4th Hot Topics in Privacy Enhancing Technologies (HotPETs)

Selected papers*

Waterloo, Canada

July 29, 2011

*HotPETs has no official proceedings. This document compiles selected papers and is published online only to fuel discussions during the workshop. Selected papers are not included in PETS proceedings so as not to preclude later publication of a full paper.

Discover new privacy-enhancing technologies. Question controversial ideas. Challenge philosophical discussions. Explore the future of privacy. Welcome to HotPETs 2011!

Held in conjunction with the PET Symposium, HotPETs is a forum for hot new ideas in privacy. The venue aims at collecting early stage works. It provides a unique chance to receive feedback from the audience of PETS and stimulate future research directions. We were excited to receive 19 submissions and after some tough decisions, 10 remain for presentation in Waterloo.

This is the fourth HotPETs. First, in Leuven, we had an exciting panel debating legal aspects of privacy. A year later, in Seattle, Alessandro Acquisti discussed the economics of privacy. Last year, in Berlin, Alexander Dix enlightened us about privacy policies. This year, Sid Stamm, Mozilla's lead privacy engineer, will present the keynote and hopefully fuel numerous discussions about privacy issues on the web.

We hope that HotPETS 2011 lives up to previous editions standard: informal, lively, inspiring, sometimes controversial and definitely thought-provoking.

We thank authors of submitted papers, presenters, PETS program chairs Simone Fischer-Huebner and Nicholas Hopper, and PETS general chairs, Katrina Hanna and Ian Goldberg.

Have a cool time at HotPETs!

Carmela Troncoso
Julien Freudiger
Waterloo, July 29, 2011

Program

9:00 Opening Remarks

9:05 Session 1: Web Privacy

Email Clients as Decentralized Social Apps in Mr. Privacy. *M. Fischer, T. Purtell, M. Lam (Stanford University)*

More Privacy for Cloud Users: Privacy-Preserving Resource Usage in the Cloud. *D. Slamanig (Carinthia University of Applied Sciences)*

Targeted, Not Tracked: Client-side Solutions for Privacy-Friendly Behavioral Advertising. *M. Bilenko, M. Richardson, J. Tsai (Microsoft Corporation)*

10:20 Break

10:45 Invited Speaker: Dr. Sid Stamm (Mozilla)

12:00 Lunch

13:45 Session 2: Chronicles of Tor

Simulation of Circuit Creation in Tor: Preliminary Results. *W. Boyd, N. Danner, D. Krizanc (Wesleyan University)*

Distributed Privacy-Aware User Counting. *F. Tschorsch, B. Scheuermann (University of Wurzburg, Germany)*

P3: A Privacy Preserving Personalization Middleware for recommendation-based services. *A. Nandi, A. Aghasaryan, M. Bouzid (Bell Labs Research, Alcatel-Lucent)*

14:50 Break

15:05 Session 3: Dude, Where's my Privacy?

Demographic Profiling from MMOG Gameplay. *P. Likarish (University of Iowa), O. Brdiczka, N. Yee, N. Ducheneaut, L. Nelson (PARC)*

Privacy: Gone with the Typing! Identifying Web Users by Their Typing Pattern. *P. Chairunnanda, N. Pham, U. Hengartner (University of Waterloo)*

15:55 Break

16:25 Session 5: The Silence of the Onions

Psychic Routing: Upper Bounds on Routing in Private DTNs. *J. Anderson, F. Stajano (University of Cambridge)*

Sleeping Dogs Lie on a Bed of Onions But Wake When Mixed. *P. Syverson (Naval Research Laboratory)*

17:15 Closing Remarks

Email Clients as Decentralized Social Apps in Mr. Privacy

Michael H. Fischer T. J. Purtell Monica S. Lam

Computer Science Department,
Stanford University, Stanford, CA 94305
{mfischer,tpurtell,lam}@cs.stanford.edu

Abstract. This paper proposes Mr. Privacy, a social application framework built on top of email, that encourages open competition and provides privacy for users. Applications built on Mr. Privacy are "social apps" that look nothing like email. Email is used only as a transport mechanism and personal database. We choose email because it is more pervasive than any social network and it uses standardized open protocols enabling inter-operability across vendors. Consumers can pick their email providers or even host their own servers. We have developed a prototype Mr. Privacy platform for the Android, iOS, and Firefox. On top of Mr. Privacy, we created applications which share GPS locations, music playlists, and contextual discussions of websites. Preliminary results suggest that the email protocols suffice for building these kinds of social applications. This model supports data privacy and ownership, and facilitates inter-operability and competition.

Keywords: distributed social network, privacy, email

1 Introduction

Online social networking is no longer just about visiting a social network webpage. We can now socialize *in situ*, which means that our friends are with us on online while we browse the web, play music, and play a games. Facebook is a leader in *in situ* social networking; their social plugins enable websites to gain access to Facebook's over 600 million active users. More than two million websites have integrated with Facebook, and over 250 million people engage with Facebook on external websites.

In return for access to the large social graph, companies are sharing their customer relationships with Facebook and giving the social networking partner detailed knowledge of our browsing history and *everything* that we wish to share with our friends. Social networks are now brokers of highly-marketable detailed profiles of large numbers of users[1].

All the social networks available today are *social intranets*; individuals must sign up to the same proprietary network before they can interact socially. The owner of a social intranet controls the application platform; it can decide which

applications are allowed and can demand compensation for the use of its social graph. Because of network effect, there may one day be a single monopoly owning the majority of the world's social graph. Lack of open competition stifles innovation and offers consumers less choice. In a *social internet*, there is no single owner of the global social graph. With open competition, vendors may differentiate from each other by offering EULAs (end-user license agreements) that respect privacy, or offering paid services that allow users to opt out of profiling. Application developers are free to innovate without being subjected to the control of a single-party application platform owner.

1.1 Open Social Sharing

A social network offers many important functions such as discovering friends, "stalking" people of interest, status broadcast, photo sharing, and selective interactions with friends like playing games and sharing favorite web pages. This paper focuses on the latter and asks if social applications can be shared by friends using different service providers, no different from how inter-operability is provided by telecom and email providers today. Open standards encourage competition, promote innovation, offer consumers choice, and generally lead to wider adoption. Imagine an open standard for playing music: friends can pick their service providers and music providers of their choice without having to sign up with a Big Brother social network which intercepts all interactions.

It is not easy to create an open social application platform that can challenge the status quo. To attract developers, such a platform needs a large user base and a programming abstraction that is as capable and easy to use as the comparable centralized API. It is not possible to start such a platform from scratch–social networking is sticky, individuals cannot change their network on their own and still interact with their friends. Furthermore, while it is desirable to let people own their data if desired, the solution must also accommodate the general public who would give up data ownership for free services.

1.2 Email as a Distributed Back End

This paper presents what we believe is a plausible solution to this thorny problem. We propose to build this platform on top of email. Email is a mature, scalable, and open infrastructure used by over 1 billion users. We can communicate with anybody as long as we know his or her email address. We need not sign up to join the same social network. All the shared information is stored as email messages. While most people get their email accounts from a few large companies, individuals and corporations do have the freedom to use paid, advertisement-free email services or to run their own servers.

We have created a prototype of such a system called Mr. Privacy. Mr. Privacy includes a collection of APIs to support social networking functions. Applications can get access to a user's social contacts and interact with them using simple data access operations for application-defined data types. Mr. Privacy hides the low-level details by translating these operations into email protocols, SMTP (Simple

Mail Transfer Protocol) [15] and IMAP (Internet Message Access Protocol) [7]. With Mr. Privacy,

1. users do not know that Mr. Privacy applications are email clients because they sport a user interface similar to any other social applications, and
2. developers need not know they are writing email clients either. They simply create their application-specific data structures, store them, and retrieve them from a database. The database turns out to be distributed, implemented on top of email, and the developers need not worry about hosting or scaling it!

Mr. Privacy overcomes the difficulty of creating a new infrastructure by bootstrapping with email. Applications can be built using the email system as it is today, allowing users to interact with their email contacts through social user interfaces. Even though the email protocol is not optimal, this allows for experimentation with open and distributed social networking. Demonstrated success will hopefully entice email providers to collaborate in optimizing the protocol.

1.3 Contributions

This paper introduces the concept of using email to create an open and standard platform for *in situ* social networking. By turning email into a distributed database, we are leveraging a mature and open system with an even larger installed base than the largest online social network today. Not only does it support better data privacy and ownership, its openness facilitates competition that will lead to better products for consumers.

We have created a prototype of our proposed idea called Mr. Privacy that runs on three platforms: Android, iPhone, and Firefox. We have created three proof-of-concept applications. A GPS sharing application that supports location sharing without having our whereabouts tracked by a central authority. A social music application that illustrates how friends can share music playlists while getting their music from different sources. And finally, a SocialBar extension for Firefox that let friends share comments on any webpage they visit, without having to give away their browsing history and conversations to a single third-party.

While the lack of server-side support of message filtering added significant complexity to our implementation, we found that the SMTP and IMAP protocols with the appropriate extensions are adequate as data storage and transport for the class of social applications we studied.

2 Mr. Privacy Design Rationale

Mr. Privacy is designed to provide developers the primitives needed to build social applications that can inter-operate across email service providers. These primitives are built around IMAP and SMTP to allow developers to use email to store and transport information. As social computing is quickly evolving, Mr.

Privacy is designed to be open and simple, both in interface and implementation, to allow for easy integration across platforms and innovations at the application level.

2.1 Specification

An application developer interacts with Mr. Privacy using four simple API calls that are implemented across device platforms.

- SHARE: Transmits a JSON object as a Mr. Privacy message with the specified tag to a set of recipients.
- LIST: Retrieves a list of JSON objects with the specified tag newer than a certain reference object.
- GET: Retrieves a particular Mr. Privacy JSON object.
- WAIT: Waits for a JSON object with the specified tag that are newer than a certain reference object to arrive.

Mr. Privacy messages are formatted to allow applications that are built using Mr. Privacy to programmatically read the messages. To allow for server-side search, Mr. Privacy messages have the following subject header: "<message subject>[Mr Privacy][<Tag>]". The tag is the name of the application that owns the data in the message. The message body allows the data to be both humans and machine readable. The human readable component of the email is layered using html for rich email clients, such as desktop clients, and text for text-only email clients, such as mobile phones. The machine readable component of the message encodes JSON formatted data for Mr. Privacy applications to use. The layering of information is done using multiple "multipart/alternative" MIME messages. A typical email client reads through each layer of a message and displays the highest fidelity layer that it can. Additional attachments can contain other types of data such as photos. The JSON object encoding the Mr. Privacy data can reference these extra attachments to support multimedia sharing applications.

The human readable version provides a few other benefits. Firstly, users can receive information promptly even if they are not running the application. Secondly, and most importantly, it helps make social applications "viral". If a user of an application shares a piece of data with another user who does not currently have the application installed, the email message acts as an "invitation" to use the application.

The Mr. Privacy platform keeps the user's inbox free from Mr. Privacy messages. When a new message arrives in the inbox, Mr. Privacy checks the header to see if it is for a Mr. Privacy application. If it is, the message is moved into a special folder. This is done automatically as long as a Mr. Privacy application is connected to the IMAP server.

2.2 Discussion

Data privacy and ownership. The primary benefit of an email-based architecture is providing privacy and giving data ownership back to the users. A social

application in this model can be just client software. It is not necessary to have a central server that mediates all communication; the contact information is stored on the clients' machine and all the data shared are stored with the users' respective mail provider. Higher-level privacy protection is generally expected of email providers than social network providers, as a wealth of private information is stored in email today. Note that users will be sharing their information with the vendors of their friends' choice as well, they still need to be vigilant about not sending confidential information to untrusted vendors. Nonetheless, having an open system allowing inter-operability is much preferred to having a monopoly that has the ability to modify the terms of use unilaterally.

Large installed user base and mature infrastructure. Applications written using Mr. Privacy can enjoy a large user base immediately and have no problems scaling by leveraging email. In addition, various services are offered around email identities. Users can have multiple email accounts to prevent others from linking activities of our different personas through avatars. OpenID lets us log in to many web services using our email identities without having to create a new account [16]. Webfinger lets us attach public metadata to our email accounts [20]. The information is stored with our email providers so no single server monitors who is retrieving the information. Webfinger will allow Mr. Privacy to transition to an optimized non-email protocol by allowing a dedicated data service to be linked to our email identities.

Standard and extensible data representation. The social data transport provided by Mr. Privacy allows for both normal social networking and long-tail applications. Consider for example a social application for patients in a medical study. Doctors could let their patients use a Mr. Privacy application on mobile phone to enable database functionality without having to set up a server thus avoiding the addition of a new dimension of HIPPA-compatible IT support requirements. Standard Facebook-like capabilities can be provided by transporting ActivityStreams [3] JSON objects via Mr. Privacy. Because the data are not owned by one provider, users are free to use any viewer application they wish. This brings up the need for access control on subsets of the structured social data that will be stored in email.

