we assume for simplicity that the CP provides *one* type of resource and has already conducted the `ProviderSetup` procedure. Furthermore, we denote by NW the normal-world and by SW the secure-world of the client's platform.

Activate$(C(\mathsf{params}_{act}, \mathsf{cpparams}_{pub}), CP(\mathsf{cpparams}_{pub,priv}))$**:** The NW sends params_{act} to the CP, who verifies them for validity and returns `true` or `false` to C. In case of `true`, C imports $\mathsf{cpparams}_{pub}$ into SW and C's SW runs an `ObtainLimit` protocol for CC limit L (contained in params_{act}) with CP. The client ends up with storing cstate_{pub} in the NW and cstate_{priv} in the SW. If the CP returns `false` in the first interaction, the operation terminates.

Spend$(C(\mathsf{cstate}_{pub,priv}, n), CP(\mathsf{cpparams}_{pub,priv}))$**:** The NW sends the number n of desired CCs along with cstate_{pub} to SW. If enough CCs are still available ("in" the token) then SW runs a `Consume` protocol to consume n CCs with CP, otherwise it returns `false` and the operation terminates. Thereby, CP obtains a proof that cstate_{pub} represents a valid token-signature pair, there are still enough CCs available and the token id not already contained in the blacklist BL. If any check fails, the operation terminates. If all checks succeed, the SW updates cstate_{priv} and obtains an updated token-signature pair cstate_{pub}, which is stored in NW.

SplitCredits$(C(\mathsf{cstate}_{pub,priv}, m), CP(\mathsf{cpparams}_{pub,priv}))$**:** The NW sends m along with cstate_{pub} to SW. Let us assume that $n > m$ whereas n represents the number of remaining CCs in cstate_{priv} for simplicity. Essentially, the operation works identical to the Spend operation, but the m resources are not consumed. Instead, an additional cstate'_{pub} is returned to C. At the end of this operation C holds cstate_{priv}, cstate'_{priv} (in SW) as well as cstate_{pub} and cstate'_{pub} (in NW), whereas the former token represents $n - m$ and the latter m CCs.

Transfer$(C_1(\mathsf{cstate}_{pub,priv}), C_2(\cdot))$**:** The SW of C_1 exports cstate_{priv} and the public and private state information $\mathsf{cstate}_{pub,priv}$ are transferred to C_2 who imports cstate_{priv} into his SW and cstate_{pub} into his NW. The SW of C_1 deletes cstate_{priv}. Note that C_1 transfers all n CCs represented by cstate_{priv} to C_2.

Refresh$(C(\mathsf{cstate}_{pub,priv}), CP(\mathsf{cpparams}_{pub,priv}, \mathsf{cpparams}'_{pub,priv}))$**:** C's NW sends cstate_{pub} to the SW and SW sends cstate_{pub} along with cstate_{priv} (representing n CCs) to CP. Now, CP can check whether cstate_{pub} represents a valid token-signature pair for n CCs. If this is true, CP engages in an `ObtainLimit` protocol with respect to new parameters $\mathsf{cpparams}'_{pub,priv}$ with C and issues a token for n CCs.

We additionally observe the following:

SplitCredits: A client may have several motives to invoke a SplitCredits operation. For instance, a client may want to "split" off some CC's from his token to obtain a new token, in order to give one of the tokens to someone else, e.g. as a gift. Another scenario is that a client pays n CCs for some computation, but the computation actually only requires $m < n$ CCs. Then, after having conducted the computation, CP issues some kind of voucher in form of a new token to C for $n - m$ CCs.

Refresh: The version of the Refresh operation presented here is the simplest one. Essentially, the client is issued fresh CCs with respect to new CL signature parameters, i.e. every validity period is represented by distinct signature parameters. Note, that providing unlimited validity of tokens would not scale well, since CP would have to store the entire blacklist for double-spending detection. More flexibility can be achieved if validity periods are encoded directly into the tokens as it is proposed as an extension in [20].

Spend: For every Spend operation at least one CC is removed from circulation and at least one new entry in the blacklist BL is required to prevent double-spending. Naturally, the amount of CCs in circulation must be known to the CP as he must be able to manage a blacklist. Consequently, the maximum amount of credits issued is bound by the maximum size of the blacklist. The policy of the CP must enforce a periodical Refresh operation by the clients – in effect accounting periods – to allow periodical clearing of the blacklist (of the expired period).

5 Implementation

For practical evaluation we prototyped the core anonymous payment operations of our approach on multiple software and hardware platforms. Some are good approximations of the current generation of smartphones.

5.1 Specifications

On the software side, our scheme was implemented in the high-level language Java and uses the jPBC 1.2.0 library[8], a library for Pairing-Based Cryptography (PBC) in Java. As an alternative to the pure Java implementation there is also a C implementation of PBC[9], which can be called from Java via a Java-to-C wrapper. In the following we denote these two setups as Java "-J" and Native C accelerated "-C" variants. Our platforms ran either Linux (Li) or Android (An) as operating system. The platforms used to measure execution speed were as follows:

Pc* Laptop HP 8440p Elitebook, Intel i7M620 @2,67 Ghz, running Android for x86 2.3.5 (RC1 20110828) of the Android_x86 porting effort [1], or Ubuntu 11.10 with IcedTea6 1.11pre (OpenJDK 64-Bit Server VM (build 20.0-b11)

[8] http://gas.dia.unisa.it/projects/jpbc/
[9] http://crypto.stanford.edu/pbc/

Ek* Freescale i.MX51 evaluation kit [15], Freescale MX515D @800Mhz[10], running Android 2.3.4 (build R10.3.2_3), or Ubuntu 10.04 LTS with IcedTea6 1.8.10 (OpenJDK Zero VM (build 14.0-b16))

SpGs Smartphone Google Nexus S, Samsung Exynos 1 GHz (ARM Cortex-A8), running Android 2.3.6.

SpS2 Smartphone Samsung Galaxy S2, Samsung Exynos 1.2 GHz dual-core (ARM Cortex-A9), running Android 2.3.3.

5.2 Results

For performance evaluation we focused on the Activate and Spend operations conducted by the client. This is due to the fact that the remaining protocols only require negligible computational resources or are based on one of the two aforementioned protocols, i.e. perform identically. The first one being run only once for initialisation of the credits token, the latter being run everytime credits are spend at the CP. Table 1 shows the results from our implementation on the platforms presented in Section 5.1. We measured from 4 to 16 bits for a practical cloud credits limit of $2^4 = 16$ to $2^{16} = 65536$ credits. The credits token is only valid at one specific provider and these limits enable many basic use cases. A larger limit is always possible, if the resulting additional computation time is acceptable for the client.

Table 1. Execution time of Activate and Spend 4 bits to 16 bits [s]

	PcLi-C	PcAn-C	PcLi-J	SpS2-C	SpGs-C	EkAn-C
Activate	0.06	0.15	0.35	0.64	0.86	1.12
Spend 4 bits	0.16	0.35	0.82	1.43	1.94	2.54
Spend 16 bits	0.34	0.77	1.72	2.99	4.11	5.32
	EkLi-C	PcAn-J	SpS2-J	SpGs-J	EkAn-J	EkLi-J
Activate	1.09	1.80	6.76	11.1	15.7	19.8
Spend 4 bits	2.53	3.60	13.3	20.9	29.6	39.7
Spend 16 bits	5.42	6.87	24.4	41.7	54.7	77.3

Figure 3 provides a more detailed analysis of the Spend protocol for tokens containing 2^x CCs. As can be seen from the figure, the time required for the run of a Spend protocol grows linearly[11] in the number of bits x of CCs in the activated token, which is due to required zero-knowledge range proofs. We do not provide explicit timings for computations of the cloud provider, since he

[10] The processor on this board is based on ARM's Cortex-A8 core and supports advanced security features such as ARM's TrustZone and secure boot facilities. Unfortunately, most parts of the documentation is only available under NDA from Freescale. For this prototype effort this is sufficient.

[11] With the non-linearities in the measurements caused by OS system services running in the background.

uses state-of-the-art servers and the computations are very efficient (the most expensive operation of the CP, i.e. `Consume` for a limit of 2^{30} CCs, reported in [20] takes about 1 second).

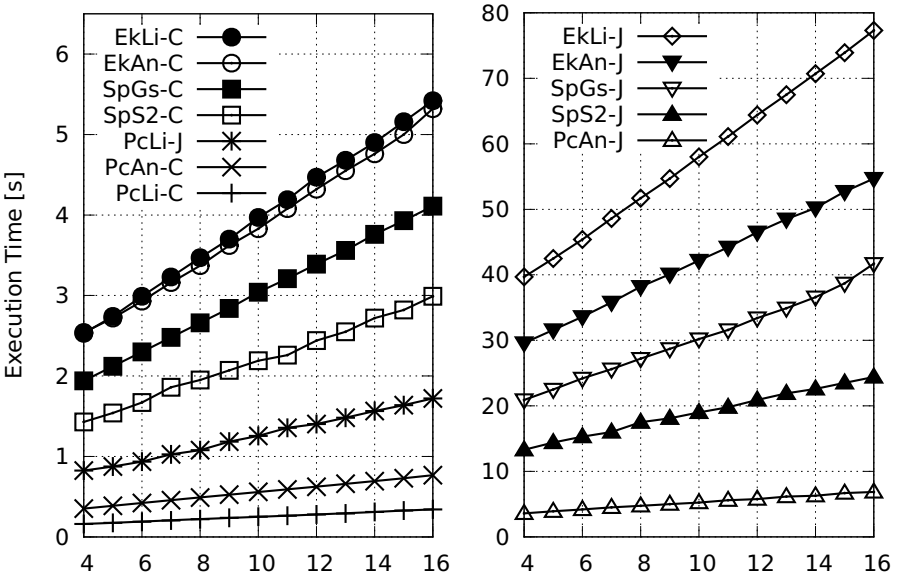


Fig. 3. `Spend` execution speed, x-axis shows credits token limit $L = 2^x$ [bits]

Our results clearly show that the *PC** versions dominate, the C versions take full advantage of the raw processor power. We interpret the difference between *PcLi-C* and *PcAn-C* due to the first being 64bit and the second a 32bit platform. They are closely followed by *PcLi-J*, a pure Java version executed by a server JVM optimized over many years. The next 4 places are claimed by the remaining C builds, as expected by their platform processor powers: 1.2GHz *SpS2-C* before 1.0GHz *SpGs-C* and 800MHz *Ek*-C*. The *EkLi-C* build runs almost identical to *EkAn-C*. The Java versions trail, again as expected by their processor speed. For the last platform, EkLi-J, the ARM JVM port appears to be quite unoptimized.

We must note that all Androids except *PcAn-J* and *SpS2-J* exhibited slight process memory leakage during execution. Despite continuous garbage collector runs memory usage grew. We assume this issue to add GC runtime overhead, therefore their execution time should actually be a little lower. So far we have been unable to determine the cause of this problem.

The main Java classes of our basic test code only consume 25 kB in size. The supporting Java libraries (*jpbc-api.jar*, *jpbc-plaf.jar*, *jpbc-pbc-jni.jar*) require 371 kB. The native C support libs (*libgmp.so*, *libjpbc-pbc.so*, *libpbc.so*) sizes vary according to the specific platform: Linux_x86_64 772 kB, Android_x86_32 688 kB, Linux_Arm 588 kB and Android_Arm 800 kB.

5.3 Discussion

The results of our practical evaluation support the feasibility of our scheme. While the Java numbers may appear to be very high at first glance, the C core accelerated versions are multiple times faster. To achieve this, first we replaced the default JNA wrapper for the Java-to-C bridge with our own custom JNI coded wrapper to allow the C code to be accessible on Android. Second, the ARM code was compiled to take advantage of ARM processor *Neon* SIMD and ARMv7 instructions features[12]. Consequently, an optimized, C core enhanced, Android build runs fast enough on current generation smartphones to allow Spend(ing) operations without long delays for end users.

Our prototype uses off-the-shelf libraries for crytography which were not optimised at all as initial evaluation of execution speed was our first objective. Consequently, our test code plus support libraries and JavaVM are way too large to fit into a TrustZone environment (Section 3.2). However, we expect a standalone implementation of the algorithm along with a small JavaVM to fit into a small TrustZone environment. This remains future work.

6 Discussion

In the following we reflect on supplemental aspects which were not discussed during the presentation of our scheme or our scenario.

Trusted Computing Attestation. In Section 2.2 we use Trusted Execution Technology to attest the cloud servers provided to the client. This technology is mass-market available and TXT integration has already been demonstrated for Linux based servers [21]. Thus, if the software image to be rented to the client is agreed upon, a remote client can ask for a remote attestation proof (*TPM_Quote*) from the server to confirm what specific software image was actually booted. Attacks of the TXT components require physical intervention [25] and consequently raise the bar for manipulation[13].

The execution of multiple parallel running client VMs on a cloud server and subsequent attestation of the security of the system is an active topic of Trusted Computing research. The current TPM chip generation was not designed for this use case, an updated revision of the TPM hardware "V2" is expected to make this use case practical.

Identification at Credits Purchase. An obvious privacy problem is a potential camera at the reseller's place, which may collect a photo of the client's face. This threat is mitigated with the possibility of scratch-off cards and the ability to Transfer credits between trustworthy friends.

Isolated Secure World. By definition computation in the SW is isolated from the rest of the platform and only reachable via a well-defined, narrow interface.

[12] GCC CFLAGS="-march=armv7-a -mfloat-abi=softfp -mfpu=neon"

[13] Or good old runtime software bugs which allow priviledge escalation.

Consequently, an obvious problem is to decide from inside the SW whether a request from the NW is authorized – or not. This problem is non-trivial, but one solution would be to require a trusted input and display path which provides user-interaction (e.g. PIN entry) if explicit authorization for sensitive operations in the SW is required. We assume that users are acting in their own best interests when using their own smartphones, which is reasonable. Nevertheless, this still leaves the problem of potential malware on the client's platform which would be able to circumvent this feature. Note that this isolation also impacts the privacy provided by our network connection, as the Tor connection is currently anchored in the NW. Thus, how could the limited SW verify whether an anonymizing network connection to the CP is properly used?

Honesty of Clients. If two clients exchange cloud credits by means of a Transfer operation, say C_1 gives n CCs to C_2, then C_2 has no means to verify whether C_1 has properly deleted the transferred credits. Essentially, if C_1 is dishonest the first-come-first-served principle applies and whoever is the first one to spend credits from the token will be able to continue spending. We may, however, assume that only clients who trust each other exchange CCs, in which case this is not a problem. Furthermore, from the perspective of CP, even if C_1 does not delete his CCs, then this does not mean any harm to the CP, since only n CCs can be consumed in total and clients cannot create "extra" credits.

Forward Secrecy. What happens when a smartphone is stolen, lost or seized? If Spend is not protected by additional secret information, e.g., a PIN, then someone in possession of the smartphone is able to spend all credits left within the currently activated token. Nevertheless, we note that even if the adversary is able to extract information from the secure world, he will not be able to link previous actions of the smartphone's holder – since no "history data" of the randomization processes of the underlying scheme is stored.

7 Related Work

To the best of our knowledge an approach related to the one presented in this paper has not been considered before. Nevertheless, there are three lines of work whose combination leads to the kind of work presented here. We will briefly elaborate on this below.

Privacy in Trustworthy Mobile Platforms. With the growing popularity of mobile computing, their ubiquitous application and the resulting privacy issues, there arises the necessity to provide functionalities of traditional privacy enhancing cryptographic protocols. However, due to limited storage capacities and processing power this task is non trivial and requires clever design. Recent works include the design of anonymous authentication for mobile devices by modifying direct anonymous attestation (DAA) and using hardware security features to prevent copying and sharing of private credentials [23]. Another implementation of DAA on mobile platforms with TPMs is presented in [12].

Privacy in Cloud Computing. Privacy is considered as one of the main issues in cloud computing. Besides known problems regarding user's privacy in traditional web applications, additional aspects imposed by the heavy use of virtualization seem to be novelties. For instance, sharing of resources among different users may potentially lead to the construction of covert or side channels which allows to infer activity patterns of other users (cf. [9]). User's access patterns represent privacy sensitive information that should also be hidden from the cloud. Recent works are mainly focusing on storing and sharing data in the cloud. In [14] an oblivious RAM (ORAM) based approach is presented, which allows users to outsource a set of data items to the cloud and provide read and write permission to users as follows. Users can only access (and read in plain) data items when accurate permissions were obtained and can learn nothing about other items. Additionally, users and even the cloud provider observing all accesses cannot infer which user is accessing which data items how often. Another approach based on dynamic accumulators was recently proposed in [19]. Here, no ORAM is employed and thus the cloud (and other users) may learn which data items are accessed, but each access is anonymous and unlinkable to each other. Hence, usage patterns of users can also not be inferred. Independent of the latter approach [18] also proposed a discretionary access control model for data outsourced in the cloud which hides access patterns.

Anonymous Payments. Concepts for anonymous and untraceable electronic payments are around for quite a long time [8]. While these first schemes were based on blind signatures and the cut-and-choose paradigm, over the years several improvements, especially for off-line e-cash, e.g. compact e-cash [5] or divisible e-cash [4], allowing to spend 2^n coins from a "single coin" accumulating all coins, have been proposed. Recently, the use of anonymous payments for overlay networks like Tor [2,10], which use lightweight payment protocols for micropayments, have been proposed. Unlike all aforementioned schemes, which assume a bank, a set of payees and a set of payers, in our scenario we have one bank and one payee represented as the same entity. Thus, we do not need to employ offline schemes, but use a kind of online payment scheme. Since we do not need properties of e-cash schemes such as double-spender identification as well as spending with arbitrary payees (this usually adds a non trivial computational overhead), we employ a scheme tailored to payment for cloud resources in our work.

8 Conclusion and Outlook

In this paper we present a privacy preserving cloud resource payment scheme for resource constrained mobile devices such as smartphones. We discuss a concrete scenario for a setting which includes clients, credit resellers and a cloud provider. The client and cloud provider take advantage of the state-of-the-art security enhanced TrustZone and TXT hardware platforms. Besides theoretical considerations, we also prototype the core operations on state-of-the-art platforms. Our results suggest that an optimized C implementation of our scheme is already fast enough for deployment on the current generation of smartphones.

Future work includes the use of an instantiation of an AABCRS based on the strong RSA version of the CL signature scheme [6], which should provide a significant performance boost. Other interesting aspects are the extension of the scenario to multiple cloud providers such that credits can be spend at different CPs (potentially including a "bank" as within traditional payment systems) and the consideration of alternative (more efficient) payment mechanisms as for instance proposed in [10]. The problem of practically realizing such scenarios with smartphone secure processing and secure data storage technologies remain an active area of research.

Acknowledgements. We thank the anonymous reviewers for their helpful feedback on the paper. In particular we thank Thomas S. Benjamin for his many suggestions for improving this paper. This work has been supported by the European Commission through project FP7-SEPIA, grant agreement number 257433. The second author has been supported by an internal grant (zentrale Forschungsförderung – ZFF) of the Carinthia University of Applied Sciences.

References

1. Android x86 Team: Android-x86 - porting android to x86 (2011), http://www.android-x86.org/
2. Androulaki, E., Raykova, M., Srivatsan, S., Stavrou, A., Bellovin, S.M.: PAR: Payment for Anonymous Routing. In: Borisov, N., Goldberg, I. (eds.) PETS 2008. LNCS, vol. 5134, pp. 219–236. Springer, Heidelberg (2008)
3. ARM Ltd.: TrustZone Technology Overview (2011), http://www.arm.com/products/esd/trustzone_home.html
4. Au, M.H., Susilo, W., Mu, Y.: Practical Anonymous Divisible E-Cash from Bounded Accumulators. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 287–301. Springer, Heidelberg (2008)
5. Camenisch, J.L., Hohenberger, S., Lysyanskaya, A.: Compact E-Cash. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 302–321. Springer, Heidelberg (2005)
6. Camenisch, J.L., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
7. Camenisch, J.L., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
8. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO, pp. 199–203. Plenum Press (1982)
9. Chen, Y., Paxson, V., Katz, R.H.: What's New About Cloud Computing Security? Tech. Rep. UCB/EECS-2010-5, University of California, Berkeley (2010)
10. Chen, Y., Sion, R., Carbunar, B.: XPay: Practical Anonymous Payments for Tor Routing and other Networked Services. In: WPES, pp. 41–50. ACM (2009)
11. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, IETF (2008), http://tools.ietf.org/html/rfc5246

12. Dietrich, K., Winter, J., Luzhnica, G., Podesser, S.: Implementation Aspects of Anonymous Credential Systems for Mobile Trusted Platforms. In: De Decker, B., Lapon, J., Naessens, V., Uhl, A. (eds.) CMS 2011. LNCS, vol. 7025, pp. 45–58. Springer, Heidelberg (2011)
13. Dingledine, R., Mathewson, N., Syverson, P.F.: Tor: The Second-Generation Onion Router. In: USENIX Security Symposium, pp. 303–320 (2004)
14. Franz, M., Williams, P., Carbunar, B., Katzenbeisser, S., Peter, A., Sion, R., Sotakova, M.: Oblivious Outsourced Storage with Delegation. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 127–140. Springer, Heidelberg (2012)
15. Freescale Semiconductor Inc.: i.MX51 evaluation kit (2010), http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MCIMX51EVKJ
16. Grawrock, D.: Dynamics of a Trusted Platform: A Building Block Approach. Intel Press (2009)
17. McCune, J.M., Perrig, A., Reiter, M.K.: Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication. In: IEEE Symposium on Security and Privacy (2005)
18. Raykova, M., Zhao, H., Bellovin, S.: Privacy Enhanced Access Control for Outsourced Data Sharing. In: Financial Cryptography and Data Security. LNCS. Springer (2012)
19. Slamanig, D.: Dynamic Accumulator based Discretionary Access Control for Outsourced Storage with Unlinkable Access. In: Financial Cryptography and Data Security. Springer (2012)
20. Slamanig, D.: Efficient Schemes for Anonymous Yet Authorized and Bounded Use of Cloud Resources. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 73–91. Springer, Heidelberg (2012)
21. Toegl, R., Pirker, M., Gissing, M.: acTvSM: A Dynamic Virtualization Platform for Enforcement of Application Integrity. In: Chen, L., Yung, M. (eds.) INTRUST 2010. LNCS, vol. 6802, pp. 326–345. Springer, Heidelberg (2011)
22. Trusted Computing Group: TCG TPM Specification Version 1.2 (2007), https://www.trustedcomputinggroup.org/developers/
23. Wachsmann, C., Chen, L., Dietrich, K., Löhr, H., Sadeghi, A.-R., Winter, J.: Lightweight Anonymous Authentication with TLS and DAA for Embedded Mobile Devices. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 84–98. Springer, Heidelberg (2011)
24. Wiegele, P., Winter, J., Pirker, M., Toegl, R.: A flexible software development and emulation framework for ARM TrustZone. In: Proceedings of The Third International Conference on Trusted Systems (INTRUST 2011). Springer (2012)
25. Winter, J., Dietrich, K.: A Hijacker's Guide to the LPC Bus. In: Petkova-Nikova, S., Pashalidis, A., Pernul, G. (eds.) EuroPKI 2011. LNCS, vol. 7163, pp. 176–193. Springer, Heidelberg (2012)

Fault-Tolerant Privacy-Preserving Statistics

Marek Jawurek and Florian Kerschbaum

SAP Research
Karlsruhe, Germany
{marek.jawurek,florian.kerschbaum}@sap.com

Abstract. Real-time statistics on smart meter consumption data must preserve consumer privacy and tolerate smart meter failures. Existing protocols for this *private distributed aggregation model* suffer from various drawbacks that disqualify them for application in the smart energy grid. Either they are not fault-tolerant or if they are, then they require bi-directional communication or their accuracy decreases with an increasing number of failures. In this paper, we provide a protocol that fixes these problems and furthermore, supports a wider range of exchangeable statistical functions and requires no group key management. A key-managing authority ensures the secure evaluation of authorized functions on fresh data items using logical time and a custom zero-knowledge proof providing differential privacy for an unbounded number of statistics calculations. Our privacy-preserving protocol provides all the properties that make it suitable for use in the smart energy grid.

Keywords: Privacy, Smart Grid, Statistics, Aggregation, Stream, Fault-Tolerance.

1 Introduction

In the smart energy grid there is a conflict between privacy of consumers and utility for service providers. Utility providers can use real-time household electricity consumption data for forecasting future consumption. This consumption forecasting allows them more efficient and more stable operation of the electricity grid. However, real-time consumption data also closely reflects any activity in the household involving electrical appliances. Thus, for the consumer, it represents a privacy invasion [10,11]. Previous studies [8,9,13,15,16,17,19,22] have shown that and how information about a household and its inhabitants can be inferred from its high-resolution energy consumption data. Furthermore, any viable solution for forecasting consumption must also anticipate failing smart meters or communication links. A single failure must not prevent the real-time calculation of statistics.

The ability to calculate statistics in real-time, i.e., in the presence of failures, can also benefit many other real-world applications like public health and clinical research on patient information or any collection and monitoring where privacy-sensitive data is processed.

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 221–238, 2012.

In this paper we provide a protocol in the *fault-tolerant, private distributed aggregation model*: Many data producers (e.g. smart meters) constantly produce sensitive data items (e.g. hourly smart meter consumption measurements). An untrusted data consumer (e.g. service provider) calculates statistics, e.g., for forecasting future consumption, over subsets of these data items in the presence of failing data producers. Not even a collusion of malicious data producers and consumer may lead to the disclosure of sensitive data items.

Our protocol roughly works as follows: We use homomorphic encryption for aggregation and employ an (w.r.t. privacy) untrusted, possibly distributed key-managing authority that provides differentially private decryption services to the data consumer while neither learning data items nor statistics results.

Recently, [2] also presented a protocol for this model. Without fault-tolerance it has been considered in [23,25]. In comparison to these existing protocols our contributions can be summarized as follows:

- The accuracy of the calculated statistics is higher. In our protocol the accuracy is independent of the number of data producers, the number of data items and the number of failures.
- We do not require synchronized clocks, but only rely on logical time.
- Our protocol enables the calculation of a wider range of statistical functions (weighted sums). The statistical function can be chosen and exchanged intermittently by the data consumer without notification to the data producers.
- We do not require any group key management. Data producers may join or leave without interaction with other participants.
- We only require uni-directional communication channels between data producers and data consumers. This implies a reduced attack surface of the smart meter.

The remainder of this paper is structured as follows: Section 2 describes the contributions of our protocol in comparison to [2,23,25]. Section 3 introduces the prerequisites used in our protocol. In Section 4 we introduce a naive version of our protocol to achieve differentially private, fault-tolerant statistics in the semi-honest model. Then, in Section 5 we present the final protocol and our custom zero-knowledge proof for the malicious model. Finally, we present related work (Section 6) and conclude with a summary in Section 7.

2 Contributions

We compare our protocol to the protocols in [2,23,25]. We favorably compare in existing criteria, but also extend their set of criteria. Our extended Table 1 illustrates the differences.

2.1 Communication

We assume that many (unsynchronized) data producers constantly produce sensitive data items. They send these items to a data consumer over a uni-directional

Table 1. DP: differential privacy CDP: Computational differential privacy AO: Aggregator Obliviousness $C \rightarrow S$: client-to-server uni-directional $C \Leftrightarrow S$: interactive between client and server

Scheme	Avg comm. per user	Comm. model	Error	Fault-tolerant	Group key management required	Synchr. clocks	Security model
Naive DP	$O(1)$	$C \rightarrow S$	$O(\sqrt{n})$	Yes	No	No	DP
[23]	$O(1)$	$C \Leftrightarrow S$	$O(1)$	No	Yes	Yes	CDP AO
[25]	$O(1)$	$C \rightarrow S$	$O(1)$	No	Yes	Yes	CDP AO
[2] Sampling	$O(\frac{1}{\phi^2 * n})$	$C \Leftrightarrow S$	$O(\phi * n)$	Yes	Yes	Yes	DP
[2] Binary	$O(\log n)$	$C \rightarrow S$	$\tilde{O}((\log n)^{\frac{3}{2}})$	Yes	Yes	Yes	CDP
this paper	$O(1)$	$C \rightarrow S$	$O(1)$	Yes	No	No	DP AO

communication channel. Every data producer has constant communication cost per data item. The data consumer queries the key-managing authority for decryption over a bi-directional channel. Thus, the communication cost for the key-managing authority is linear in the number of calculated statistics.

[23] requires bi-directional communication between data producers (users) and the data consumer (aggregator) so that the users can cooperate in decrypting the threshold decryption system. [25] only requires uni-directional communication links between data producers and the consumer and has the same communication cost as our scheme. In [2] data producers also communicate uni-directionally with data consumers but have higher communication cost. This is due to redundant information provided by data producers for fault-tolerance.

2.2 Accuracy

In our protocol the accuracy of the data consumer's calculated statistics is independent of the number of data producers or data producer failures. The only error in accuracy is introduced deliberately to ensure differential privacy and that is $O(1)$ with respect to the number of data producers or failures.

[25] and [23] also introduce $O(1)$ error for differential privacy while [2] introduces polylogarithmic error dependent on the number of data producers.

2.3 Fault-Tolerance

Fault-tolerance in our protocol is introduced by a selection process of the data consumer. The data consumer can arbitrarily select a subset of available data items as input to the statistics calculation. Consequently, our scheme tolerates an arbitrary and unbounded number of failing data producers. The key-managing authority can also be distributed (as we describe in Section 4.3) and we only require a majority of key-managing authority instances to be available during a run of the protocol.

[25] and [23] do not tolerate any failures of data producers (users). The former relies on blinding shares of zero which requires all shares to be present. The latter requires the data producers to participate in a threshold decryption of the final result. This offers some fault-tolerance, but the accuracy of the result degrades with failures. [2] offers fault-tolerance, but the error of the results grows sub-linearly in the number of absent users. A bound on the maximum number of failing users needs to be pre-arranged.

2.4 Group Key Management

In our protocol, there is no group or elaborate key management. Data producers may join or leave independently at any time without any new key distribution or setup phase.

[25] requires a new distribution of the blinding shares for all data producers whenever a new producer joins. [23] requires a new distribution of key shares for the threshold decryption. [2] introduce a dynamic join & leave protocol which tolerates joining of data producers up to a certain limit. Beyond, they need to increase the tree height to accommodate more data producers. In such a case, the trusted dealer needs to distribute one additional secret key to existing users and $O(\log n)$ secret keys to new users.

2.5 Synchronization

Our protocol does not require synchronized clocks. We use logical time in order to chronologically order and thus prevent re-use of data items. All data items are encrypted under the same key.

All protocols [2,23,25] depend on communication rounds and require synchronization, although they do not explicitly mention this.

2.6 Security

Our scheme offers aggregator obliviousness (AO), i.e., the aggregator will not learn anything else but the final result. Input data and intermediate results are not available to him. Furthermore, we ensure differential privacy (DP) [4] for statistics.

Although [23,25] provide aggregator obliviousness they only guarantee the weaker notion of computationally differential privacy (CDP). In these protocols the differential privacy has to be ensured by the data producers while in our scheme the key-managing authority ensures this. [2] offers protocols with differential privacy and computational differential privacy, but does not ensure aggregator obliviousness. The data consumer (aggregator) learns some intermediate values (even if there is no fault) in order to calculate the result in the presence of failures.

3 Prerequisites

3.1 Differential Privacy

The definition of differential privacy according to [4] is the following:

Definition 1. *A randomized function K gives ϵ-differential privacy if, for all data sets D_1 and D_2 differing on at most one element and all $S \subset Range(K)$,*

$$Pr[K(D_1) \in S] \leq \exp(\epsilon) \cdot Pr[K(D_2) \in S]$$

It means, that the probability for any result of K changes only slightly (less than $\exp(\epsilon)$ if single elements are included/excluded in the set of inputs to K. In consequence, even with knowledge of the data set, the function result and arbitrary auxiliary information, it is hard for an attacker to identify whether an element is present or not. Thus, also the actual value of the element is protected.

Applied to smart metering that means, that if every statistics function on smart meter consumption data is differentially private, the individual readings can not be recovered by the receiver of the statistics's results, e.g. the service provider. Thus, differentially private statistics functions protect consumer privacy.

We will transform any function f into an ϵ-differential private version of itself by adding random noise according to a symmetric geometric distribution [6]. Specifically, if δ is the f's sensitivity, we add a sample from distribution $Geom(exp(\frac{\epsilon}{\delta}))$ to each function result $f(x)$ to make it differentially private. The parameter ϵ must be chosen according to the use case at hand. It represents the desired trade off between accuracy of the function K and how well it preserves privacy.

3.2 Paillier Cryptosystem

We only give a basic introduction to the Paillier cryptosystem, further information – including security proofs – can be found in [20]. This cryptosystem defines two functions:

- $E(m, r) \to c$, encrypts a message $m \in \mathbb{Z}_n$ with random value $r \in \mathbb{Z}_n^*$ to the ciphertext $c \in \mathbb{Z}_{n^2}^*$. All encryptions in our protocol are performed by data producers and use the key-managing authority's public key.
- $D(c) \to (D_v(c), D_r(c)) \to (m, r)$, decrypts ciphertext c to a tuple (m,r). In our protocol, all decryptions are performed by the key-managing authority using its private key.

We use this cryptosystem's ability to also recover the random parameter r during decryption.

The Paillier cryptosystem also has the following homomorphic properties:

$$D(E(m_1, r_1)E(m_2, r_2) \mod n^2) = (m_1 + m_2, r_1 \cdot r_2) \mod n \quad (1)$$

$$D(E(m, r)^k \mod n^2) = (km, r^k) \mod n \quad (2)$$

In the following, whenever we refer to the encryption E or the decryption function D applied to either singular values or vectors of ciphertexts it yields what makes most sense in the respective context: Singular values are encrypted (decrypted) to singular values and a vector of values is encrypted (decrypted) to a vector of values. The function $\boldsymbol{X} \cup v$ appends the scalar value v to the vector $\boldsymbol{X}$. Also, for improved readability, we write $f \circ h(X)$ for the sequential composition of functions f and h.

4 Protocol Description

In this Section we first give a description (Section 4.1) of a naive version of our fault-tolerant, differentially private statistics protocol. It provides differential privacy for a semi-honest data consumer, but may fail in case of a malicious data consumer. Then, in Section 4.2 we analyze this deficiency and explain the necessary restrictions that we must employ, namely the freshness of used data items and correctness of computation.

Finally, in the next Section 5, we present the full protocol with a focus on our custom zero-knowledge proof that provides fault-tolerant differential privacy in the presence of malicious data consumers with a distributed key-managing authority.

4.1 Naive Protocol

We restrict ourselves, without loss of generality, to one round of communication from data producers to the data consumer and one subsequent function evaluation and decryption. In the general case, the protocol step *Preparation* i.e., the creation of data items, is executed repeatedly in parallel and unsynchronized to the protocol steps that implement calculation and the decryption (*Calculation* and *Decryption*). Therefore, in the general case, in protocol step *Calculation* the data consumer could choose among all input values that he has received during the entire system run time, i.e. multiple rounds of data item creation. Furthermore, also without loss of generality, we assume, that data consumer and key-managing authority have pre-arranged a function f that the data consumer wishes to evaluate. We also assume, that in a setup phase all data producers obtain the public key that corresponds to the private key only held by the key-managing authority.

Preparation: Every data producer j encrypts its value v_j with a random number r_j. Every data producer sends $E(v_j, r_j)$ to the data consumer.

Calculation: The data consumer now chooses a vector

$$\boldsymbol{V} = (E(v_1, r_1), \ldots, E(v_m, r_m))$$

out of all encrypted data items that it received and *has never used before*. The data consumer calculates any statistics function of the form

$$f((x_1, \ldots, x_m), c) = (\sum_{i=1}^{m} a_i \cdot x_i) + c$$

by evaluating f's homomorphic counterpart

$$f_h(\boldsymbol{V}, E(c, r_c = 1)) = (\prod_{i=1} E(v_i, r_i)^{a_i}) \cdot E(c, r_c)$$

f_h is derived from f using the homomorphic relationship (see Equations 1, 2), the a_i are constants and c is a data consumer-chosen variable.

Then, the data consumer sends $f_h(\boldsymbol{V}, E(c, 1))$ to the key-managing authority for decryption to receive the plaintext of the statistics result.

Decryption: The key-managing authority decrypts $f_h(\boldsymbol{V}, E(c, 1))$ which yields (according to Equations 1,2):

$$D_v \circ f_h(\boldsymbol{V}, E(c, 1)) = f \circ D_v(\boldsymbol{V}, E(c, 1))$$

Note that the data consumer can use its addend c to prevent the disclosure of the statistics result to the key-managing authority.

Then, it applies the function $m_f(s, x)$ which adds a random sample according to the sensitivity of f (see Section 3.1) and a seed s to the decrypted function result in order to make the statistics function evaluation differentially private. Finally, it returns the differentially private statistics result's plaintext: $m_f(s) \circ D_v \circ f_h(\boldsymbol{V}, E(c, 1))$

4.2 Malicious Behavior

The naive protocol version described in Section 4.1 will provide differential privacy in the semi-honest model [7]. We assume that the data consumer abides by the protocol and computes the function as agreed with the key-managing authority. This includes that he only chooses data items that he has never used before. However, this freshness property of the data items is not ensured by the protocol. Thus, in the presence of malicious data consumers, the naive protocol cannot protect data items from re-use by the data consumer.

The re-use of data items has severe implications for differential privacy of continuous statistics calculation: It is easy (see Section 3.1) to determine the necessary parameter for the distribution of the random noise that makes a single function evaluation ϵ-differentially private. In [18] McSherry has analyzed how the differential privacy of combined function evaluations (queries) over intersecting data sets relates to the differential privacy of the individual functions: Let a function f_i provide ϵ_i-differential privacy. If all functions f_i cover disjoint subsets, the differential privacy for the combined function is $max_i \epsilon_i$. However, if several functions span the same set of values, the differential privacy for the combination of those functions is $\sum_i \epsilon_i$.

Thus, if we do not prevent the re-use of data items, the differential privacy will add up. Eventually, it will be insufficient for the protection of data items. In the PINQ framework [18] the aggregate knowledge gain of the attacker, i.e. the sum of differential privacy over all combined functions, is limited. This is achieved by giving every user a specific, upfront budget of differential privacy.

Once this budget is used up, no more statistics calculations (and decryptions) are allowed.

This approach is incompatible with continuous calculation of statistics. In the smart meter statistics context but also for other applications the number of statistics calculations must be unbounded. Thus, in order to limit the aggregate knowledge gain of an attacker we have to prohibit the re-use of data items and thus make it equivalent to calculations over disjoint subsets.

In order to turn the naive protocol into a secure protocol in the malicious model, the data consumer must prove to the key-managing authority that it used fresh data items and that it correctly evaluated a known statistics function. The evaluated statistics function f must be part of the data consumer's proof, because f's sensitivity determines the variance of the random noise in differential privacy (see Section 3.1).

However, Goldreich's compiler approach [7] to turn the naive protocol secure in the malicious model would not be feasible: The resulting generic zero-knowledge proof (ZKP) would be prohibitively costly spanning the entire history of data items.

Therefore, we subdivide the proof in two parts: The first part covers the fault-tolerance of the calculated statistics, i.e., the choice of k fresh data items out of n total data items. We model this choice with the function $\text{fresh}_k(n)$. Section 4.3 describes how a key-managing authority can guarantee this freshness by keeping state. Furthermore, in Section 4.4, we implement a distributed key-managing authority to make it resilient to failures.

The second part of the proof covers the the evaluated statistics function f. In Section 5 we present the final protocol that employs a custom zero-knowledge proof that covers the evaluated statistics function and completes the freshness guarantee.

Thus, in total, the data consumer proves to the key-managing authority with both parts that it honestly evaluated $f \circ \text{fresh}_k(n)$.

4.3 Freshness of Data Items

As identified in Section 4.2 the key-managing authority must only allow decryption of function results that incorporate fresh data items.

This can only be accomplished by keeping state with the key-managing authority. The main challenges with keeping state are:

1. Keep the state as small as possible and not dependent on the number of decryption requests or the frequency of data item creation.
2. Allow for a distributed key-managing authority which makes the protocol resilient to failure of the key-managing authority.

In order to keep the state at the key-managing authority manageable we further restrict the freshness property: Data items originating from the same data producer can only be used in chronological order. Items may be skipped however. The function $P_j(v)$ at data producer j returns the position (logical time) of data item v within the total order of data items of data producer j.

The key-managing authority remembers for every data producer which data item was used last in any statistics calculation. At the key-managing authority, the function $C(j)$ returns for data producer j the position (logical time) of the latest data item that was used in a statistics calculation in j's total order of data items.

Preparation: In addition to the naive protocol, every data producer j also sends $P_j(v_j)$ to the data consumer.

Decryption Request: Let p_i denote the data producer, then the data consumer compiles data item information $\boldsymbol{I}$ about the input values $\boldsymbol{V}$ (see the naive protocol) he used for the current statistics calculation:

$$\boldsymbol{I} = ((p_0, P_0(v_0), \ldots, (p_m, P_m(v_m)))$$

We provide integrity protection for $\boldsymbol{I}$ using signatures in Section 5.

The data consumer sends a decryption request with the encrypted statistics result $f_h(\boldsymbol{V}, E(c, 1))$ and $\boldsymbol{I}$ to the key-managing authority.

Validation of Freshness: The key-managing authority verifies the freshness of every used data item: For every $(p_j, P_j(v_j)) \in \boldsymbol{I}$ it checks whether $C(p_j) < P_j(v_j)$. If successful, $C()$ is updated so that: $C(p_j) \leftarrow P_j(v_j)$ and the key-managing authority proceeds with the decryption and the addition of random noise as in the naive protocol.

4.4 Distributed Key-Managing Authorities

It may be desirable to distribute the key-managing authority to make it resilient against failure.

However, a key-managing authority like described in the previous Section 4.3 cannot be simply split into several instances because of its state $C()$. A malicious data consumer could send decryption requests over intersecting subsets of data items to different instances of the key-managing authority. These instances would be unable to guarantee freshness of data items which would break our measures for ensuring differential privacy (as explained in Section 4.2).

Communication between different instances, in order to synchronize state, would also be prohibitively costly: Upon every decryption request an instance would have to query all other instances for the current state of their function C. We also would have to cover merging of different functions C and how to cope with failed instances.

Therefore, we propose the following augmented protocol for a distributed key-managing authority:

The total number of key-managing authority instances is $n = 2t + 1$ and we assume that at least a majority of $t + 1$ instances are alive at any time. Every instance holds its own version of function $C()$.

Decryption Request: The data consumer prepares the same decryption request for all key-managing authority instances. As in the non-distributed

protocol, it sends the encrypted statistics result $f_h(\boldsymbol{V}, E(c,1))$ and the information about the used input values $\boldsymbol{I}$ to, unlike in the non-distributed protocol, all key-managing authority instances.

Validation of Freshness: As in the non-distributed case, every instance verifies the freshness of used data items: For every $(p_j, P_j(v_j)) \in \boldsymbol{I}$ it checks whether $C(p_j) < P_j(v_j)$. If successful, $C()$ is updated: $C(p_j) \leftarrow P_j(v_j)$.

The k-th instance decrypts $f_h(\boldsymbol{V}, E(c,1))$ and applies the function $m_f(s,x)$ to make it differentially private. We describe at the end of this Section how all instances can ensure they apply the same random noise. This yields

$$x = m_f(s) \circ D_v \circ f_h(\boldsymbol{V}, E(c,1))$$

The k-th instance now creates a share s_k of secret x using a random polynomial in Shamir's secret sharing scheme. In order to ensure that all instances choose the same random polynomial, we apply the same mechanism as for the noise (see below). This resulting secret share s_k is finally returned to the data consumer.

Assembly: The data consumer collects the secret shares from at least $t+1$ key-managing authority instances and reassembles the differentially private statistics result's plaintext.

The idea of this protocol is that the data consumer has to contact at least $t+1$ instances of the key-managing authority to obtain enough shares for its statistics. Every instance in this majority will then also update their state C. This means, that subsequently, a malicious data consumer will fail to find a disjoint majority set of instances and thus cannot cannot create partitions of key-managing authority instances with different states.

Note that all key-managing instances operate completely independent of each other. On the one hand, this eliminates any synchronization problems. On the other hand, every instance must create valid shares of the same x in order to allow successful assembly of x at the data consumer. We propose a mechanism that chooses the randomness for the random noise and for the secret sharing polynomial based on a pseudo-random function with identical seed across all instances. We then use the commonly available information $\boldsymbol{I}$ as seed.

5 Zero Knowledge Proof

5.1 Properties

Already in the naive protocol version we shown how a key-managing authority can decrypt a homomorphically encrypted function result and add random noise in order to guarantee ϵ-differential privacy in the presence of a semi-honest data consumer. However, this relies on the key-managing authority's knowledge about the sensitivity of the evaluated function (see Section 3.1) and on the freshness of data items (Section 4.2). In Section 4.3 we describe how a key-managing authority can guarantee freshness with manageable state and how such a key-managing authority can be distributed (Section 4.4).

In order to complete the final protocol that ensures differential privacy over an unbounded number of statistics calculations in case of a malicious data consumer we require guarantees for the following properties:

1. The key-managing authority can verify the correctness of the provided data item information $\boldsymbol{I}$ in order to guarantee freshness.
2. The key-managing authority can verify the correct evaluation of a known statistical function on data items provided by data producers. This allows the key-managing authority an appropriate choice of random noise for making the statistics result differentially private.

In the following (Section 5.2) we describe the final fault-tolerant, privacy-preserving statistics protocol between the data consumer and the distributed key-managing authority that provides guarantees for these properties.

5.2 The Final Protocol

Like in the naive protocol version we assume, without loss of generality, a pre-arranged statistics function f and a pre-distributed key-managing authority's public key. The function $P_j(v)$ at data producer j returns the position (logical time) of data item v within the total order of data items of data producer j. At the key-managing authority (at every instance), the function $C(j)$ returns for data producer j the position (logical time) of the latest data item that was used in a statistics calculation in j's total order of data items.

Preparation: Every data producer $j \in 1, \ldots, n$ encrypts its data item v_j with a random number r_j. Furthermore, the data producers encrypt the r_j with another chosen random value r'_j and create signatures over the random's ciphertext and the data item identification tuple $(j, P_j(v_j))$: $S_j(E(r_j, r'_j), (j, P_j(v_j)))_j$. Every data producer sends $(j, P_j(v_j))$, $E(v_j, r_j)$, $E(r_j, r'_j)$ and $S_j(E(r_j, r'_j), (j, P_j(v_j)))_j)$ to the data consumer.

Calculation: The data consumer now chooses a vector

$$\boldsymbol{V} = (E(v_1, r_1), \ldots, E(v_m, r_m))$$

out of all encrypted data items that it received and corresponding vectors of encrypted randoms $\boldsymbol{R}$, of data item information $\boldsymbol{I}$ and signatures $\boldsymbol{S}$:

$$\begin{aligned}
\boldsymbol{V} &= (V_1, \ldots, V_m) | V_i \in \{E(v_1, r_1), \ldots, E(v_n, r_n)\} \\
\boldsymbol{R} &= (R_1, \ldots, R_m) | R_i \in \{E(r_1, r'_1), \ldots, E(r_n, r'_n)\} \\
\boldsymbol{I} &= (I_1, \ldots, I_m) | I_i \in \{(1, P_1(v_1)), \ldots, (n, P_n(v_n))\} \\
\boldsymbol{S} &= (S_1, \ldots, S_m) | \\
&\quad S_i \in \{S(E(r_1, r'_1), (1, P_1(v_1))_1, \ldots, S(E(r_n, r'_n), (n, P_n(v_n)))_n\}
\end{aligned}$$

Then, exactly like in the naive protocol version, it calculates the pre-arranged function f of the form

$$f((x_1, \ldots, x_m), c) = (\sum_{i=1}^{m} a_i \cdot x_i) + c$$

by evaluating f's homomorphic counterpart

$$f_h(\boldsymbol{V}, E(c, r_c = 1)) = (\prod_{i=1}^{m} E(v_i, r_i)^{a_i}) \cdot E(c, r_c)$$

f_h is derived from f using Equations 1 and 2, the a_i are constants and c is a data consumer-chosen input variable.

If $\boldsymbol{V} = (E(v_1, r_1), \ldots, E(v_m, r_m))$ and f_h and f' have been derived from the same function f the following holds:

$$D_r \circ f_h(\boldsymbol{V}, E(c, 1)) = f' \circ D_r(\boldsymbol{V} \cup E(c, 1)) \tag{3}$$

Then, the data consumer sends $f_h(\boldsymbol{V}, E(c, 1))$, $\boldsymbol{R}$, $\boldsymbol{I}$ and $\boldsymbol{S}$ to each available key-managing authority instance for decryption.

Decryption: Every key-managing authority instance performs two checks: First, for the freshness guarantee: For every $I_j = (j, P_j(v_j)) \in \boldsymbol{I}$ and $R_j \in \boldsymbol{R}$ it checks the signature S_j and the freshness of every used data item: $C(j) < P_j(v_j)$. If successful, $C()$ is updated so that: $C(j) \leftarrow P_j(v_j)$.

Second, for correct function evaluation with data producers' data items: It derives function f' from f. f' represents the homomorphic operations on the random part of the ciphertext of f (see Equations 1,2):

$$f'(\boldsymbol{V}) = \prod_{i=1}^{m} D_r(E(v_i, r_i))^{a_i}$$

It checks if (according to Equations 3):

$$D_r \circ f_h(\boldsymbol{V}, E(c, 1)) = f' \circ D_v(\boldsymbol{R})$$

The key-managing authority instance accepts the proof if both checks pass. Then, like in the naive protocol it applies the function $m_f(s, x)$ which adds a random sample according to the sensitivity of f (see Section 3.1) and seed s to the decrypted function result: $x = m_f(s) \circ D_v \circ f_h(\boldsymbol{V}, E(c, 1))$. We use a pseudo-random function on the data item information $\boldsymbol{I}$ as seed s in order to create the randomness, so that every instance of the distributed key-managing authority will add the same random sample and thus obtains the same result x. This allows the k-th key-managing authority instance to finally create a share s_k of x (using the same seed s, but a different pseudo-random function) and return that to the data consumer.

Remark 1. Note that, if the data consumer supplies different input $\boldsymbol{I'}$ to one key managing authority, then its resulting share will be uniformly randomly distributed. Our ZKP only covers malicious behavior in the computation of the statistical function f by the data consumer and not its decryption. The data consumer can always prevent the correct decryption of the statistics, but this only prevents him from obtaining the result.

Assembly: The data consumer assembles the decryption shares from at least $(t+1)$ key-managing authority instances and reassembles the decrypted, differentially-private, statistics result:

$$D_v(m_f(\boldsymbol{I}) \circ_h f_h(\boldsymbol{V}, E(c,1)))$$

Theorem 1. *Our zero knowledge proof is complete, sound and honest-verifier zero-knowledge.*

Proof. Following Remark 1, we can assume that the vector $\boldsymbol{I}$ is static and omit it from the proof. For completeness: Assume that the data consumer, according to protocol step *Calculation*, chooses $\boldsymbol{V}$,$\boldsymbol{R}$ and $\boldsymbol{S}$, computes the pre-arranged, linear function f_h with $\boldsymbol{V}$ and supplies the function result $f_h(\boldsymbol{V}, E(c,1))$, $\boldsymbol{R}$ and $\boldsymbol{S}$ to the key-managing authority. Then the key-managing authority will verify that $\boldsymbol{R}$ only contains fresh entries and that those entries have valid signatures in $\boldsymbol{S}$ and according to the last check of the protocol's step *Decryption* it will accept the proof if: $D_r \circ f_h(\boldsymbol{V}, E(c,1)) = f' \circ D_v(\boldsymbol{R})$
Which is true if $\boldsymbol{R}$ contains the corresponding entries to $\boldsymbol{V}$ (using Equation 3):

$$\begin{aligned} & D_r \circ f_h(\boldsymbol{V}, E(c,1)) = f' \circ D_r(\boldsymbol{V} \cup E(c,1)) \\ =& f'((D_v(R_1), \ldots, D_v(R_m)) \cup 1) \\ =& f' \circ D_v(\boldsymbol{R}) \end{aligned}$$ □

For soundness, we show by contradiction: If our ZKP was not sound, IND-CPA game for the Paillier cryptosystem would be solvable. By providing a simulator that reduces the IND-CPA challenge to the problem of forging the ZKP we will show that forging the ZKP must be hard as well.

For soundness of the prover we need to differentiate between data producers (system environment) and consumer (prover). We use – in this and the subsequent part of the proof – a signature oracle for the data consumers that returns any valid signature. In this part of the proof, the data consumer has no access to this oracle.

Our simulator first interacts with the IND-CPA challenger. It provides two values as input: r_0 and r_1. The challenger returns the encryption of a randomly chosen value $c_x = E(r_0|r_1, r'_x)$. Without loss of generality, assume that the function f takes m data items from data producers as inputs. Now, the simulator takes $(m-1)$ triples $\{v_i, r_i, r'_i\}$ and encrypts and signs them, resulting in $(m-1) \times \{E(v_i, r_i), E(r_i, r'_i), S(E(r_i, r'_i), (j, P_j(v_i)))\}$. The simulator also signs c_x and some data item information tuple y (using the oracle) and supplies this incomplete ZKP $Z = \{(m-1) \times \{E(v_i, r_i), E(r_i, r'_i), S(E(r_i, r'_i), (j, P_j(v_i)))\} \cup \{c_x, S(c_x, y)\}$ to the data consumer. Note that in this setup the data consumer needs to complete the ZKP, e.g. by supplying the missing (input value) ciphertext: $E(v_x, r_0|r_1)$. Then the data consumer engages in communication with the key-managing authority, with our simulator listening in their communication. The data consumer passes the full ZKP $Z \cup E(v_x, r_0|r_1)$ to the key-managing authority and obtains the decrypted function result $f(\boldsymbol{V})$. The simulator still

knows all v_i (except v_x) and can then infer v_x from the function result. Consequently, the simulator can also infer a plausible $E(v_x, r_0|r_1)$. Now, the simulator performs the last check: It creates $E(v_x, r_0)$ and $E(v_x, r_1)$ and compares them to $E(v_x, r_0|r_1)$. It thus solves the IND-CPA problem. □

For honest-verifier zero-knowledge we give a simulator of the authority's (verifier's) view only from its input and output. We stress, that for zero-knowledge of the verifier we can view the data producers and consumer as one entity. The verifier – and not the prover – holds the private keys for the encryption of the values submitted during the proof. Consequently we need to simulate the plaintexts of the encryption. Furthermore, we need to take care of the randomization of the encryption, since it is not always uniformly distributed.

In order to simulate $f_h(\boldsymbol{V}, E(c, 1))$ the simulator uniformly chooses two random values $x \in \mathbb{Z}_n$ and $r \in \mathbb{Z}_n^*$. The simulated value is $E(x, r)$. Note that, since c can be drawn uniformly from $\mathbb{Z}_n$, so can x. Then, in order to simulate $\boldsymbol{R}$ the simulator uniformly chooses $m-1$ random values $r_i \in \mathbb{Z}_n^*$ $(1 \leq i \leq m-1)$. It sets $r_m = r(\prod_{i=1}^{m-1} r_i)^{-1}$. The simulated values are the ciphertexts $E(r_i, r_i')$ where the r_i' are drawn uniformly from $\mathbb{Z}_n^*$. All the random values in the message of the ZKP are identically distributed – including their arithmetic relationship.

Last, in order to simulate the signatures $\boldsymbol{S}$ the simulator invokes the signature oracle on the simulated vector $\boldsymbol{R}$. If this oracle works properly, all signatures will be valid. This completes our simulator.

6 Related Work

Attacks on Smart Metering Privacy: Several works [8,9,15,16,17,19,22] have covered the area of behavior analysis from energy consumption traces. The authors of [17] developed a system which inferred behavior events from the electrical consumption data and evaluated the performance of their approach with control data from video surveillance. This enables them to construct a sample disclosure metric that "...associates data quality (accuracy of readings, time resolution, types of readings, and so on) from a particular source with the information that the data could reveal." [15] focuses on detecting and characterizing different appliances according to load signatures.

Privacy-Preserving Smart Meter Billing: A cryptographic approach to privacy in smart meter billing has been presented in [12]. A privacy component homomorphically calculates the price locally in the household and only reports the final price and a cryptographic proof over commitments on the meter readings to the supplier. The proof allows the supplier to verify the correct calculation and tariff without ever receiving plaintext values. Another cryptographic approach very similar to [12] is described in [24]. It focuses on realizing a variety of different tariff types with a cryptographic solution and reducing the complexity of the calculations in the smart meter.

Private Distributed Aggregation Model: In the following we describe two general protocols [25,23] in the *private distributed aggregation model* and one [2] in the *fault-tolerant, private distributed aggregation model*.

[25] describes a system where users provide homomorphically encrypted data items with individually added random noise. The aggregator homomorphically aggregates these data items and decrypts them. During aggregation individual amounts of random noise cancel each other out except for a specific amount that guarantees computational differential privacy.

In [23] bi-directional communication between users and the aggregator is required. First, users supply their homomorphically encrypted and randomized value to the aggregator for aggregation. Then, the aggregator sends the aggregated perturbed value back to all users for decryption. Every user removes its individually added random noise (except for differential privacy) and replies with a decryption share of the the final result which are finally combined by the aggregator.

In [2] a system of intersecting user groups allows the aggregator to compensate for user failures with the help of redundant information. As every user's data item is represented in the aggregate of several groups it can be recovered even if some group aggregates cannot be recovered due to user failures.

Privacy-Preserving Smart Metering Statistics: Furthermore, we provide an overview on different (non-fault-tolerant) approaches that ensure privacy for different applications of smart metering data: leakage detection [5] by homomorphic encryption, general aggregation [14] and comparison by secret-sharing, aggregation by third party [1,21] or aggregation [1] by randomization.

In [5] a privacy-preserving detection algorithm for leakages in electricity distribution has been proposed. Using homomorphic encryption and secret sharing several smart meters engage in a protocol with a (potentially malicious) substation that searches for differences between its own measurement and the sum of all smart meter readings. While both compute the sum in a private manner, our approach (obviously) does not require a measurement of aggregate consumption, requires only uni-directional communication and tolerates failing smart meters.

In [14] the authors propose several protocols for privacy-preserving aggregation or comparison on smart metering data based on secret-sharing. They also provide an extensive analysis w.r.t. cryptographic verifiability, their computational and communicational complexities and their applicability for settlement and profiling applications. Their work differs from ours mainly in their requirement of smart meters to form groups over which aggregates are computed which also implicates meter to meter communication and implicitly assumes no failures.

Furthermore, in [1] a model for measuring privacy in smart metering is developed and subsequently two different solutions to privacy are presented: A trusted third party approach, where aggregation takes place at the third party and alternatively the approach of masking individual values with added noise directly at the smart meters which is canceled out in the sum over all meters at the supplier. The trusted third party of their first approach is able to calculate arbitrary statistics. However, as it has to perform all calculations instead

of one decryption, as in our case, it requires more computational power and storage than in our protocol. Their second approach has reduced variance in the function result with increasing number of meters in contrast to our approach. Ours, however, is independent of the total amount of smart meters contributing readings and allows the calculation of arbitrary linear functions.

Finally, in [21] the authors suggest to use the electrical grid infrastructure as a trusted third party to anonymize up-to-date consumption values constantly sent out by smart meters. This third party verifies the authenticity of the data, removes the identifying information and forwards it to the consumer of this data. This solution provides privacy by anonymizing information at a trusted third party before forwarding them to the data consumer which requires a similar computational and storage effort as in [1].

Application of Differential Privacy in Smart Metering: In [3] the authors build upon a previous work [24] and additionally apply differential privacy to hide any information leakage that is implied by the billing amount itself. Using differential privacy the consumer presents a randomized bill to the supplier that is higher then the actual one. Their approach especially targets the remaining information leakage implied by the bill calculation function itself, in which it is similar to ours. However, we target forecasting where smart meter readings are required in real-time and thus aggregation over time is not an option.

7 Summary and Conclusion

We provide a protocol in the *fault-tolerant, private distributed aggregation model* that allows a data consumer to calculate unbounded statistics (weighted sums) over homomorphically encrypted sensitive data items from data producers. Our protocol is fault-tolerant, as the data consumer can choose to calculate over an arbitrary subset of all available data items, i.e., failing data producers do not prevent the statistics calculation. It is also privacy-preserving, because a (possibly distributed) key-managing authority ensures differential privacy before responding to the data consumer's decryption request for the homomorphically encrypted statistics result. Our protocol is secure against malicious data consumers (aggregators) and features aggregator obliviousness, differential privacy and a uni-directional communication model between data producers and data consumers. In comparison to the other existing protocol [2] in this model the accuracy of our statistics calculation is higher, particularly in the presence of failures. We are also more flexible, since we allow the non-interactive, intermittent change of the statistics function and we do not require group key management.

In summary, our protocol provides the best suitable foundation for privacy-preserving, real-time energy consumption forecasting using smart meters. Future work is to extend the availability of consumption data to other data consumers, such as regulators that need to verify single data items, and to incorporate other data sources, such as weather forecast and television programs.

Acknowledgments. Marek Jawurek's work in this paper was partly funded by the German Federal Ministry of Economics and Technology (BMWi) as part of

the MEREGIOmobil project with reference number 01ME09007 and the MEREGIO project with reference number 01ME08006.