Social rules of engagement. Mr. Privacy applications, with user interfaces similar to social applications rather than traditional email clients, need not abide by the rules of engagement established for email. Mr. Privacy changes the contract for email by isolating the social data messages from the normal inbox. We also expect these applications to provide an intuitive way for users to specify a white list. Not only does this prevent spam, it also provides the socially pleasing "read it if you feel like it" paradigm.

Performance. Using Mr. Privacy, when a message is sent to multiple users, each user receives a separate copy. A centralized social network, on the other hand, needs to keep only one copy of the item. Luckily because of the presence of spam, many email providers have optimizations to eliminate duplicate email messages.

Lack of server side computation. IMAP supports only data queries and provides no other functionality typically expected of a social networking server. This makes it difficult for Mr. Privacy to handle public interactions or support friend-of-a-friend interactions. We have chosen email primarily for bootstrapping, extensions to email are expected in the future to support more social networking features. In the meantime, we can leverage existing services to overcome Mr. Privacy's deficiencies. For example, we can selectively make certain parts of our interactions public, such as sharing a photo with Flickr or status update on Twitter.

Invalidating information. Once an item is sent from one user to another, it cannot be invalidated. Invalidation is available in a centralized social network because it has control over all the data. While Mr. Privacy could send out a recall message to be interpreted by applications, users could still read their own email and gain access to the invalidating information.

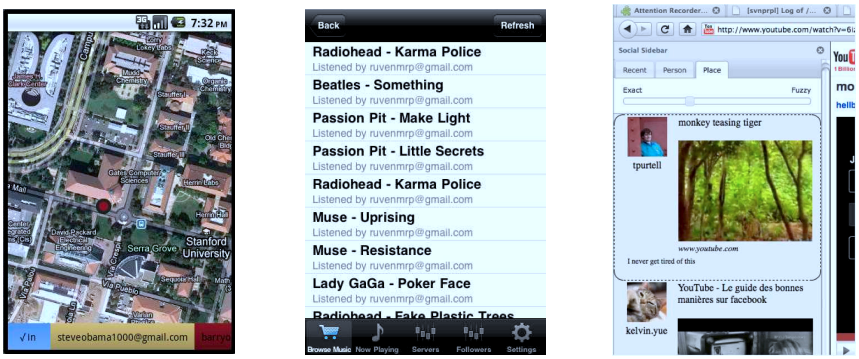
3 Prototype

We explored the implementation of a Mr. Privacy framework on three different platforms: iOS, Android, Firefox. In the ideal implementation, Mr. Privacy is a core platform service. A user trusts Mr. Privacy with with full email access, and Mr. Privacy restricts the data social applications are able to use. The Android implementation of the Mr. Privacy platform was built on the JavaMail framework, which connects directly to a mail server directly via SMTP and IMAP. Unfortunately, the iOS platform disallows local services, so each application must request credentials from the user. The Firefox browser version of Mr. Privacy is a custom mail client implemented in Javascript using the native socket transport functionality exposed to plugins.

Typical social networking applications are built on top of a centralized database. Since we cannot change email servers at all, all application-specific functionality must be provided on the clients. This fundamental shift of responsibility affects both developers and users, so we built three demonstration applications to test out the acceptability of the Mr. Privacy concept from these two points of view.

3.1 GPS Sharing

Consider physical check-in services like Facebook Places, Four Squares, and Google Latitude. While it is fun to let our friends know all the new places we are visiting, making such sensitive information like locations public is potentially dangerous. The "PleaseRobMe.com" website, for example, collects information about when people are away based on public status information and can be used by burglars to pick their victims [17]. We have created an application for the Android phone called Mr. GPS that allows individuals to exchange GPS locations using Mr. Privacy. The application sports a UI similar to other check-in services, as shown in Fig. 1(a).



(a) (b) (c)

Fig. 1. (a) GPS sharing using Mr. Privacy on an Android phone, (b) a shared playlist on Jinzora Mobile on iOS, (c) SocialBar showing comments in Firefox.

3.2 Social Music with Jinzora Mobile

As a prototype, we added playlist sharing via Mr. Privacy to Jinzora Mobile, an open-source music application on the iPhone that allows users to stream music from their PCs[12]. At 30 seconds into each song, Jinzora Mobile shares a datum with the registered friends indicating that the song is being played. Users can now select a new option "Recently Played by Friends", which uses Mr. Privacy *list* and *get* commands to fetch the list of music their friends have played. Users can view these plays, as shown in Fig. 1(b), and tap on them to start listening. By using a standard playlist format, other music players can also be built upon Mr. Privacy to share playlists across all the music services.

3.3 Contextual Social Browsing

To allow users to curate content for each other with privacy, we have developed SocialBar, a FireFox sidebar extension built on top of Mr. Privacy. SocialBar allows us to discover new content, explore a friends interests, and most importantly, discuss the pages we are viewing with friends. SocialBar provides a better *in situ* browsing experience than a portal-based one because it is built into the browser, as shown in Fig. 1(c). Our friends are available on the side as we browse the web. It is particularly engaging if we can continue the conversation on the side as we view live content. Another important advantage of SocialBar is that we can socialize on any web page. For example, academicians can discuss the content of research websites, which normally do not have any integrated social features. We can even leave notes to our friends, and ourselves for that matter, on files of a web-accessible file repository!

3.4 Suitability of Email Protocols

Deploying our applications gave several insights into the viability of the Mr. Privacy concept. We summarize the lessons learned below.

Not all users have a suitable email service. Mr. Privacy requires an IMAP service provider to allow for isolation of data messages to support the "read it if you feel like it" model. POP does not have the support for folders required to hide the messages from the user. Some providers do not offer IMAP access, but there are free alternatives like Yahoo! Mail and Gmail that will suffice.

Servers tamper with messages as a normal part of existing infrastructure. We had originally assumed that Mr. Privacy messages received would be identical to the ones sent. To our surprise, this turned out not to be the case, for example, when mails were sent to mailing lists. This can be handled by not relying on a specific MIME layout of the received messages. Also, spam detection systems occasionally quarantine messages or alter them in minor ways.

Basic IMAP does not provide the full gamut of features required. Mr. Privacy must provide alternative implementations to handle the variations in IMAP support. For example, Mr. Privacy takes advantage of the IMAP extension called "IDLE" so it can be alerted when new messages arrive in a particular folder. Similarly, Mr. Privacy uses the IMAP "CREATE" command to make a new folder on the server. Some mail services, notably AOL mail, do not support this.

Mail systems may have standard protocols, but individual implementations may have different performance characteristics. Mr. Privacy relies on existing servers and therefore must work with existing implementations. It is important that we take advantage of the techniques that servers use to optimize typical email usage. For example, Mr. Privacy takes advantage of server-side search. The first technique we explored was adding Mr. Privacy tags to message headers. This worked well for small test accounts, but would take minutes on accounts with greater than 10,000 messages. Instead, we now tag Mr. Privacy messages by adding "Mr. Privacy" to the subject and use subject search in IMAP to retrieve relevant messages. Because existing mail server implementations have an index dedicated to accelerating subject searches this search filter was dramatically faster. On large inboxes search times were reduced from minutes to seconds.

We have to make compromises in our design and implementation because we are using a legacy protocol. Nonetheless, we found that we were able to provide sufficient functionality for the kinds of social applications we have built while requiring minimal Mr. Privacy related code.

4 Related Work

A growing number of efforts are underway to provide choice in social networking. Google's Open Social allows developers to create a social application once and deploy it on different social networks [9]. However, users cannot interact across networks. This model only reduces the development effort for supporting multiple social data providers. Mozilla's Contacts abstracts existing networks

at the browser level [14] based on the W3C Contacts specification [19]. OneSocialWeb uses federated XMPP and server extensions to create a federated social network [5]. Google's Wave was a collaboration tool that used XMPP and server extensions to enable users to have rich discussion threads across providers [10]. Diaspora [11] and Appleseed [2] define a P2P protocol for social networking. PeerSoN explores building social functionality on top of distributed storage, such as OpenDHT [4].

Various other projects have proposed techniques to improve the usability and effectiveness of email. Flores et al. created The Coordinator, a messaging tool that uses structured requests to capture the essence of language thus enabling social actions [8]. Cockbrun et al. explored Mona, a novel conversation based platform to enhance email for collaborative work [6]. Rodden et al. designed Mailtrays, a system that automatically organizes and filters incoming messages to match the current needs of the user [18]. Semantic Email examined how a better user interface can be presented on top of existing email infrastructure. A management agent orchestrates sending, receiving, and reprocessing messages in order to simplify tasks for the user [13].

5 Conclusions

This paper provides an alternative social application platform to the current status quo where a single company owns all the social data. By building on top of email, our proposed Mr. Privacy platform leverages the billions of email accounts that already exist, the open email protocols that enable inter-operability, and a mature and scalable infrastructure.

Application developers can choose to write their social applications as pure client software if they wish, letting all the data be stored with the users' preferred service providers and leaving scalability issues to email providers. Users can choose an email service provider or host their own email server if they are concerned about data ownership.

We have developed a prototype Mr. Privacy platform for Android, iOS, and Firefox. On top of Mr. Privacy, we created applications which share GPS locations, music playlists, and contextual discussions of websites. There is no single third-party company monitoring all our activities. Furthermore, these applications are using the social contacts in individuals email address books and there is no third-party social network owner that has complete control of the platform.

SocialBar is available at `http://mobisocial.stanford.edu/socialbar/`. Additionally, the source code including the Mr. Privacy client library is publicly released at `https://github.com/Mobisocial/socialbar/`.

Acknowledgment

The authors would like to thank Ruven Chu for his development of the Jinzora Mobile application. This research is supported in part by the NSF POMI (Pro-

grammable Open Mobile Internet) 2020 Expedition Grant 0832820, the Stanford Clean Slate Program, and the Stanford MobiSocial Computing Laboratory.

References

1. ABC News. Microsoft Deepens Facebook Ties in Web Search Battle. http://abcnews.go.com/Business/wireStory?id=11876359.
2. The Appleseed Project, 2010. http://opensource.appleseedproject.org/.
3. M. Atkins, W. Norris, C. Messina, M. Wilkinson, and R. Dolin. Activity Streams Concepts and Representations (draft), 2010. http://activitystrea.ms/head/json-activity.html.
4. S. Buchegger, D. Schiöberg, L.-H. Vu, and A. Datta. PeerSoN: P2P social networking: early experiences and insights. In *SNS '09: Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, pages 46–52, New York, NY, USA, 2009.
5. D. Cheng and O. Griffin. OneSocialWeb, 2010. http://onesocialweb.org/.
6. A. Cockburn and H. Thimbleby. Reducing user effort in collaboration support. In *Proceedings of the 1st International Conference on Intelligent User Interfaces*, IUI '93, pages 215–218, New York, NY, USA, 1993.
7. M. Crispin. Internet Message Access Protocol - Version 4rev1, 1996.
8. F. Flores, M. Graves, B. Hartfield, and T. Winograd. Computer systems and the design of organizational interaction. *ACM Trans. Inf. Syst.*, 6:153–172, April 1988.
9. Google. Opensocial, 2010. http://code.google.com/apis/opensocial/.
10. Google. Wave, 2010. http://wave.google.com/.
11. D. Grippi, M. Salzberg, R. Sofaer, and I. Zhitomirskiy. Diaspora, 2010. http://www.joindiaspora.com/.
12. Jinzora, 2010. http://jinzora.com/.
13. L. McDowell, O. Etzioni, A. Halevy, and H. Levy. Semantic email. In *WWW '04: Proceedings of the 13th International Conference on World Wide Web*, pages 244–254, New York, NY, USA, 2004.
14. Mozilla. Contacts, 2010. https://mozillalabs.com/blog/2010/03/contacts-in-the-browser/.
15. J. Postel. Simple Mail Transfer Protocol, 1982.
16. D. Recordon and D. Reed. OpenID 2.0: A Platform for User-Centric Identity Management. In *DIM '06: Proceedings of the Second ACM Workshop on Digital Identity Management*, pages 11–16, 2006.
17. The Register. Burglars used social network status updates to select victims, 2010. http://www.theregister.co.uk/2010/09/13/social_network_burglary_gang/.
18. T. Rodden and I. Sommerville. *Building conversations using mailtrays*, pages 159–172. North-Holland Publishing Co., Amsterdam, The Netherlands, 1991.
19. W3C. Contacts. http://www.w3.org/TR/2010/WD-contacts-api-20100121/.
20. Webfinger, 2010. http://code.google.com/p/webfinger/.