References

1. Bohli, J.-M., Ugus, O., Sorge, C.: A privacy model for smart metering. In: Proceedings of the First IEEE International Workshop on Smart Grid Communications (in Conjunction with IEEE ICC 2010) (2010)
2. Chan, T.-H.H., Shi, E., Song, D.: Privacy-preserving stream aggregation with fault tolerance. In: Proceedings of the 16th International Conference on Financial Cryptography and Data Security, FC 2012 (2012)
3. Danezis, G., Kohlweiss, M., Rial, A.: Differentially private billing with rebates. Cryptology ePrint Archive, Report 2011/134 (2011), http://eprint.iacr.org/
4. Dwork, C.: Differential Privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part II. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
5. Garcia, F.D., Jacobs, B.: Privacy-Friendly Energy-Metering via Homomorphic Encryption. In: Cuellar, J., Lopez, J., Barthe, G., Pretschner, A. (eds.) STM 2010. LNCS, vol. 6710, pp. 226–238. Springer, Heidelberg (2011)
6. Ghosh, A., Roughgarden, T., Sundararajan, M.: Universally utility-maximizing privacy mechanisms. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 351–360. ACM, New York (2009)
7. Goldreich, O., Warning, A.: Secure multi-party computation (1998)
8. Hart, G.: Nonintrusive appliance load monitoring. Proceedings of the IEEE 80(12), 1870–1891 (1992)
9. Hart, G.W.: Residential energy monitoring and computerized surveillance via utility power flows. IEEE Technology and Society Magazine (June 1989)
10. Heck, W.: Smart energy meter will not be compulsory. NRC Handelsblad (online) (April 2009), http://www.nrc.nl/international/article2207260.ece/Smart_energy_meter_will_not_be_compulsory
11. Jamieson, A.: Smart meters could be 'spy in the home'. Telegraph (UK) (online) (October 2009), http://www.telegraph.co.uk/finance/newsbysector/energy/6292809/Smart-meters-could-be-spy-in-the-home.html
12. Jawurek, M., Johns, M., Kerschbaum, F.: Plug-in privacy for smart metering billing. CoRR, abs/1012.2248 (2010)
13. Jawurek, M., Johns, M., Rieck, K.: Smart metering de-pseudonymization. In: ACSAC, pp. 227–236 (2011)
14. Kursawe, K., Danezis, G., Kohlweiss, M.: Privacy-Friendly Aggregation for the Smart-Grid. In: Fischer-Hübner, S., Hopper, N. (eds.) PETS 2011. LNCS, vol. 6794, pp. 175–191. Springer, Heidelberg (2011)
15. Lam, H., Fung, G., Lee, W.: A novel method to construct taxonomy electrical appliances based on load signaturesof. IEEE Transactions on Consumer Electronics 53(2), 653–660 (2007)
16. Laughman, C., Lee, K., Cox, R., Shaw, S., Leeb, S., Norford, L., Armstrong, P.: Power signature analysis. IEEE on Power and Energy Magazine 1(2), 56–63 (2003)
17. Lisovich, M.A., Mulligan, D.K., Wicker, S.B.: Inferring personal information from demand-response systems. IEEE Security and Privacy 8(1), 11–20 (2010)

18. McSherry, F.: Privacy integrated queries: an extensible platform for privacy-preserving data analysis. Commun. ACM 53(9), 89–97 (2010)
19. Molina-Markham, A., Shenoy, P., Fu, K., Cecchet, E., Irwin, D.: Private memoirs of a smart meter. In: Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building, BuildSys 2010, pp. 61–66. ACM, New York (2010)
20. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
21. Petrlic, R.: A privacy-preserving concept for smart grids. In: Sicherheit in vernetzten Systemen: 18. DFN Workshop, pp. B1–B14. Books on Demand GmbH (2010)
22. Prudenzi, A.: A neuron nets based procedure for identifying domestic appliances pattern-of-use from energy recordings at meter panel. IEEE Power Engineering Society Winter Meeting 2, 941–946 (2002)
23. Rastogi, V., Nath, S.: Differentially private aggregation of distributed time-series with transformation and encryption. In: Proceedings of the 2010 International Conference on Management of Data, SIGMOD 2010, pp. 735–746. ACM, New York (2010)
24. Rial, A., Danezis, G.: Privacy-preserving smart metering. Technical report, Microsoft Research (November 2010)
25. Shi, E., Chan, T.-H.H., Rieffel, E.G., Chow, R., Song, D.: Privacy-preserving aggregation of time-series data. In: NDSS (2011)

Evading Censorship with Browser-Based Proxies

David Fifield[1], Nate Hardison[1], Jonathan Ellithorpe[1], Emily Stark[2], Dan Boneh[1], Roger Dingledine[3], and Phil Porras[4]

[1] Stanford University
[2] Massachusetts Institute of Technology
[3] The Tor Project
[4] SRI International

Abstract. While Internet access to certain sites is blocked in some parts of the world, these restrictions are often circumvented using proxies outside the censored region. Often these proxies are blocked as soon as they are discovered. In this paper we propose a browser-based proxy creation system that generates a large number of short-lived proxies. Clients using the system seamlessly hop from one proxy to the next as these browser-based proxies appear and disappear. We discuss a number of technical challenges that had to be overcome for this system to work and report on its performance and security. We show that browser-based short-lived proxies provide adequate bandwidth for video delivery and argue that blocking them can be challenging.

1 Introduction

While the Internet began as a research network open to all types of data, many nations now filter Internet traffic. The OpenNet Initiative, which tracks public reports of Internet filtering, lists a large number of countries that filter Internet traffic. Some countries block sites like YouTube and Facebook while others block access to web content containing materials they consider objectionable. The list of countries includes well-publicized examples in Asia and the Middle East as well as Australia and several European countries. Over half of the 74 countries tested in 2011 imposed some degree of filtering on the Internet [1].

Censored users try to bypass the censor by connecting to sites through a proxy. Several proxy systems have emerged to help users circumvent censorship. Most notable among these is Tor [2], which, while originally designed to provide anonymity, has also seen wide use in circumvention. Other proposals include Telex [3] and Ultrasurf [4]. The existence of circumvention systems makes the censor's job harder: The censor must block access to all circumvention tools, in addition to any resources it would ordinarily block, if it is to remain effective. Our goal is to enable access to circumvention even in the face of such blocking.

Proxy-based circumvention systems generally need to solve three problems:

1. **Rendezvous.** A rendezvous protocol lets a user in the censored region send and receive a small amount of information (a few bytes) from the circumvention system to outside the censored region, for the purpose of introducing a

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 239–258, 2012.

user to a proxy. Rendezvous protocols are designed for low-rate traffic and are intended to be difficult to block. Tor, for example, developed rendezvous protocols to distribute the IP addresses of Tor bridges, which are relays whose addresses are not universally known so they are harder to block [5].

2. **Proxy creation.** The circumvention system relies on proxies outside the filtered region to relay traffic from the client to the desired site. In response the censor can masquerade as legitimate users to discover proxy addresses and promptly block them. One way to combat this Sybil attack is to constantly create new proxies outside the filtered region. As proxies get blocked, new ones take their place. Rapid proxy creation is the main topic of this paper.
3. **Camouflage.** Once the client has the address of a non-blocked proxy, it needs to camouflage its conversation with the proxy so that the session cannot be blocked by traffic analysis. The goal of camouflage is to make the conversation look like acceptable traffic such as an e-commerce transaction, a voice conversation, or part of a multiplayer game. Concrete proposals for camouflage include obfsproxy [6] and StegoTorus [7]. We treat camouflage as an independent layer and do not discuss it in this paper.

A complete circumvention system must also address secure client software distribution, an install system, and secure integration with a web browser. The Tor Project already handles these issues quite well and we do not discuss them here.

In this paper, we focus on Tor because it is widely deployed, with hundreds of thousands of daily users [8]. Tor consists of a network of several thousand volunteer nodes, known as relays. Clients build a three-node encrypted *circuit* through the network by selecting relays from a public relay directory. The client sends data to an *entry node*, which forwards it through an intermediate and an exit node, after which the exit node sends the data to the destination host on the public Internet. More details about the Tor design are given in [2]. The fact that relays are listed in a public directory makes them easy to block [9] – a countermeasure to this blocking is Tor *bridges*, relays whose addresses are not made universally known. Bridges, too, have been found susceptible to partial enumeration and blocking [10–12].

Our Contributions. In this paper we propose a new approach to rapid proxy creation. The core idea is to use the power of the web to create millions of short-lived proxies, each proxy being active for only a few minutes. To do this, we enlist the help of volunteer web sites (e.g., personal home pages) outside the filtered region that want to support an open Internet. These volunteer web sites are unrelated to the destination web site that the censored user is trying to reach.

A volunteer web site simply embeds a small "Internet Freedom" badge on its web pages (see Fig. 1). The badge is a tiny user interface on top of some JavaScript code. When a web browser outside the filtered region visits the volunteer site, it runs the JavaScript program, which relays traffic to and from the filtered region through the visitor's web browser. In effect, the browser visiting the volunteer's site becomes a short-lived proxy. As soon as the visitor navigates away from the page, the browser unloads the badge and the proxy disappears leaving no trace on the visitor's machine. Surprisingly, browsing the web through

these ephemeral browser-based proxies, even when hopping from one proxy to another, works quite well. Our experiments in Sect. 4 show that the filtered client can sustain more than enough bandwidth to carry a Tor tunnel.

Fig. 1. Flash proxy badge on a web page. It runs in a visitor's web browser, and the (optional) counter increments for each client served. Clicking on the badge disables it.

Our flash proxies[1] are at the opposite extreme of Tor bridges. Tor bridges are intended to serve for a long time; they are created at a relatively low rate and there are only a few thousand of them. Our proxies are only active while the visitor's browser is viewing the volunteer web page – often just a few minutes – but a new proxy is created every time someone visits a volunteer page, potentially creating a pool of millions of active proxies at any given time. Building a reliable transport using these ephemeral proxies presents interesting challenges discussed in Sect. 3. For completeness we discuss new strategies for rendezvous in Sect. 6.

2 Threat Model and Assumptions

Our setting includes four key players. The **client** owns a computer in the filtered region and is trying to access a web site outside the filtered region. We assume the client has complete control of his computer and, in particular, can install arbitrary software on the computer. The **target web site** (e.g., Facebook) is located outside the filtered region and is generally oblivious to the circumvention effort. That is, it does not cooperate with nor try to prevent circumvention. The **circumvention tool** attempts to relay traffic back and forth from the client to the target host. The tool may include client software as well as network infrastructure inside and outside the filtered region. The **adversary** (censor), who tries to filter traffic, must allow acceptable Internet traffic such as banking and e-commerce, but tries to block all objectionable traffic, including possibly blocking systems used by the circumvention tool. The definition of what is acceptable and objectionable is up to the adversary.

The adversary achieves its goal by installing hardware and software at Internet Service Providers (ISPs) in the filtered region. It can inspect all network traffic

[1] The word "flash" is meant to evoke an idea of quickness and ephemerality. Our first implementation of the system used Adobe Flash, which partly inspired the name. The current implementation uses JavaScript rather than Flash.

to and from the filtered region and block any packet it wishes. However, the adversary operates under the following three constraints:

Line Rate. It must operate at line rate and cannot noticeably slow down legitimate traffic to and from the filtered region. In other words, it has little time to make its decision whether to block or allow packets and flows.

No Control of Clients. We assume the adversary does not have software installed on the client computer. This assumption matches current reality where filtering happens at the network and not at the end host. If the adversary can force users to run filtering software – either by law or by using technologies such as Trusted Computing (TCG) – then the circumvention problem becomes much harder, though not impossible. Some circumvention tools are designed for a public-kiosk environment where the client is browsing the web over a public terminal that may have filtering software installed. These tools, however, are much harder to use and therefore in this paper we assume the user is in full control of his computer.

Minimal Collateral Damage. To the extent possible, the adversary tries to avoid collateral damage. It tries to minimize the impact to acceptable flows, such as those used for banking and e-commerce. The adversary cannot simply shut down all Internet traffic to the filtered region, as this would halt all banking and e-commerce in the region. While some governments, including Egypt, Libya and Syria, have recently implemented filtering by literally shutting down all Internet traffic, most deployed Internet filters try to minimize collateral damage so as not to hurt commerce.

3 Rapid Proxy Creation Using Flash Proxies

Next we present an architecture for creating a large number of short-lived proxies. The flash proxy system uses browsers all over the Internet as ephemeral proxies. The design is in large part dictated by a limitation of web socket technologies, namely that they can only make outgoing TCP connections, and cannot receive connections as a normal proxy would. In the flash proxy model, the proxy connects to the client, not the other way around.

Our implementation uses the Tor pluggable transports support introduced in Tor version 0.2.2.32 [13]. Pluggable transports are designed to enable different circumvention and tunneling schemes without having to modify core Tor code. Our system requires a program called a *client transport plugin* running on the client and a *server transport plugin* running on the relay. The functioning of these pieces is described further below. The integration of pluggable transports in Tor means that using these auxiliary programs is fairly painless.

Two of the five system components (the Tor client and relay) are the same as are used for any Tor connection. The remaining three (the proxy, facilitator, and transport plugins) are specific to the flash proxy system (see Fig. 2).

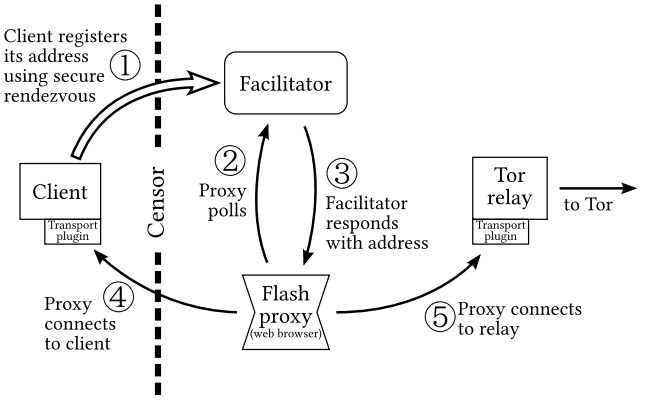


Fig. 2. Flash proxy architecture. When a client wants service, it registers with the facilitator and waits. A flash proxy appears and polls the facilitator for a client address. Once the proxy obtains a client address it connects to both the client and to the Tor relay and proxies traffic back and forth for the client.

Tor Client. The censored user runs a Tor client, configured to use a **client transport plugin**.

Flash Proxy. The proxy itself is a small application that runs in a web browser and is hosted on a volunteer web site. We have made two implementations of the flash proxy, one using Adobe Flash and one using JavaScript with WebSocket. We have presented the proxy as a small "Internet Freedom" badge as shown in Fig. 1. Whenever someone visits the page, the badge begins running in their browser. When they navigate away the badge is unloaded leaving no trace. The badge communicates with the **facilitator** to find the addresses of clients that need a connection. Once it has a client address, it connects to the **client transport plugin**, then to the **server transport plugin** running on the **Tor relay**, and begins proxying data between them. The badge itself runs in the background and has no impact on the visitor's interaction with the volunteer site. It is important to understand that censored users do not need to see the badge – in fact web sites including the badge can be blocked by the censor – because it is visitors to the web site, not the web site itself, who provide the proxy access.

Facilitator. The facilitator keeps track of client registrations and hands them out to **proxies** when requested. When the **client transport plugin** starts it registers with the facilitator using a robust rendezvous protocol such as one of those described in Sect. 6. The facilitator may be blocked by the censor; the purpose of the rendezvous protocol is to allow the client to send its IP address – just a few bytes – to the facilitator despite blocking. The flash proxies that communicate with the facilitator do not go through the censor and are therefore not impacted by blocking the facilitator. After registration, when a browser-based flash proxy is available for service, the **facilitator** assigns it to serve a censored client.

Transport Plugins. The Tor client and relay both run their own transport plugin to read from and write to the flash proxy tunnel. In the case of WebSocket, the transport plugins respond to the HTTP handshake, encode and decode binary data, and put data into WebSocket frames. The client transport plugin has the additional important duty of bridging Tor's "connect outward to a proxy" expectation and the "receive a connection from a proxy" reality of the flash proxy architecture: it receives connections from the Tor client (using ordinary TCP), and on another port receives connections from flash proxies (using WebSocket), maintaining a pool of connections of each type. Whenever there is at least one connection in both pools, the transport plugin links them up and begins to relay data traffic back and forth. When a flash proxy disappears, the transport plugin begins to use another proxy connection from the pool. The Tor client is mostly unaware that a new flash proxy is put in place.

Tor Relay. Any Tor relay can be used as the entrance to the Tor network, as long as it runs the server transport plugin. The relay may be blocked from the point of view of the client.

3.1 Establishing Connections

Programs running in a web browser, whether they use WebSocket, Flash, or other socket-like technologies, share a limitation: they cannot open a listening socket and wait passively for connections; they can only initiate new outgoing connections. This forces an inversion of the usual proxy model: It is the client (with a full complement of socket operations) that listens for a connection, and the flash proxy (limited by running in a web browser) that connects to it. This inversion is the source of most of the complexity of the model.

A less important restriction is that web browser security policies generally prevent programs from making connections to arbitrary destinations. The browser requires some positive cooperation from the destination that indicates that the connection should be allowed. For us this poses no problem as the three destinations the proxy connects to – the facilitator, the client transport plugin, and the server transport plugin – are cooperative. What this means in concrete terms for the WebSocket implementation is that the facilitator must allow cross-origin resource sharing (CORS) [14] by sending an Access-Control-Allow-Origin header field, and that the client and server must be able to answer the WebSocket handshake and proxy the tunneled data to Tor. For Flash it means that the endpoints must serve a crossdomain policy [15], a small chunk of XML specifying which connections should be allowed.

The fact that the flash proxy may not receive connections has one important advantage. The flash proxy operator (i.e., web browser) may be behind network address translation (NAT) or a firewall that doesn't allow incoming connections, and it doesn't affect the architecture. The unfortunate corollary is that clients must be able to receive connections, which generally means not being behind NAT or else being able to configure port forwarding. We feel that this is at least the right way to allocate the burden, if it must fall somewhere. We expect flash

proxies to greatly outnumber censored clients; running a proxy should be as simple as possible while clients are already motivated to take technical steps for secure communication. Ideally flash proxies would be usable even behind NAT without any special configuration using NAT punching techniques; Sect. 5.2 discusses difficulties and solutions.

Figure 2 illustrates the components operating in sequence in a sample session:

1. The client starts Tor and the client transport plugin, and sends a registration to the facilitator using a secure rendezvous mechanism. The transport plugin begins listening for a remote connection.
2. A flash proxy comes online and polls the facilitator.
3. The facilitator returns a client registration, informing the flash proxy where to connect.
4. The proxy makes an outgoing connection to the client, and this connection is received by the client's transport plugin.
5. The proxy makes an outgoing connection to the transport plugin on the Tor relay, and begins sending and receiving data between the client and relay.
6. Sooner or later, the flash proxy disappears and breaks the connection between the the client and the relay. The client's transport plugin then switches immediately to another available proxy. In the unlikely event that none are available, the transport plugin waits until one becomes so. In this process, existing tunneled TCP streams are broken, but see Sect. 5.2 for ideas on transparently keeping TCP connections intact.

The client transport plugin maintains a pool of up to five live proxy connections, in order to make switching between them faster. Only one of the proxies will be used at a time, but when the one being used disappears, the socket handshake is already finished with one of the reserves, for a lower delay in establishing a new connection. The transport plugin discards reserve proxies as they go offline, and the facilitator replenishes the reserve as long as there is extra capacity, so there is no danger of uselessly switching to a defunct proxy.

The flash proxy has features for quality of service and good network behavior. A single proxy limits itself to 10 simultaneous client–relay pairs. Its built-in adjustable rate limit avoids using too much of the operator's bandwidth. The proxy checks the platform it's running on, and disables itself if it is on a mobile phone or similar device where bandwidth that may be limited or expensive.

The goal of the facilitator is to fairly allocate proxies to clients and attempt to provide good service for all. The facilitator also seeks to even the load carried by each proxy whenever there is spare proxy capacity. The facilitator keeps track of the number of clients given to each proxy. When a new client arrives, it is given to the proxy with the least load, with ties being broken randomly. (In the usual case we expect most proxies to be idle most of the time, so this will be a random selection from among all the idle proxies.) While there is unused proxy capacity, the facilitator seeks to provide each client with a small number of redundant proxies for faster switching. When one of the proxies goes offline (which can be detected because the proxy ceases polling), the facilitator will

attempt to assign a new proxy to the client that it had been serving. As we discuss in Sect. 6, client communication with the facilitator is low-bandwidth and infrequent; therefore there is no explicit message for a client to "unregister" itself. Instead, the facilitator estimates that a client no longer needs service when a proxy reports that it has been unable to connect to a client, and a reasonable timeout elapses. (The timeout is to allow other proxies an attempt to serve the same client, in case the failed proxy is experiencing network problems.)

One unusual feature of this architecture is that the client using the proxy does not know in advance where it is connecting to. In fact, the client transport plugin accepts a dummy destination address only to comply with the pluggable transports protocol, then ignores it and tunnels to the proxy's other endpoint, namely the Tor relay. In effect, the first hop of a Tor circuit is made blindly, and then the client may choose the second and third hop from the directory of relays. Section 5.1 explains why this does not affect security (briefly, the use of Tor allows authenticating the destination after the connection is made). We also note that this is the same situation a client finds itself in when using a bridge address it has not seen before.

As a practical matter, there may be more than one facilitator, to diffuse the risk of one's being breached, and to distribute load. However we do not rely on there being multiple facilitators for the purpose of evading blocking; we assume that all facilitators will be permanently blocked by the censor. The censored clients send data only over a secure rendezvous channel whose characteristics – being very low-bandwidth, write-only, and infrequently used – make it much harder to block. Section 6 discusses potential rendezvous protocols in more detail.

4 Experimenting with Ephemeral Flash Proxies

4.1 Throughput

We measured the maximum throughput of a single proxy, independent of Tor and any other network bottleneck, by running transport plugins, facilitator, proxy, and web server all on the local host. We then started between 1 and 50 simultaneous HTTP downloads of a 10 MB file directly over TCP, through a WebSocket proxy, and through a Flash-based proxy. The test was run on the Linux kernel version 3.2 in a QEMU instance with a 2.27 GHz CPU. The proxies ran in Firefox 8.0.1 with Flash Player 11.1 r102. The proxies' bandwidth and connection limits were disabled. Figure 3 shows the time taken for each of the simultaneous clients to download the file. Each column contains multiple dots, but in the cases of Flash sockets and direct download the dots approximately coincide.

Both flavors of proxy are slower than a direct download. The WebSocket measurements show download times being roughly proportional to the number of clients. There is much variance, perhaps due to unfairness in the way that the browser schedules WebSocket message events.

A surprising effect is seen with Flash sockets: the time taken to download per client is almost constant, up to about 16 clients. We suspect some kind of internal limit within Flash Player that restricts individual connections to

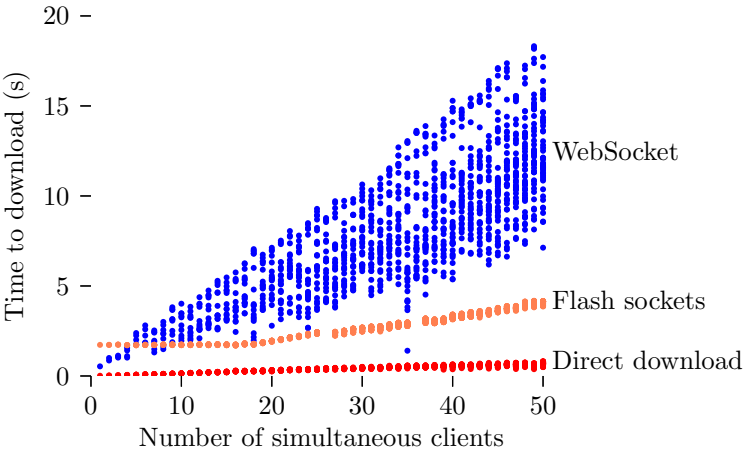


Fig. 3. Time taken to download a 10 MB file through many simultaneous connections

no more than about 6 MB/s. Because of this restriction, WebSocket is faster than Flash sockets for small numbers of clients. Eventually natural limits on bandwidth become more restrictive than this artificial limit, and the download time becomes approximately linear.

The WebSocket API does not expose a "read" procedure to read from the socket; rather, one registers a "message" callback function that is called whenever data have already been read. Received packets not immediately handled by the application are buffered. At high data rates this buffer can grow without bound, possibly eventually crashing the proxy. This issue is not specific to flash proxies, and has not been a problem at Tor rates, but it leaves proxies somewhat exposed to attacks by malicious clients or relays. WebSocket provides a way to control the size of the write buffer; a complementary mechanism for the read buffer would be sufficient to solve this. Flash sockets don't have this problem as they buffer data at the operating system kernel level, so the TCP window prevents too much data from being received at once.

These results show that once connected, a flash proxy can provide more bandwidth than is commonly used by a Tor circuit (which is in the range of hundreds of kilobytes per second [16]), and even enough for online video sites. With few clients, browser-based proxies are stable and have predictable performance.

4.2 Switching between Proxies

It is expected that a Tor client will have to switch between flash proxies frequently as they go offline. To measure the effect this has on performance, we set up two browsers running the flash proxy program, driven by a script that turned each proxy on for 10 seconds and then off for 6, so that the two proxies' periods of operation would overlap by 2 seconds in a cycle (see Fig. 4). There was

Fig. 4. Alternation of proxies. Solid bars indicate when a proxy is operating.

always at least one proxy available, so any delay was caused purely by the need to switch between them. We did a test over the public Tor network, retrieving the same 5 MB file used by the torperf measurement tool [17]. (It is not possible to use torperf directly because it does not retry downloads.) This table shows the difference between normal Tor, an uninterrupted flash proxy, and alternating flash proxies. The direct Tor connection was configured to use the same entry bridge that the flash proxies use. Because the performance of the Tor network changes over time, all the tests were run in direct succession.

The switching experiment was run 20 times. Figure 5 shows the measurements for each test. Switching between proxies causes a visible increase in the mean time to download, but most of the variance is caused by Tor itself.

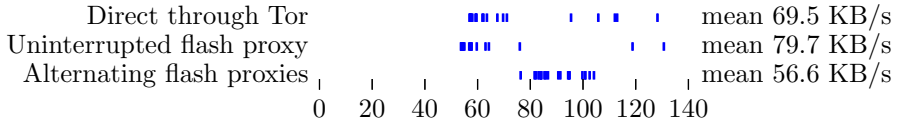


Fig. 5. Time taken for each trial in the switching experiment. Each data set contains 20 measurements.

In these results we see that a flash proxy that stays online gives performance roughly the same as connecting directly to Tor. However, overall speed is 20 to 40 percent lower when using proxies that are unreliable, because of the overhead of building new Tor circuits. The results of the throughput experiments show that a flash proxy can provide more bandwidth than Tor can use, so it is not surprising that downloading though a flash proxy is as fast as downloading over Tor. On the other hand, reestablishing a broken connection is more expensive. In this test, the time taken to restart the download was variable, but commonly left only 3–6 seconds of useful downloading time out of each 8-second cycle.

4.3 Capacity

The flash proxy badge has several properties that, we believe, make it easy to adopt. It can be installed just by pasting an HTML snippet. It does not require any special access or configuration on the web server, nor cooperation from ISPs or the servers being proxied to. Web pages displaying the badge do not have to be in any particular network position. The sites displaying the badge may even be blocked by the censor, because it is the viewers of the site, not the site itself, that provide a circumvention bridge. In this section we seek to predict how many censored users can be helped, given a certain number of flash proxies.

Here we make some simplifying assumptions. First, we approximate the pattern of visits to a web page as a Poisson process, with different arrivals being independent, and the times between arrivals being exponentially distributed. This is not a very strong assumption; there is evidence that the request arrival process for a single web page is Poisson [18], even though the process for individual packets is not [19]. Second, we assume that traffic has reached a steady state, without edge effects from web pages appearing or disappearing, and without variability due to time of day or other factors.

Under the Poisson arrival approximation, the traffic to a single web page is governed by two parameters: λ, the mean number of arrivals per unit time; and μ, the mean visit duration. The times between arrivals are exponentially distributed with density function $\lambda e^{-\lambda t}$. The mean inter-arrival time is $1/\lambda$. We do not assume anything about the distribution of visit durations (which correspond to proxy lifetimes), other than that a mean exists and that the process generating them is strictly stationary (unchanging over time).

Different web pages have different traffic characteristics, but we may treat a group of pages uniformly using the same two parameters, for the following reasons. The exponential distribution has the property that the minimum of several exponentials with rates $\lambda_1, \ldots, \lambda_k$ is also exponentially distributed with rate $\lambda_1 + \cdots + \lambda_k$. The minimum time to the next arrival is exactly what inter-arrival time is, so the inter-arrival times from several web pages come from their own exponential distribution. The aggregate mean visit duration across several web pages is just the mean of all of their individual visit durations. The flash proxy is configured to serve up to 10 clients at a time; this may be handled by multiplying the arrival rate by 10. In this section we think of each proxy as handling only one client at a time.

If proxy badges collectively provide service with parameters λ and μ, then the expected number of operating proxies (and hence the number of clients that can be served) at any time is $\lambda\mu$ (which is Little's law [20]). For example, if 3 proxies come online every second ($\lambda = 3$) and each lasts 60 seconds on average ($\mu = 60$), then we expect 180 proxies to be operating at once on average.

Let us substitute some measured numbers into this formula. We wrote a program to record the visits of viewers capable of running Flash, and installed it on a personal home page for about two weeks. For each visitor, the program recorded (*start-time*, *end-time*, *bandwidth*, *latency*). The program recorded 784 visits. we discounted the two longest-lived connections of approximately 1.9 and

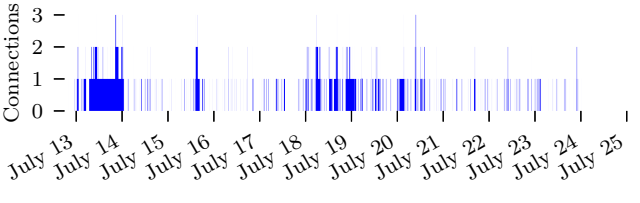


Fig. 6. Measured simultaneous connections on a personal home page

3.1 days (the next longest connection after those was 15 hours). We also ignored 28 connections that had a measured bandwidth of 0, and an apparent outlier that came much later than any other connection. This left 753 entries.

Figure 6 shows the number of simultaneous visitors to the web page for each day of the experiment. There were at times as many as three visitors viewing the page at once, but only about 17% of the time was there even one visitor. This web page acting alone would not be able to provide good proxy service, because of the long periods during which there are no visitors and hence no flash proxies. It would take several such pages working together to fill in the gaps and provide continuous service with high probability.

We estimate the quantities λ and μ by taking the sample means of inter-arrival time and visit duration, respectively:

Mean inter-arrival time	$1/\lambda$	1407.6 s
Mean arrival rate	λ	0.00071/s
Mean duration	μ	285.8 s

If 100 web pages like the one we tested were to install the flash proxy badge, then we estimate an overall arrival rate of $\lambda' = 100 \cdot 10 \cdot \lambda \approx 0.71$ per second (recall that one proxy can handle 10 simultaneous clients). The combined mean visit duration is unchanged: $\mu' = \mu$. The expected number of clients that these 100 web pages can support is $\lambda'\mu' \approx 203$.

The number of clients scales linearly with the number of proxies, so 1,000 web pages with similar traffic characteristics would be able to support 2,030 clients on average, with the same expected duration. Pages with more visitors (lower inter-arrival time) or longer visit durations will provide better service.

4.4 Field Testing

We tested connecting to the Tor network from within China over flash proxies in December, 2011. We were running one proxy, and a few others were run by unknown web users. The proxies worked as expected, and we could use Tor despite its well-known blocking by China. This test used only a simple HTTP-based rendezvous, and not any of the advanced rendezvous methods from Sect. 6, so it could have been blocked by IP address. Nevertheless, the test shows that the proxies work as they are supposed to, once the rendezvous step is completed.

5 Discussion

5.1 Security and Privacy

Flash proxies can in principle be used to reach any kind of tunnel, such as an HTTP proxy or SSH tunnel. Tunneling through Tor, however, brings attractive security features. The most obvious is enhanced anonymity: It is not easy for a network observer, including the flash proxy itself, to know the final destination

of traffic. Users who would like to use Tor for circumvention, but cannot because Tor relays and bridges are blocked, therefore do not have give to up anonymity when using a flash proxy transport.

A second feature provided by Tor is encryption. Once a proxy has connected to its two endpoints, it sends and receives only ciphertext. This is important not only for the client, but for the temporary operator of the proxy. It would not be friendly toward proxy operators to allow them to send plaintext that could potentially run afoul of a corporate firewall, for example. Note, however, that encryption can also be obtained by directing flash proxies to forward traffic to an SSH server in the open Internet, in which case the client runs an SSH client instead of a Tor client.

The third feature is authentication. Even though a malicious flash proxy can connect to any endpoint it wants, it cannot in this way trick a client into connecting somewhere unexpected. The most it can do is deny service. If the proxy connects to something that is not a Tor relay, the Tor client will fail to make a connection and show error messages. Once a connection is made to the first relay, the proxy cannot interfere with the client's circuit construction because it is already within a layer of encryption. A flash proxy operated by a malicious adversary gets to choose the first relay, so it could always choose a relay it controls, and thereby always see the first hop of a circuit. However, such an adversary can already do something similar simply by running its own bridges and waiting for connections from ordinary bridge users.

How might the flash proxy system be attacked? In our threat model, the adversary "wins" if it is able to prevent access to sites that it would block normally. The adversary can also control some fraction of all flash proxies and relays. Here we list a number of attacks and possible mitigations.

Client Enumeration. The censor can query the facilitator and get a list of IP addresses of presumed circumventors. This is a consequence of the fact that flash proxies connect to censored clients, instead of the other way around. Note however that the adversary is already in a position to learn the addresses of users of a circumvention tool, just by sniffing at the firewall, but having a centralized facilitator makes it easier. One possible mitigation of this attack is sheer numbers. If there are many more proxies than there are clients, then most proxies will not have any client to serve at all. (The facilitator will just return "no client address" when queried.) The adversary will be competing against all legitimate flash proxies in querying the facilitator to learn client addresses, and most attempts will be fruitless. With successful deployment we expect to have millions of available proxies at any given time meaning that most, including many of the adversary's, will be idle.

Flooding Client Registrations. The facilitator accepts client registrations from anywhere as a consequence of the indirect rendezvous mechanism. An adversary can flood the facilitator with fake client addresses. When a legitimate proxy retrieves one of these fake client addresses, it will waste time trying to give service to that presumed client. This temporarily removes an otherwise useful proxy from the system. It should be noted, though, that legitimate registrations

will still get through and will eventually be picked up by a proxy. Mitigating this attack requires enough proxies to absorb the busywork created by the adversary. It is also possible to limit the number of registrations accepted over a period of time from a given source address.

Exhausting Client Registrations. The adversary can pose as multiple flash proxies to the facilitator, and ask for addresses of clients, which it then ignores. The aim is to cause the facilitator to think that the client address has been given to enough proxies that it need not be given out any more, and prevent legitimate proxies from seeing the address. This too is mitigated by numbers. If there are enough legitimate proxies, the adversary will have difficulty claiming all the registration "slots" for a particular client. As long as one of the proxies is legitimate, the client will be able to get service.

Protocol Fingerprinting. The fact that they run through a browser means that flash proxy tunnels look different from ordinary TCP connections at the network level. With WebSocket, there is the HTTP handshake to begin the connection, followed by data in a structured framing format. Flash sockets are distinguishable because they begin with a crossdomain policy request. Even if obfuscation is used to hide the Tor protocol, it must all happen within the framework provided by browser sockets. The unblockability of the system rests in part on the type of connections used by the proxies (e.g., WebSocket) also carrying enough ordinary traffic that the censor will be reluctant to block that type of connection wholesale. It remains to be seen whether WebSocket will have this level of popularity, but support for WebSocket in major browsers is a promising trend.

5.2 Usability

Usability is listed as a security requirement in the design of Tor [2]. In this section we examine how much additional effort is required to use flash proxies on the part of users and relay operators, and how this affects usability.

Relay Operators. On the part of Tor relay operators, the only additional requirement is to run the flash proxy server transport plugin. This is a matter of installing the plugin and adding a line to the relay's configuration file.

Clients. Our programs are designed to work with the Tor pluggable transports design, so configuration is fairly easy. A user must run the transport plugin, add a few lines to the Tor configuration file, and then be able to register with the facilitator. Our transport plugin is written in Python, which is an additional requirement beyond plain Tor. If the user is not able to receive direct TCP connections, the more technical step of enabling port forwarding must be taken. A Windows installer can automate this process and we plan to build one as interest increases.

Limitations. The fact that proxies are expected to disappear causes a qualitative difference in network behavior when using flash proxies. These differences

vary in importance depending on the application being used. Basic web browsing, with relatively short-lived connections, works quite well. When connections are short, there is a smaller chance that they will be interrupted. When they are interrupted, fixing a partially downloaded page only requires refreshing the page. Some browsers, such as Firefox and Chrome, automatically restart failed downloads making this seamless. Large web downloads and video work less well, because it is more likely that a download will be interrupted in the middle. Again, browsers that automatically restart failed downloads from the point of failure make this less of a problem.

The fact that clients must not be behind NAT is an impediment to usability. A NAT traversal mechanism that works within our threat model would be a great benefit. Typical NAT traversal technologies, such as STUN (Session Traversal Utilities for NAT) [21] and RTMFP (Real Time Media Flow Protocol) [22], rely on a stable third-party server to facilitate the connection, which is trivially defeated by the censor blocking the third party by IP address. (Also we believe it is better to avoid informing a third party of each flash proxy connection if it can be avoided.) Tricks involving low-level packet manipulation, for example pwnat [23], are not available to browser sockets. Ideally, any NAT traversal scheme will not require both the client and the proxy to know each other's IP address, so that facilitator registration can remain unidirectional. New technologies like WebRTC [24] may fill this need in the future, if they become sufficiently popular that flash proxies' use of them does not stand out as unusual.

Applications that inherently rely on long-lived connections, e.g. SSH, have a poor user experience. The session ends completely whenever a proxy goes offline, losing state and requiring a new login. For SSH, the most important property of a proxy is long lifetime, which flash proxies in the wild usually do not provide. On the other hand, protocols that use long connections but do not maintain much server-side state can still work tolerably well with enough application support. For example, an Internet Relay Chat (IRC) client that automatically reconnects after losing its proxy server will be usable with only brief interruptions. The flash proxy system can be extended with buffers in the transport plugins, to enable connection continuity across different flash proxy sessions.

5.3 Deployment Scenarios

Our proposal hinges on volunteer web pages hosting the flash badge. Recall that while the badge is running in the visitor's browser it has no impact on the visitor's experience at the site. Nevertheless, web site admins will likely want to configure the badge to best suit their objectives. For this reason the badge is highly configurable for different scenarios:

- **Opt-in vs. Opt-out:** Sites that maintain user accounts (e.g., Facebook) can add a check-box to user profiles allowing users to specify whether they want to participate in this system or not. The Flash object will only be served to users who check the box. The default settings for the checkbox is a matter of policy. Moreover, the badge itself can be configured to only begin proxying after the visitor clicks on the badge.

- **Geographic Limitations:** The badge can be configured to only serve clients in certain geographies.
- **Proxy Targets:** If Tor is not used, the badge can be configured to only proxy to specific domains such as YouTube and Facebook.
- **Connectivity:** The badge is already configured to shut down when running on a mobile device so as not to use up the host's data plan. It can similarly be configured to shut down when it detects a low-bandwidth connection so as not to interfere with the host's browsing experience.

We envision two types of deployments. Commercial sites that are already blocked, such as YouTube, can deploy the badge on their home page so that their uncensored users can help their censored users reach the site. In these deployments, the badge will only forward traffic to YouTube, and possibly only from regions where YouTube is known to be censored.

Another type of deployment can come from people who are concerned about Internet filtering and choose to deploy the badge on their home page and blogs. Visitors to those pages will help censored users connect to Tor. People who choose to deploy the badge on their blogs can customize it as they wish, possibly serving clients only in certain geographies or forwarding traffic only to certain domains.

6 Rendezvous Protocols

The flash proxy system relies on a robust rendezvous mechanism that lets clients in the censored region register their IP address with the facilitator. If the censor could simply block the facilitator then the flash proxy system would break down.

The flash proxy rendezvous problem is related to, but somewhat different from the rendezvous problem in Tor. In the Tor system, rendezvous is used to communicate the address of an unblocked Tor bridge *into* the censored region. In the flash proxy system rendezvous is used to communicate the address of a client *out of* the censored region.

Our facilitator design is reasonably modular, so that the facilitator itself does not need to understand all of the potential rendezvous methods, or know in advance which will be used. The censored client runs a program implementing a certain rendezvous method; an uncensored recipient that understands the protocol then forwards the registration to the facilitator on the client's behalf. In this way, new rendezvous methods can be tested without redeploying the facilitator, and without requiring the cooperation of those who run the facilitator.

We experimented with two methods for communicating a small amount of information out of the censored region. The first uses cloud-based storage servers outside the blocked region. The second uses cooperating web sites. We note that many other systems, including Skype and Telex [3] can be used for rendezvous.

Storage Servers. Cloud storage systems like S3 and Box.net provide a good opportunity for rendezvous. These services are difficult to block due to large collateral damage caused by blocking them. We experimented with a system that uses a variety of such cloud storage systems. As long as one system among many is unblocked, the information will get out.

The idea is that the facilitator signs up for an account on all cloud storage servers that the system will use. It sets permissions so that anyone may write to the storage server, but only the facilitator may read from it. Clients in the censored region who want service write their IP address to all the storage servers over HTTPS and the facilitator retrieves them by periodically polling the servers.

Our experiments showed that this method has high latency. For example, it can take several seconds for data written to S3 in one geographic location to become available for reading in another location. Hence, while this channel is insufficient for general network access, it is fairly well suited for rendezvous.

Web Sites. Our second approach, which is more speculative, is to embed rendezvous messages in standard HTTP requests.

A client wishing to register with the facilitator would send an HTTP request to a participating web server where one of the HTTP headers (e.g., a cookie header) contains a crafted random string. The web server would be configured with a secret key that lets it recognize the pattern in the HTTP request and forward the request to the facilitator. The censor, who does not have the secret key, cannot recognize that the request encodes a flash proxy registration request. If many web servers participate in this scheme then the censor's only hope for blocking these messages (besides blocking all web sites) is to attempt to compile a list of cooperating sites to block, by crawling the web and trying to use each site to rendezvous with the facilitator. If many web sites participate in this scheme, then blocking all of them will cause considerable collateral damage.

To experiment with this approach, we modified the Apache web server to accept rendezvous requests for the flash proxy system. This modification could be packaged as an Apache module for ease of deployment. Users might discover participating web sites through word of mouth or social media; or if there are enough of them, a browser plugin could discover them automatically during the course of normal web browsing. When the client wishes to register with the facilitator it chooses a random web site that is participating in this rendezvous mechanism. Next, it encrypts the message $(0^{32} \| \text{IPaddr})$ with that server's public key and embeds the resulting ciphertext in an HTTP request sent to the web server. The ciphertext is embedded in a session cookie header that is normally used when interacting with this web site. The censor cannot tell the difference between a real session cookie and a rendezvous request, since both appear to be random strings. The client uses this header in a request for a page that does not exist (which can be generated by choosing a random string or a random English word). When the web server receives an HTTP request with a session cookie that results in a 404 response, the server tries to decrypt the contents of that session cookie. If it detects the 0^{32} pattern, it forwards the encoded IP address to the facilitator. The server sends a 200 response so that the client knows that the rendezvous request was successful. Note that the public-key decryption step is only used on HTTP requests that result in a 404 response.

In our prototype, we used identity-based encryption (IBE) [25], in which a server's public key is its domain name. With IBE, clients do not need to use some other mechanism to learn a server's public key before using it for rendezvous.

7 Related Work

Blocking resistance is an ongoing area of research. It appears that there is no single, simple solution, but rather many different problems must be solved together for effective blocking resistance.

Simple proxy systems suffer from the problem that they typically reside on static IP addresses and are expected to be relatively long-lived, making them easy to block. It will always be a race between users striving to find new usable proxy addresses and the censor to block them. Additionally, single-hop proxies cannot provide the same security properties that Tor can; for example they may be susceptible to subversion of the proxy server itself.

Infranet [26] is a design to conceal traffic that would otherwise be blocked within seemingly normal HTTP traffic. A user's ordinary innocuous browsing provides cover for messages sent and received to some other blocked server. Infranet requires the cooperation of unblocked web servers. We note that Infranet's covert channel could be used as a method of rendezvous.

In some cases, the mere presence of statistical anomalies in traffic, such as an encrypted stream, could be enough to cause a censor to block access to a resource, merely on suspicion that the opaque stream may be used for circumvention. This illustrates the need for protocol camouflage, making circumvention traffic look like "normal" traffic, at least from the point of a censoring firewall. The Tor Project made a proposal for pluggable transports [13], which allow using a variety of different camouflage or circumvention techniques, depending on what is mutually supported by the client and entry node.

Recently Wustrow et al. introduced Telex [3], a system that allows cryptographically "tagging" normal TLS streams so that an ISP-level router may redirect it to a blocked destination. Telex is different from other proposals in that it involves action by entities in the middle of the network path, not only at the edges. Unlike with Infranet, unblocked web sites do not need to know about or participate in the circumvention.

The Tor Project has enhanced blocking resistance by introducing non-public bridges in addition to public relays [5]. Bridge addresses go not into the main relay database, but into a special bridge database. The set of bridges is partitioned and each partition is distributed over a different channel, for example email, HTTPS, or non-electronic means. The bridge database hands out only a few bridge addresses at a time, with some additional restrictions making it hard for one user to learn many or all of them. The goal is to prevent a scarce and precious resource (bridge addresses) from being completely enumerated, while still allowing anyone to learn the few addresses they need to get connected.

8 Conclusions

We have introduced flash proxies, a new method of producing many short-lived proxies that for the purpose of censorship circumvention. Rather than attempting to hide the addresses of proxies, we aim to create so many that it is not feasible to

block them all. Performance experiments are promising and the system is ready to be deployed. The flash proxy badge is installed on the project home page, and visitors are already acting as flash proxies. There are already sufficiently many active flash proxies to provide intermittent service. The project code is open source and available from https://crypto.stanford.edu/flashproxy/.

Acknowledgments. We are grateful for many helpful conversations on this topic with Drew Dean, Pat Lincoln, Ian Schuler, and Vinod Yegneswaran; and to Steve Beaty for help in testing.

The work is supported by the Defense Advanced Research Project Agency (DARPA) and Space and Naval Warfare Systems Center Pacific under Contract No. N66001-11-C-4022. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Project Agency and Space and Naval Warfare Systems Center Pacific. Distribution Statement "A:" Approved for Public Release, Distribution Unlimited.

References

1. The OpenNet Initiative: OpenNet Initiative Internet censorship data (2011), http://opennet.net/research/data
2. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium (August 2004)
3. Wustrow, E., Wolchok, S., Goldberg, I., Halderman, J.A.: Telex: Anticensorship in the network infrastructure. In: Proc. 20th USENIX Security Symposium (2011)
4. Ultrareach Internet Corp.: Ultrasurf proxy, http://www.ultrasurf.us/
5. Dingledine, R., Mathewson, N.: Design of a blocking-resistant anonymity system. Technical Report 2006-1, The Tor Project (November 2006)
6. Kadianakis, G., Mathewson, N.: Obfsproxy architecture (2011), https://www.torproject.org/projects/obfsproxy
7. Weinberg, Z., Wang, J., Yegneswaran, V., Briesemeister, L., Boneh, D., Wang, F. (StegoTorus: A camouflage proxy for the Tor anonymity system)
8. Tor Metrics Portal: Users (2011), https://metrics.torproject.org/users.html
9. Lewman, A.: Tor partially blocked in China (September 2009), https://blog.torproject.org/blog/tor-partially-blocked-china
10. McLachlan, J., Hopper, N.: On the risks of serving whenever you surf: Vulnerabilities in Tor's blocking resistance design. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2009). ACM (November 2009)
11. Wilde, T.: Knock knock knockin' on bridges' doors (January 2012), https://blog.torproject.org/blog/knock-knock-knockin-bridges-doors
12. Winter, P., Lindskog, S.: How China is blocking Tor. Technical report, Karlstad University (April 2012)
13. Appelbaum, J., Mathewson, N.: Pluggable transports for circumvention (October 2010), https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/180-pluggable-transport.txt
14. W3C: Cross-origin resource sharing (April 2012), http://www.w3.org/TR/cors/
15. Adobe Systems Incorporated: Adobe Flash Player 9 security (July 2008), http://wwwimages.adobe.com/www.adobe.com/content/dam/Adobe/en/devnet/flashplayer/pdfs/flash_player_9_security.pdf

16. Tor Metrics Portal: Time in seconds to complete 5 MiB request (2012), https://metrics.torproject.org/performance.html
17. Loesing, K.: torperf measurements-HOWTO (2011), https://gitweb.torproject.org/torperf.git/blob_plain/HEAD:/measurements-HOWTO
18. Arlitt, M.F., Williamson, C.L.: Internet web servers: workload characterization and performance implications. IEEE/ACM Transactions on Networking 5 (October 1997)
19. Paxson, V., Floyd, S.: Wide area traffic: the failure of Poisson modeling. IEEE/ACM Trans. Netw. 3, 226–244 (1995)
20. Little, J.D.C.: A proof of the queuing formula L = λW (1960)
21. Rosenberg, J., Mahy, R., Matthews, P., Wing, D.: Session Traversal Utilities for NAT (STUN). RFC 5389 (Proposed Standard) (October 2008)
22. Adobe Systems Incorporated: Real Time Media Flow Protocol (October 2008), http://labs.adobe.com/technologies/cirrus/
23. Müller, A., Evans, N., Grothoff, C., Kamkar, S.: Autonomous NAT traversal. In: 10th IEEE International Conference on Peer-to-Peer Computing (P2P) (2010)
24. W3C: WebRTC 1.0: Real-time communication between browsers (January 2012), http://dev.w3.org/2011/webrtc/editor/webrtc.html
25. Lynn, B.: PBC library, http://crypto.stanford.edu/pbc/
26. Feamster, N., Balazinska, M., Harfst, G., Balakrishnan, H., Karger, D.: Infranet: Circumventing web censorship and surveillance. In: Proceedings of the 11th USENIX Security Symposium (2002)

Exploring the Ecosystem of Referrer-Anonymizing Services

Nick Nikiforakis, Steven Van Acker,
Frank Piessens, and Wouter Joosen

IBBT-DistriNet, KU Leuven, 3001 Leuven, Belgium
firstname.lastname@cs.kuleuven.be

Abstract. The constant expansion of the World Wide Web allows users to enjoy a wide range of products and services delivered directly to their browsers. At the same time however, this expansion of functionality is usually coupled with more ways of attacking a user's security and privacy. In this arms race, certain web-services present themselves as privacy-preserving or privacy-enhancing. One type of such services is a Referrer-Anonymizing Service (RAS), a service which relays users from a source site to a destination site while scrubbing the contents of the referrer header from user requests.

In this paper, we investigate the ecosystem of RASs and how they interact with web-site administrators and visiting users. We discuss their workings, what happens behind the scenes and how top Internet sites react to traffic relayed through such services. In addition, we present user statistics from our own Referrer-Anonymizing Service and show the leakage of private information by others towards advertising agencies as well as towards 'curious' RAS owners.

Keywords: referrer, anonymization, online ecosystem.

1 Introduction

In the infant stages of the Internet, privacy and anonymity were mostly unnecessary due to the small size of the online community and the public nature of the available data. Today however, this has changed. People have online identities, are connected to the Internet almost permanently and they increasingly store their sensitive documents, photos and other data online in the cloud. Our new online way of life provides interesting opportunities to those who seek to exploit it. In an extreme case, corrupt regimes trying to find out what their citizens are doing and thinking, want to violate both online privacy and anonymity [9]. The threat however, need not be a far-fetched scenario or exclusive to the paranoid: large companies and organizations are also interested in the online habits of the masses for various reasons, e.g. targeted advertising.

Projects like The Onion Router (TOR) [11,25] and the Invisible Internet Project (I2P) [16] provide online anonymity to their users by routing Internet traffic through a number of relays, thus making it harder for the endpoint to trace

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 259–278, 2012.

the source of the traffic. The application-layer however, on top of the network-layer where TOR or I2P reside, could still carry information that can compromise a user's anonymity or privacy. This is especially so when a web-browser is used, because browsers leak a wealth of information about their users. A study [13] by the EFF's Panopticlick project [1] shows that, based on data typically provided by a browser, a web-site visitor can be uniquely identified in the majority of cases. Private details can be extracted even in the cases where users utilize their browsers' private modes [2] or spoof their user-agent information [15].

One particularly sensitive piece of data, transmitted with almost every HTTP request but commonly overlooked, is the referrer information in the 'Referer' [1] header, which can be used to trace the page where a visitor came from. Online services known as Referrer-Anonymizing Services (RASs) scrub this referrer information from HTTP requests, providing both anonymity to web-sites hosting links as well as privacy to users following those links. In this paper, we take a closer look at RASs. We first perform a manual analysis of popular RASs and record their workings and architectural choices. Through a series of experiments we approach RASs from three different perspectives: the perspective of sites utilizing RASs, the RASs themselves, and the destination sites receiving traffic relayed through a RAS. In the first experiment, we determine what type of sites make use of a RAS and for what reason. The second experiment analyzes the data that RASs have access to and whether they actually protect the privacy of visitors and the anonymity of linking sites. In the last experiment, we observe the reactions of popular web-sites when they are exposed to incoming links relayed through a RAS. From these experiments, we can conclude that in several cases, user privacy is sacrificed for the linking site's anonymity and that not all RASs can be trusted with private data.

The main contributions of this paper are:

- Large-scale study of RASs and their common features
- Experimental evidence of privacy and anonymity violations from RASs
- Identification of types of RAS users and the rationale behind their usage
- Analysis of third-party site responses towards traffic relayed through RASs showing that RAS-relayed traffic is occasionally not well-received

The rest of the paper is structured as follows: In Section 2 we provide background information about referrers and how they can be used but also abused. In Section 3 we describe how RASs work in general followed by a taxonomy of 30 real-world RASs in Section 4. In Section 5 we study how well these 30 RASs protect the anonymity and privacy of their users. In Section 6, we categorize users of RASs and their purpose based on data gathered by our own RAS. In Section 7, we expose popular Internet sites to requests anonymized by RASs and measure their behavior. We discuss related work in Section 8 and we conclude in Section 9.

[1] The correct spelling is 'referrer'. The misspelled word 'referer' was introduced by mistake by Phillip Hallam-Baker [17] and later adopted into the HTTP specification.

2 Background

In this section we briefly go over the workings of the referrer header and we list some valid use-cases as well as abuse-cases for this header.

2.1 Referrer Header

In the HTTP protocol, all client-side requests and server-side responses have headers and optionally a data body. At the client-side, each request contains headers that, at minimum, ask for a specific resource from the web-server, in a `GET` or `POST` manner and in the context of a specific web-site (`Host`), since typically a single web-server serves more than just one web-site. On top of these headers, browsers add a wide range of other headers, the most common of which are headers specifying the user's cookies towards a specific web-site, the user-agent and the encoding-schemes accepted by the current browser.

An HTTP header that is less known but as present in requests as all aforementioned headers, is the '`Referer`'. The HTTP referrer header is automatically added by the browser to outgoing requests, and identifies the URI of the resource from which the current request originated [24]. For instance, if a user while being on `www.example.com/index.php?id=42`, clicks on a link to `www.shopping.com`, her browser would emit a request similar to the following one:

```
GET / HTTP/1.1
Host: www.shopping.com
User-Agent: Mozilla/5.0 (X11; Linux i686)
Accept: text/html,application/xhtml+xml
Proxy-Connection: keep-alive
Referer: http://www.example.com/index.php?p=42
```

In this request, the user's browser provides to `www.shopping.com` the exact location of the page containing the clicked link, resulting in the request towards their servers. This behavior is true not only when a user clicks on a link, but also on all the non-voluntary requests that a browser automatically initiates while parsing a page. For example, all requests created while fetching remote images, scripts, cascading style sheets and embedded objects will contain the referrer header. The referrer header is traditionally omitted in one of the following cases:*(i)* when users manually type a URI in their browser's address bar, *(ii)* when users click on an existing browser bookmark and *(iii)* when users are on a HTTPS site and click on an HTTP link.

In HTML5, the web-programmer can add a special 'noreferrer' attribute to selected anchor link tags that will cause the browser not to emit the referrer header when these links are clicked [26]. At the time of this writing, from the most popular three browsers (Mozilla Firefox, Google Chrome and Internet Explorer), Google Chrome is the only browser which supports this new 'noreferrer' attribute. We believe that this lack of browser support will only amplify the hesitation of web-developers in trying and adopting new security/privacy mechanisms [31]. For this reason, we do not expect widespread use of this 'noreferrer' attribute any time in the near future.

2.2 Referrer Use-Cases

In this section we provide a non-exhaustive list of legitimate uses of the HTTP referrer for the web-server which receives the referrer-containing request:

- **Advertising Programs:** In many cases, a web-site will buy banner/link placement space on more than one third-party web-sites. Using the referrer header, the advertised site can assess the percentage of visitors coming from each third-party site and use this information to either renew or cancel its advertising contracts.
- **CSRF Protection:** Cross-site Request Forgery (CSRF) is a type of attack where the attacker abuses the established trust between a web-site and a browser [30]. In the typical scenario, a victim who has an active session cookie with a web-application is lured into visiting a malicious site which initiates arbitrary requests towards that web-application in the background. The victim's browser appends the session cookie to each request thus validating them towards the web-server. Due to the way this attack is conducted, the referrer header in the malicious requests will not be the same as when the requests are conducted by the user, from within the web-application. Thus, a simple countermeasure against CSRF attacks is to allow the requests containing the expected referrer header and deny the rest.
- **Deep-Linking Detection:** Deep-linking or 'hotlinking' is the practice of linking directly to an object on a remote site without linking to any other part of the remote site's content. In practice, this behavior is unwanted, when the object that is deep-linked was originally given to users after a series of necessary steps (e.g. giving access to a music file after filling out an online survey) [8]. By checking the referrer header before releasing the object, the site can protect itself from users who did not go through the expected series of steps. Unfortunately this approach can be easily circumvented by users who change the referrer header values of their requests to match the expected value of the remote site, using a modified browser.
- **Access-Control:** Using the same reasoning as in deep-linking, a site can enforce access control to individual pages by making sure that the visiting user arrives there only from other selected destinations. This technique is also used to provide special offers when a site is visited from another site that would normally not be visible to regular users. Wondracek et al. [28] discovered that this technique is used by traffic brokers in adult-content web-sites. As in the previous use case, this sort of access-control can be bypassed by a user with malicious intent.
- **Statistics Gathering:** In large and complex web-sites, the web-developers seek to understand whether the content layout is facilitating the user into finding the data that they need. Through the use of the referrer, web-applications can track users between pages (without the need for cookies) and find the most common visitor paths.

2.3 Referrer Abuse-Cases

The same referrer information that can be used for legitimate reasons, can be abused by web-site operators to assault a user's privacy and in certain cases even perform user impersonation attacks.

- **Tracking Visitors:** Traditionally, users associate tracking with tracking cookies. A web-site that wishes to track its users between page loads, or collaborating sites that wish to track users as they transition from one to the other can do so through the referrer header even if users delete their cookies on a regular basis.
- **Session Hijacking:** As described in Section 2.1, the referrer header contains not only the domain of the page where each request originated but the full URI of that page. That becomes problematic when web-sites use GET parameters to store sensitive data, as is the case in sites that add session information to URIs instead of cookies. In this case, the HTTP referrer will contain the session identifier of the user on the originating site, allowing a malicious operator of the target web-site to impersonate the user on the originating site [19].
- **Sensitive-Page Discovery:** It is common for web-developers to hide the existence of sensitive files and scripts in directories that are accessible from the Web but are not linked to by any visible page of the web-site. These files or directories sometimes have obfuscated names to stop attackers from guessing them. Penetration-testing tools such as *DirBuster*[2] that attempt to guess valid directories using dictionary and brute-force attacks, attest towards this practice. In such cases, if one of the sensitive pages contains an external link or external resource, the exact location of that page can be leaked to the remote web-server through the referrer header.

3 Referrer-Anonymizing Services

In Section 2 we described the possible uses and abuses of the HTTP referrer. Given the wealth of information that the referrer header provides, one can think of many scenarios where the user doesn't want to release this header to remote web-servers.

Today, users can achieve this either by configuring their browser not to send the referrer header, or through the use of Referrer-Anonymizing Services when clicking on links from sensitive web-pages. While the former approach is available on many modern browsers, it works as an all-or-nothing setting in the sense that the user cannot selectively allow the transmission of the referrer header. This can be problematic when a user navigates to web-applications that use the referrer header as a CSRF countermeasure. In such cases, the user wouldn't be able to use the web-application, unless they re-enable the transmission of the referrer header. An additional problem is for web-site owners that wish to link to third-party sites but not at the expense of uncovering their identity. A controversial

[2] http://sourceforge.net/projects/dirbuster/

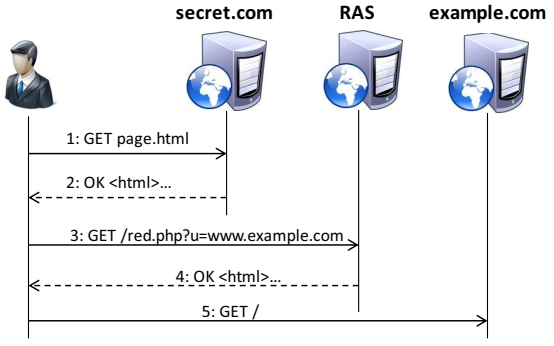


Fig. 1. HTTP messages involved in the use of a Referrer-Anonymizing Service

but popular example are 'warez forums'[3], where the descriptions of the pirated software or multimedia are usually given by linking back to the legitimate web-sites. In these cases, the web-site operators cannot rely on privacy-aware users, but must use a solution that will seamlessly work for all. This can be achieved through the use of Referrer-Anonymizing Services.

A Referrer-Anonymizing Service (RAS) is a web-service that is responsible for anonymizing the referrer header of a user before that user reaches a remote web-server. Note that, for security reasons, a web-site is not allowed to arbitrarily change a user's referrer header. The referrer header is created and emitted by the browser and thus the only way to anonymize this header is for the RAS to place itself between the site that links to an external resource, and that resource. By doing so, the RAS appears in the referrer header of the user's browser, instead of the original web-site, thus effectively anonymizing the original web-site. This technique is conceptually similar to the anonymizing techniques applied by Crowds [23] and TOR [11] where instead of the user's IP address, the link-embedding web-site is hidden.

Figure 1 shows the series of steps involved when using a RAS. In steps 1 and 2, a user requests and receives a page from `secret.com`. This page has a link that, if clicked, will eventually lead to `example.com`. However, since `secret.com` wishes to remain anonymous, it uses a RAS instead of linking directly to `example.com`. In step 3, the user clicked on the link expecting that it leads to `example.com`. However, the link creates a `GET` request towards a RAS with `example.com` as its argument. In response, the RAS generates a page (step 4) that will automatically redirect the user to `example.com` either directly, or after a timeout. In both cases, as far as the user's browser is concerned, the final request towards `example.com` originated not from `secret.com` but from the web-site of the RAS. Thus, the request depicted in Step 5, will have the redirect-causing web-page of RAS as its referrer, effectively hiding the original source of the link. Note that in the aforementioned process, `secret.com` will still reveal its presence to an external entity, but it chooses to reveal itself to the RAS instead of `example.com`.

[3] 'warez' is slang for pirated software.

The RAS can redirect the user's browser to `example.com` using one of the following ways:

- **HTTP MOVE messages:** When a web-server receives a request for a resource, it can emit a 301/302 `HTTP MOVE` message, that informs the user's browser of the 'move' of the resource, and provides it with the new location. Upon the receipt of such a message, a browser automatically initiates a new request towards the instructed location, thus completing the redirect.
- **HTML Meta-Refresh tag:** One tag of the HTML specification allows the web-developer to 'refresh' a page after a configurable number of seconds. The refresh can load the same page, or a new one. For example, `<meta http-equiv="refresh" content="5;url=http://www.example.com">` instructs the user's browser to replace the current page with the main page of `example.com` upon the expiration of 5 seconds.
- **JavaScript:** The same effect can be achieved using JavaScript, by setting the value of the `window.location` property to the desired site.

4 Taxonomy of RASs

In this section we analyze and compare common features of real-world Referrer-Anonymizing Services. We obtained a list of 30 functional RASs (listed in the appendix) by using a well-known search engine and searching for phrases related to their services, such as 'referrer anonymization' and 'hiding referrer'. The popularity of some of these services is evidenced by their high ranking in the Alexa top sites list. For instance, the most popular RAS, `anonym.to`, currently ranks higher than well-known sites such as `blackberry.com` and `barnesandnoble.com`. We summarize the discovered features of the studied RASs in Table 1.