More Privacy for Cloud Users: Privacy-Preserving Resource Usage in the Cloud

Daniel Slamanig

Carinthia University of Applied Sciences, Primoschgasse 10, 9020 Klagenfurt, Austria.
d.slamanig@cuas.at

Abstract. Contrary to many other approaches to security and privacy in the cloud, we are interested in the problem of hiding behavioral information of users consuming their cloud resources, e.g. CPU time or storage space, from the cloud provider. More precisely, we are looking for solutions that allow users to purchase a contingent of resources from a cloud provider and to anonymously consume their resources till their limit is reached (in case of storage they can also reclaim these resources back anonymously). We present a definition of such *anonymous yet authorized and bounded cloud resource schemes* along with an instantiation based on Camenisch-Lysyanskaya signatures. Then, we extend the scheme to another scheme providing even more privacy for users (by even hiding the issued resource bound during interactions and thus providing full anonymity to users) and provide some useful extensions for both schemes. We also underpin the practical efficiency of our schemes by means of experimental results obtained from an implementation.

Keywords. Anonymous resource consumption, privacy, cloud computing, Camenisch-Lysyanskaya signatures, proofs of knowledge.

1 Introduction

Cloud computing is an emerging paradigm, but some significant attention remains justifiably focused on addressing security and privacy concerns. Reasons are among others that customers have to trust the security mechanisms and configuration of the cloud provider and the cloud provider itself. Recently, different cryptographic solutions to improve privacy have been proposed, which mainly focus on private storage, private computations and private service usage.

Storing data encrypted seems to be sine qua non in many cloud storage settings, since cloud providers, having access to the storage infrastructure, can neither be considered as fully trustworthy nor are resistant to attacks. Kamara and Lauter [26] propose several architectures for cryptographic cloud storage and provide a sound overview of recent non-standard cryptographic primitives like searchable encryption and attribute-based encryption, which are valuable tools in this context. Other issues are data privacy and verifiability when outsourcing data and performing computations on these data using the cloud as computation infrastructure. The recent introduction of fully homomorphic encryption [24] is a promising concept for performing arbitrary computation on encrypted data. However, up to now these concepts are far from being practical [25]. Another interesting issue from a privacy perspective is to hide user's usage behavior (access patterns and frequencies) when accessing services, e.g. cloud storage services. More precisely, users may not want the cloud provider to learn how often they use a service or which resources they access, i.e. they want to have anonymous and unlinkable access. Nevertheless, cloud providers can be assumed to have access restricted to authorized users and additionally users may want to enforce (attribute-based) access control policies. There are quite different existing approaches to realize this, e.g. one can employ anonymous credential systems [4] or oblivious transfer [8, 9].

In this paper we introduce an additional aspect which may be valuable when moving towards privacy friendly cloud computing and seems to be valuable when used in conjunction with the aforementioned approaches. In particular, we focus on the anonymous yet authorized and bounded use of cloud resources like CPU time (e.g. CPU per hour) or storage space. Although we illustrate our concept by means of the resource storage space in this paper, one may use this approach for arbitrary resources.

Our contribution. We consider a setting where users should be able to register and obtain a resource bound (limit) from a cloud provider (CP) in form of a partially blindly signed token. This token includes an identifier, the already consumed resources and the limit, wheres the limit in fact is the only value signed in clear. This limit determines how much of a resource, e.g. CPU time, storage space, a user is allowed to consume. Then, users should be able to consume their resources in an anonymous and unlinkable yet authorized fashion. For instance, if a user wants to consume l resources, he has to convince the CP that he possesses a signed token with a valid identifier (double-spending protection) and that his consumed resources (including l) do not exceed his bound. If this holds, the anonymous user is allowed to consume the resources and obtains an updated signature for a token corresponding to a new identifier and updated consumed resources. Note, due to the anonymity and unlinkability properties, the CP is unable to track how much a user has already consumed, however, can be sure that he solely consumes what he has been granted. Furthermore, when the resource represents storage space a user may also reclaim resources when he deletes data, whereas these actions should also be anonymous and unlinkable.

We for the first time consider this problem and provide a definition for the concept of anonymous yet authorized and bounded cloud resource schemes. Furthermore, we present an efficient instantiation of such schemes. Then, we extend the scheme to another scheme providing even more privacy for users (by even hiding the issued resource bound during interactions) and provide some useful extensions for both schemes. Our schemes are obtained using recent signature schemes due to Camenisch and Lysyanskaya [12, 13] along with efficient zero-knowledge proofs for proving properties of signed messages. We note that many of the approaches discussed subsequently employ similar features of CL signatures as our approach does. But the signer controlled interactive update of signed messages discussed in Section 4.1, which is an important functionality behind our protocols, seems to be novel. Furthermore, we note that we base our concrete scheme on groups of known order and the essential ingredient is the pairing based CL signature scheme [13]. We want to emphasize that one could as well base the construction on hidden order groups. Then, one can use the strong RSA based CL signature [12], Fujisaki-Okamoto commitments [23] and the range proof proposed by Boudot [7]. But within our construction we achieve much shorter proofs and signatures.

Related work. Pairing based CL signatures [13] and it's strong RSA based pendant [12] are useful to construct various privacy enhancing cryptographic protocols. Among them are anonymous credential systems and group signatures [13] as well as privacy protecting multi-coupon systems [16, 18], anonymous subscriptions [6], electronic toll pricing [5], e-cash systems [11] and n-times anonymous authentication schemes [10] based on compact e-cash or unclonable group identification schemes [21] which achieve similar goals as in [10]. To solve our problem, the most straightforward solutions seems e-cash, i.e. CP issues k coins to a user and a user can use one coin per resource unit. However, to achieve a suitable granularity this induces a large amount of "small valued coins" which makes this approach impractical. The same holds for compact e-cash schemes [11], where a user can withdraw a wallet of 2^l coins at a time and thus the withdrawal procedure is much more efficient. However, in compact e-cash coins from the wallet can only be spend one by one and the above problem still exists. In divisible e-cash [15, 3], which allows a user to withdraw a wallet of value 2^l in a single withdraw protocol, spending a value 2^m for $m \leq l$ can be realized more efficient than repeating the spending 2^m times. However, in the former solution even for a moderate value of $l = 10$ the spending of a single coin requires 800 exponentiations which makes it very expensive. The latter approach is more efficient, but statistical, meaning that users can spend more money than what they withdraw.

Multi-coupons [16, 18] represent a collection of coupons (or coins or tokens) which is issued in a single withdraw protocol and every single coupon of the MC can be spend in an anonymous and unlinkable fashion. But in our scenario, they suffer from the same problem as e-cash.

Recently, Camenisch et al. proposed an interesting protocol for unlinkable priced oblivious transfer with rechargeable wallets [9]. This does not exactly fit our scenario but could be mapped to it. However, [9] do not provide an efficiency analysis in their work and their protocols seem to be quite costly. Their rechargeable wallets are an interesting feature and such an idea is also supported by our second scheme.

2 Definition

Below, we present a description of the problem along with a model of anonymous yet authorized and bounded cloud resource scheme. We note, that we do not provide formal security arguments for the presented schemes in this extended abstracts, but they are presented in the full version.

2.1 Problem Description and Motivation

In our setting we have a cloud provider (CP) and a set of users U. Our main goal is that users are able to purchase a contingent of resources (we focus on storage space here) and CP does not learn anything about the resource consumption behavior of users. In particular, users can store data at the CP as long as there are still resources from their contingent available. The CP is in any interaction with the user convinced that a user is allowed to consume (or reclaim) resources and cannot identify the user nor link any of the user's actions. Clearly, if the resource is storage space and the data objects contain information on the user then this may break the anonymity property. Nevertheless, then we can assume that data is encrypted which seems to be sine qua non in many cloud storage settings.

Our main motivation is that it is very likely that only a few large cloud providers will own large portions of the infrastructure of the future Internet. Thus, these cloud providers will eventually be able to link data and information about resource consumption behavior of their consumers (users) allowing them to build extensive dossiers. Since for many enterprises such a transparency can be too intrusive or problematic if these information are available to their competitors we want to hide these information from cloud providers. As for instance argued in [19], activity patterns may constitute confidential business information and if divulged could lead to reverse-engineering of customer base, revenue size, and the like.

2.2 Definition of the Scheme

An anonymous yet authorized and bounded cloud resource scheme is a tuple (`ProviderSetup`, `ObtainLimit`, `Consume`, `Reclaim`) of polynomial time algorithms or protocols between users U and cloud provider CP respectively:

- `ProviderSetup`. On input a security parameter k, this algorithms outputs a key pair sk and pk of a suitable signature scheme and an empty blacklist BL (for double-spending detection).
- `ObtainLimit`. In this protocol a user u wants to obtain a token t for a resource limit of L units from the CP. The user's output is a token t with corresponding signature σ_t issued by CP. The token contains the limit L and the actually consumed resources s (wheres both may be represented by a single value $L' := L - s$). The output of CP is a transcript T_{OL} of the protocol.
- `Consume`. In this protocol user u wants to consume l units from his remaining resources. The user shows value $t.id$ of a token t and convinces the CP that he holds a valid signature σ_t for token t. If the token was not already spend ($t.id$ is not contained in BL), the signature is valid and there are still enough resources left, i.e. $s' + l \leq L$ (or $L' - l \geq 0$), then the user's output is `accept` and an updated token t' for resource limit L and actually consumed resources $s' + l$ (or $L' - l$) with an updated signature $\sigma_{t'}$ from CP. Otherwise the user's output is `reject`. The output of CP is a transcript T_C.
- `Reclaim`. In this protocol user u wants to reclaim l units, e.g. he wants to delete some data of size l. The protocol is exactly the same as the `Consume` protocol. Except for the `accept` case the updated token t' contains $s' - l$ (or $L' + l$) as the actually consumed resources and the transcript is denoted as T_R. We emphasize that u needs to prove by some means that he is allowed to reclaim l resources, e.g. when deleting some data, the user needs prove knowledge of some secret associated with the data during the integration. Otherwise, users could simply run arbitrary many `Reclaim` protocols to illicitly reclaim resources and indirectly improve their actual resource limit.

3 Preliminaries

An essential ingredient for our construction are honest-verifier zero-knowledge proofs of knowledge (Σ-protocols). We use the notation from [14], i.e. a proof of knowledge of a discrete logarithm $x = \log_g y$ to the base g will be denoted as $PK\{(\alpha) : y = g^\alpha\}$, whereas Greek letters always denote values whose knowledge will be proven. We note, that compositions of single Σ-protocols using conjunctions and disjunctions can be efficiently realized [20]. Furthermore, the non-interactive version of a (composed) proof obtained by applying the Fiat-Shamir transform [22] is denoted as a signature of knowledge or SPK for short.

3.1 Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_t$ be two groups of prime order p, let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$ a bilinear map between these two groups. The map e must satisfy the following properties:

1. Bilinear: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$ we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degenerate: $e(g, g) \neq 1$.
3. Computable: There is an efficient algorithm to compute $e(u, v)$ for any $u, v \in \mathbb{G}$.

Though the group operation in $\mathbb{G}$ is in general an additive one, we express both groups using multiplicative notation. This notion is commonly used, since $\mathbb{G}_t$ is always multiplicative and it is more easy to capture the sense of cryptographic protocols.

3.2 Pedersen Commitments

Pedersen commitments [30] represent a widely used commitment scheme working in any group $\mathbb{G}$ of prime order p. Let g, h be random generators of $\mathbb{G}$, whereas $\log_g h$ is unknown. To commit to a value $s \in \mathbb{Z}_p$, one chooses $r \in_R \mathbb{Z}_p$ and computes $C(s, r) = g^s h^r$, which unconditionally hides s as long as r is unknown. To open the commitment, one simply publishes $(s, r, C(s, r))$ and one verifies whether $g^s h^r = C(s, r)$ holds. For simplicity, we often write $C(s)$ for a commitment to s instead of $C(s, r)$. We note that the Pedersen commitment inherits an additive homomorphic property, i.e. given two commitments $C(s_1, r_1) = g^{s_1} h^{r_1}$ and $C(s_2, r_2) = g^{s_2} h^{r_2}$ then one is able to compute $C(s_1 + s_2, r_1 + r_2) = C(s_1, r_1) \cdot C(s_2, r_2)$ without either knowing any of the hidden values s_1 or s_2. Furthermore, note that a proof of knowledge $PK\{(\alpha, \beta) : C = g^\alpha h^\beta\}$ of the ability to open a Pedersen commitment can be realized using a proof of knowledge of a DL representation of C with respect to the elements g and h [28].