4.1 Redirection Mechanism

By manually visiting and recording the redirection mechanisms of the 30 RASs, we found out that 73% of them were redirecting their users using the meta-refresh mechanism, 20% using JavaScript and 7% using a combination of both 302 and meta-tags. The use of the HTML meta-refresh is the most common mechanism because it doesn't require JavaScript to execute and because it allows the RAS to delay the redirect in order to show advertising banners to the visiting users. The sites that used JavaScript to redirect a visitor, used it together with a timeout function to emulate the effect of the meta-refresh mechanism.

The HTTP MOVE messages were the least used among the services for the two reasons. Firstly, redirects occurring through a 301/302 HTTP message retain the original referrer header and are thus not suited for use from RASs. The services that did utilize them, always combined them with a meta-header, where the 302 message would redirect the user to another page on the RAS's web-site which would then use a HTML meta-refresh tag. Secondly, even if the browser would clear out the referrer header, the HTTP MOVE mechanism doesn't allow for a delayed redirect, thus the services cannot use it to show advertisements.

Table 1. Common features of Referrer-Anonymizing Services

Common Feature	Percentage of RASs
Redirection:	
HTML meta-refresh	73%
JavaScript	20%
HTTP MOVE+ meta-refresh	7%
Ads	36.66%
Mass Anonymization	50%

An interesting observation is the diverse redirection behavior that different browsers display. When using Mozilla Firefox (version 9) and Google Chrome (version 16), a redirect implemented through JavaScript retains the referrer that caused the redirect. That is not a problem for RASs since the page that causes the redirect is not the original page that wishes to remain hidden, but a page of the RAS (step 4 in Figure 1). On the other hand, the same redirection mechanism in Microsoft's Internet Explorer 8, clears out the referrer. Contrastingly, Firefox and Internet Explorer clear out the referrer header in case of a HTML meta-refresh but Chrome still retains the redirect-causing referrer. From a point of user privacy, the complete clearing of the referrer is the best option for the user since the web-server cannot distinguish between users coming from web-sites that protect themselves and users who typed in the URIs or clicked on their browser bookmarks. However, the same mechanism that protects a user's privacy may negatively affect a user's security, as later explained in Section 8.

4.2 Delay and Advertising

Since all RASs that we encountered were providing their redirection services for free, there is a high probability that they attempt to capitalize on the number of incoming users through the use of advertising. From our set of 30 services, 11 (36.66%) were displaying advertising banners to the users waiting to be redirected to the destination web-site. From these services, 10 of them were constructing advertisements on the fly (through the use of client-side scripting and a chain of redirections) and only one had the same banners, statically embedded in its web-site. We also noticed that the sites that included advertising had, on average, a higher delay than the non-advertising web-sites which sometimes didn't delay the user at all. More specifically, the RASs with advertisements were redirecting the user after an average delay of 11.8 seconds whereas the non-advertising RASs were redirecting the user after an average delay of 2 seconds.

An interesting observation for the RASs that create dynamic advertisements is that all the requests towards the advertising agencies contain a referrer header which is the URI of the RAS page where the user is waiting to be redirected to the destination site. Since all RASs work by receiving the destination URI (`example.com` in Figure 1) as a GET parameter, the various advertising agencies get access, not only to the IP addresses of the RAS visitors, but also to their eventual destination. By combining this knowledge with other data, they may

be able to associate users with sites, even if the destination web-site doesn't collaborate with a specific advertising agency. Thus, in one third of the cases, the privacy of individual users is sacrificed for the anonymity of the linking site.

4.3 Mass Anonymization

Most RASs have a simple API for use of their services. All a RAS needs, is a remote URI to which it will redirect users while clearing, or substituting, their referrer header. For this reason, all RASs work in a stateless fashion. Unlike URL shortening services, where a user needs to first visit the service and generate a new short URL for their long URL, a user can utilize a RAS without first visiting the RAS's web-site. In our example in Figure 1, the administrator of `secret.com` can create an anonymized-referrer link to `example.com` simply by making a GET request with `example.com` in the `u` parameter.

This stateless nature of RASs allows for mass-anonymization of links without the hassle of registering each and every link to a remote service. From the 30 RASs that we analyzed, 50% were providing a mass-anonymization option through the use of an anonymizing script. This script, which is supposed to be included by remote web-sites, iterates over all `<a>` elements of the current HTML page and converts all links to RAS-links. Additionally, the scripts usually provide a white-list option where domains that do not need be anonymized (such as the links to local pages within a web-site) can be listed and excluded from the anonymization process. While we didn't encounter a misuse, a site including a remote anonymizing script is implicitly trusting the RAS providing it to not include malicious JavaScript along with the anonymizing functionality.

4.4 Background Activity

Apart from advertisements, RASs can use the browsers and IP addresses of visiting users to conduct arbitrary requests before redirecting the users towards their final destination. This activity can range all the way from harmless but unwanted to malicious. In our analysis of 30 RASs, we found that 2 services were performing unexpected actions that were hidden from the user.

The first service had embedded an invisible iframe that performed a search request with the keyword 'myautopass' using Google Search. While the RAS cannot steal any private data from that iframe (since the Same-Origin Policy disallows such accesses), the requests were made by the user's browser and user's IP address. As far as Google Search is concerned, tens of thousands of people [4] search for that word on a daily basis, an action which most likely affects the ranking and trend of that keyword, even if the shown links are never clicked.

The second service, instead of redirecting the user to the desired web-site, created a frameset with two frames. In the first frame, which spanned the entire page, it loaded the requested site and on the second frame it loaded a local page of that RAS. In this scenario, while the user gets access to the remote page, they never actually leave the site of the RAS. This 'sticky' behavior is common

[4] We obtained the RAS's estimated number of daily visitors using `quantcast.com`

in anonymizing web-proxies which request a page from a remote server on the user's behalf and then present the result to the user. To the remote web-server, the request appears as coming from the anonymizing proxy's IP address, thus hiding the user's IP address. Note however that in the case of RASs, this 'sticky' behavior adds no more privacy to the visiting user, since the requests are all made from the client-side and thus using the user's browser and IP address.

By analyzing the contents of the second frame, we observed that through a series of redirects, the site was opening the registration page of a file-hosting web-site, with a specific affiliate identifier. It is unclear how the operators of that RAS were expecting to fill in the registration page in an invisible frame but this method hints towards the RAS's attempt to capitalize on visiting users for monetary gains in affiliate programs.

5 Information Leakage

Whenever a user utilizes an online service, there is always the possibility that the service will retain information from the user's activity and use that information in the future, possibly for financial gains. In Section 4.2 we showed that 36.66% of all tested Referrer-Anonymizing Services used ads as a way of getting monetary compensation for their free services. In almost all cases, the ads were created dynamically, initiating a GET request for a script or an image from the RAS to the advertising company. Through the referrer header, these requests reveal to the advertising agencies which page the user is on (the RAS waiting page) and the page that the user intends to go to (the destination GET argument given to the RAS). This is problematic for two reasons: first, if the destination URI contains a secret parameter tied to a specific resource (e.g. a session identifier, a file identifier for file-hosting services or a document identifier for online collaboration platforms) this identifier will be disclosed to an untrusted third party (the advertising agency). The second reason is that advertising agencies gain more information about users and the sites they visit even if the destination sites do not collaborate directly with them.

Another instance of the same problem is encountered between the source site and the RAS. The administrator of a RAS is able to view the referrer headers of traffic towards their service and can thus discover the original pages that relay visitors through them (e.g `secret.com` in Figure 1). If the source site hosted the link to the RAS on a page with sensitive data in its URL (both path and GET parameters) – e.g. `secret.com/admin.php?pass=s3cr3t` – this will be available for inspection to the utilized RAS.

In order to measure whether the various advertising agencies of RASs make use of users' referrer headers and whether the administrators of RASs access sensitive source pages, we conducted the following experiments.

5.1 Experimental Setup

We first created and registered `fileleaks.co.cc`, a web-site supposedly providing leaked sensitive documents and then developed two crawlers that visited all

the RASs daily and requested a redirection towards URIs within our site. e.g. `http://anonym.to?http://fileleaks.co.cc/index.php?filename=[POPULAR_TOPIC]&dlid=[PER_RAS_UNIQUE_ID]`. The `dlid` contained a random identifier that was unique for all tested RASs and allowed us to accurately detect which RASs leaked our destination site.

The first crawler was simply requesting the URI in a `wget`-like way and proceeding to the next RAS. In this case, our destination URI could be leaked only through the web-server logs of the target RAS since no scripts or images were rendered. The second crawler was actually an instrumented instance of Firefox that automatically visited each site and waited for 10 seconds before moving on to the next target. The key difference between the two crawlers is that the latter one was a functional browser which executed all needed image and JavaScript requests to fully render each page. These two crawlers allowed us to roughly distinguish between URIs leaked by the web-administrator of a RAS and URIs leaked through the referrer header sent to advertising agencies.

To detect RAS administrators looking for sensitive pages in the referrer headers of their logs, we added a fake password-protected administrative panel to our site and programmed an additional `wget`-like crawler which constantly visited all RASs, pretending that the request for anonymization was originating at `http://fileleaks.co.cc/admin/index.php?password=[SECRET_PASS]&pid=[PER_RAS_UNIQUE_ID]`. The fake administrative script was logging all accesses and the `pid` GET parameter was used to distinguish leakage between the tested RASs as in our first set of crawlers.

5.2 Results

Leakage of Destination Site. In a 30-day period our monitors on `fileleaks.co.cc` recorded a total of 250 file requests using unique URIs that were made available to the Referrer-Anonymization Services as destination sites. By decoding each URI we identified that the URIs were leaked by 3 different RASs. Interestingly, all the recorded URIs were communicated to the services through our instrumented Firefox crawler and not through the `wget`-like crawler, implying that the URIs were most likely leaked by subsequent JavaScript and image requests of each RAS-waiting page. For privacy and ethical reasons, we do not identify the services by name and we refer to them as RAS1, RAS2 and RAS3.

The unique URI of RAS1 was found in 1.2% of the requests. It appears that the service leaked the URIs directly to a specific search engine, which at a later time requested the files with the exact parameters originally provided to RAS1. RAS2 and RAS3 were both leaking the requests towards services running on Amazon's Elastic Compute Cloud. The requests leaked by RAS2 (88.8% of the total requests) were revealing, through their user-agent, that they were crawlers working on behalf of a specific advertising company which specializes in identifying which pages would prove to be the best ones for ad placement for any given product or service. The last set of requests (10% of the total) with leaked

identifiers from RAS3 were also originating from hosts on Amazon's Cloud but their user-agent didn't provide any identifying details.

In total, our experiment showed that 10% of the tested RASs were, knowingly or not, leaking the referrer-header of their users to third-party advertising agencies who were recording them and using them at a later time. Given the risks associated with the leakage of a visiting user's IP address and destination site, we believe this to be a significant privacy risk.

Leakage of Originating Site. In the same period, our administrative-panel monitor recorded three visits from the same visitor. In the third visit, the user provided the exact `password` and `pid` combination that our third crawler was providing one of the 30 tested RASs through its referrer header. It is important to realize that given the nature of our crawler, the referrer header containing the `fileleaks.co.cc` administrative panel URI (and password as GET parameter) could only be retrieved from the RAS web-server's logs since the pages were always retrieved but never rendered in a real browser. Thus, no advertising agencies, or legitimate Web traffic scripts could ever access our referrer header. This shows that the administrator of one of the thirty RASs was actively searching the logs of the RAS in an effort to identify 'interesting' source sites and then manually inspect them.

6 User Categorization

In the previous section, we explored the various features of RASs and recorded which traits are prevalent and for what reasons. While this gives us an insight of the motives behind these services, it doesn't provide any specific details on the users who use these services or the actual reasons justifying their usage. In order to investigate the user-part of the RAS ecosystem, we created and advertised our own Referrer-Anonymizing Service, which we made available at `www.cleanref.us`. In a period of 25 weeks, our service received a total of 2,223 requests for referrer-anonymization. Figure 2 shows that the weekly usage of our service varied significantly. In the next sections we describe a subset of these requests according to their purpose.

6.1 Ethical Considerations

The data that was collected for this experiment are the following: For each request we recorded i) its timestamp ii) the IP address of the host performing the request, iii) its GET parameters and iv) the referring host. These were collected in a single text file on the web server, in a password-protected directory that only the authors of this paper had access to.

The data collected is a subset of the data collected by every web server on the web in standard server logs. Many web developers even share this collected information with third parties, such as Google Analytics, for the purpose of gathering usage statistics. The reason for deploying our own RAS was to identify

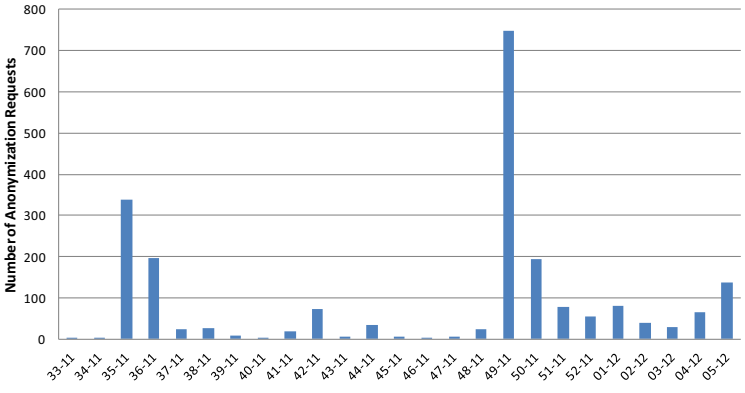


Fig. 2. Weekly number of anonymization requests that our RAS received during our study

potential abuse of such services. Since reporting to users up front that they would be part of an experiment would defeat the purpose of the experiment, our RAS did not explicitly warn users of the data collection. Because of the nature of the data collected, and the fact that these data are collected by every web server, we believe this lack of user warning to be acceptable.

We are not planning on releasing the data to the general public, and we will delete all data after publication of the paper.

6.2 Hiding Advertising Infrastructures

The greatest part of the first peek of Figure 2 corresponds to 317 requests conducted by 127 unique IP addresses. By analyzing our data, we discovered that for two days in the 35th week of 2011, our service was part of two advertising campaigns and the requests were from users who were the targets of these campaigns. While a significant number of countries was involved, the campaigns seem to have been targeting users mostly in Vietnam and the US, since these countries received 31.49% and 16.53% of the total traffic, respectively. It is unclear whether the advertising campaign was conducted through spam emails or through browser pop-up windows, however the fact that the advertiser used a Referrer-Anonymizing Service shows that they wished to conceal the exact method of driving traffic by hiding the referrer header from the final web-servers.

In our referrer logs for those days, we found 3 types of links. One type of link, was used to drive traffic directly from our service towards the destination web-site. The second type of link, was a chain of RASs all connected together in a way that allowed each RAS to redirect the user to next one, until the user is redirected to their final destination. For example, when the link:

```
http://cleanref.us/?u=http://www.refblock.com?http://cloakedlink.com
/zqkzqrfvgs
```

is clicked (or opened in a pop-up) by a user, her browser will request a page from `cleanref.us` which will redirect her to `refblock.us` which in turn will redirect the user to `cloackedlink.com`. `cloackedlink.com` is the end of the chain and when resolved, it will redirect the user to the actual site. The combination of multiple RASs allows the advertiser to hide its presence behind multiple layers of indirection. For instance, using this method, an inspection of the referrer does not reveal whether `cleanref.us` was the first part of a chain or whether another RAS redirected the user to our service using a redirection method that completely cleaned the referrer header. The last type of link, pointed to a service that received an obfuscated string as its only parameter:

```
http://linkfoobar.net/track-link.php?id=aHR0cDovL3d3dy5jbGVhbnJlZi51c
y8/dT1odHRwOi8vd3d3LnJlZmJsb2NrLmNvbT9odHRwOi8vY2xvYWtlZGxpbmsuY29tL3
pxa3pxcmZ2Z3M=
```

The `id` argument passed to the anonymized `linkfoobar.net` web-site is a Base64-encoded string that, when decoded, makes a chain of RASs similar to our previous example. As the name suggests, this is most likely the first part of the chain where the advertiser first tracks that someone clicked on a tracked link and then proceeds to redirect the user to the final destination through a chain of RASs.

By combining the knowledge of all three categories of referrer URIs found in our logs, it is evident that the advertiser mixes the order of RASs in their chains in order to reveal only part of their traffic and infrastructure to each RAS. Thus in some cases, our service was the last in the chain of RASs, sometimes in the middle and occasionally the first service that began the chain of RASs after the user's click was registered by `linkfoobar.net`.

6.3 Remote Image Linking

Throughout our experiment we noticed several groups of requests (e.g. weeks 42-11, 04-12 and 05-12 in Figure 2) towards image files, some of them located on popular image hosting sites. By sampling some of the destination URLs, we noticed that the requested images were almost always of an adult nature. For the sampled requests, we also reviewed the referrer header when it was available. The observed linking sites fell into two categories. In the first category, the sites linking to adult images through our RAS were forums where the users could write new posts and include links to images. In this case, the RAS was added to hide the linking site from the image hosting site, since the uploaded images didn't conform with the rules of the latter. Some of the requests were using more than one RAS chained together as shown in Section 6.2.

In the second case, the linking sites were personal sites that were requesting images either from image hosting sites or directly from adult-content sites forming a client-side image collage. As in the former category, the owners of such pages were hiding the exact location of their personal pages from the sites hosting the linked images.

6.4 Web-Mashups

A web-mashup is a combination of data and functionality from more than one remote service, which has more value than any one of the remote services by itself. The highest peek in Figure 2 stems from the adoption of our service from a book price-comparison web-application. The destinations of these requests are popular online bookstores and other price-comparison sites in Europe and the US. Each request contained the ISBN number of a different book. In a period of 9 weeks, the web-application initiated a total of 1,273 requests for anonymization of which 1,198 (94.10%) formed a unique combination of ISBN number and third-party service. Given the vast majority of unique queries for books, we believe that the requests happen once and their results are cached in the web-application. The usage of a RAS in between the book price-comparison web-application and the other online bookstores, allows the former to retrieve information from the latter without revealing its location or purpose.

7 Tracking of Anonymizers

In the previous sections, we analyzed the existing Referrer-Anonymizing Services, we listed some ways that one can legitimately or illegitimately use them and provided empirical data on the types of users that they attract. The final part of the RAS ecosystem are the destination sites (`example.com` in Figure 1). It is interesting to know, whether popular web-sites are aware of the existence of RASs and if they are, how do they react towards traffic relayed through them.

In order to identify differences in served content, we conducted an automated experiment involving the top 1,000 sites of the Internet according to Alexa. The first time we visited each Alexa link, we provided the URL of a popular search engine as our referrer header, simulating a user who followed a search result from that search engine. We then repeated the same request 30 times, each time providing as a referrer, the URL of one of our 30 RASs.

Given the dynamic nature of the Web, simply comparing the pages of different visits or their hashes is insufficient to differentiate between changes that were due to the web-site's reaction to a certain referrer header and usual expected changes, such as different timestamps or randomly generated data embedded in the source-code of the page. In order to overcome this obstacle we used a combination of two methods. The first method was to apply Arc90's *Readability* algorithm [3] which attempts to separate and show the most important content on a page while hiding the less important. In the second method, we recorded the number and type of HTML input elements present on the page. The rationale behind the second method was that, even if a page legitimately changes between successive visits, the number of visible and invisible input elements should not change. If any one of the two methods provided different results between the first search-engine-referred visit of the site and any of the the RAS-referred ones, the site and its HTML code was recorded and the results were manually verified.

From a total of 1,000 web-sites we discovered that three of them were using the referrer header to provide radically different content. The first, `facebook.com`,

Fig. 3. Facebook's main page when visited through a specific RAS

was serving a very different page when our requests claimed to come from one of the 30 studied RASs. By manually checking the resulting page, we realized that instead of Facebook's usual login-screen, we received a page that alerted us that we had most likely been a victim of a phishing attack, and was inviting us to start the procedure of resetting our password (Figure 3). Interestingly, `facebook.com` reacted this way only when the referrer header was coming from that specific RAS and provided the normal content for the remaining 29 RASs. This could mean that Facebook was reacting to a real phishing attack where the attackers were trying to hide the location of the phishing page by redirecting their victims back to `facebook.com` through that specific RAS after the user credentials had been stolen.

The second web-site was a photo gallery where users can upload pictures and retrieve links that they can later use in various web-sites such as forums and blogs. When the site was visited through one of the 30 RASs, instead of giving us access to its main page, it provided us with a GIF image that stated that the site had blocked access to the linking page. This picture will be provided to any image requests that pass through that RAS. This verifies the behavior that we discovered in our own RAS, where visitors were linking to adult content uploaded to generic image galleries and hiding the linking site through the anonymization of their referrer header. The third site was a general information portal which was consistently providing a 403 HTTP 'forbidden' error when visited through a specific RAS, the same RAS blacklisted by the image gallery site.

The above experiment shows that even though only a small fraction of the tested web-sites reacted visibly to specific referrers, their behavior was always 'negative' when the referrer appeared to be a RAS. This attests towards the argument that RASs are associated more with illegal activity and less with a legitimate user's privacy concerns.

8 Related Work

Although the practice of cleaning the 'Referer' header through a RAS is common knowledge, we are unaware of any research into the operation or usage of these

services, with regard to online privacy and anonymity. The related work that we are aware of, falls in the following 4 categories:

Online privacy and anonymity. Online privacy and anonymity are important for numerous reasons. The Internet was not built to provide anonymous communication mechanisms, which lead to the creation of various projects that provide forms of anonymous networking. The Onion Router (Tor) [25] project and the Invisible Internet Project (I2P) [16] are the most famous of these networks.

Experiments. It is easy to state that a security problem is real and dangerous. Providing evidence backing up this claim is often difficult since it involves covertly tracking the behavior of attackers and victims in an ethical way.

In our experiments in Section 5, we used enticing links to lure visitors to our own *fileleaks.co.cc* in a honeypot-like way. Honeypots [22] have been traditionally used to study attacking techniques and post-exploitation trends. Yuil et al. [29] introduce Honeyfiles as an intrusion detection tool to identify attackers. Honeyfiles are bait files that are stored on, and monitored by, a server. These files are intended to be opened by attackers and when they do so, the server emits an alert. Similarly, Bowen et al. [7] use files with 'stealthy beacons' to identify an insider thread. We have used these techniques in the past to detect whether attackers are violating the assumed privacy in file-hosting services [20].

Referrer abuse. Referrer headers were designed to identify which URL a visitor is coming from. This information is of great benefit to content-providers because it can provide some insight in the browsing habits of visitors, as discussed in Section 2.2. The referrer data however, is optional and can be spoofed (e.g. RefSpoof [12]), prompting the inception of *referrer spam* [27]: sending the URL for a third-party web-site in the referrer header so it will show up in the logs of a visited web-site.

Because of their use to track visitor movements, everyone naturally expects referrer headers to contain URLs. This expectation can result in the development of a web-application which displays the contents of the referrer header, without sufficient sanitization. In such an application, the referrer header can be abused to carry an XSS attack as documented in [14].

Many Cross-site Request Forgery (CSRF) countermeasures depend on the referrer header to determine whether a request was sent from a trusted location. The authors of some of these CSRF countermeasures, aware of visitors that disable the transmission of referrer information, will implement lenient referrer validation [5], which will allow requests without referrer header in order not to break the web-application that is being protected. This deliberate loophole allows an attacker to launch a CSRF attack by relaying a request from an untrusted location through a RAS, which will remove the referrer information. Because the request in this attack has no referrer information, it is allowed by the CSRF countermeasure and the attack can succeed.

Solutions dealing with referrers. There are only two parties that can benefit from non-disclosure of referrer information: the visiting browser and the author of the web-site on which a link is hosted. The referrer-leakage problem can thus be solved by either party.

Referrer-Anonymizing Services attempt to solve privacy and anonymity issues that arise because a visitor's browser is leaking information through the referrer header by design. The author of a web-site linking to an external web-page used to not have any other means to prevent the referrer header from exposing their web-site. One way the author of a web-site could prevent the referrer header from exposing their web-site, was to host their web-site using HTTPS. The HTTP protocol specification [24] advises that referrer information should not be sent when navigating from an HTTPS web-site to an HTTP site. However, browsers are free to preserve the referrer if the destination is also an HTTPS web-site, even if the destination site is situated on a different domain.

Recognizing the need for a better solution, the WHATWG has included the 'noreferrer' link type [26] in the HTML5 specification. By annotating certain HTML elements with this link type, a web-page author will prevent referrer information from leaking when clicking the annotated link. RASs protect the web-site's anonymity as much as they protect a visitor's privacy. Therefore it makes sense for an Internet user to disable referrer information to safeguard that privacy at the source. Many modern web-browsers provide means to disable referrer header transmission [18,6,21]. For other browsers, the referrer can be filtered out using a client-side proxy like e.g. Privoxy [10]. Due to the privacy problems associated with the referrer header, the 'Origin' header [4] has been proposed because it only leaks the origin (scheme, hostname and port number) of a URL to a remote web-site instead of the full URL.

9 Conclusion

In this paper, we explored the ecosystem of Referrer-Anonymizing Services and classified their functionality, their user-base and their abuse. We showed that in several cases, RASs were taking advantage of their position and leaked private user information to advertising companies. Conversely, we discovered that users were occasionaly using RASs to hide illegal or unethical activity and we revealed that some popular Internet sites do not respond well to RAS-relayed traffic. Overall we showed that, while protecting a user's privacy through the anonymization of the referrer header is desirable, not all RASs are equally noble and thus care should be taken when choosing one. At the same time, browser developers have the responsibility to facilitate a complete migration away from such services through the support of privacy-preserving HTML5 tags.

Acknowledgments. This research is partially funded by the IBBT, the Research Fund K.U.Leuven, the B-CCENTRE, the EU-funded FP7 project NESSoS and the IWT project SPION.

References

1. Panopticlick, http://panopticlick.eff.org/
2. Aggrawal, G., Bursztein, E., Jackson, C., Boneh, D.: An analysis of private browsing modes in modern browsers. In: Proc. of 19th Usenix Security Symposium (2010)
3. Readability — Arc90 Lab, http://lab.arc90.com/2009/03/02/readability/
4. Barth, A.: The Web Origin Concept, http://tools.ietf.org/html/draft-abarth-origin-09
5. Barth, A., Jackson, C., Mitchell, J.C.: Robust defenses for cross-site request forgery. To appear at the 15th ACM Conference on Computer and Communications Security, CCS 2008 (2008)
6. Beverloo, P.: List of Chromium Command Line Switches, –no-referrers, http://peter.sh/experiments/chromium-command-line-switches/#no-referrers
7. Bowen, B.M., Hershkop, S., Keromytis, A.D., Stolfo, S.J.: Baiting Inside Attackers Using Decoy Documents. In: Chen, Y., Dimitriou, T.D., Zhou, J. (eds.) SecureComm 2009. LNICST, vol. 19, pp. 51–70. Springer, Heidelberg (2009), http://www.springerlink.com/index/H4833U76H771L873.pdf
8. Chu, Z., Wang, H.: An investigation of hotlinking and its countermeasures. Computer Communications 34, 577–590 (2011)
9. Clayton, R.C., Murdoch, S.J., Watson, R.N.M.: Ignoring the Great Firewall of China. In: Danezis, G., Golle, P. (eds.) PET 2006. LNCS, vol. 4258, pp. 20–35. Springer, Heidelberg (2006)
10. P. Developers. Privoxy, http://www.privoxy.org.
11. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium (2004)
12. ebricca. refspoof Firefox extension, https://addons.mozilla.org/en-US/firefox/addon/refspoof/
13. Eckersley, P.: How Unique Is Your Web Browser? In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 1–18. Springer, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-14527-8_1
14. Recent referer xss vulnerabilities, http://evuln.com/xss/referer.html
15. Fioravanti, M.: Client fingerprinting via analysis of browser scripting environment. Technical report (2010)
16. I2P Anonymous Network, http://www.i2p2.de/
17. ietf-http-wg mailinglist. Re: Referer (sic) from Phillip M. Hallam-Baker (March 09, 1995), http://lists.w3.org/Archives/Public/ietf-http-wg-old/1995JanApr/0109.html
18. MozillaZine. Network.http.sendRefererHeader - MozillaZine Knowledge Base, http://kb.mozillazine.org/Network.http.sendRefererHeader
19. Murphey, L.: Secure session management: Preventing security voids in web applications (2005)
20. Nikiforakis, N., Balduzzi, M., Van Acker, S., Joosen, W., Balzarotti, D.: Exposing the lack of privacy in file hosting services. In: Proceedings of the 4th USENIX Conference on Large-Scale Exploits and Emergent Threats, LEET 2011. USENIX Association, Berkeley (2011)
21. Opera. Disabling referrer logging - Opera Knowledge Base, http://www.opera.com/support/kb/view/93/
22. Provos, N.: A virtual honeypot framework. In: Proceedings of the 13th Conference on USENIX Security Symposium, SSYM 2004, vol. 13, pp. 1–1. USENIX Association, Berkeley (2004)

23. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. ACM Trans. Inf. Syst. Secur. 1, 66–92 (1998)
24. RFC 2616 - Hypertext Transfer Protocol
25. Tor Project: Anonymity Online, http://www.torproject.org
26. WHATWG. HTML - Living standard, http://www.whatwg.org/specs/web-apps/current-work/multipage/links.html#link-type-noreferrer
27. Wikipedia. Referrer spam, http://en.wikipedia.org/wiki/Referrer_spam
28. Wondracek, G., Holz, T., Platzer, C., Kirda, E., Kruegel, C.: Is the internet for porn? an insight into the online adult industry. In: Proceedings of the Ninth Workshop on the Economics of Information Security, WEIS (2010)
29. Yuill, J., Zappe, M., Denning, D., Feer, F.: Honeyfiles: deceptive files for intrusion detection. In: Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, pp. 116–122 (June 2004)
30. Zeller, W., Felten, E.W.: Cross-site request forgeries: Exploitation and prevention. Technical report (2008)
31. Zhou, Y., Evans, D.: Why Aren't HTTP-only Cookies More Widely Deployed? In: Proceedings of 4th Web 2.0 Security and Privacy Workshop (W2SP 2010) (2010)

A Appendix

Table 2. Alexa Ranking of 30 tested RASs – gray color denotes RASs showing ads

URI	Ranking	URI	Ranking	URI	Ranking
anonym.to	661	nolink.in	152,160	anonym2.com	846,517
hidemyass.com	904	cloakedlink.com	187,162	sock.im	933,195
referer.us	7,662	th3-0utl4ws.com	257,867	hidelinks.net	1,170,906
anonymizeit.com	10,212	savanttools.com	305,880	anon.projectarchive.net	2,591,907
lolinez.com	35,526	a.myurl.in	311,033	1-url.net	4,032,638
nullrefer.com	37,227	privatelink.de	338,512	crypo.net	5,009,009
linkblur.com	62,993	carcabot.com	347,260	anonym.ehmig.net	5,510,217
refblock.com	63,834	linkanonymizer.net	433,913	hidehelp.com	9,470,830
dereferer.ws	104,803	spoofurl.com	645,526	devgizmo.com	No Data
anonymous-link.net	118,261	refhide.com	679,101	theybetrollin.com	No Data

Risk Communication Design: Video vs. Text

Vaibhav Garg, L. Jean Camp, Katherine Connelly, and Lesa Lorenzen-Huber

Indiana University
{gargv,ljcamp,connelly,lehuber}@indiana.edu

Abstract. There are significant differences between older and younger adults in terms of risk perception and risk behaviors offline. The previously unexplored existence of this dissimilitude online is the motivation for our work. What are the risk perceptions of older adults? How are these correlated with the classic dimensions of risk perception offline? Can we leverage episodic memory, particularly relevant for older adults, to increase the efficacy of risk communication? We conduct a survey based experiment with two groups: video (n=136) and text (113). We find that leveraging episodic memory using video risk communication can improve the ability of elders to avoid phishing attacks and downloading malware. The applicability of the dimensions of risk were different based not only the risk but also the mode of risk communication.