3.3 Range Proofs

An elegant proof that a number hidden within a Pedersen commitment lies in an interval $[a, b]$ in the setting of prime order groups was presented in [27]. Although this proof might be impractical in general, since it requires $O(\log\ b)$ single bit-proofs, it is efficient for the application that we have in mind due to small values of b. The basic idea is to consider for a number $x \in [0, b]$ its binary representation $x = x_0 2^0 + x_1 2^1 + \ldots + x_{k-1} 2^{k-1}$, whereas $x_i \in \{0, 1\}$, $0 \leq i < k$. Thereby, $k = [\log_2\ b] + 1$ represents the number of digits, which are necessary to represent every number within $[0, b]$. Now, in essence one proves that the binary representation of x lies within the interval $[0, 2^k - 1]$. This can be done by committing to each x_i using an Okamoto commitment [29] (essentially a Pedersen bit commitment) along with a proof that this commitment hides either 0 or 1 and demonstrating that for commitments to x and all x_i's it holds that $x = x_0 2^0 + x_1 2^1 + \ldots + x_{k-1} 2^{k-1}$. The concrete range proof is a Σ-protocol for a proof of knowledge

$$PK\{(\alpha_0, \ldots, \alpha_{k-1}) : \bigwedge_{i=0}^{k-1} (C_i = h^{\alpha_i} \vee C_i g^{-1} = h^{\alpha_i})\}$$

or $PK\{(\alpha, \beta) : C = g^\alpha h^\beta \wedge\ (0 \leq \alpha \leq b)\}$ for short.

3.4 Camenisch-Lysyanskaya Signature Scheme

Camenisch and Lysyanskaya have proposed a signature scheme in [13] which satisfies the usual correctness and unforgeability properties of digital signatures and is provably secure under the LRSW assumption for groups with bilinear maps, which implies that the DLP is hard (cf. [13]). We present the CL signature scheme below:
Key Generation. Let $\mathbb{G}$ and $\mathbb{G}_t$ be groups of prime order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$ a bilinear map. Choose $x, y, z_1, \ldots, z_l \in_R \mathbb{Z}_p$. The private key is $sk = (x, y, \{z_i\})$ and the public key is $pk = (X, Y, \{Z_i\}, e, g, \mathbb{G}, \mathbb{G}_t, p)$, whereas $X = g^x$, $Y = g^y$ and $Z_i = g^{z_i}$.
Signing. On input message $(m_0, \ldots, m_l)$, sk and pk, choose $a \in_R \mathbb{G}$, compute $A_i = a^{z_i}$, $b = a^y$, $B_i = (A_i)^y$ and $c = a^{x+xym_0} \prod_{i=1}^{l} A_i^{xym_i}$. Output the signature $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$.
Verification. On input of $(m_0, \ldots, m_l)$, pk and $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$ check whether

- A_i's are formed correct: $e(a, Z_i) = e(g, A_i)$
- b and B_i's are formed correct: $e(a, Y) = e(g, b)$ and $e(A_i, Y) = e(g, B_i)$
- c is formed correct: $e(X, a) \cdot e(X, b)^{m_0} \prod_{i=1}^{l} e(X, B_i)^{m_i} = e(g, c)$

What makes this signature scheme particularly attractive is that it allows a receiver to obtain a signature on committed messages (using Pedersen commitments), while the messages are information-theoretically hidden from the signer (messages here means elements of the message tuple). Additionally, the receiver can randomize a CL signature such that the resulting signature is unlinkable to the original signature. Furthermore, receivers can use efficient zero-knowledge proofs to prove knowledge of a signature on committed messages. We will elaborate on the aforementioned functionalities more detailed in Section 4.1 and will show how to extend this functionality to interactive updates of signatures, the signed commitments and messages respectively.

4 Scheme

In this section we present our scheme along with an optional modification in order to increase the privacy in some settings even further. We start with the presentation of an important observation of CL signatures which is central to our constructions. Then, we first give a high level description followed by a detailed description of the schemes. Additionally, we present an performance evaluation of a prototypical implementation which supports the efficiency of the schemes. Finally, we present some extensions as well as system issues.

4.1 Interactive Update of Signed Messages

As already noted, CL signatures allow signing of committed messages (using Pedersen commitments), while the signer does not learn anything about them. Assume that the signer holds a private key $sk = (x, y, z)$ and publishes the corresponding public key $pk = (X, Y, Z, e, g, \mathbb{G}, \mathbb{G}_t, p)$.
Blind signing. If a receiver wants to obtain a blind signature for message m, he chooses $r \in_R \mathbb{Z}_p$, computes a commitment $C = g^m Z^r$ and sends C along with a signature of knowledge $SPK\{(\alpha, \beta) : C = g^\alpha Z^\beta\}$ to the signer (the ability to open the commitment is necessary for the security of the scheme, cf. [13]). If the verification of the proof holds, the signer computes a signature $\sigma = (a, A, b, B, c)$ for the commitment C by choosing $k \in_R \mathbb{Z}_p$, setting $a = g^k$ and computing $\sigma = (a, a^z, a^y, a^{yz}, a^x C^{kxy})$ and sends σ to the receiver.
Verification. In order to show the signature to a verifier, the receiver randomizes the signature by choosing $\mathsf{r}, \mathsf{r}' \in_R \mathbb{Z}_p$ and computing $\sigma' = (a', A', b', B', c')$ as $\sigma' = (a^{\mathsf{r}}, A^{\mathsf{r}}, b^{\mathsf{r}}, B^{\mathsf{r}}, c^{\mathsf{r}\mathsf{r}'})$ and sends σ' with the message m along with a signature of knowledge $SPK\{(\gamma, \delta) : v_\sigma^\gamma = \mathbf{v} v_r^\delta\}$ to the verifier. Therefore, both need to compute $v_\sigma = e(c', g)$, $\mathbf{v} = e(X, a') \cdot e(X, b')^m$ and $v_r = e(X, B')$. The verifier checks the proof and checks whether A' as well as b' and B' were correctly formed. Note, that the proof can be conducted by means of a standard DL-representation proof [17], which can easily be seen by rewriting the proof as $SPK\{(\gamma, \delta) : \mathbf{v} = v_\sigma^\gamma (v_r^{-1})^\delta\}$.
Remark. Observe, that we can realize a concept which is similar to partially blind signatures. However, in contrast to existing partially blind signature schemes [1], where the signer can integrate some common agreed upon information in the signature, here, the signer *arithmetically*

adds a message to the "blinded message" (hidden in the commitment). Therefore, during the signing, the signer simply updates the commitment to $C' = Cg^{m_S}$ and uses C' instead of C during signing. The receiver then obtains a signature for message $m + m_S$, whereas m_S is determined by the signer and m is hidden from the signer.

Update. The interesting and from our point of view novel part is that a signer can use a somewhat related idea to "update" a randomized signature without showing the message. Assume that a receiver holds a randomized signature σ' for message (m', r) whereas $m' = m + m_S$ and wants the signer to update the signature such that it represents a signature for message $(m' + m'_S, r + 1)$. Since showing m', as within the signature above, would destroy the unlinkability due to both messages are known, the receiver can solely prove that he knows the message in zero knowledge and both can then interactively update the signature. Therefore in the verification the receiver provides a signature of knowledge $SPK\{(\alpha, \beta, \gamma) : v_\sigma^\alpha = \mathbf{v} v_{m'}^\beta v_r^\gamma\}$ to the verifier, whereas $v_\sigma = e(g, c')$, $\mathbf{v} = e(g, a')$, $v_{m'} = e(g, b')$ and $v_r = e(g, B')$, which convinces the signer that the receiver possesses a valid signature for unknown message (m', r). Then, for the update, i.e. to *add* m'_S it is sufficient for the signer to compute $\tilde{C}_{m'+m'_S} = a'^{m'_S} A'$ and send it to the receiver. The receiver computes $C_{m'+m'_S} = (\tilde{C}_{m'+m'_S})^{\mathsf{r}'}$ and provides a signature of knowledge $SPK\{(\alpha, \beta, \gamma) : v_\sigma^\alpha = \mathbf{v} v_{m'}^\beta v_r^\gamma \ \wedge \tilde{C}_{m'+m'_S} = (C_{m'+m'_S})^\alpha\}$. Note that this proof convinces the signer that the receiver has randomized the commitment of the signer using the same random factor (r') as within the randomization of the signature. Then, the signer computes the updated signature $\sigma'' = (a^{\tilde{\mathsf{r}}}, A''^{\tilde{\mathsf{r}}}, b'^{\tilde{\mathsf{r}}}, B''^{\tilde{\mathsf{r}}}, (c'(C_{m'+m'_S})^{xy})^{\tilde{\mathsf{r}}})$ for $\tilde{\mathsf{r}} \in \mathbb{Z}_p$ and gives $\sigma'' = (a'', A'', b'', B'', \tilde{c}'')$ to the receiver. The receiver sets $c'' = (\tilde{c}'')^{\mathsf{r}'^{-1}}$ and now holds a valid signature for message $(m' + m'_S, r + 1)$ which he can in turn randomize. Therefore, observer that in the signature tuple only the last element actually includes the messages and we have $c' = c^{\mathsf{r}\mathsf{r}'} = (a^x C'^{kxy})^{\mathsf{r}\mathsf{r}'} = (a^{x+xy(m'+zr)})^{\mathsf{r}\mathsf{r}'}$ and $(C_{m'+m'_S})^{xy} = (a^{xy(m'_S+z)})^{\mathsf{r}}$. By taking these results together we have a well formed signature component $c'' = (a^{x+xy(m'+m'_S+z(r+1))})^{\mathsf{r}\mathsf{r}'}$. The remaining elements of the signature are easy to verify for correctness.

Remark. This functionality can easily be extended to signatures on arbitrary tuples of messages, will be a building block for our scheme and may also be of independent interest. Note that issuing a new signature in every step without revealing the hidden messages would not work and thus we use this "update functionality".

4.2 High Level Description of the First Scheme

Before presenting the detailed protocols, we provide a high level description. The aim of our construction is to let the user solely prove in each `Consume` protocol that enough storage space is available. In this setting, the user does not provide any useful information about the actual consumed space to the verifier, but the verifier learns only the fact that the user is still allowed to consume storage space.

ProviderSetup. The cloud provider generates a key-pair (sk, pk) for the CL signature scheme, publishes pk, initializes an empty blacklist BL and fixes a set $\mathcal{L} = \{L_1, \ldots, L_n\}$ of space limits.

ObtainLimit. A user chooses a limit $L \in \mathcal{L}$ and obtains a CL signature σ_t for a token $t = (C(id), C(s), L)$, whereas the initially consumed storage space is set to be $s = 1$.

Consume. Assume that the user holds a token signature pair $t = ((C(id), C(s), L), \sigma_t)$. Note, that id (the token-id) and s were signed as commitments and thus the signer is not aware of these values. When a user wants to integrate a data object d, the user computes $C(id')$ for the new token, randomizes the signature σ_t to σ'_t and proves that σ'_t is a valid signature for id and L (by revealing these two elements) and an unknown value s that satisfies $(s + |d|) \in [0, L]$ or equivalently $s \in [0, L - |d|]$, i.e. when integrating the new data object d the user needs to prove that after adding of $|d|$ space units at most L storage space will be consumed. If id is not contained in BL and this proof succeeds, the signature will be updated to a signature for $C(id + id')$, $C(s + |d|)$ and L. Consequently, the provider adds id to BL and the user obtains an updated signature for a token $t' = (C(id + id'), C(s + |d|), L)$. Otherwise, the cloud provider

will reject the integration of a new data object.

Reclaim. Assume that the user holds a token signature pair $t = ((C(id), C(s), L), \sigma_t)$. When a user wants to delete a data object d, as above, the user computes $C(id')$ for the new token, randomizes the signature σ_t to σ_t' and proves that he is allowed to delete d and that σ_t' is a valid signature for id and L (by revealing these two elements). If id is not contained in BL and the signature is valid, the user obtains a signature for a token $t' = (C(id + id'), C(s - |d|), L)$. Otherwise, the cloud provider will reject to delete d.

4.3 Detailed Description of the First Scheme

Subsequently, we provide a more detailed description of our protocols providing the technical details.

ProviderSetup: The cloud provider generates a key-pair for the CL signature scheme to sign tokens of the form $t = (id, s, L)$. More precisely, the cloud provider signs tokens of the form $t = (id, r_{id}, s, r_s, L)$, but we usually omit the randomizers for the ease of presentation. Consequently, the cloud provider obtains the private key $sk = (x, y, z_1, z_2, z_3, z_4)$ and publishes the public key $pk = (X, Y, Z_1, Z_2, Z_3, Z_4, e, g, \mathbb{G}, \mathbb{G}_t, p)$. Furthermore, he initializes an empty blacklist BL and fixes a set $\mathcal{L} = \{L_1, \ldots, L_n\}$ of limits that can be obtained by users.