Keywords: Risk, Privacy, Video, Mental Models, Phishing, Malware.

1 Introduction

Emerging digital technologies are typically neither designed by or for older adults, increasing older adults' susceptibility to privacy violations. Older adults tend to be less experienced with technology than younger adults. With certain privacy risks, such as phishing, there is a tangible financial cost to the victims, their families and service providers. Older adults are the fastest growing demographic in United States [13] and they own a disproportional amount of financial assets [10]. They are also more susceptible to financial fraud offline [4], a vulnerability that may transfer online as well. Along with the rest of the population, older adults are being encouraged to use digital technologies to conduct financial transactions - technologies with which they are often unfamiliar. For example, from 2011, the United States Internal Revenue Services (IRS) will no longer mail tax forms to every household, encouraging online filing instead. For those who wish to file their taxes with paper forms, the forms would be available at local post offices. However older adults often have less mobility than young adults, and may not find the post office accessible. In addition, tax returns filed online are processed faster, further encouraging taxpayers to do so. However, online forms are unfamiliar to many, especially older adults. Thus, when an older adult receives a phishing email claiming to be from the IRS that requests them to click on a link to resolve discrepancies with their tax filing, they are likely to believe there has been a legitimate problem and will follow the link. Much has been said

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 279–298, 2012.

about younger adults, their use of social media, and their disregard for privacy. But older adults may (or may not) be very different in their privacy preferences [31] and in fact may be more likely to engage in risky behavior [51].

There have been two approaches to inform privacy decisions in end users: education [42] and risk communication [2]. Education is a long-term effort that targets risk comprehension and therefore attitudes. Older adults, however, have limited cognitive plasticity, affecting their ability and desire to understand the mechanics of privacy risks such as phishing. Since the goal is to inform behaviors, risk communication might be more pertinent. Thus, we examine the construction of perceived risk online for older adults. Research in risk communication for privacy risks is typically conducted with college undergraduates. Designing for older adults may have additional constraints [23] that arise from limitations on information retention and retrieval [46].

We compare textual and video risk communication. Both media types are built using physical mental models, based on previous research [6,25]. We begin the description of our work by placing it in the historical context of risk perception, and classic developments in risk communication in Section 2. Building on that context, we detail our methodology in Section 3. We present the results in Section 4. After discussing the results in Section 5, we conclude in Section 6.

2 Background and Related Work

The post World War II school of thought, regarding privacy was that of confidentiality, engendered from the privacy needs of the Cold War [39]. Thus, privacy was treated as an extension of security and privacy solutions were mostly technical, e.g. encryption. Information sharing can, however, be beneficial when it, for example, improves market inefficiencies [40]. That individuals share information does not imply a lack of concern for privacy. It merely emphasizes the limitations of privacy as confidentiality [28].

Privacy then becomes not just the ability to hide information, but also the ability to **control** information sharing. To that end there have been both technical solutions like Tor, as well as policy solutions like Do Not Track. Unfortunately, privacy solutions are often limited by their usability [50]. For example, deploying a Tor client requires a certain level of technical competence [18], which may hinder adoption [20]. This might be particularly critical for older adults who are less technically versed. At the same time, usability evaluations are usually done with younger adults, typically college undergraduates.

Previous risk studies have focused on the usability of privacy-enhancing technologies [14,45,19]. Here we focus of purposeful and accidental information-sharing, using phishing and malware respectively. Specifically we examine downloaded malware instead of drive-by malware because downloading requires user action. The goal is not to educate users so that they understand the underlying technical risks but rather to give them the skill set necessary to avoid the risk.

Privacy solutions, even when usable, are frequently not adopted [3]. Even privacy protections that require no technological expertise are not adopted. A

recent example of this is the Do Not Track initiative. In terms of usability, all it requires is a click of a button to indicate that the user does not desire to be tracked. Despite the simplicity of it's design, figures indicate that less than 2% of Firefox users opt in [8]. At the same time, survey reports indicate that users are uncomfortable being tracked online. This disconnect between attitudes and behaviors, termed "the privacy paradox" [37], is also encountered offline and should not be surprising [43]. One explanation is our failure to contextualize privacy solutions appropriately [36]. The risk of information sharing is most salient during survey-based elicitation of attitudes. However, in context it would be the benefits of information sharing that are most easily available.

Bonneau et al. argue that it is rational for online service providers to hide the "privacy control interface and privacy policy to maximize sign-up numbers and encourage data sharing from the pragmatic majority of users" [11]. Thus, it is not a lack of user concern for privacy, rather a lack of clear signals that makes privacy solutions impotent [15]. The paradigm of rationality fails systematically and predictably for both security [41,26] and privacy decisions [3]. Our research seeks to address the deeper problem that privacy solutions are limited due to the designers' assumptions about a rational end-user by providing guidance on the systematic, potentially predictable, patterns of irrationality. How does rationality fail? And how can such failures be predicted by building on the foundations built by researchers focusing on offline risks? Ideally, privacy risks would be evaluated as the product of probability of occurrence and the magnitude of implications [9]. However, privacy risk decisions are often acted upon by external factors, e.g. control [12]. Brandimarte et al. noted that participants were more willing to share information when they had control over the publication of information, even though they had no control over access to that information or third party use. Thus, perceived control alleviates aversion to risk. Privacy decisions are then subjective, with end-users balancing perceived risk with perceived benefit. Fischhoff et al. [22] developed the canonical nine-dimensional model of perceived risk. This model has been used extensively to study perceived risk offline for a diverse set of risks, e.g. health risks [32] and environmental risks [24]. It has been used online to examine insider threats [21] as well as security risks [25]. The nine dimensions consist of voluntary, immediacy, knowledge to exposed, knowledge to expert, control, newness, common-dread, chronic-catastrophic, and severity. These have the potential to impact privacy decisions as well. Information sharing on social networks is done voluntarily, implying control and alleviating perceived risk. However, behavioral advertising raises privacy concerns as information is being involuntarily revealed [38]. The benefits of information sharing are immediate, while the consequences of privacy violations may appear delayed [1]. Thus, regrets about privacy risks might appear later [49]. Privacy risks are often not known to the end-user. Even when they are known, end-users may put too much value in expert opinion. Increased trust in expert systems can increase risky behavior. For example, drivers with ABS-enabled cars drive closer to other vehicles [33]. Similarly, perceptions of control can increase risky behavior [12]. Online privacy risks are newer than

those offline. For example, information aggregation reveals information about end-users that would not be as easily accessible by individual bits of data. Humans understand averages better than aggregates [48]. When an end-user shares a single piece of information on Facebook, Google Plus, and Linked In, they evaluate it as one piece of information being shared on average, rather than as three pieces of information being shared. It would be even more difficult for end-users to account for a fourth piece of information being revealed as a result of the first three pieces being combined.

Common risks are dreaded less than rarely encountered risks. For example, terrorism appears more threatening than E. coli, despite the far higher death rates for the latter. Likewise, privacy risks are commonly encountered and thus may appear less threatening. Perceived risk is also directly proportional to the number of people at risk. Severity of risk consequences is also directly correlated with perceptions. Thus, decisions concerning risk are not strictly rational.

Risk decisions can be improved by a soft paternalistic nudge [2]. Nudging must impinge both the intuitive as well as the rational decision processes [44]. The effectiveness of intuitive systems is based on the decision system's ability to recognize risk [34]. Individuals respond irrationally to online risks in a manner analogous to offline risks. Ability to identify risk online is driven by the mental models that are used for representation [16]. Accessible mental models would make it easier for end-users to associate online risk with offline risks that they would be more familiar. When encountering an unfamiliar risk, individuals will be guided by their perceptions, not by the calculus of risk [17].

Previous work argued that security experts use five mental models: physical, criminal, warfare, medical, economic. End-users find physical mental models to be most accessible [6,25]. Thus, grounding risk communication designs in physical mental models would make user intuitions more informed.

Simultaneously, the effectiveness of a rule-based system is driven by its ability to extract the relevant information. The design of risk communication must then address information coding, storage, and retrieval. This is especially relevant for older adults who may have less cognitive plasticity compared to the younger adults [52]. The impact of aging on memory is severe [5], with retrieval becoming more difficult as older adults increasingly experience irrelevant intrusions (i.e. interruptions, unrelated information, or background noise) [29]. One approach to address bounds on cognition is by the use of richer media [30]. Videos, for example, are stored in the episodic memory [47] rather than semantic [35] as coding is based in context rather than content [46]. Previous research indicates that richer media can facilitate cognition [7], targeting episodic memory [47] rather than semantic [35]. Simultaneously, the use of appropriate mental models can make risk information more accessible [16]. Traditional online risk communication, however, uses text. Thus, there is a need explore video-based risk communication designs that privacy risks for older adults.

In this paper we present the design of narrative driven risk communication videos grounded in physical mental models. We present the evaluation of these videos by comparing them to traditional text based risk communication. We

have designed these videos for older adults, thus our participant pool consists of adults older than 65 years of age. We examine the differences in perceived risk as experienced through different risk communication media. In the next section we present our methodology.

3 Methodology

We used an expressed preferences methodology to evaluate the difference between text and video based risk communication for phishing and malware risks. We conducted a survey-based experiment with two groups. Group 1, referred to as the video group, saw the risk information as a physical mental models based narrative presented in a video. Group 2, referred to as the text group read the risk information in a text form. Participants were randomly assigned to each group. Participant risk was minimized by following the ethics guidelines, for expedited studies, from Internal Review Board (IRB).

Participant recruitment and survey deployment was conducted by Knowledge Networks. Knowledge Network provides access to a representative sample of the U.S. population. They cover both online and offline populations, listed and unlisted phone numbers, households with or without phones as well as households with only cellular phones. Sampling frame is constructed using address based sampling (ABS) and random digit dialing (RDD). This is similar to methodology used by CDC for national immunization surveys. Participants are chosen at random from the sampling frame. Eligibility criteria is applied, i.e. participants need to be over 65. Over-sampling and under-sampling concerns are addressed by the use of post stratification weights. Despite a rigorous methodology there are always limitations. For example, even with ABS and RDD combined the sampling frame is constructed from 98% of the US population. However, the results can be considered to be reasonably generalizable.

Participants began by providing the consent to participate according to IRB guidelines. Participants provided demographic information: age, frequency of Internet use, frequency of cellphone use, as well as whether the participants lived alone or with other people. Participants were then shown text based or video based risk communication respectively. For each group half the participants rated their attitude towards the risk, i.e. which did they consider higher, the risk of responding or that of not responding. For example, for phishing emails the risk of responding would imply clicking on a link and providing personal information. The other half predicted their behavior in response to being exposed to the risk, i.e. are they more likely to respond (or not)?

Risk Communication Design: Participants were communicated the risk of phishing and malware. These risks were chosen due to their implications for identity theft, which could result in personal financial loss. This loss is more detrimental to older adults, as their ability to replace lost income is limited. The risk information was presented in text form to a subset of the participants, text group, and in video form to the rest, video group. Here we will discuss the design of both the text and video based risk communication. Due to space limitation

we will discuss only phishing. A detailed discussion of the design process can be found in [27].

In order to minimize psychological risks, such as anxiety, we created a false persona of an older adult: Mr. Cullen. Participants were told that Mr. Cullen is a retired older adult who had just received an email. Then the participants were presented with a canonical phishing email:

Dear Mr. Cullen,
We are from the IRS and we are writing regarding your retirement funds and bank accounts. It has come to our attention that there might be some discrepancies with respect to some of the transactions made from your accounts. We are conducting an investigation into this. We would like to get some information from you. Please click on the link at the bottom of the e-mail and answer a few questions. Please make sure that you have your bank account number, password, and your social security number as you may be asked about them.
www.IRS.com
Regards IRS

The participants were then informed that Mr. Cullen clicked on the associated link and provided the relevant information.

To present this information in the video form, we first identified the key characteristics of phishing from the above email. First, phishing emails appear legitimate. Secondly, they try to scare the recipient. Finally, they ask for the email recipient's financial information. Based on these characteristics we developed a mental models based narrative grounded in a physical analogy. This narrative was later developed into a video. For consistency, the victim in the video was also an older adult. The attacker had to be a legitimate financial entity. Again for consistency we chose an IRS agent, or rather an attacker pretending to be one. Thus, the agent fashioned credentials that appeared authentic. The agent contacted the older adult at home and informed him that he was under investigation due to financial discrepancies. The agent then asked for the older adult's financial information. The older adult, wanting to comply with the investigation, provided the information and, thus, was phished.

Risk Assessment: Text based risk communication would impinge perceived risk differently from risk videos[1]. Half the participants identified whether they felt the risk of responding to the risk was higher than *not responding*. The remaining participants predicted their behavior, i.e. if they would respond to the risk or *not respond*. Participants rated the perceived benefit of responding to the risk on a seven point Likert scale (1=Not beneficial; 7=Highly beneficial). Participants also rated the perceived risk of responding on seven point Likert scale (1=Not risky; 7=Highly risky).

Risk of responding was also evaluated on a nine dimensions of perceived risk identified by Fischhoff et al. [22]. Since its inception in 1978, this model has been used extensively to study risks in a diversity of domains including health

[1] Phishing Video: http://www.youtube.com/watch?v=4ZQ9pFTCdy4
Malware Video: http://www.youtube.com/watch?v=6zHJoZqrCB0

and environmental risks. Participants were asked to rate the perceived risk of responding to phishing and malware risks on each of the nine dimensions. The rating was on a five point like scale and the dimensions were defined as:

1. Voluntary: To what extent does Mr. Cullen have a choice in being exposed to this risk? (1=Voluntary; 5=Involuntary)
2. Immediacy: Is the risk from the threat immediate or does it occur at a later time? (1=Immediate; 5=Delayed)
3. Knowledge to the exposed: How much would a person like Mr. Cullen know about the implications of this risk? (1=Knows a lot; 5=Knows nothing)
4. Knowledge to the expert: How much would an expert know about the implications of this risk? (1=Knows a lot; 5=Knows nothing)
5. Control: To what extent can you control (or mitigate) the risk? (1=Uncontrollable; 5=Controllable)
6. Newness: Is this a new risk resulting from new technologies or is it a new version of an old risk? (1=Old; 5=New)
7. Common-Dread: Is this risk commonplace or rarely encountered? (1=Common; 5=Rare)
8. Chronic-catastrophic: Does this risk affect only Mr. Cullen or does it affect many people? (1=(Mr. Cullen) Individual; 5=(Many People) Global)
9. Severity: In the worst possible outcome, how severe would the consequences be? (1=Not Severe; 5=Severe)

4 Results

There were a total of 249 participants. There were 113 participants in the text group and 136 participants in the video group. 49 of the participants lived alone. 26 of those participated in the text group, while 23 participated in the video group. 199 participants lived with other people. 86 of those participated in the text group, while 113 participated in the video group. 122 of the participants were men, 56 of whom were in the text group and 66 in the video group. There were 127 women participants, of which 57 read the text and 70 saw the video.

The mean time for completing the text survey was 219.3186 minutes, while that for completing the video survey was 303.6103 minutes. There was no statistically significant difference between the mean time taken by each group, $p < .05$. The high values from mean time to complete are due to a handful of participants taking over 6000 minutes to complete the survey, i.e. several days. Thus, we also computed the medians.

The median time of completion for the text survey was 21 minutes and that for the video survey was 30.5 mins. Medians were compared by using Mann-Whitney or two-sample Wilcoxon test. This non-parametric test makes two assumptions: (1) the distribution is continuous, and (2) that the shape of the distribution of both samples is similar. Since time is a continuous variable assumption one is justified. For the second assumption, both distributions were heavily right skewed. The lower bound was 0 while the upper bound was less than 6500 minutes. On

conducting a one sided test, the time to complete was significantly different between text and video; w=4851.5, p-value=2.762e-07. Participants in the video group took more time to complete the survey than those in the text group.

Risk Attitudes: The risk of responding to phishing emails is often underestimated. Phishing emails create anxiety in recipients, thereby making the risks of not responding salient, while appearing to alleviate the risks of responding; e.g. 'If you do not contact us on the following, your account may be closed'. Thus, participants were asked if they considered the risk of responding to be greater to than the risk of not responding to phishing. 62 participants answered this question for text while 65 answered this for video. 38 participants in the text group indicated responding to the email was more risky, while 48 participants indicated responding to the agent with the information was more risky. Test of proportions did not indicate a statistically significant difference; p-value= 0.1439.

Similarly, participants were asked if they considered the risk of responding to be greater to than the risk of not responding to malware. 62 participants answered this question for text while 64 answered this for video. 53 participants in the text group indicated responding was more risky, while 62 participants in the video group indicated that responding was risker. One sided test of proportions was *statistically significant*, p=0.02565.

Predicted Behavior: Even when the risk of responding is perceived to be greater, participants may choose to respond, as the implications might, for example, appear more controllable [12]. Thus, we asked participants to indicate their likely action in response to the phishing attack if they had indicated that responding was riskier than not responding (n=51 for text and n=72 for video). All text participants said that they would not respond, while 71 video participants said they would not respond. Test of proportions did not indicate a statistically significant difference; p-value= 1.

We also asked participants to indicate their likely action in response to the malware scenario if they had indicated that responding was riskier than not (n=52 for text and n=72 for video). 49 text participants said they would not respond, while 69 video participants said they would not respond. Test of proportions was not statistically significant; p-value=0.2858.

Perceived Benefit: Participants rated the benefit of responding to phishing risk on a seven point Likert scale (1=not beneficial at all; 7=highly beneficial). The mean benefit for the text group was 1.372881, while that for the video group was 1.828125. An independent two sample T-test did not indicate statistically significant difference. Participants rated the benefit of responding to malware risk on a seven point Likert scale (1=not beneficial at all; 7=highly beneficial). The mean benefit for the text group was 1.271186, while that for the video group was 1.281250. An independent two sample T-test did not indicate statistically significant difference.

Perceived Risk: Participants rated the risk of responding to phishing on a seven point Likert scale (1=not risky; 7=highly risky). The mean risk for the text group was 6.814815, while that for the video group was 6.830986. An independent two

sample T-test did not indicate statistically significant difference. Participants rated the risk of responding to malware on a seven point Likert scale (1=not risky; 7=highly risky). The mean risk for the text group was 6.018868, while that for the video group was 6.555556. An independent two sample T-test indicates *statistically significant* difference, p=0.002392.

Table 1. Mean Risk Ratings of Fischhoff's nine dimensions

	Phishing			Malware		
Text vs. Video	Text	Video	p-value	Text	Video	p-value
Voluntary	2.389381	1.992593	0.02683	1.651786	1.955882	0.04183
Immediacy	1.876106	2.318519	0.005024	2.063063	2.977941	8.346e-07
Exposed	3.723214	3.830882	> 0.05	3.732143	4.082707	0.004020
Expert	1.153153	1.110294	> 0.05	1.099099	1.186567	> 0.05
Control	1.783784	2.514706	7.385e-05	3.027027	2.851852	> 0.05
Newness	2.765766	2.463235	> 0.05	3.442478	3.432836	> 0.05
Common Dread	1.781818	1.733333	> 0.05	1.928571	2.373134	0.0007834
Chronic-Catastrophic	3.919643	4.066176	> 0.05	3.814159	4.073529	0.04495
Severity	4.855856	4.867647	> 0.05	4.548673	4.850746	0.0001627

4.1 Nine Dimensional Model

Participants rated the risk of responding to phishing and malware on Fischhoff's nine dimensions of perceived risk. Table 1 reports the mean rating given for each dimension as well as the p-values of one sided T-tests between text and video. Participants rated the voluntary nature of risk (1=Voluntary; 5=Involuntary). Text and video were statistically different for both phishing as well as malware. While for phishing the risk of responding was more voluntary for the video group, for malware risk of responding was more voluntary for text. Participants rated the immediacy of the impact of risk (1=Immediate; 5=Delayed). The consequences of responding were more delayed for the video than for text, in both phishing as well as malware. Participants rated the knowledge a typical victim would have regarding the risk (1=Knows a lot; 5=Knows nothing). There was no statistical difference between text and video for phishing. However, knowledge to the exposed was higher in the video group for malware. The relationship of perceived risk with voluntary, immediacy, and knowledge to exposed is shown in figure 1.

Participants rated the knowledge an expert would have regarding the implications of phishing or malware (1=Knows a lot; 5=Knows nothing). There was no statistically significant difference between text and video for either phishing or malware. Participants rated the extent to which they can control the risk consequences (1=Uncontrollable; 5=Controllable). Phishing risk was more controllable for video than for text, while the difference was not statistically significant for malware. Participants rated the newness of phishing or malware (1=Old; 5=New). There was no statistically significant difference between text

and video for either phishing or malware. The relationship of perceived risk with knowledge to expert, control, and newness is shown in figure 2.

Participants were asked to rated how common they considered the risk to be (1=Common; 5=Rare). Participants also rated whether the risk impact just the victim or if the risk was global (1=Individual; 5=Global). Finally, participants rated the severity of the consequences of phishing (1=Not Severe; 5=Severe). The difference between text and video was not statistically significant for phishing on these three dimensions. However, for malware the difference was statistically significant. Video risk was more rare, catastrophic, and severe, compared to text. The relationship of perceived risk with common-dread, chronic-catastrophic, and severity is shown in figure 2.

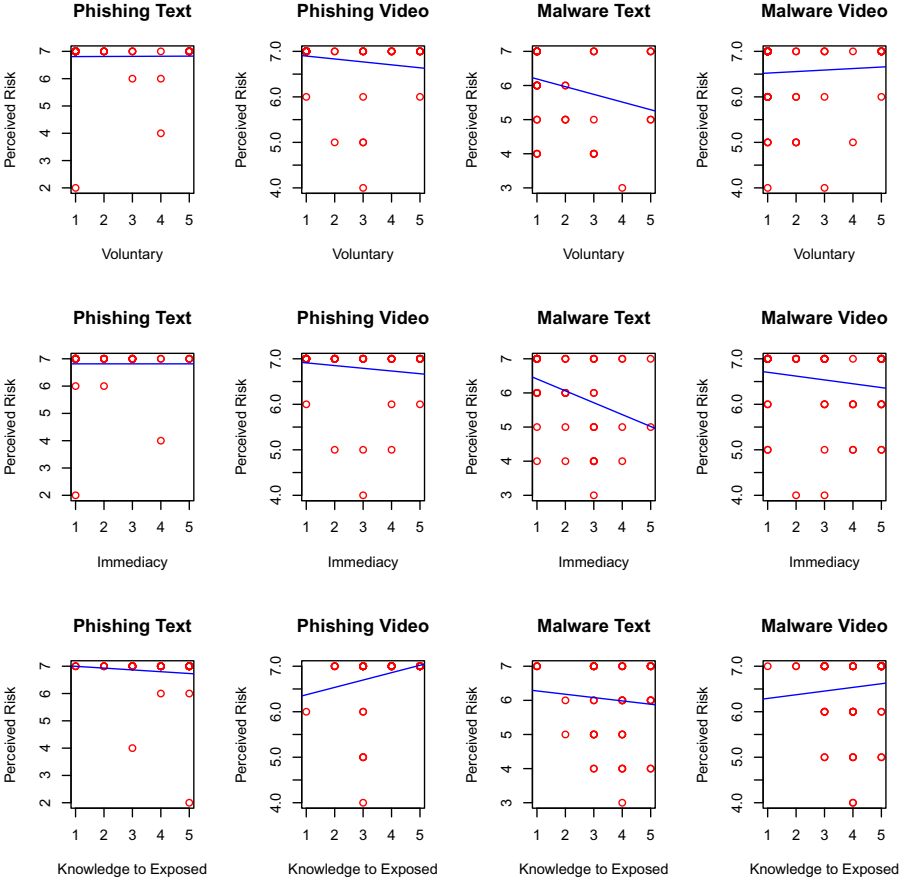


Fig. 1. Perceived Risk vs. Voluntary, Immediacy, and Knowledge to Exposed

4.2 Regression Analysis: Perceived Risk vs. Nine Dimensions Model

Figures 1-3, note that the relationship between perceived risk and the nine dimensions is linear. Thus, linear regression analysis is applicable. We then identified the best fit model, i.e. the subset of the nine dimensions that best explain variance in perceived risk. Thus, we isolated the relevant dimensions evaluate their ability to explain perceived risk.

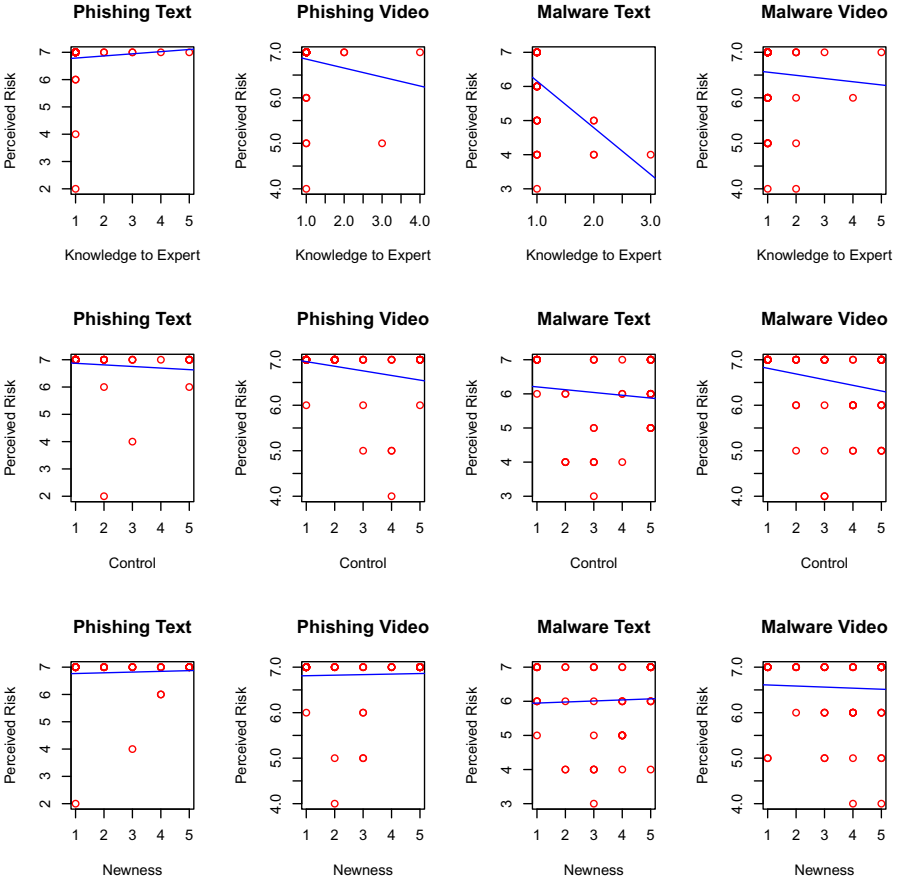


Fig. 2. Perceived Risk vs. Knowledge to Expert, Control, and Newness

We considered additional variables: (1) time taken to complete the survey, (2) whether the participant lived alone or not, (3) frequency of Internet use, and (4) frequency of cellphone use. These will be referred to as confounding variables. We measure the models explanatory power by examining *adjusted* R-square values, rather than raw R-square values. *Adjusted* R-square: 1) accounts for collinearity of independent variables, and 2) adjusts for extra variables. We first conducted the analysis for phishing text. The dependent variable was perceived risk, while

the independent variables were the nine dimensions. The R-square value was negative with the complete nine dimensional model. Reducing the dimensions improved the model's explanatory power. Best fit for the model was given by severity, common dread, knowledge to the exposed, knowledge to expert, and chronic-catastrophic; R square=0.03208, p=0.27776. Adding the confounding variables further improved the model's explanatory power. The best fit for the model was given by immediacy, common dread, knowledge to expert, frequency of internet use, and cellphone use; R-square=0.1921, p=0.01468.

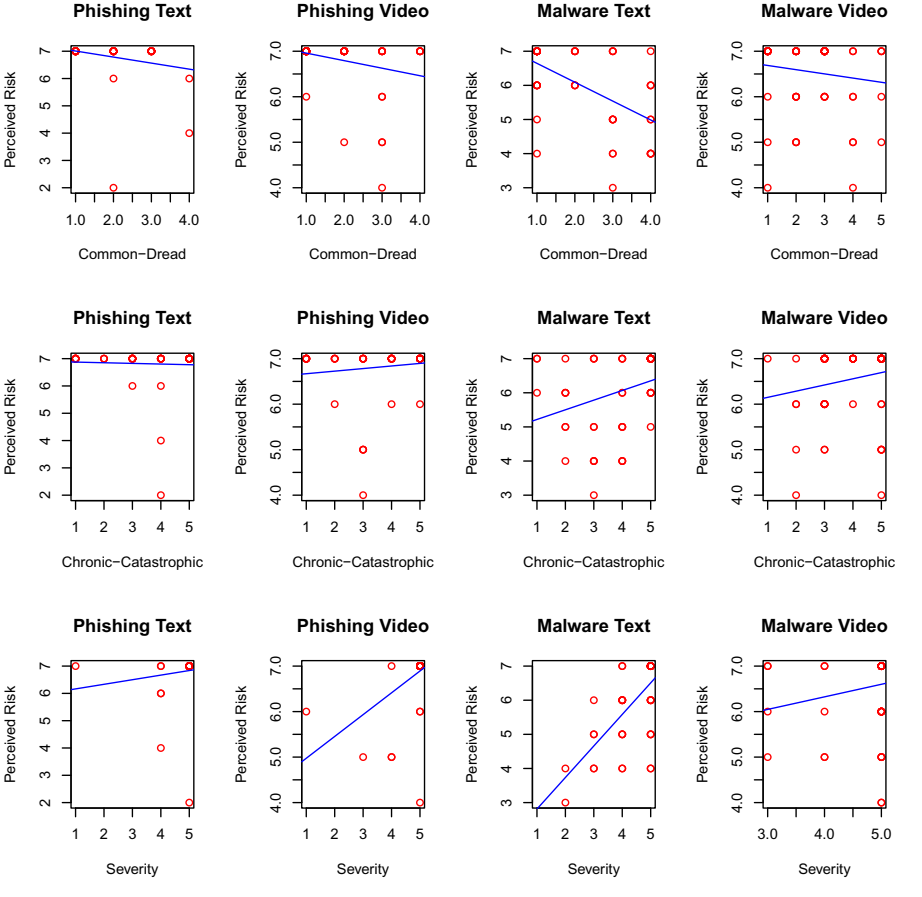


Fig. 3. Perceived Risk vs. Common-Dread, Chronic-Catastrophic, and Severity

Next we conducted the analysis for phishing video. With just the nine dimensions as the independent variables the model has good explanatory power; R square= 0.2494, p= 0.001391. Severity was statistically significant. Best fit for the model was given by voluntary, knowledge to the exposed, control, severity;

R square= 0.2849, p=2.633e-05. Knowledge to the exposed and severity were statistically significant. Adding the confounding variables did not increase the explanatory power of the model.

For malware, again, we first analyzed text. With perceived risk as the dependent variable and the nine dimensions as independent variable the model had significant explanatory power; R-square =0.6082, p=7.981e-07. Knowledge to expert, severity, and chronic-catastrophic were statistically significant. Best fit was given by voluntary, severe, newness, common-dread, knowledge to exposed, knowledge to expert, and chronic-catastrophic; R-square=0.6511 p-value=3.133e-09. Again knowledge to expert, severity, and chronic-catastrophic were statistically significant. Adding the confounding variables did not increase the model's explanatory power.

However, for malware video perceived risk was not explained by the nine dimensional model; R-square value was negative. Reducing the dimensions improved the performance slightly. Best fit was given by control, immediacy and chronic-catastrophic; R-square=0.06659 p=0.05139. Adding the confounding variables and reducing dimensions, the best fit was given by control, knowledge to the exposed, and frequency of Internet use; R-square=0.09577, p=0.02179. Control was the only statistically significant dimension.