ObtainLimit: A user registers with the cloud provider and obtains a space limit $L_i \in \mathcal{L}$ (we do not fix any concrete protocol for this task here since no anonymity is required). After the user has registered and both have agreed on the value L_i (which we denote as L below for simplicity), they proceed as depicted in Protocol 1.

1. The user chooses a token-identifier $id \in_R \{0,1\}^{l_{id}}$ and randomizers $r_{id}, r_s \in_R \mathbb{Z}_p$ for the commitments and we let the user start with value $s = 1$. Then, he computes the commitments $C_{id} = g^{id} Z_1^{r_{id}}$ and $C_s = Z_2^s Z_3^{r_s}$ and sends them along with a signature of knowledge

$$SPK\{(\alpha, \beta, \gamma) : \; C_{id} = g^{\alpha} Z_1^{\beta} \wedge C_s = Z_2 Z_3^{\gamma}\} \tag{1}$$

   to prove the ability to open the commitments, whereas the second part in the proof also convinces the cloud provider that $s = 1$.
2. If the verification of the signature of knowledge in (1) holds, the cloud provider computes a CL signature for (C_{id}, C_s, L) as follows: He chooses $k \in_R \mathbb{Z}_p$, computes $a = g^k$, $b = a^y$, $A_i = a^{z_i}$, $B_i = A_i^y$ for $1 \leq i \leq 4$ and $c = a^x (C_{id} C_s Z_4^L)^{kxy}$ and sends $\sigma = (a, \{A_i\}, b, \{B_i\}, c)$ to the user.
3. The user verifies whether the signature is valid and if this holds the user is in possession of a valid signature σ for a token $t = (id, s, L)$, whereas the cloud provider is not aware of id and knows that $s = 1$. Furthermore, the user locally randomizes the signature σ to $\sigma' = (a', \{A_i'\}, b', \{B_i'\}, c')$ by choosing $\mathsf{r}, \mathsf{r}' \in \mathbb{Z}_p$ and computing $\sigma' = (a^{\mathsf{r}}, \{A_i^{\mathsf{r}}\}, b^{\mathsf{r}}, \{B_i^{\mathsf{r}}\}, c^{\mathsf{r}\mathsf{r}'})$.

Remark. All further actions are fully anonymous and in practice also unlinkable, since we can assume that one limit will be issued to a quite large number of users (and the limit is the only information that could potentially be used for linking)!

Prot. 1: The `ObtainLimit` protocol.

Consume: A user holds a randomized signature $\sigma' = (a', \{A_i'\}, b', \{B_i'\}, c')$ for a token $t = (id, s, L)$ and wants to integrate a data object d. The protocol to integrate a data object and obtain a new token is depicted in Protocol 2.

1. The user sends the randomized signature σ', the "visible part" (id, L) of the token t and a data object d along with a signature of knowledge

$$SPK\{(\alpha, \beta, \gamma, \delta) : v_\sigma^\alpha = \mathbf{v} v_{r_{id}}^\beta v_s^\gamma v_{r_s}^\delta \ \wedge \ (0 \leq \gamma \leq 2^{l_L - l_{|d|}} - 1)\} \tag{2}$$

for the validity of the randomized signature containing a proof that still enough space is available to the cloud provider. It must be noted, that the presentation of the proof in (2) represents a shorthand notation for the signature of knowledge

$$SPK\{(\alpha, \beta, \gamma, \delta, \epsilon, \epsilon_1, \ldots, \epsilon_k, \zeta, \zeta_1, \ldots, \zeta_k) : \ \mathbf{v} = v_\sigma^\alpha (v_{r_{id}}^{-1})^\beta (v_s^{-1})^\gamma (v_{r_s}^{-1})^\delta \ \wedge$$
$$C = g^\beta Z_1'^\zeta \ \wedge$$
$$C = \prod_{i=1}^{k} (g^{\epsilon_i} Z_1'^{\zeta_i})^{2^{i-1}} \ \wedge$$
$$\bigwedge_{i=1}^{k} (C_i = Z_1^{\zeta_i} \vee C_i g^{-1} = Z_1'^{\zeta_i})\}$$

Essentially, besides the DL-representation proof for the validity of the randomized signature, we use an additional commitment $C = g^s Z_1'^r$ to the value s with a new randomizer r computed as

$$r = r_1 2^0 + r_2 2^1 + \ldots + r_k 2^{k-1} \ \text{MOD} \ p$$

for r_i's chosen uniformly at random from $\mathbb{Z}_p$ and the single commitments for the range proof are $C_i = g^{s_i} Z_1'^{r_i}$. It also must be mentioned, that k represents $l_L - l_{|d|}$, the binary length of $L - |d|$. Furthermore, note that in case of $s = 1$, i.e. in the first execution of the `Consume` protocol, it would not be necessary to provide a range proof. However, when performing a range proof, the initial `Consume` protocol is indistinguishable from other protocol executions and thus provides stronger privacy guarantees.
2. The cloud provider checks whether $id \notin BL$. If id is not blacklisted, the cloud provider verifies the validity of the signature for the part (id, L) of the token t. Therefore, the cloud provider locally computes the values

$$v_\sigma = e(g, c'),\ v_{r_{id}} = e(X, B_1'),\ v_s = e(X, B_2'),\ v_{r_s} = e(X, B_3') \text{ and}$$

$$\mathbf{v} = e(X, a') \cdot e(X, b')^{id} \cdot e(X, B_4')^L$$

from pk, (id, L) and σ' and verifies the signature of knowledge (2) Additionally, he checks whether the A_i''s as well as b' and B_i''s are correctly formed.
3. A positive verification convinces the cloud provider that enough storage space is available to integrate d and a signature for an updated token t' can be computed in cooperation with the user as follows: Firstly, we need an observation regarding the signature σ'. Note, that the only element of the signature that depends on the message is c', which can be rewritten as

$$c' = (a^{x+xy(id+z_1 r_{id}+z_2 s+z_3 r_s+z_4 L)})^{\mathsf{r}'} = (a^{x+xyid} A_1^{xyr_{id}} A_2^{xys} A_3^{xyr_s} A_4^{xyL})^{\mathsf{r}'}$$

and in order to update a signature for the id-part (to construct a new id for the new token) it is sufficient to update a and A_1. To update the s-part, which amounts to update the currently consumed space, it is sufficient to update A_2 and A_3. The latter update needs to be computed by the cloud provider to be sure that the correct value $|d|$ is integrated and the former one needs to be computed by the user to prevent the cloud provider from learning the new token identifier. Hence, the cloud provider computes $\tilde{C}_{s+|d|} = A_2'^{|d|} A_3'$ and sends $\tilde{C}_{s+|d|}$ to the user, who verifies whether $|d|$ has been used to update the commitment. The user in turn chooses a new identifier and randomizer $id', r_{id'} \in_R \mathbb{Z}_p$, computes $C_{id+id'} = (a'^{id'} A_1'^{r_{id'}})^{\mathsf{r}'}$, $C_{s+|d|} = (\tilde{C}_{s+|d|})^v = (A_2'^{|d|} A_3')^{\mathsf{r}'}$ and sends $(C_{id+id'}, C_{s+|d|})$ along with a signature of knowledge:

$$SPK\{(\epsilon, \zeta, \eta, \phi, \iota, \kappa) : C_{id+id'} = a'^\epsilon A_1'^\zeta \ \wedge$$
$$\tilde{C}_{s+|d|} = (C_{s+|d|})^\eta \ \wedge \ \mathbf{v} = v_\sigma^\eta (v_{r_{id}}^{-1})^\phi (v_s^{-1})^\iota (v_{r_s}^{-1})^\kappa\}$$

to the cloud provider. Note, that the user additionally to the knowledge of the ability to open the commitments proves that he has randomized the commitment $\tilde{C}_{s+|d|}$ to a commitment $C_{s+|d|}$ using the same randomization factor (r') as used to randomize the signature σ without revealing this value. After positive verification of this signature of knowledge, the cloud provider chooses $\tilde{\mathsf{r}} \in_R \mathbb{Z}_p$ and computes an updated signature

$$\sigma'' = (a'^{\tilde{\mathsf{r}}}, \{A_i'^{\tilde{\mathsf{r}}}\}, b'^{\tilde{\mathsf{r}}}, \{B_i'^{\tilde{\mathsf{r}}}\}, (c'(C_{id+id'} C_{s+|d|})^{xy})^{\tilde{\mathsf{r}}}) \tag{3}$$

and sends this updated signature $\sigma'' = (a'', \{A_i''\}, b'', \{B_i''\}, \tilde{c}'')$ to the user. The user sets $c'' = (\tilde{c}'')^{\mathsf{r}'^{-1}}$ and obtains a valid signature for a token $t' = (id + id', s + |d|, L)$ or more precisely a token $t' = (id + id', r_{id} + r_{id'}, s + |d|, r_s + 1, L)$, which he verifies for correctness (in Appendix A we show that σ'' is indeed a valid signature). Consequently, the user can randomize σ'' and run a new `Consume` protocol for a data object d' with token $t' = (id + id', s + |d|, L)$.

Prot. 2: The `Consume` protocol.

Reclaim: Reclaiming resources, i.e. deleting a data object, is achieved by a slight adaption of the `Consume` protocol. In step 1, instead of the SPK (2) the user provides the subsequent

signature of knowledge (the proofs that enough space is available is not necessary)

$$SPK\{(\alpha,\beta,\gamma,\delta) : v_\sigma^\alpha = \mathbf{v} v_{r_{id}}^\beta v_s^\gamma v_{r_s}^\delta\}$$

And in step 3, the cloud provider computes $\tilde{C}_{s-|d|} = A_2'^{p-|d|} A_3'$ instead of $\tilde{C}_{s+|d|} = A_2'^{|d|} A_3'$.

Remark. As we have already mentioned, a cloud provider should only perform a `Reclaim` protocol if the user is able to prove the possession of the data object d (and we may assume that only owners delete their data objects). It is not the focus of this paper to provide a solution to this task. However, a quite straightforward solution would be to commit to some secret value for every data object and the cloud provider requires a user to open the commitment or prove knowledge that he is able to open the commitment to delete a data object. This problem is somewhat similar to what Halevi et al. recently denoted as proofs of ownership (PoWs) [32]. However, their focus although also in the cloud storage setting is (client-side) deduplication, i.e. storing exactly the same file only once thus avoiding unnecessary copies of repeating data. This is usually achieved by sending a hash of a file and the cloud checks if this hash is already registered. In order to avoid security problems in this setting, they employ PoWs which require users to prove that a user actually holds a file. This is somewhat similar but diametric to proofs of data possession (PDPs) [2] and proofs of retrievability (PORs) [31] respectively.

4.4 A Modified Scheme (Scheme 2) Providing Even more Privacy for Users

In order to increase privacy further, it may be desirable that the initially issued limit L is hidden from the CP during `Consume` or `Reclaim` protocols. We, however, note that if the number of initial tokens associated to CP-defined limits in $\mathcal{L}$ is huge, the respective anonymity sets may be of reasonable size for practical application and this adaption may not be necessary. Nevertheless, we provide an adaption of our protocols which removes the necessity to include L, does only include the available amount of resources (denoted as s) and hides this value s from the CP during any further interactions. We present the modification below:

ProviderSetup. Now, tokens are of the form $t = (id, r_{id}, s, r_s)$ and thus the private key is $sk = (x, y, z_1, z_2, z_3)$ and the public key is $pk = (X, Y, Z_1, Z_2, Z_3, e, g, \mathbb{G}, \mathbb{G}_t, p)$.

ObtainLimit. The user computes commitments $C_{id} = g^{id} Z_1^{r_{id}}$ and $C_s = Z_3^{r_s}$ and provides $SPK\{(\alpha,\beta,\gamma) : C_{id} = g^\alpha Z_1^\beta \wedge C_s = Z_3^\gamma\}$. The element c of the signature is now computed by the CP as $c = a^x (C_{id} C_s Z_2^L)^{kxy}$ and the user can randomize this signature for token $t = (id, r_{id}, L, r_s)$ as usual.

Consume. Here the user only provides id of the actual token and a signature of knowledge

$$SPK\{(\alpha,\beta,\gamma,\delta) : v_\sigma^\alpha = \mathbf{v} v_{r_{id}}^\beta v_s^\gamma v_{r_s}^\delta \wedge \ (2^{l_{|d|}} - 1 \leq \gamma \leq 2^{l_\mathbf{L}} - 1)\}$$

In this setting $\mathbf{L}$ does not represent a user-specific limit but *the maximum* of all issued limits, whereas this proof convinces the CP that enough resources to integrate d are still available (note that the local computations of the CP for the verification of the signature in step 2 have to be adapted, which is however straightforward). In step 3, the update of the signature remains identical to the first scheme with the exception that the CP computes the commitment as $\tilde{C}_{s-|d|} = A_2'^{p-|d|} A_3'$, which updates the remaining resources, e.g. in the first run of the `Consume` protocol to $s := L - |d|$.

Reclaim. The reclaim protocol remains identical to the first scheme with the exception that $\tilde{C}_{s+|d|} = A_2'^{|d|} A_3'$.

4.5 Performance Evaluation

In this section we provide a performance evaluation of our first scheme. We have implemented the user's and the cloud provider's parts of the protocols in Java using the `jPBC`[1] library version 1.2.0 written by Angelo De Caro. This library provides a Java porting of as well as a Java wrapper

[1] `http://libeccio.dia.unisa.it/projects/jpbc/`

for the Pairing-Based Cryptography Library (PBC)[2] developed by Ben Lynn in C. In particular, we have used the Java PBC wrapper which calls the PBC C library and is significantly faster than the pure Java implementation. All our experiments were performed on an Intel Core 2 duo running at 2.6 GHz with 3GB RAM on Linux Ubuntu 10.10.

As the cryptographic setting we have chosen a symmetric pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_t$ constructed on the supersingular elliptic curve $y^2 = x^3 + x$ over a prime field $\mathbb{F}_q$ where $|q| = 512$ bits and $q \equiv 3 \pmod 4$. The group $\mathbb{G}$ represents a subgroup of $E(\mathbb{F}_q)$ of order $r = 160$ bits. The embedding degree is $k = 2$ and thus $\mathbb{G}_t$ is a subgroup of $\mathbb{F}_{q^2}$ and with our choice of the parameters we obtain a DL security of 1024 bit. For the non-interactive proofs of knowledge we have used the SHA-256 hash function.

Experiments. Our setting for the experiments is as follows: For the computational performance we have taken the average over 100 experiments, with limits $L = 10^i$, $i = 3, \ldots, 9$ each. Thereby, within every of the 100 experiments per limit, the user has conducted 10 `Consume` as well as 10 `Reclaim` operations with $|d|$ sampled uniformly at random from $[1, 10^{i-2}]$. Figure 1 presents the performance of the `ObtainLimit`, the `Consume` and the `Reclaim` protocols from a computational and bandwidth perspective, whereas point compression for elements in $\mathbb{G}$ is used to reduce the bandwidth consumption. As one can see, all protocols are highly efficient from the user's as

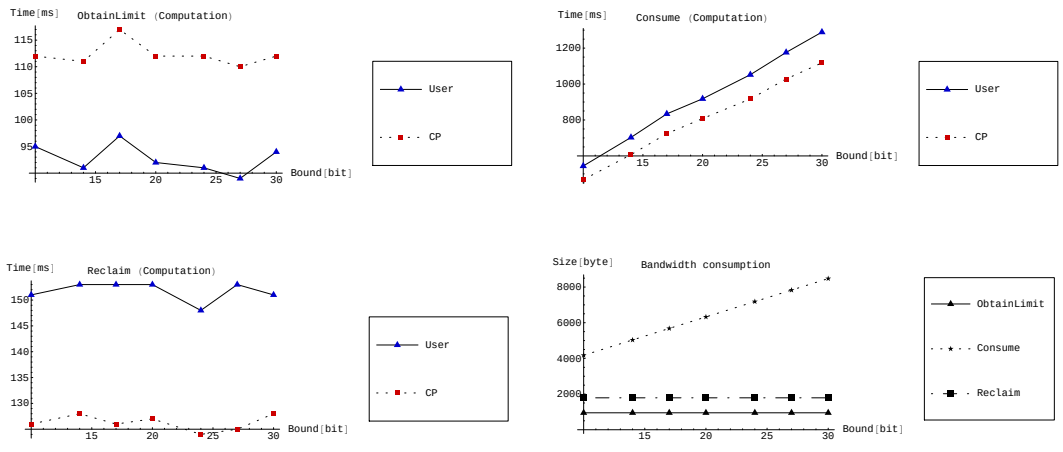


Fig. 1. Experimental results from a Java implementation of our first scheme.

well as the cloud provider's perspective, both in the computational effort and the bandwdith consumption. This holds, although the code has not been optimized for performance and pre-computations have not been used. Hence, our evaluation shows that from the efficiency point of view our protocols are entirely practical.

4.6 Extensions and System Issues

Below, we present two extensions of our schemes which seem to be valuable when deploying them for practical applications.

Limited validity. One could rightly argue that in a large scale cloud the double spending detection of token identifiers using a database (blacklist) does not scale well. In order to overcome this limitation, we can extend our schemes such that a resource limit associated to a token only has a limited validity. Then, before the validity ends a user has to provide the actual token, i.e. the identifier and the available resources (either s and L or solely s in the second scheme) along with the corresponding signature. Then the user runs a new `ObtainLimit` protocol with the CP. Note that in case of the first scheme users should not end up with a new limit L which is very likely to be unique and thus users should take one of the predefined limits. We now sketch how

[2] `http://crypto.stanford.edu/pbc/`

this adaption for the first scheme looks like (for the second one it is analogous): The keys of the CP are adapted such that the public key is $pk = (X, Y, Z_1, Z_2, Z_3, Z_4, Z_5, Z_6, e, g, \mathbb{G}, \mathbb{G}_t, p)$. Token are augmented by elements (V, r_V) which represent the validity period, e.g. an encoding in Unix time. In the `ObtainLimit` protocol the user additionally computes $Z_6^{r_V}$ (and proves knowledge of this DL) and the c part of the signature is adapted to $c = a^x(C_{id}C_s Z_4^L Z_5^V Z_6^{r_V})$ whereas the CP here integrates the validity V. The remainig ideas stay the same with exception that in the `Consume` protocol, the SPK needs to be adapted to

$$SPK\{(\alpha, \beta, \gamma, \delta, \epsilon, \zeta) : v_\sigma^\alpha = \mathbf{v} v_{r_{id}}^\beta v_s^\gamma v_{r_s}^\delta v_V^\epsilon v_{r_V}^\zeta \wedge$$
$$(0 \leq \gamma \leq 2^{l_L - l_{|d|}} - 1) \wedge (2^{l_{\mathtt{time}}} - 1 \leq \epsilon \leq 2^{l_{\mathbf{p}}} - 1)\}$$

whereas $\mathbf{p}$ represents the maximum validity period and `time` the representation of the actual date and time (in the `Reclaim` we only need the second range proof). For the update of the signature and the token respectively, the user has additionally to compute $C_V = (A'_5 A'^{r'_V}_6)^{\mathsf{r}'}$ and augment the prove of knowledge in step 3 of Protocol 2 to

$$SPK\{(\zeta, \eta, \phi, \iota, \kappa, \lambda, \mu, \nu, \xi) : C_{id+id'} = a'^\zeta A_1'^\eta \ \wedge C_V = A_5'^\phi A_6'^\iota \ \wedge$$
$$\tilde{C}_{s+|d|} = (C_{s+|d|})^\phi \ \wedge \ \mathbf{v} = v_\sigma^\phi (v_{r_{id}}^{-1})^\kappa (v_s^{-1})^\lambda (v_{r_s}^{-1})^\mu (v_V^{-1})^\nu (v_{r_V}^{-1})^\xi\}$$

Note that these modifications do influence the overall performance of the `Consume` protocol approximately by a factor of two, which though performs very good in practice when compared with our experimental results.

Elasticity. Clouds extremely benefit from users being able to request resources "on the fly". In our first scheme this can only be achieved by means of requesting additional tokens, i.e. running additional `ObtainLimt` protocols for the required resource, and users have then to manage a list of tokens. Although this seems to be an issue which can be handled in practical applications, the second scheme (modification to hide the limit L) allows for such updates. Therefore we can simply use the `Reclaim` protocol of Section 4.4 (we may denote it as `Recharge` in this case), whereas $|d|$ is simply replaced by the amount of resources to be extended.

5 Conclusion

In this paper we have investigated the problem of anonymous yet authorized and bounded use of cloud resources, which means that users should be able to register and obtain a resource limit from a cloud provider such that this limit determines how much of a resource, e.g. CPU time, storage space, a user is allowed to consume. Then, users should be able to consume (or reclaim) their resources in an anonymous and unlinkable fashion, but the ability of users to consume resources should be constrained by their issued limit. We have presented a scheme, it's modification providing even more privacy, have presented extensions valuable for practical application and have supported the efficiency of the proposed scheme by a performance analysis based on a prototypical implementation. Concluding we present anonymity revocation as an open problem and then we briefly discuss future work.

Anonymity revocation. It is not clear to us how anonymity revocation could be suitably realized in this setting. We argue that it does not seem to be meaningful to use identity escrow within every transaction, i.e. to employ techniques known from group signatures (which could though be realized using CL signatures). It is absolutely not clear who would have the power to perform anonymity revocation. In contrast, if at all, it seems more suitable to employ techniques like used within e-cash [11] or (n-times) anonymous authentication [11, 21]. Mapped to our scenario this would mean that the identity of an anonymous user is solely revealed if the user tries to consume more resources than allowed (although this is prevented by the protocols). However, it is not clear to us how to achieve this, since in the aforementioned approaches spend protocols or authentications are atomic and in our setting we do not know in advance how often a user will consume or reclaim resources. We leave this functionality as an open problem for future work.

In order to gain more insights into system issues and to gather experience on limitations in practical use we are working on the integration of our schemes with the full functionality (all extensions) into the Eucalyptus cloud[3] using Amazon's S3 storage service to.

References

1. Abe, M., Okamoto, T.: Provably Secure Partially Blind Signatures. In: CRYPTO '00. LNCS, vol. 1880, pp. 271–286. Springer (2000)
2. Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.X.: Provable Data Possession at Untrusted Stores. In: 14th ACM Conference on Computer and Communications Security, CCS 2007. pp. 598–609. ACM (2007)
3. Au, M.H., Susilo, W., Mu, Y.: Practical Anonymous Divisible E-Cash from Bounded Accumulators. In: Financial Cryptography and Data Security 2008. LNCS, vol. 5143, pp. 287–301. Springer (2008)
4. Backes, M., Camenisch, J., Sommer, D.: Anonymous Yet Accountable Access Control. In: ACM Workshop on Privacy in the Electronic Society, WPES '05. pp. 40–46. ACM (2005)
5. Balasch, J., Rial, A., Troncoso, C., Preneel, B., Verbauwhede, I., Geuens, C.: PrETP: Privacy-Preserving Electronic Toll Pricing. In: 19th USENIX Security Symposium. pp. 63–78. USENIX Association (2010)
6. Blanton, M.: Online Subscriptions with Anonymous Access. In: 3rd ACM Symposium on Information, Computer and Communications Security, ASIACCS '08. pp. 217–227. ACM (2008)
7. Boudot, F.: Efficient Proofs that a Committed Number Lies in an Interval. In: EUROCRYPT 2000. LNCS, vol. 1807, pp. 431–444. Springer (2000)
8. Camenisch, J., Dubovitskaya, M., Neven, G.: Oblivious Transfer with Access Control. In: 16th ACM Conference on Computer and Communications Security, CCS '09. pp. 131–140. ACM (2009)
9. Camenisch, J., Dubovitskaya, M., Neven, G.: Unlinkable Priced Oblivious Transfer with Rechargeable Wallets. In: 14th International Conference on Financial Cryptography and Data Security, FC 2010. LNCS, vol. 6052, pp. 66–81. Springer (2010)
10. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to Win the Clone Wars: Efficient Periodic n-Times Anonymous Authentication. In: 13th ACM Conference on Computer and Communications Security, CCS '06. pp. 201–210. ACM (2006)
11. Camenisch, J., Hohenberger, S., Lysyanskaya, A.: Compact E-Cash. In: EUROCRYPT '05. LNCS, vol. 3494, pp. 302–321. Springer-Verlag (2005)
12. Camenisch, J., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Security in Communication Networks, SCN' 02. LNCS, vol. 2576, pp. 268–289. Springer (2002)
13. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer (2004)
14. Camenisch, J., Stadler, M.: Efficient Group Signature Schemes for Large Groups (Extended Abstract). In: CRYPTO '97. LNCS, vol. 1294, pp. 410–424. Springer (1997)
15. Canard, S., Gouget, A.: Divisible E-Cash Systems Can Be Truly Anonymous. In: EUROCRYPT 2007. LNCS, vol. 4515, pp. 482–497. Springer (2007)
16. Canard, S., Gouget, A., Hufschmitt, E.: A Handy Multi-coupon System. In: Applied Cryptography and Network Security 2006. LNCS, vol. 3989, pp. 66–81. Springer (2006)
17. Chaum, D., Evertse, J.H., van de Graaf, J.: An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In: EUROCRYPT '87. LNCS, vol. 304, pp. 127–141. Springer (1987)
18. Chen, L., Escalante, A.N., Löhr, H., Manulis, M., Sadeghi, A.R.: A Privacy-Protecting Multi-Coupon Scheme with Stronger Protection Against Splitting. In: Financial Cryptography and Data Security 2007. LNCS, vol. 4886, pp. 29–44. Springer (2007)
19. 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)
20. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols. In: CRYPTO '94. LNCS, vol. 839, pp. 174–187. Springer (1994)
21. Damgård, I., Dupont, K., Pedersen, M.Ø.: Unclonable Group Identification. In: EUROCRYPT '06. LNCS, vol. 4004, pp. 555–572. Springer (2006)
22. Fiat, A., Shamir, A.: How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In: CRYPTO '86. LNCS, vol. 263, pp. 186–194. Springer (1987)
23. Fujisaki, E., Okamoto, T.: Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations. In: CRYPTO '97. LNCS, vol. 1294, pp. 16–30. Springer (1997)