4.3 Factor Analysis

We were also interested in the underlying structure of the nine dimensional model, specifically how it different for video vs. text as well as phishing vs. malware. Thus, we are interested in the differences between not just media but also the nature of the risks. To analyze the underlying structure we conducted exploratory factor analysis. To determine the number of factors, we conducted Scree test; figure 4. We consider eigenvalues greater than 1. Thus, we consider four factors for each scenario.

We conducted factor analysis using R's inbuilt factors analysis function factanal with varimax rotation and pairwise deletion of missing values. Tables 2, 3, 4, and 5 show the factor loadings for the different factors, the communalities for the nine dimensions, and the variance explained by the four factors.

5 Discussion

The hypotheses that this research examines are two: 1) videos are more effective risk communication media than text for older adults, and 2) perceived risk can be grounded in Fischhoff's nine dimensional model. To test the first hypothesis we examined the difference in participants' attitude towards responding to risk, predicted behavior, ranking of perceived benefit and perceived risk.

Unlike voluntary, immediacy has the same relationship with perceived risk for text as well as video, for phishing as well as malware. When the risk is immediate

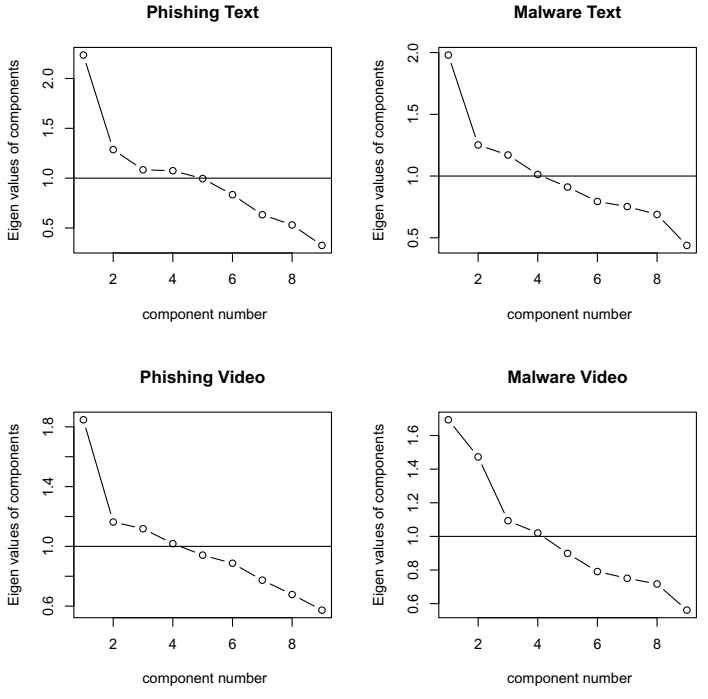


Fig. 4. Scree Test

Table 2. Phishing Text Factor Loadings

	Factor 1	Factor 2	Factor 3	Factor 4	Communality
Voluntary			0.979	0.189	0.995
Immediacy	0.157	0.232	0.112	0.211	0.136
Knowledge to Exposed			0.182		0.040
Knowledge to Expert	0.570				0.340
Control	0.175	0.979			0.995
Newness		0.186	0.106		0.047
Common-Dread	0.123	0.411	-0.222	0.873	0.995
Chronic-Catastrophic				-0.145	0.027
Severity	-0.957	-0.150		-0.226	0.995
Proportion Var	0.146	0.139	0.120	0.103	
Cumulative Var	0.146	0.285	0.405	0.508	

it is perceived to be more risky, when the effects are delayed, less so, figure 1. Thus, a lower value for immediacy informs perceived risk better. Thus, text was better than video at informing perceived risk. For both phishing and malware, participants in the text group perceived the impact of risks to be more immediate than delayed.

Table 3. Phishing Video Factor Loadings

	Factor 1	Factor 2	Factor 3	Factor 4	Communality
Voluntary				0.199	0.043
Immediacy	-0.111	0.233	0.151	0.354	0.215
Knowledge to Exposed			-0.347		0.124
Knowledge to Expert	-0.192	0.148	0.129		0.082
Control	-0.144	0.971			0.965
Newness		0.128			0.025
Common-Dread	-0.195		0.817	0.109	0.721
Chronic-Catastrophic	0.132			-0.629	0.417
Severity	0.991		-0.105		0.995
Proportion Var	0.124	0.116	0.094	0.065	
Cumulative Var	0.124	0.239	0.334	0.399	

Table 4. Malware Text Factor Loadings

	Factor 1	Factor 2	Factor 3	Factor 4	Communality
Voluntary	0.219	-0.319			0.157
Immediacy		-0.355	0.119		0.154
Knowledge to Exposed			-0.160		0.032
Knowledge to Expert	0.977	-0.113	0.138		0.995
Control			0.792		0.629
Newness		0.279	-0.173	0.363	0.246
Common-Dread	0.187	-0.292	0.165	0.581	0.485
Chronic-Catastrophic		0.139		-0.383	0.168
Severity		0.801	0.120	-0.203	0.706
Proportion Var	0.119	0.118	0.085	0.075	
Cumulative Var	0.119	0.237	0.322	0.397	

Table 5. Malware Video Factor Loadings

	Factor 1	Factor 2	Factor 3	Factor 4	Communality
Voluntary				0.278	0.086
Immediacy		0.802			0.649
Knowledge to Exposed			0.390	0.292	0.244
Knowledge to Expert			-0.372		0.142
Control	0.114	0.159	-0.339		0.155
Newness		0.225		0.418	0.232
Common-Dread	0.719	0.247		0.266	0.650
Chronic-Catastrophic	-0.439			0.214	0.245
Severity	-0.254			0.430	0.256
Proportion Var	0.089	0.089	0.071	0.046	
Cumulative Var	0.089	0.178	0.249	0.295	

For knowledge to the exposed, the results were mixed. There was no statistical difference in means values for phishing text and video. However, in general higher knowledge led to higher perceived risk. The importance of this is reinforced in the linear regression model for phishing video, where knowledge to exposed is one of the dimensions in the best fit model, and is also statistically significant. For malware, knowledge to the exposed was rated lower in text than in video. However, in text lower knowledge led to higher perceived risk, while in video the relationship was reversed. Knowledge to exposed was present in the best fit models of both text and video, but was only statistically significant in text. Thus, it is unclear if text or video are better at leveraging this dimension. Knowledge to the expert was similar between text and video, for both phishing and malware. Knowledge to the expert, in general tends to reduce perceived risk. It would then lead to more risk taking behaviors. Knowledge to the expert was in the best fit model for phishing text and malware text. It was not statistically significant for phishing text, however, for malware text it was significant. Thus, text based risk communication, for malware, may lead to more risk taking behaviors based on knowledge to experts.

Control had a consistent relationship with perceived risk. Uncontrollable risks were perceived more risky, figure 2. For phishing, text group perceived phishing risk to be more uncontrollable than video group. Control was in the best fit model for both phishing video and malware video. It was statistically significant for malware video. Thus, video based risk communication would lower perceived risk on the control dimension.

Newness was similar for both text and video, for phishing as well as malware. Newness was only in the best fit model of malware text. However, it was not statistically significant. In general, newer risks were perceived to be more risky. However, there does not seem to be much evidence of newness having a significant impact on perceived risk.

Common-dread had a consistent relationship with perceived risk. Common risks were perceived to be less risky than those rarely encountered. The difference between phishing text and phishing video was not significant. The difference for malware was significant. Text group perceived malware to be more common than video. Common-dread was in the best fit models for text but not for video. Thus, text based risk communication may alleviate perceptions of risk by making them appear common. Chronic-catastrophic did not have a significant impact on phishing text. For phishing video and malware, both text and video, risks that impact more people were perceived to be more risky. The difference between phishing text and video was not statistically significant. The difference for malware was significant. Malware was seen to impact more people for video than for text. Chronic-catastrophic was in the best fit model for both malware text and malware video. It was statistically significant for malware text. Thus, text based risk communication might alleviate perceptions of malware risk.

Severity had a significant impact on perceived risk. More severe risks were perceived as more risky. The difference between phishing text and video was not significant. For malware the difference was significant. Malware risks were

perceived more severe for video than for text. Severity was in the best fit model for malware text, and was statistically significant. Malware appears less severe in text than video. Thus, perceived risk might be lower for text than video.

6 Conclusion and Future Work

Recall the purpose of our experiments was to test two hypotheses. The first is that grounding risk communication in mental models empowers older adults (who are not familiar with information technology) to avoid common threats. We built on previous findings for mental models, using physical mental models. We concluded that video was more effective in making risk salient and perceived as severe; yet the prediction of behaviors was not significantly different between text and videos.

The second hypothesis was that online risk perceptions could be understood using the classic dimensions of offline risks. The perceptions differed based on the presentation of the risk and the risk itself; however, in any case there were multiple significant dimensions. Knowledge to experts was stronger in the case of text; thus text may perversely increase risk-seeking. Perceptions of severity were greater with video than text. The perception of a risk as voluntary for phishing made the risk more salient in the case of text and less so in video. Text made the risk appear more immediate and in the case of malware, less voluntary. Neither risk, reasonably so, was seen as particularly dreadful. Perceptions of severity had by far the greatest impact on perceived risk. Neither knowledge to the exposed nor newness were significant in either case.

The use of mental models proved effective in providing strategies for risk mitigation and illustrating the severity of risk; thus mental models rather than exact technical information are more powerful in decreasing potentially hazardous information-sharing. The use of video to leverage episodic memory proved somewhat mixed; yet on the most significant variable (i.e., severity) video proved more effective. Risk communication is more effective when grounded in the risk itself (e.g., phishing) than in the technical vector in which the risk is embedded. We conclude that targeting risk communication using videos and mental models has the potential to be extremely effective in the (also extremely vulnerable) online population of older adults.

Acknowledgements. We would like to thank Dr. Kelly Caine who assisted us at various stages of the research. We also acknowledge the insights provided by the Stat/Math center at Indiana University. Any mistakes are, however, authors' own responsibility.

This material is based upon work supported, in part, by the National Science Foundation under Grants NSF IIS 1036918 and NSF CNS 0943382. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Acquisti, A.: Privacy in electronic commerce and the economics of immediate gratification. In: Proceedings of the 5th ACM Conference on Electronic Commerce, pp. 21–29. ACM (2004)
2. Acquisti, A.: Nudging privacy: The behavioral economics of personal information. IEEE Security & Privacy 7(6), 82–85 (2009)
3. Acquisti, A., Grossklags, J.: Privacy and rationality in individual decision making. IEEE Security & Privacy 3(1), 26–33 (2005)
4. Anderson, K.: Consumer fraud in the United States: An FTC survey. Federal Trade Commission (2004)
5. Anderson, N., Craik, F.: Memory in the aging brain. The Oxford Handbook of Memory, 411–425 (2000)
6. Asgharpour, F., Liu, D., Camp, L.J.: Mental Models of Security Risks. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 367–377. Springer, Heidelberg (2007)
7. Aslan, A., Bäuml, K.H., Pastötter, B.: No inhibitory deficit in older adults' episodic memory. Psychological Science 18(1), 72 (2007)
8. Bachman, K.: Study: Internet user adoption of dnt hard to predict. Tech. rep., AdWeek (March 2012), http://www.adweek.com/news/technology/study-internet-user-adoption-dnt-hard-predict-139091
9. Bernstein, P.: Against the gods: The remarkable story of risk. John Wiley & Sons Inc. (1998)
10. Bertoni, D.: Identity Theft: Governments Have Acted to Protect Personally Identifiable Information, But Vulnerabilities Remain: Congressional Testimony. DIANE Publishing (2009)
11. Bonneau, J., Preibusch, S.: The privacy jungle: On the market for data protection in social networks. Economics of Information Security and Privacy, 121–167 (2010)
12. Brandimarte, L., Acquisti, A., Loewenstein, G.: Misplaced confidences: Privacy and the control paradox. In: Workshop of Economics and Information Security (WEIS). Harvard University (2010)
13. Breau, C.: Projected population of the united states, by age and sex: 2000 to 2050. Tech. rep., U. S. Census Bureau (2000), http://www.census.gov/population/www/projections/usinterimproj/
14. Brodie, C., Karat, C.M., Karat, J., Feng, J.: Usable security and privacy: a case study of developing privacy management tools. In: Proceedings of the 2005 Symposium on Usable Privacy and Security, SOUPS 2005, pp. 35–43. ACM, New York (2005)
15. Camp, L.J.: Reliable, usable signaling to defeat masquerade attacks. ISJLP 3, 211 (2007)
16. Camp, L.J.: Mental models of privacy and security. IEEE Technology and Society Magazine 28(3), 37–46 (2009)
17. Camp, L., McGrath, C., Genkina, A.: Security and morality: A tale of user deceit. In: Models of Trust for the Web (MTW 2006), Edinburgh, Scotland, vol. 22 (2006)
18. Clark, J., Van Oorschot, P., Adams, C.: Usability of anonymous web browsing: an examination of Tor interfaces and deployability. In: Proceedings of the 3rd Symposium on Usable Privacy and Security, pp. 41–51. ACM (2007)
19. Cranor, L., Garfinkel, S.: Security and usability: Designing secure systems that people can use. O'Reilly Media, Inc. (2005)

20. Dingledine, R., Mathewson, N.: 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)
21. Farahmand, F., Spafford, E.H.: Understanding insiders: An analysis of risk-taking behavior. Information Systems Frontiers, 1–11 (2010)
22. Fischhoff, B., Slovic, P., Lichtenstein, S., Read, S., Combs, B.: How safe is safe enough? a psychometric study of attitudes towards technological risks and benefits. Policy Sciences 9(2), 127–152 (1978)
23. Fisk, A.D., Rogers, W.A., Charness, N., Sharit, J.: Designing for older adults: Principles and creative human factors approaches, vol. 2. CRC (2009)
24. Flynn, J., Slovic, P., Mertz, C.K.: Gender, race, and perception of environmental health risks. Risk Analysis 14(6), 1101–1108 (1994)
25. Garg, V., Camp, L.J.: End user perception of online risk under uncertainty. In: 45th Hawaii International Conference on System Sciences. IEEE (2012)
26. Garg, V., Camp, L.J.: Heuristics and biases: Implications for security and privacy (2012), http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1933957
27. Garg, V., Camp, L.J., Lorenzen-Huber, L.M., Connelly, K.: Risk communication design for older adults. In: ISG*ISARC 2012. International Society for Gerontechnology (in press, 2012)
28. Gürses, S., Berendt, B.: Pets in the surveillance society: A critical review of the potentials and limitations of the privacy as confidentiality paradigm. Data Protection in a Profiled World, 301–321 (2010)
29. Hasher, L., Stoltzfus, E., Zacks, R., Rypma, B.: Age and inhibition. Journal of Experimental Psychology: Learning, Memory, and Cognition 17(1), 163–169 (1991)
30. Herron, C., York, H., Corrie, C., Cole, S.: A comparison study of the effects of a story-based video instructional package versus a text-based instructional package in the intermediate-level foreign language classroom. Calico Journal 23(2), 281 (2006)
31. Hoofnagle, C.J., King, J., Li, S., Turow, J.: How Different are Young Adults from Older Adults When it Comes to Information Privacy Attitudes and Policies? SSRN eLibrary (2010)
32. Johnson, B.B., Slovic, P.: Presenting uncertainty in health risk assessment: initial studies of its effects on risk perception and trust. Risk Analysis 15(4), 485–494 (1995)
33. Jonah, B.A., Thiessen, R., Au-Yeung, E.: Sensation seeking, risky driving and behavioral adaptation. Accident Analysis & Prevention 33(5), 679–684 (2001)
34. Kahneman, D., Frederick, S.: Representativeness revisited: Attribute substitution in intuitive judgment. Heuristics and Biases: The Psychology of Intuitive Judgment, 49–81 (2002)
35. Lövdén, M., Rönnlund, M., Wahlin, Å., Bäckman, L., Nyberg, L., Nilsson, L.G.: The extent of stability and change in episodic and semantic memory in old age: Demographic predictors of level and change. The Journals of Gerontology Series B: Psychological Sciences and Social Sciences 59(3), 130 (2004)
36. Nissenbaum, H.F.: Privacy in context: Technology, policy, and the integrity of social life. Stanford Law & Politics (2010)
37. Norberg, P.A., Horne, D.R., Horne, D.A.: The privacy paradox: Personal information disclosure intentions versus behaviors. Journal of Consumer Affairs 41(1), 100–126 (2007)
38. Phelps, J., Nowak, G., Ferrell, E.: Privacy concerns and consumer willingness to provide personal information. Journal of Public Policy & Marketing, 27–41 (2000)

39. Posner, R.A.: Right of privacy, the. Ga. L. Rev. 12, 393 (1977)
40. Posner, R.A.: The economics of privacy. The American Economic Review 71(2), 405–409 (1981)
41. Schneier, B.: The psychology of security. Communications of the ACM 50(5), 128 (2007)
42. Sheng, S., Magnien, B., Kumaraguru, P., Acquisti, A., Cranor, L., Hong, J., Nunge, E.: Anti-phishing phil: The design and evaluation of a game that teaches people not to fall for phish. In: Proceedings of the 3rd Symposium on Usable Privacy and Security, pp. 88–99. ACM (2007)
43. Sherman, S.J.: On the self-erasing nature of errors of prediction. Journal of Personality and Social Psychology 39(2), 211–221 (1980)
44. Sloman, S.A.: The empirical case for two systems of reasoning. Psychological Bulletin 119(1), 3 (1996)
45. Smetters, D.K., Grinter, R.E.: Moving from the design of usable security technologies to the design of useful secure applications. In: Proceedings of the 2002 Workshop on New Security Paradigms, NSPW 2002, pp. 82–89. ACM, New York (2002)
46. Spencer, W.D., Raz, N.: Differential effects of aging on memory for content and context: A meta-analysis. Psychology and Aging 10(4), 527 (1995)
47. Tulving, E.: What is episodic memory? Current Directions in Psychological Science 2(3), 67–70 (1993)
48. Tversky, A., Kahneman, D.: Judgment under uncertainty: Heuristics and biases. Science 185(4157), 1124 (1974)
49. Wang, Y., Komanduri, S., Leon, P., Norcie, G., Acquisti, A., Cranor, L.: I regretted the minute I pressed share.: A qualitative study of regrets on Facebook. In: Symposium on Usable Privacy and Security (2011)
50. Whitten, A., Tygar, J.: Why Johnny cant encrypt: A usability evaluation of PGP 5.0. In: Proceedings of the 8th USENIX Security Symposium, vol. 99 (1999)
51. Wild, K., Boise, L., Lundell, J., Foucek, A.: Unobtrusive in-home monitoring of cognitive and physical health: Reactions and perceptions of older adults. Journal of Applied Gerontology 27(2), 181–200 (2008)
52. Willis, S., Schaie, K., Martin, M.: Cognitive plasticity. Handbook of Theories of Aging, 295–322 (2009)

Use Fewer Instances of the Letter "i": Toward Writing Style Anonymization

Andrew W.E. McDonald, Sadia Afroz, Aylin Caliskan,
Ariel Stolerman, and Rachel Greenstadt

Drexel University, Philadelphia, PA
{awm32,sa499,ac993,ams573,greenie}@cs.drexel.edu

Abstract. This paper presents Anonymouth, a novel framework for anonymizing writing style. Without accounting for style, anonymous authors risk identification. This framework is necessary to provide a tool for testing the consistency of anonymized writing style and a mechanism for adaptive attacks against stylometry techniques. Our framework defines the steps necessary to anonymize documents and implements them. A key contribution of this work is this framework, including novel methods for identifying which features of documents need to change and how they must be changed to accomplish document anonymization. In our experiment, 80% of the user study participants were able to anonymize their documents in terms of a fixed corpus and limited feature set used. However, modifying pre-written documents were found to be difficult and the anonymization did not hold up to more extensive feature sets. It is important to note that Anonymouth is only the first step toward a tool to acheive stylometric anonymity with respect to state-of-the-art authorship attribution techniques. The topic needs further exploration in order to accomplish significant anonymity.

Keywords: stylometry, privacy, anonymity, machine learning.

1 Introduction

The Privacy Enhancing Technologies community has long been interested in tools that enable people to participate in anonymous or pseudonymous speech[1]. Current anonymity and circumvention systems focus strongly on location-based privacy but do not address many avenues for the leakage of identification through the content of data. In particular, writing style as a marker of identity is not addressed in current circumvention tools. Given the high accuracy of even basic stylometry systems this is not a topic that can afford to be overlooked.

Stylometry is a form of authorship recognition that relies on the linguistic information found in a document. While stylometry existed before computers and artificial intelligence, the field is currently dominated by AI techniques such as neural networks and statistical pattern recognition. State-of-the-art stylometry

[1] Selected Papers in Anonymity: http://freehaven.net/anonbib/

S. Fischer-Hübner and M. Wright (Eds.): PETS 2012, LNCS 7384, pp. 299–318, 2012.

approaches can identify individuals in sets of 50 authors with over 90% accuracy [1]. Recent work has scaled stylometry methods to over 100,000 authors [2]. Stylometry is currently used in intelligence analsyis and forensics. The 2009 Technology Assessment for the State of the Art Biometrics Excellence Roadmap (SABER) commissioned by the FBI stated that, "As non-handwritten communications become more prevalent, such as blogging, text messaging and emails, there is a growing need to identify writers not by their written script, but by analysis of the typed content [3]."

The stylometry field has focused on creating new methods that attempt to classify unknown works using known sets of authors, with little attention being given to the question of what happens when an adversary tries to intentionally circumvent the classification system that has been established. This paper aims to provide a framework for researching and accomplishing writing style anonymization: Anonymouth. Work by Brennan and Greenstadt has shown that non-expert human subjects can defeat stylometry simply by consciously hiding their writing style or imitating the style of another author [4]. However, when analyzing the Brennan-Greenstadt Adversarial Stylometry Corpus, we find that some authors are more capable of composing anonymous documents than others. Furthermore, a case study of the long-term pseudonymous blog, "A Gay Girl in Damascus," showed that even when authors were skilled at hiding their style, doing so with consistency was difficult [5]. There is currently active research in finding stylometry methods that work on adversarial passages such as those in the Brennan-Greenstadt corpus. Even if these methods succeed at identifying these adversarial passages, they should be benchmarked against an adaptive attack where the adversary has access to the features and tools used to identify the text. Lastly, the limited research in circumventing stylometry has focused on creating anonymous documents, whereas the Anonymouth framework provides a mechanism to study the modification/anonymization of documents that were written without anonymity in mind.

This paper includes three key contributions.

1. Our Anonymouth framework defines the steps necessary to anonymize documents. Anonymouth's novel feature clustering and prioritization algorithms enable it to identify the changes necessary to anonymize a document relative to a set of author documents and a set of linguistic features. We show that modifying the features as suggested does result in anonymized documents.
2. We have implemented this framework via two tools, JStylo and Anonymouth, that have been released under an open source license (GPL 3) and can serve as a research platform for stylometry and adversarial stylometry[2]. This software not only performs authorship attribution, but also calculates the features that are most identifying and the ways the feature vectors must change to provide anonymity. The software also provides suggestions to users to help them anonymize their style. We found that 80% of user study participants

[2] Available at `https://psal.cs.drexel.edu/index.php/JStylo-Anonymouth`

were able to anonymize their documents in terms of a data corpus and feature set that is known to and chosen by the user before anonymization.
3. We have performed a user study to investigate whether and how users can edit previously written documents so they obscure their authorship. We show that this problem is harder than starting from scratch. Anonymouth can suggest the right changes, but they are difficult to implement. The methods used to aid a user in anonymizing his document need further development in order for Anonymouth to be effective against state-of-the-art feature sets.

2 Related Work

Anonymization plays an important role in data privacy. Perfect anonymity is hard to achieve. Private information about an individual can be revealed not only from his name and physical and virtual addresses, but also from browser configuration [6], netflix movie ratings [7], and even from the public outputs of a recommender system [8]. Anonymity at the network level can be achieved through onion routing systems like Tor [9]. But the privacy concerns of writing style are still not well-analyzed. Writing style is a serious threat to anonymity and free speech. With the improvement of authorship recognition techniques, it is possible to identify authorship of a document even among 100,000 authors [2].

Authorship attribution can be circumvented by changing writing style. All authorship attribution techniques are based on the fact that people always write in their regular style. Brennan et al. showed that current authorship attribution techniques perform less than random chance if people hide their writing style by imitating someone else or by obfuscating their regular style [4]. Though it is possible to change writing style, it is hard to maintain a separate style consistently in anonymous writings [5].

Rao and Rohatgi suggested round trip machine translation (for example, English → German → English) as a possible method for document anonymization [10]. But because of improvement in machine translation, empirical results have shown that round trip machine translation is not effective in obfuscating writing style[3].

Anonymization by obfuscating writing style was first explored by Kacmarcik et al. [11]. Their approach was to identify the features that a typical authorship attribution method uses to attribute authorship and then adjust the frequencies of these features to make them less effective. They used the Federalist papers and found that 14 changes per 1000 words are sufficient to reduce the likelihood of identifying an author as himself. Our work differs from this work in several ways. First, they did not change the actual documents, only modified the feature sets to prove obfuscation is possible to circumvent attribution. Anonymouth helps the user to change the actual document. Second, their feature set was limited, only function words were used. Anonymouth supports both the Basic-9 [4] feature set and the Writeprints [1] features. Third, only the 12 disputed

[3] http://events.ccc.de/congress/2009/Fahrplan/events/3468.en.html

Federalist papers were analyzed, whereas Anonymouth allows obfuscation of any written document.

There is considerable prior work in authorship attribution [1,12,13]. But currently no tool is available to allow circumvention of the authorship attribution techniques to achieve anonymity. Anonymouth is the first research to explore the idea of changing writing style to anonymize a written document. Anonymouth makes a user aware of the idiosyncrasies of his writing style. It allows users to choose a background corpus in terms of which a document can be anonymized. It indicates features that are unique to the user and suggests how a feature value can be adjusted to achieve a sufficient level of anonymity.

3 Problem Statement

An author A has a document D that he wants to anonymize. The author selects a set of his own writing D_{pre} and a set B of N authors where $A \notin B$. Author A also chooses a feature set F and authorship attribution method M. The goal is to create a new document $D\prime$ from D where the feature values F are changed sufficiently so that $D\prime$ does not appear to be written by A. To evaluate the level of anonymity, D_{pre} is used. $D\prime$ is anonymized if a classifier trained on D_{pre} and documents written by B, B_{pre}, attributes authorship of $D\prime$ to A with a probability p less than random chance, i.e. $p \leq \frac{1}{N+1}$.

4 Approach

Our writing style anonymization framework consists of two platforms: JStylo and Anonymouth. JStylo is a standalone platform for authorship attribution. It is used as an underlying feature extraction and authorship attribution engine for the anonymization framework. Anonymouth is the writing style anonimization platform. It uses the extracted stylometric features and classification results obtained through JStylo and provides suggestions to users to anonymize their writing style.

4.1 JStylo: An Authorship-Attribution Platform

JStylo uses NLP techniques to extract linguistic features from documents, and supervised machine learning methods to classify those documents based on the extracted features. JStylo first "learns" the style of known candidate authors based on documents of those authors, and the style of a given set of anonymous documents. It then attributes authorship of the anonymous documents to any of the known authors. JStylo is a Java-based open-source software with a graphic user interface and an extendable API.

Structure and Usage. The main work-flow of JStylo consists of four consecutive phases: defining a problem set, defining a feature set, selecting classifiers and running the analysis.

A problem set is defined by a training corpus, constructed of documents of all potential authors (as it is supervised learning), and a set of documents of unknown authorship whose authorship are to be determined.

A feature set is defined by a set of various stylistic features to be extracted from the text. Currently there are just above 50 different configurable features available, spanning over different levels of the text, like parts-of-speech in the syntactic level or word frequencies in the lexical level.

The current version of JStylo supports three pre-defined feature sets: Basic-9, Writeprints, and Writeprints (limited). The Basic-9 feature set consists of the nine features that were used in the neural network experiments in [4]. The Writeprints feature set consists of the features used for the Writeprints technique [1]. The Writeprints (Limited) feature set consists of the same features used for Writeprints, where feature classes with potential of exceeding 50 features (e.g. letter bigrams, with a potential of 26^2 features) are limited to the top 50 features. The documents in the training set are mined for the selected features, which are later used for training the classifier, basically profiling the stylistic characteristics of each candidate author. The same features are mined in the test set, for later classification by the trained classifiers.

Each feature is defined by 1) optional text pre-processing tools that allow various filtering methods, to be applied before the feature extraction (e.g. stripping all punctuation); 2) the "core" of the feature which is the feature extractor itself; 3) optional feature post-processing tools to be applied on the features after extraction (e.g. picking the top features frequency-wise); and 4) optional normalization baselines and factoring multipliers (e.g. normalizing over the number of words in each document). The components in 1-3 are based on the JGAAP API [14].

The classifiers available for selection are a subset of Weka [15] classifiers commonly used, such as support vector machine, Naïve Bayes, decision tree, etc. There are several analysis configurations available, the main choice being either to run a 10-fold cross validation analysis over the training corpus or to train the classifiers using a training corpus and classifying the test documents.

JStylo as a Stylometry Research Platform. The main advantages and novelties of JStylo are 1) allowing integration of multiple features to represent various stylistic characteristics of documents, and 2) a high level of feature-set customizability, where each feature can be configured with its own text pre-processing tools, feature extractors, feature post-processing tools and normalization methods. Its user-friendly graphic interface and Java API allow a high level of usage across both linguistic researchers and computer scientists, providing a convenient platform for stylometry research.

Details of the performance and accuracy of JStylo as a stylometry research platform are discussed in section 6.1.

4.2 Anonymouth: An Authorship–Anonymization Framework

Anonymouth aims to use the tools of authorship attribution to systematically render them ineffective on a text, while preserving the message of the document

in question to the highest degree possible. The task of actively changing the document is however, at this point, left to the user. For Anonymouth to be able to read in a document and output an anonymized version satisfying the constraint that the meaning be preserved, it would need a deep understanding of the structure of the English language (assuming English text), knowledge of almost all words, and a reasonable grasp of things like metaphors and idioms - which is quite a tall order.

After initialization via JStylo, Anonymouth performs an iterative two-step approach to achieve writing style anonymization. These steps are: 1) feature clustering and preferential ordering, and 2) feature modification and document reclassification.

Initialization. Anonymouth requires[4] the user (A) to input three sets of documents: 1) a single document consisting of approximately 500 ± 25 words, the `documentToAnonymize` (D); 2) a set (at least 2, though preferably more) of sample documents written by the user, totaling 6500 ± 500 words, the `userSampleDocuments` (D_{pre}); and 3) a corpus — preferably made up of at least 3 different authors — of sample documents, *not* written by the user, containing 6500 ± 500 words per author, the `otherSampleDocuments` (B_{pre}). The `userSampleDocuments` are used to determine where the `documentToAnonymize`'s features should *not* be, while the `otherSampleDocuments` are used to determine where the `documentToAnonymize`'s features could be moved to.

After an initial classification has been produced (by JStylo), four groups of features result: 1) those extracted from D, `toAnonymizeFeatures`; 2) those extracted from D_{pre}, `userSampleFeatures`; 3) those extracted from B_{pre}, `otherSampleFeatures`; and 4) a combination of the two previous groups, `userAndOtherFeatures`. Anonymouth then runs Weka's information gain method on the `userAndOtherFeatures` to select the top f features according to information gain. These top f features will be used in the subsequent computations to generate suggestions for changing writing style. Among the top f features, any that *are not present* in D are excluded from the suggestions Anonymouth deliveres. Resultantly, f becomes $f\prime$. This is done to provide effective suggestions because it cannot be freely assumed that any given feature can be reasonably added to the document. This only applies to JStylo's Writeprints feature sets, where without excluding the non-existing features from suggestions (as an extreme example), a user might be asked to include the word, "Electromagnetic" — when that particular word has no business appearing in the document the user is interested in anonymizing.