[3] `http://open.eucalyptus.com/`

24. Gentry, C.: Fully Homomorphic Encryption using Ideal Lattices. In: 41st Annual ACM Symposium on Theory of Computing, STOC 2009. pp. 169–178 (2009)
25. Gentry, C., Halevi, S.: Implementing Gentry's Fully-Homomorphic Encryption Scheme. Cryptology ePrint Archive, Report 2010/520 (2010), http://eprint.iacr.org/
26. Kamara, S., Lauter, K.: Cryptographic Cloud Storage. In: Financial Cryptography Workshops 2010. LNCS, vol. 6054, pp. 136–149. Springer (2010)
27. Mao, W.: Guaranteed Correct Sharing of Integer Factorization with Off-Line Shareholders. In: Public Key Cryptography 1998. LNCS, vol. 1431, pp. 60–71. Springer (1998)
28. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: CRYPTO '92. LNCS, vol. 740, pp. 31–53. Springer (1992)
29. Okamoto, T.: An Efficient Divisible Electronic Cash Scheme. In: CRYPTO '95. LNCS, vol. 963, pp. 438–451. Springer (1995)
30. Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: CRYPTO '91. LNCS, vol. 576, pp. 129–140. Springer (1992)
31. Shacham, H., Waters, B.: Compact Proofs of Retrievability. In: ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer (2008)
32. Shai Halevi, Danny Harnik, B.P.A.S.P.: Proofs of Ownership in Remote Storage Systems. Cryptology ePrint Archive, Report 2011/207 (2011), http://eprint.iacr.org/

A Correctness of Update and Randomization in Consume

We need to show, that $\sigma'' = (a'', \{A_i''\}, b'', \{B_i''\}, c'')$ is a valid signature for $t' = (id + id', r_{id} + r_{id'}, s + |d|, r_s + 1, L)$. Therefore, we firstly take a closer look at the signature component c'' and then show that the signature verification for σ'' works and consequently σ'' represents a valid CL signature.

Regarding the component c'', observe that we have

$$c = a^{x}(C_{id}C_s Z_4^L)^{kxy} = a^{x}(g^{id}Z_1^{r_{id}}Z_2^{s}Z_3^{r_s}Z_4^{L})^{kxy} = a^{x+xy(id+z_1 r_{id}+z_2 s+z_3 r_s+z_4 L)}$$

and

$$c' = c^{\mathsf{r}\mathsf{r}'} = (a^{x+xy(id+z_1 r_{id}+z_2 s+z_3 r_s+z_4 L)})^{\mathsf{r}\mathsf{r}'}$$

as well as

$$(C_{id+id'}C_{s+|d|})^{xy} = ((a'^{id'}A_1'^{r_{id'}})^{\mathsf{r}'}(A_2'^{|d|}A_3')^{\mathsf{r}'})^{xy} = (a^{xy(id'+z_1 r_{id'}+z_2|d|+z_3)})^{\mathsf{r}\mathsf{r}'}$$

Thus, by taking these results together we obtain

$$\begin{aligned} c'' &= ((c'(C_{id+id'}C_{s+|d|})^{xy})^{\bar{\mathsf{r}}})^{\mathsf{r}'-1} = (((a^{x+xy(id+z_1 r_{id}+z_2 s+z_3 r_s+z_4 L)}a^{xy(id'+z_1 r_{id'}+z_2|d|+z_3)})^{\mathsf{r}\mathsf{r}'})^{\bar{\mathsf{r}}})^{\mathsf{r}'-1} \\ &= (a^{x+xy(id+id'+z_1(r_{id}+r_{id'})+z_2(s+|d|)+z_3(r_s+1)+z_4 L)})^{\mathsf{r}\bar{\mathsf{r}}} \end{aligned}$$

and we can write all components of σ'' similar to c''. More precisely, we can represent the remaining components as $(a^{\mathsf{r}\bar{\mathsf{r}}}, \{A_i^{\mathsf{r}\bar{\mathsf{r}}}\}, b^{\mathsf{r}\bar{\mathsf{r}}}, \{B_i^{\mathsf{r}\bar{\mathsf{r}}}\})$. Now, it remains to show that the verification relations for signature σ'' work. Recall, therefore we need to show that the A_i'''s, b'', B_i'''s and c'' are formed correctly:

- A_i'''s need to satisfy $e(a'', Z_i) = e(g, A_i'')$: This can easily be verified, since

$$e(a'', Z_i) = e(a^{\mathsf{r}\bar{\mathsf{r}}}, g^{z_i}) = e(g, g^{kz_i\mathsf{r}\bar{\mathsf{r}}}) = e(g, (a^{z_i})^{\mathsf{r}\bar{\mathsf{r}}}) = e(g, A_i^{\mathsf{r}\bar{\mathsf{r}}}) = e(g, A_i'')$$

- b needs to satisfy $e(a'', Y) = e(g, b'')$: This holds, since

$$e(a'', Y) = e(a^{\mathsf{r}\bar{\mathsf{r}}}, g^{y}) = e(g^{k\mathsf{r}\bar{\mathsf{r}}}, g^{y}) = e(g, g^{ky\mathsf{r}\bar{\mathsf{r}}}) = e(g, a^{y\mathsf{r}\bar{\mathsf{r}}}) = e(g, b^{\mathsf{r}\bar{\mathsf{r}}}) = e(g, b'')$$

- B_i'''s need to satisfy $e(A_i'', Y) = e(g, B_i'')$: This holds, since

$$e(A_i'', Y) = e(A_i^{\mathsf{r}\bar{\mathsf{r}}}, g^{y}) = e(g^{kz_i\mathsf{r}\bar{\mathsf{r}}}, g^{y}) = e(g, g^{kz_i y\mathsf{r}\bar{\mathsf{r}}}) = e(g, A_i^{y\mathsf{r}\bar{\mathsf{r}}}) = e(g, B_i^{\mathsf{r}\bar{\mathsf{r}}}) = e(g, B_i'')$$

- c'' needs to satisfy

$$e(X, a'') \cdot e(X, b'')^{id+id'} \cdot e(X, B_1'')^{r_{id}+r_{id'}} \cdot e(X, B_2'')^{s+|d|} \cdot e(X, B_3'')^{r_s+1} \cdot e(X, B_4'')^{L} = e(g, c'')$$

  which holds, since by rewriting the left hand side and using bilinearity we obtain

$$\begin{aligned} &e(g, a)^{\mathsf{r}\bar{\mathsf{r}}} \cdot e(g, a^{xy(id+id')})^{\mathsf{r}\bar{\mathsf{r}}} \cdot e(g, a^{xyz_1(r_{id}+r_{id'})})^{\mathsf{r}\bar{\mathsf{r}}} \cdot e(g, a^{xyz_2(s+|d|)})^{\mathsf{r}\bar{\mathsf{r}}} \cdot e(g, a^{xyz_3(r_s+1)})^{\mathsf{r}\bar{\mathsf{r}}} \cdot \\ &e(g, a^{xyz_4 L}) = e(g, a^{x+xy(id+id'+z_1(r_{id}+r_{id'})+z_2(s+|d|)+z_3(r_s+1)+z_4 L)})^{\mathsf{r}\bar{\mathsf{r}}} = e(g, c'') \end{aligned}$$

Targeted, Not Tracked: Client-side Solutions for Privacy-Friendly Behavioral Advertising

Mikhail Bilenko, Matthew Richardson, and Janice Y. Tsai

Microsoft Corporation
1 Microsoft Way, Redmond, WA 98052
{mbilenko,mattri,janice.tsai}@microsoft.com

Abstract. The current discussion of potential *Do Not Track* regulation for online advertising is worrisome for the advertising industry, as it may significantly limit the capability for targeted advertising, a key revenue source for online content. The present discourse conflates the behavior tracking and ad targeting processes, leading to the presumption that providing privacy must come at the cost of eliminating advertisers' targeting capability. This paper focuses on a family of methods that facilitate behavioral targeting while providing consumer privacy protections. This is achieved by differentiating between client-side and server-side tracking. Client-side solutions provide for mechanisms and policies that address the privacy concerns over lack of user control over data while providing advertising platforms with the ability to target users. We compare and contrast several client-side methods along several dimensions of user privacy, adoption effort, and trust. A novel client-side profiling method is proposed that differs from prior work in not requiring installation of additional software by the user and providing compatibility with existing ad serving infrastructure. Empirical evaluation of the method on large-scale real-world datasets demonstrates the potential for high targeting performance of client-side techniques. We hope that by considering such middle-ground approaches, the present debate will converge towards solutions that satisfy both advertisers' desire for targeting and users' desire for privacy.

1 Introduction

Privacy concerns related to online advertising have grown over the past several years among web users, as reflected in coverage of the issues surrounding "behavioral tracking" in the popular press (e.g. the Wall Street Journal's "What They Know" series [15]). These concerns have attracted government attention, resulting in regulatory proposals [2], workshops [1], as well as Congressional hearings and legislation. The combination of popular sentiment, media attention and government involvement is likely to result in action to protect consumer privacy in online advertising in the near future. A series of discussions that has taken place recently involving legislators, advertising and technology industries, privacy advocates, and regulatory bodies has resulted in the *Do Not Track* (DNT) solution gaining the greatest momentum.

DNT relies on incorporation of privacy-protecting features in web browsers that either discourage or block communication with third parties that perform behavioral advertising. Different DNT implementations proposed by browser manufacturers can be grouped into three categories: domain blocking, opt-out cookies, and HTTP headers. Domain blocking allows the user to specify domains which the browser should never contact. In contrast, with opt-out cookies and HTTP headers the browser contacts the target domain but informs it that the user wishes not to be tracked. These latter two solutions require the user to trust that the target domain will comply. Evaluating these DNT variants in the context of the regulatory framework put forth by the Federal Trade Commission (FTC) [2] demonstrates that none of them currently meet all criteria in their formal interpretation: each of the three strikes a different balance between ease-of-use, universality and enforceability. Crucially, DNT has so far also failed to win the endorsement of the online advertising industry, which continues to advocate for self-regulation being sufficient.

In this position paper, we argue that the current discussions of *Do Not Track* are hampered by the lack of clear distinction between *tracking* (collection and aggregation user behavior data) and *targeting* (use of this data during ad selection). Demarcation of the two processes is crucial, as they are increasingly being performed by multiple parties, whose interactions are increasingly non-trivial both technically and financially. Additionally, because any party performing either tracking or targeting can also be the content publisher (first party), policies that do not distinguish these differences are inherently ambiguous and hence ineffective.

Differentiating between tracking and targeting has significant implications for protecting both consumer and industry interests. To advertisers and ad platforms, tracking is only a means to an end of increasing advertising effectiveness, which is achieved by targeting. For users, there appears to be a gap in attitudes towards data collection and targeted advertising. Survey results reported by Hallerman [9] indicate that 55% of respondents are very or somewhat comfortable with ad targeting, while another survey by McDonald and Cranor [11] found that over two-thirds of respondents have agreed or strongly agreed that "someone keeping track of my activities online is invasive". While these results should be viewed in the context of the fact that most users lack fundamental understanding of how tracking and targeting work [10], they nonetheless indicate that the two processes are perceived differently.

There *is* a family of solutions that differentiates between tracking and targeting. In contrast to the existing DNT discussions and browser-based solutions, these *client-side tracking* proposals protect the privacy of users while still enabling advertisers to target ads to them. This makes such solutions attractive to both users and advertisers, and are an important category of solution that should be considered in any future discussions on DNT and behavioral targeting. Client-side tracking solutions store behavioral data on the user's machine, giving users complete control over their data, ensuring that their behavioral information can be edited or deleted permanently as needed. In addition to provid-

ing contrastive analysis of previously proposed client-side tracking mechanisms, this position paper describes a new approach, *Client-only Profiles(CoP)*, demonstrating how it can be implemented using a recently proposed machine learning method for constructing compact profiles [4]. Unlike previously proposed solutions, CoP does not require a user-installed browser plugin, and relies on very minor modifications to the existing advertising infrastructure. We argue that client-side profiling strikes a balance between user privacy, advertiser revenue, and user and advertiser adoption, and that future exploration in this direction can yield effective alternatives to the binary policies currently being discussed.