Feature Clustering and Preferential Ordering. Knowing what features to change in order to anonymize a document says nothing about how much

[4] In its present state as a research platform rather than a software designed for an end-user, this is the case. However, these limitations are by no means absolute.

to change them, nor does it indicate how much they can be changed and still represent a coherent document that adheres to the rules of grammar. The cause–and–effect relationship among the stylometric features is comparable to that of a field of Dominoes: altering the sentence count of a text will impact the average sentence length; which will affect the Gunning-Fog Readability Index — which happens to be inversely related to the Flesch-Kincaid Reading Ease Score; all of which will inevitably change the character count and will (probably) change the number of occurrences of the three letter word "and". Because of this, it is hard to decide exactly what changes can/should be made in an existing document. However, individually grouping the values of every feature across all B_{pre} seems to provide a fairly decent guideline. It allows Anonymouth to decide how to change each of the $f\prime$ features based upon where the 'real' document's features lie with respect to both one another as well as the user's normal distribution for each feature. The clustering of all instances of each feature assists Anonymouth in selecting physically realizable 'target' values to represent the 'suggested' final document configuration that the user should aim to achieve in order to evade authorship detection. The mechanism behind this selection process is presented through the rest of this section.

Objects containing `otherSampleFeatures` and their respective document names are then fed into a modified k-means clustering algorithm (described in Algorithm 1). The algorithm clusters the objects with respect to each Object's value with, $k = numAuthors$ (where $numAuthors$ is the total number of authors), means, using a modified k-means++ [16] initialization algorithm on a per feature basis spanning across all documents represented by `otherSampleFeatures`. The most significant change to the k-means algorithm is that if any clusters exist with less than three elements after the algorithm converges, the algorithm is re-initialized with $k = k - 1$ means. A more accurate representation might be ak-means. The reasoning behind this adjustment is: because target values for the `documentToAnonymize` are chosen as the centroids of clusters, more elements weighing in on the target value (centroid) increases the potential for anonymization — as opposed to having a single element cluster and effectively copying another's writing style[5]. It remains to be seen whether it would be beneficial to scale the minimum cluster size limit as the number of documents increases; as of now, the value remains fixed.

Implementing these changes in the k-means++ and k-means algorithms creates a safety net that allows Anonymouth to deal with many potential issues that may arise while analyzing an unknown number of documents with unknown similarities/differences. Anonymouth assumes that the documents it receives will be easily clustered. It will adapt if this is not the case, and produce the most beneficial output it can.

[5] There is no guarantee that each cluster will contain documents from more than one author. However, limiting the minimum cluster size helps increase the chances of this happening. In practice, clusters have been observed to contain documents by more than one author more often than not.

Algorithm 1. The *a*k-means Clustering Algorithm (Done on a per-feature basis)

1. Initialization:
   (a) run k-means++ algorithm to initialize cluster's based on `otherSampleFeatures`, with the following exceptions:
      i. If 10,000 numbers have been tried before finding a new centroid, restart.
      ii. If all remaining unchosen values are the same, update the number of total centroids to number of current centroids, set $maxCentroidsFound = True$, and exit initialization; nothing else can be done.
   (b) Assign all instances of the current feature (one per document) from `otherSampleFeatures` to the centroid nearest to it based on one-dimensional euclidean distance. These are the initial clusters.
2. Update Centroids:
   (a) Calculate the average of the elements (features) contained within each cluster, and update that cluster's centroid with the calculated average.
3. Reorganization:
   (a) Calculate the linear distance between each element, and each existing centroid.
   (b) Assign each element to its closest centroid based on the distance calculation in (a).
   (c) If no elements moved:
      i. If $maxCentroidsFound$ is $True$, or there are at least two clusters with no less than 3 elements per cluster, algorithm has converged.
      ii. If there is only one cluster and $maxCentroidsFound$ is $False$, increment $numMeans$, and Initialize.
      iii. If there are any clusters with less than 3 elements and $maxCentroidsFound$ is $False$, decrement $numMeans$, and Initialize.
   (d) Else if elements did move:
      i. Update centroids.

Once the *a*k-means algorithm has converged, clusters are assigned a preference value based on the primary preference calculation, and placed into an $i \times j$ array after being sorted (from least to greatest).

$$p_{i,j} = numElements_{i,j} \times \mid centroid_{i,j} - userSampleMean_i \mid \qquad (1)$$

where: $p_{i,j}$ is the primary preference of feature i's jth cluster; $numElements_{i,j}$ is the number of elements in feature i's jth cluster; $centroid_{i,j}$ is the average of feature i's jth cluster's elements; and $userSampleAvg_i$ is the average of the user's sample documents, `userSampleDocuments`, for feature i. The purpose of taking the number of elements into account rather than basing a cluster's preference value off its distance from the user's average values alone is to avoid attempting to modify a user's `documentToAnonymize` to take the form of a document who's features lie in the extremes due to specific content, while refraining from unwittingly restricting the pool of potential target values due to a single feature. Ordering each feature's clusters in such a way that the most desirable cluster has the highest value also lays the groundwork that allows cluster groups to be ordered by a secondary preference calculation.

The secondary preference calculation weights features with respect to their information gain ranking, and ensures that cluster groups that appear with high frequency take precedence over those that appear less often. Because the most desirable cluster, as determined by the primary preference calculation in Eq. (1), has the highest value, weighting the secondary preference calculation in this manner is intended to assign the greatest cluster group preference to the most common cluster group that has the most impact on the features with high information gain. The centroids of the cluster group with the highest ranking are likely to be the best target values for the `documentToAnonymize`. However, because the primary and secondary preference calculations have not been completely optimized, it is possible that the actual best target cluster will be found slightly further down the list of cluster group preferences. For this reason, as well as to help validate the approach by graphically displaying the workings of Anonymouth, the Clusters tab was created.

The "Clusters" tab, as seen in Fig. 1, displays the clusters formed by the Algorithm 1, represented by the empty green ellipses which contain clusters of blue dots representing the `otherSampleFeatures`. A shaded purple ellipse displays the user's confidence interval (CI) for a given feature. CI is computed using the following formula,

$$CI = D_{pre_{mean}} \pm 1.96 \times \sigma \tag{2}$$

where, $D_{pre_{mean}}$ = average of all `userSampleDocuments`($\mathrm{D_{pre}}$), and σ = standard deviation from the mean. The visible red dot displays the present value of the same feature in the `documentToAnonymize`, which Anonymouth tries to 'put' in the most populated location as far away from the purple shaded ellipse as possible. By selecting one of the available cluster configurations from the drop-down menu, the user may view configurations from, P_{CG_0} (which should provide the greatest potential to anonymize the document) to $P_{CG_{u-1}}$ (which should provide the least potential to anonymize the document), where u is the number of unique document cluster configurations, and P_{CG_n} is the nth document cluster group's cluster group preference. Upon choosing a configuration, one cluster per feature will be shaded green, and is the target cluster for that feature within that configuration. When a cluster configuration is selected, each target cluster's centroid — represented by the empty black circle — is set to be the target value for each feature (respectively) within the `documentToAnonymize`.

One might ask, why not simply pick the cluster farthest away from the author's average value for each feature? The danger in doing this, as has been determined experimentally, is that many features are co-dependent upon one another; so, it may be physically impossible to modify a document to be represented by a set of independently chosen features. For example, it is impossible to increase the average sentence length of a document, while increasing the number of sentences (assuming the idea is to keep the document more or less the same length). Target values for features must be selected while being mindful of other features. Why, then, not just use the document with a cluster group configuration farthest from the author's standard set of values? This is done because ideally it is more feasible

to alter an existing document to look 'generic' than it is to attempt to drive it toward an extreme, which may only be that way as a result of content (including unusual errors that one might have a hard time trying to, or would not want to, reproduce). If many documents share more or less the same configuration, there is a greater chance that any given document can also be fit to share that configuration while maintaining readability. Furthermore, changing a document to look more like many other documents should be more effective in making it anonymous than simply altering it to look as much unlike the true author's work as possible.

Fig. 1. Anonymouth Clusters tab with cluster group P_{CG_0} selected, using the Basic-9 feature set, with 6 'otherSample' authors. The red circles display the present value of each feature within the user's 'documentToAnonymize', the purple ellipses depict the user's confidence interval (assuming a normal distribution) for each feature, and the shaded green ellipses show where the user's feature's will fall if all features are modified as recommended by Anonymouth.

Feature Modification and Document Reclassification. Once the targets are selected, the user is presented with a clickable list of features to change. When a feature is selected, a suggestion appears that aids the user in changing the present value of the feature to its target value. The suggestions for the Basic-9 feature set have been optimized to guide the user to change the elements in their document that will have the greatest overall impact on its classification. An example of this is, "[replace] some single use words with less than 3 syllables with words that have already been used and have 3 or more syllables", as seen in Fig. 2. Once the document has been changed so that its present values reflect the target values, the document is reclassified. If the document has reached a

sufficiently low classification[6] the document is considered anonymized. Until that point, the process loops back to 'feature clustering and preferential ordering.' Every time the features are clustered, slightly different clusters may result; which leads to changing target values. We found that in some cases (especially for the Writeprints features) clustering the features only once is a better alternative to continually re-clustering the features upon every classification.

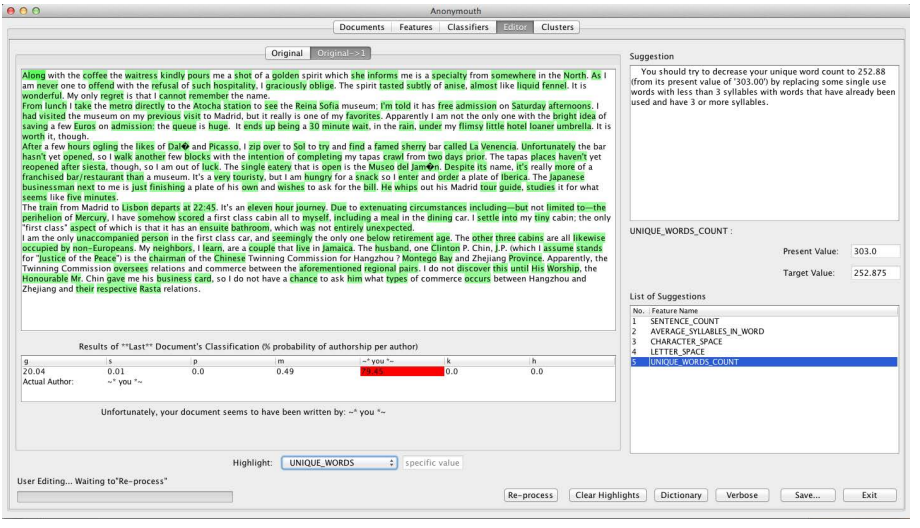


Fig. 2. Anonymouth Editor tab showing the 'Unique Words Count' suggestion selected, with unique words (only used once) highlighted, and an initial classification attributing authorship of the displayed text to the user of the Anonymouth with 79.5% probability

The Editor tab contains a 'Dictionary' which brings up an interface to Princeton's WordNet 3.0, allowing a user to search for synonyms and words containing various continuous character strings (e.g. 'ie'). A 'verbose' button will bring up a window that prints Anonymouth's standard output and error streams in real time as well. Finally, should the user want to revert back to a previous copy of the `documentToAnonymize`, tabs that display where each copy of the document originated from permit the user to trace back through processed changes, while viewing each document's classification results.

5 Anonymouth User Study

We performed a user study with 10 participants to understand the effectiveness of Anonymouth in changing writing style. We evaluated Anonymouth based on its effectiveness in anonymizing a document and its ease of use from a user's perspective.

[6] In this case, a sufficiently low classification means at or below random chance, which is $1/(numAuthors)$, where $numAuthors$ is the total number of authors.

We asked participants to anonymize their pre-written writing samples. Anonymizing a pre-written text is more difficult than writing in a changed style from the start. Stylometry methods fail to attribute authorship when people write in a different style [4]. We wanted to evaluate how much anonymity can be achieved by changing the writing style of a pre-existing document. The subjects were asked to submit 6500 words of pre-existing writing, along with a document of approximately 500 words to modify. The 6500-word sample was used as the subject's training sample. We chose to use 6500 words of writing as it has been found to be enough to leak the identity of an author [10]. As a background corpus, we used regular writing samples of six authors from the Brennan-Greenstadt adversarial corpus. Anonymouth allows the user to choose any corpus as a background corpus. But for the purpose of the study we fixed the background corpus for all users.

During the experiment, each user was asked to use the Basic-9 feature set and SMO SVM classifier as authorship attribution method. Anonymouth provides the option of choosing any feature set and any classifier. The reason we fixed the feature set is because changing an existing document with Writeprints features is very hard, and after the first two participants failed to follow the suggestion for changing Writeprints features we decided not to use it in this study. We excluded the Writeprints result of the first two participants. We only used all of the 10 participants results with Basic-9 features. SMO SVM is used for its high accuracy. The users were asked to perform classification of their document, choose the appropriate cluster from the clustering window and change the document based on Anonymouth's suggestion. We suggested the users to repeat the process for one hour, or until the result of the classification goes below random chance (which was 14% accuracy in our case).

After a user successfully anonymized his/her document or used Anonymouth for an hour, we asked them to rate several aspect of Anonymouth on a 10-point Likert scale. The survey asked basic demographics questions.

6 Evaluation and Results

This section discusses the results of the Anonymouth user study. The following subsections explain the effectiveness of JStylo in attributing authorship, effectiveness of Anonymouth in anonymizing a document, the effect of the choice of background corpus and feature set on anonymity, which features were changed by the users to achieve anonymity, and the user satisfaction survey.

6.1 Effectiveness of JStylo

To evaluate the effectiveness of JStylo as a sufficiently accurate authorship attribution engine for Anonymouth and as an authorship attribution research platform in general, we conducted experiments using the Brennan-Greenstadt Adversarial Stylometry Corpus, which includes 13 authors with 5000-word documents each. We then compared the results with those of two other state-of-the-art

authorship attribution methods in the literature: the Writeprints method and the synonym-based approach [17]. The experiments with JStylo were conducted using a SVM classifier, over two feature sets: the Basic-9 and the Writeprints (Limited). All experiments were evaluated using 10-folds cross-validation. The results are summarized in table 1.

Table 1. Authorship attribution results using Writeprints, Synonym-based and JStylo

Method	*Accuracy*
Writeprints	73.33%
Synonym-based	89.61%
JStylo with Basic-9	53.98%
JStylo with Writeprints (Limited)	92.03%

Although the Basic-9 feature set did not produce as high results as the other methods, it is still much higher than random chance (7.69%), and is used only as baseline for authorship attribution features in JStylo, or anonymization features baseline in Anonymouth. It is notable that using the Writeprints (Limited) feature set with JStylo produced the highest results across all four experiments.

6.2 Effectiveness of Anonymouth

Figure 3 shows authorship attribution accuracy of the modified and unmodified documents. Using the Basic-9 features 80% participants were able to anonymize their documents in terms of the corpus used. The first participant's (s1) original document was not attributed to him as an author. The second participant (s2) made no changes to his document. All other participants were able to anonymize their documents.

6.3 Effect of the Background Corpus on Anonymity

The background corpus, or set of reference authors and documents, is important for document anonymization with Anonymouth as the tool calculates the average value of each feature based on the background corpus and suggests changes to users based on the average feature values.

We tested if documents anonymized in terms of one background corpus are also anonymized against a different background corpus. To test this, we used a different six author subset from the Brennan-Greenstadt adversarial corpus. We also tested the results using the whole 13-author corpus. Results are shown in Figure 6.3 (a) and Figure 6.3 (b). The effectiveness of the anonymization changes if the background corpus is changed. Unfortunately, the basic 9-Feature set is not very effective at stylometry. Where possible, we pre-selected documents that were correctly classified for the anonymization with respect to the original background corpus. However, when we switched to the new background corpus, only four of these were correctly classified. Of these four, 50% (2) of the authors' documents

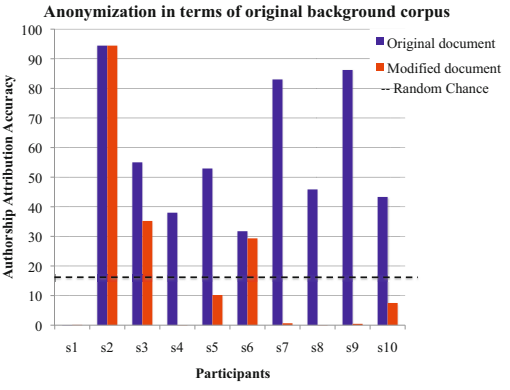


Fig. 3. Authorship attribution accuracy of modified and original documents using the original background corpus. The Basic-9 feature set and SMO SVM classifer were used. All subjects who made changes were able to anonymize their documents (8/10).

were still anonymized even in terms of a different corpus of six authors and the others remained anonymized (as shown in Figure 6.3 (a)). For the corpus of 13 authors, 5 subjects' original documents were classified correctly and all modified documents were classified incorrectly (as shown in Figure 6.3(b)).

6.4 Effect of Feature Set on Anonymity

We wanted to see if documents anonymized with one authorship attribution approach are detectable by another approach. Unfortunately in every case documents anonymized with Basic-9 features were attributed to the real author when Writeprints (Limited) feature were used. The Writeprints feature set is much larger than Basic-9, contains around 700 linguistic, content specific and structural features. Most of these features are very low level features, for example, frequencies of character uni-/bi-/tri-grams. Providing effective suggestions

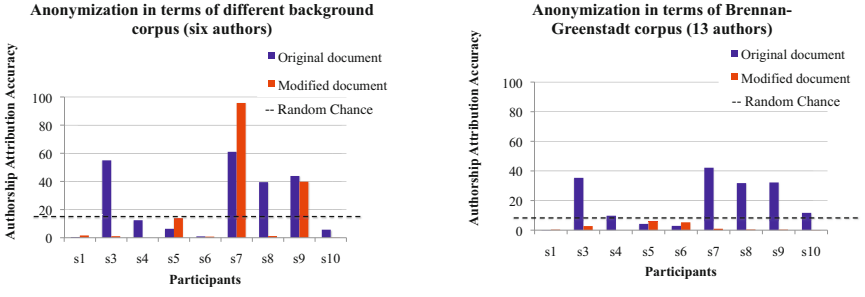


Fig. 4. Authorship attribution accuracy of modified and original documents using six different author samples as background corpus 6.3 (a) and 13 authors as background corpus 6.3 (b). The Basic-9 feature set and SMO SVM classifier were used.

for such low level features is challenging. Changing existing documents by following those suggestions to hide author specific features is also very difficult. For this reason, none of the participants in our study were able to anonymize themselves using the Writeprints (Limited) features.

We wanted to evaluate functionality of Anonymouth using the Writeprints (Limited) features to find out the minimum number of features that need to be changed to anonymize a document. To do this, we first ranked the features based on information gain ratio[??]. Then we calculated clusters of feature values using Anonymouth. We chose the top K features based on information gain ratio and changed their values with those of the first cluster, where K= 25, 50, 75, ..., 300. Result of the experiment is shown in Figure 5.

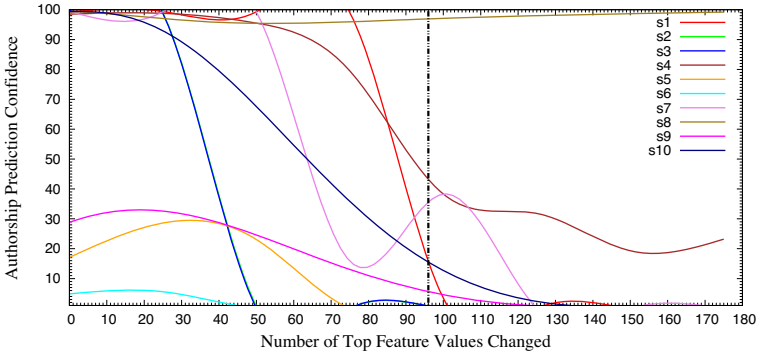


Fig. 5. Number of Writeprints (Limited) features needs to be changed to achieve anonymity. Authorship prediction accuracy decreases as the top feature values are replaced with those predicted by Anonymouth. Sufficient anonymity can be achieved after changing 13.79% (96 out of 696) of the features.

The result shows that authorship prediction accuracy decreases as the top feature values are replaced with the values predicted by Anonymouth. After changing 13.79% of the features, 90% of the documents were anonymized. This experiment shows that the core approach of Anonymouth works successfully to anonymize a document even against a robust feature set like Writeprints.

6.5 Change in Features

We compare the frequencies of different features to understand which ones people change to anonymize their writing style. The changes made to features are shown in Figure 6. We only used samples of the participants who were successful in anonymizing their documents. This graph illustrates the changes in frequencies for each feature. The y-axis contains a list of features that have been adjusted in the passages and x-axis of the graph denotes the change in each feature. We compute the change in feature using the following formula:

Change in Feature f, $C_f = (f_{mod} - f_{ori})/(f_{ori})$
where,
f_{mod} = Value of feature f in the modified document.
f_{ori} = Values of feature f in the original document.

The amount to the right of the y-axis represents the increases in a feature and the amount to the left represents the decreases. 87.5% of the successful participants (7/8) increased average sentence length and decreased sentence count. Average syllable count was increased in 75% of the cases. Increase in complexity was also noticed in every anonymized document. This indicates that most participants made their language complicated to anonymize their documents, which is also evident by the increase of the Gunning-Fog (GF) readability index.

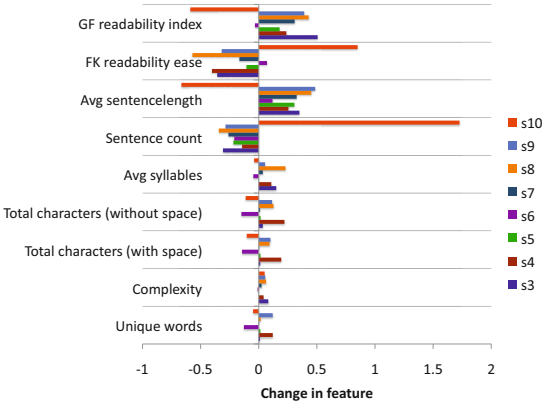


Fig. 6. Feature changes to achieve anonymity

This result differs from the feature changes in the Brennan-Greenstadt adversarial documents where participants used simpler language to obfuscated their document [5]. Ideally a document can be anonymized by using a language that is either more complex or less complex than the original writing style of an author. As seen in [5], people usually use less complex language while obfuscating their writing style, which is easily distinguishable from regular writings. Anonymouth allows user to choose his own background corpus and provide suggestions to change his writing style. Thus by choosing a diverse background corpus an author can hide both his writing style and the indication of changing style.

6.6 User Experience Survey

We had 10 participants in the study within 18-45 age limit who are daily computer users. 2 of them were females and 8 of them were males. On average, the users considered themselves to be moderately good writers. The participants all either had or were working on college degrees, most with different majors, and

90% of them were native English speakers. None of the subjects had any previous knowledge of linguistics or stylometry.

The detailed evaluation of Anonymouth was covered in the first part of the survey which had a 10-point scale, with '0' being the lowest/worst and '9' being the highest/best. The following graph summarizes the reaction of the subjects to Anonymouth that was captured in the first part of the survey.

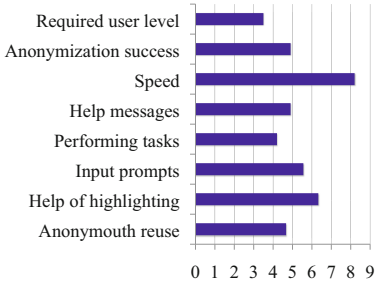


Fig. 7. Anonymouth user experience survey

On average, participants found Anonymouth was user-friendly and that it did not require any specific background knowledge to use. Anonymouth's word highlighting feature was highly rated as helpful. The speed of Anonymouth was considered very fast. Participants felt Anonymouth was moderately successful in anonymizing documents (rated 4.9 on Likert chart). 7 of the participants said they would recommend Anonymouth to other people.

7 Discussion

Although it appears to be quite challenging for a user to implement the changes that Anonymouth asks for, even when only using Brennan and Greenstadt's 'Basic-9' feature set, preliminary results suggest that when users are able to do what is asked, they *can* successfully anonymize their documents — with respect to that feature set. As shown in Fig. 3, 80% of participants were able to reduce the accuracy of the SVM classifier used with respect to the original background corpus used. Furthermore, 60% of participants succeeded in achieving a final classification probability below random chance, which for a total of 7 authors is just under 14.3%.

Initial user tests using the 'Writeprints (Limited)' feature set implemented by JStylo suggested less usability than existed when using the Basic-9 feature set in terms of users being able to perform the actions requested by Anonymouth. Due to the complex nature of the Writeprints (Limited) feature set, the user is asked to do things like add more of the letter 'i' to his/her document, or to decrease the number of occurrences of a part of speech n-gram. While no one was able to anonymize their document with respect to the Writeprints (Limited) feature set,

it has been shown that in general, if approximately 15% of possible features are changed to the values determined by Anonymouth, a document initially classified as having been written by its actual author with 98% probability, will — about 80% of the time — end up being classified as having been written by another author with over 95% probability.

This suggests that the core of Anonymouth — the methods used to determine what and how much should be changed within a document — have some merit. That is not to say that Anonymouth's core has either been optimally adjusted or is in fact the best way to decide how to anonymize a document. There is a clear separation between knowing the degree to which certain things need to be changed, and being able to execute those changes. Resulting from finding that Anonymouth's suggestions to the user regarding how to make these changes need re-working, it is quite possible that Anonymouth's algorithms will need to be re-worked as well.

7.1 Future Work

In general, it seems as though the information presented to the user should be of a higher level, such as, "re-write this sentence using the third person and in the past tense". Of course, doing just this is not the solution. In attempting to anonymize a document via a set of naïve algorithms, there appears to be a trade-off between anonymity and affect. Assuming that the author of a document has written that document in a style that he usually writes in, it is very difficult for that author to go back to another document and modify it to then appear in a different style, while retaining the document's meaning (it is assumed that in order to retain meaning, the imagery and tone would have to create the same end result). Simply stripping descriptive words, modifying tense, and altering the point of view (e.g. from third to first person) would certainly increase anonymity; though clearly at the expense of the documents impact on the audience (affect). While this is one approach that may be taken, it seems far from ideal, and as though it ought to be considered as a last resort.

To achieve its goal, Anonymouth must be able to understand what a sentence/passage means to the extent necessary to enable it to produce an output passage expressed using language constructs foreign to the original author's work that can *at least* capture the main idea and tone of the original passage. While a perfect system would be quite challenging to implement, constructing a system that offers a list of potentially reasonable alternatives to a given passage seems to be a realizable goal.

Adding these features to Anonymouth would resolve the current usability problem that limits the application of Anonymouth.

8 Conclusion

This paper presents Anonymouth, a novel framework for anonymizing writing style.Without accounting for style, anonymous authors risk identification.

This framework is necessary to provide a tool for testing the consistency of anonymized writing style and a mechanism for adaptive attacks against stylometry techniques. Our framework defines the steps necessary to anonymize documents and implements them via Anonymouth and JStylo. These are (1) Analysis of the documents using authorship attribution techniques relative to a corpus of text and a set of linguistic features (implemented via JStylo), (2) Determining the features that need to be changed (using information gain), (3) Ordering the features to be changed and determining where they need to go (using our modified k-means clustering approach), (4) Suggesting changes to achieve these changes (via Anonymouth). We have shown that these steps are effective, that users who make the suggested changes do anonymize their documents relative to the suggested feature sets, and that Anonymouth does help users reduce the accuracy of stylometry techniques. However, our user studies suggest that step 4 is quite difficult and significant research remains to determine the best way to suggest changes that are easy to apply, especially for the large and complex feature sets that result in the highest accuracy. It is not so easy to use fewer 'i's.

This paper presents the first study that evaluates modifying pre-written documents to anonymize style. We found that this was much more difficult than creating anonymous documents from scratch. Human subjects can create a document that evades multiple state-of-the-art authorship techniques in 30-60 minutes without access to those techniques [4]. However, an hour was sometimes not enough to anonymize pre-existing documents in reference to a limited feature set and the changes made did not transfer to analysis with other feature sets, showing that the choice of feature set is critical when using a tool like Anonymouth. More research is needed to better anonymize pre-existing documents as people do not often write with anonymity in mind and may wish to publish documents previously written without compromising their identity.

Acknowledgements. We thank DARPA (grant N10AP20014) and the Intel Science and Technology Center (ISTC) for Secure Computing for supporting this work. We thank our colleagues Pavan Kantharaju, Michael Brennan for their invaluable suggestions on the project. We would also like to thank our shepherd Arvind Narayanan and the anonymous reviewers for their useful feedback.

References

1. Abbasi, A., Chen, H.: Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. ACM Trans. Inf. Syst. 26(2), 1–29 (2008)
2. Narayanan, A., Paskov, H., Gong, N., Bethencourt, J., Stefanov, E., Shin, R., Song, D.: On the feasibility of internet-scale author identification. In: Proceedings of the 33rd Conference on IEEE Symposium on Security and Privacy. IEEE (2012)
3. Wayman, J., Orlans, N., Hu, Q., Goodman, F., Ulrich, A., Valencia, V.: Technology assessment for the state of the art biometrics excellence roadmap (March 2009), http://www.biometriccoe.gov/SABER/index.htm

4. Brennan, M., Greenstadt, R.: Practical attacks against authorship recognition techniques. In: Proceedings of the Twenty-First Innovative Applications of Artificial Intelligence Conference (2009)
5. Afroz, S., Brennan, M., Greenstadt, R.: Detecting hoaxes, frauds, and deception in writing style online. In: Proceedings of the 33rd Conference on IEEE Symposium on Security and Privacy. IEEE (2012)
6. Eckersley, P.: How Unique Is Your Web Browser? In: Atallah, M.J., Hopper, N.J. (eds.) PETS 2010. LNCS, vol. 6205, pp. 1–18. Springer, Heidelberg (2010)
7. Narayanan, A., Shmatikov, V.: Robust de-anonymization of large sparse datasets. In: Proceedings of the 29th IEEE Symposium on Security and Privacy, pp. 111–125. IEEE (2008)
8. Calandrino, J.A., Kilzer, A., Narayanan, A., Felten, E.W., Shmatikov, V.: You might also like: Privacy risks of collaborative filtering. In: Proceedings of the 32nd IEEE Symposium on Security and Privacy, pp. 231–246. IEEE (2011)
9. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: Proceedings of the 13th Conference on USENIX Security Symposium, vol. 13, pp. 21–21. USENIX Association (2004)
10. Rao, J.R., Rohatgi, P.: Can pseudonymity really guarantee privacy. In: Proceedings of the Ninth USENIX Security Symposium, pp. 85–96 (2000)
11. Kacmarcik, G., Gamon, M.: Obfuscating document stylometry to preserve author anonymity. In: Proceedings of the COLING/ACL on Main Conference Poster Sessions, pp. 444–451. Association for Computational Linguistics (2006)
12. Juola, P.: Authorship attribution. Foundations and Trends in information Retrieval 1(3), 233–334 (2008)
13. Uzuner, Ö., Katz, B.: A Comparative Study of Language Models for Book and Author Recognition. In: Dale, R., Wong, K.-F., Su, J., Kwong, O.Y. (eds.) IJCNLP 2005. LNCS (LNAI), vol. 3651, pp. 969–980. Springer, Heidelberg (2005)
14. Juola, P.: Jgaap, a java-based, modular, program for textual analysis, text categorization, and authorship attribution
15. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
16. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, Philadelphia, PA, USA, pp. 1027–1035 (2007)
17. Clark, J.H., Hannon, C.J.: A Classifier System for Author Recognition Using Synonym-Based Features. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007. LNCS (LNAI), vol. 4827, pp. 839–849. Springer, Heidelberg (2007)

Author Index