2 Targeting vs. Tracking

To consumers, businesses, and advertising content providers, the term **tracking** may have different connotations. For this paper, we adopt the definition put forth by the Center on Democracy and Technology in the context of online behavioral advertising [3]: "Tracking is the collection and correlation of data about the Internet activities of a particular user, computer, or device, over time and across non-commonly branded websites for any purpose other than fraud prevention or compliance with law enforcement requests." Tracking can be performed by the first party or a third party, where the first party is the functional entity with which a user reasonably expects to exchange data, while the third party is any other other functional entity.

It is crucial to distinguish between **tracking** and **targeting**: the former refers to the process of data *collection* and *processing*, while the latter focuses on the *use* of processed data for personalization in the context of a specific task, such as advertising. This distinction is effectively disregarded in the ongoing public debates. To some degree, the confusion is a reflection of the overall poor level of understanding of tracking technologies, behavioral advertising, and the risks of related information exchange [11]. However, the distinction is critical, because it represents the fact that there are numerous parties involved in data collection, processing, and its use for advertisement selection. Both the tracking and targeting steps can be performed by several entities, and any policy or technology proposal must take the complexity of ad delivery pipelines into account to be effective and unambiguous.

The moniker *Do Not Track* masks this complexity and is ambiguous on multiple levels. In addition to conflating tracking and targeting, the public policy focus has been on "Do Not Track for the Purpose of Behavioral Advertising," which neglects tracking performed for non-advertising purposes (e.g., data collected could be used for differential pricing on retail websites). While targeting is always performed by one or more parties involved in advertisement selection as explained in the next section, tracking may be performed by any of the entities involved. In addition to parties involved in targeting, users may be tracked by specialized data aggregation firms (data exchanges), which subsequently sell the data data to interested parties, including advertising platforms. In addition, tracking may also be performed by the first party, which then provides aggregated information along with the ad request.

2.1 Tracking Mechanics

The architectural details of the tracking infrastructure are not widely publicized, and have mostly been revealed via reverse-engineering studies performed by researchers and journalists [5, 13]. The lack of transparency is not surprising given the combination of intellectual property value of the underlying technology and the sensitivity of the related privacy issues.

Third-party tracking is typically performed via an element of the viewed page that sends an HTTP request to the tracking server, passing the properties of the current context and client-side identification data. The sending element is typically either a tracking pixel (small image hosted on the tracking server), or JavaScript code loaded with the page that sends the request. Identification of the user can be performed via an ID stored in a cookie, as well as via indirect methods such as relying on the first-party user ID encoded in the URL of viewed page, which is passed as the Referrer header. Additionally, it has been shown that combining standard request headers such as the IP address and UserAgent string can lead to high-precision identification [6].

User and context properties passed to the tracking server may include attributes of the viewed page. These attributes may include its category in some publisher-defined taxonomy, or a search query if the page was visited via a link from a search engine. In addition, the publisher may provide any additional user data, such as user-submitted demographic or location data, or data summarizing prior behavior of the user. Tracking servers also receive user data that it stores client-side, e.g., in its cookie, or in specialized plugin cookies (known as local shared objects), or in HTML5 browser local storage. Tracking data may also include demographic and location data (self-reported or inferred), attributes derived from user browsing activity (e.g., inferred interest categories), or specialized features (e.g., identifiers encoding specific products which the user has viewed on a retail site). The tracking platform can either store all user-related information on the client-side, which requires updating the information regularly, or utilize server-side storage.

2.2 Targeting Mechanics

The effectiveness of advertising is directly affected by the availability of data used to estimate the user's potential responsiveness to the advertisement creative or the underlying product. Such information may come in various forms: demographic and location information, aggregated information about user's past behavior, and raw behavioral information can all be factors that affect user's desirability to the advertiser. Access to additional information describing the user beyond the context in which the advertisement is served allows advertisers to modulate the prices they are willing to pay to optimize their return-on-investment (ROI).

Advertising platforms that perform their own tracking utilize the tracking information in ad selection. In recent years, an increasing volume of display advertisements is allocated via ad exchanges, also known as real-time bidding (RTB) platforms [12]. A content publisher sends an ad request to the ad exchange,

which then forwards it to multiple advertising networks. User information collected via tracking may be provided by the first party (the publisher), or by any of the involved third parties: the ad exchange, ad networks submitting the bids, or dedicated data exchanges (tracking-only companies), which may partner with any of the former entities. As a result, actual ad selection may involve not one, but multiple user profiles accumulated by a number of agents, where any of them can be stored either server-side or client-side. Matching of user IDs is an additional complication, which is resolved by a establishing the mappings across the different parties and caching them, typically performed by the exchange.

3 Targeted, But Not Tracked

This section discusses how the distinction between targeting and tracking can be exploited by technological approaches which reduce or prevent user tracking, while allowing advertising networks to retain all or most of the revenue gains achieved from targeting. These solutions make use of *client-side* aggregation of personal data, which allows the user to be targeted while leaving user in posession of their data. It is know that major concerns over behavioral advertising include users' lack of control over the data describing their past behavior, as well as the insufficient transparency of the data collection and retention, as evidenced by users' poor understanding of these processes and policies [10]. Client-side user profiling solutions respect users' desire for privacy while maintaining the advantages (both to users and service providers) that comes with behavioral advertising, and hence provide a strong alternative to clear-cut, binary solutions like *Do Not Track*.

First, we give an overview of three recently proposed solutions in this space, which require the user to install a custom plugin in their browser that performs behavioral tracking and advertisement targeting. Then, we present a novel approach, Client-only Profiles (CoP), which naturally fits into the existing advertising ecosystem. CoP does not require any installation or actions by users or advertisers, and requires only minimal changes from advertising networks. We also summarize relevant results of an empirical study for one possible algorithmic implementation on which CoP can be based: a machine learning method titled *Predictive Compact Profiles* [4]. These results show that the CoP approach can potentially retain nearly all of the revenue gains obtained from behavioral ad targeting, while preserving users' right to privacy and control of their data.

3.1 Plugin-Based Client-Side Profiling

Plugin-based client-side solutions make use of a browser extension installed on the user's machine to incorporate user preference in advertisement selection. The plugin maintains a collection of the user's browsing and behavioral data on the user's machine, and uses it to facilitate targeting during ad selection. The three primary approaches that fall into this category are Privad [8], Adnostic [14], and RePRIV [7].

Privad [8] exemplifies an approach whose goal is complete user privacy. User behavior is monitored by a plugin installed on the client machine that maintains a user profile built from it. Ad platform's server provides the plugin with all (or a large subset) of the potential ads that may be displayed, from which the plugin selects the actual ad to be displayed utilizing the profile to achieve targeting. Ad impressions and clicks are encrypted and passed through a third-party dealer which is not able to view the information, but anonymizes its source before passing it to the ad network (i.e., hides the user's IP address from the ad network). Thus, the ad network does not know which ads were shown to or clicked by which users, but obtains aggregate statistics.

Adnostic [14] takes a similar approach to Privad: a browser plugin selects the ad to display with the aid of a locally constructed profile. In contrast to Privad, Adnostic takes the view that ad impressions should be kept hidden from the service provided, but not ad clicks. This makes the ad platform less vulnerable to click fraud, but also reveals the targeting attributes of a user when that user clicks on an ad. When a user visits a webpage, the ad network sends 10 to 20 ads which can be displayed, and the client plugin selects one of these based on local targeting attributes. The information about which ad is displayed is encrypted and provided to the ad network in a form that prevents the network from knowing which ad was shown. Occasionally (monthly, for example), aggregated encrypted data is provided to a trusted third party that decrypts it and informs the network how many times each ad was viewed.

Both Privad and Adnostic make fraud detection difficult for ad platforms (though less so in Adnostic), which is a significant concern from the perspective of ad platforms. Both approaches also increase network traffic and page load times since they move a significant portion of the ad selection step to the client along which requires transferring the ad inventory. Another concern is advertiser budget constraints: with Adnostic, an advertising platform must estimate when an ad's budget will expire in advance before sending it to the client, which may lead to ads being shown too many or too few times depending on the quality of the prediction. Furthermore, these two approaches take a significant portion of control over tracking and targeting out of the hands of the advertising network, reducing its ability to innovate and experiment with new targeting methods.

RePriv [7] takes a slightly different approach to targeting without server-side tracking. It constructs user profiles from the raw browsing data on the client machine, and sends the profiles up to ad platform's server to facilitate targeting server-side. In contrast to the previous two approaches, this allows the ad network to view user data and perform whatever personalization it desires at the time the ad is requested. The ad network can also provide custom miner modules to the client that extract data from the user's raw behavior, allowing the network to develop more complex targeting mechanisms. The user has the option to review the data that will be sent to the ad network, and either approve or disapprove of its release. Since the ad network is directly selecting ads and recording clicks, this approach solves many of the difficulties that Privad and Adnostic have with regard to fraud, budgets, and innovation.

3.2 Native Client-Side Profiling

This section introduces a novel approach, Client-only Profiles (CoP), that also stores behavioral information on the client but, unlike the methods discussed in the previous section, does not require the user to download and install a custom browser plugin. Most importantly, CoP gives users control over their data while allowing platforms to target advertisements without making significant structural changes to the current delivery or pricing mechanisms.

In CoP, user behavior is maintained in aggregated form along with a cache of raw recent behavior in the browser cookie associated with the ad network. The ad network receives the cookie with the ad delivery request from the user's current context (the web page the user is loading), and returns targeted ads with an updated cookie containing a refreshed profile. As with RePriv and Adnostic, while the ad network does receive the raw user browsing behavior, it is bound by policy to discard the information once it has returned the targeted ads and the cookie to the user; it also must not store the user ID with any logs of ad impressions and ad clicks. Thus, the only record of user behavior is maintained on the client in the cookie, leaving the user with the option of deleting their profile at any time, knowing that there are no records associated with them remaining on the server. Because this method relies on policy compliance by ad networks, it is not enforceable. However, this is the same assumption that the leading DNT solutions (Opt-out cookies and HTTP header) and RePriv make. Given that policy violations are detectable (by manipulations of client-side data), and bear significant legal and public relations ramifications, compliance assumptions are reasonable. We also note that CoP can be implemented to incorporate server-side encryption of profiles, preventing their interception in non-secure HTTP traffic.

CoP requires ad networks to construct incrementally-updated user profiles to facilitate targeting. In our initial implementation of CoP for search advertising, we model the gains from ad targeting by considering *bid increments* that advertisers can specify along with their bids. Increments are triggered when the user has shown a past and has expected future interest in the ad's topic. Bid increments are commonplace in display advertising platforms, however they are based only on explicitly known demographic attributes, or broad, loosely defined segments. In search advertising, advertisers have an analogous interest in adjusting their keyword bids for users known to have had a past interest in the keyword's topic.

To maintain a profile of the user's predicted future interests, different profile constructions can be employed. Here, we summarize a machine learning based approach recently proposed in [4]. The approach utilizes features the encode recency and frequency of past behavior associated with a keyword and its neighbors (related keywords), as well as context-independent keyword and user properties. For each candidate keyword considered for inclusion in the profile, a scoring function trained via a machine learning approach predicts, for that user, the likelihood that the user will click on an advertisement associated with the keyword in the future. Based on predicted probability, top-k keywords are selected

to comprise the user's profile, which is then used to trigger bid increments during future selection. Targeting is thus performed under the restriction that the only available user information is that which is stored in the client-side cookie on the user's machine, yet is able to closely match the predictive accuracy of targeting that relies on the user's complete behavior history. In the next section, we summarize an empirical study fully described in [4] that examines whether revenue is lost in moving from traditional server-side tracking to client-side profiling based on a particular algorithmic implementation of profile construction in CoP.

3.3 Revenue Impact Analysis

Client-side vs. server-side keyword profiles were evaluated using two months of search and advertising behavior logs for 2.4 million users of the Bing search engine, sampled randomly from the overall, larger pool of bot-filtered US-based active users, where active users were defined as those users who had used the search engine (issued at least one query) on at least 30 of the 60 days in the time period. The first six weeks of data were used for training the machine-learning based predictor used for keyword selection for profiles. Training is performed by simulating the profile construction process and utilizing the subsequent behavior to obtain a training label (ad click or lack thereof) for every keyword that was a candidate for inclusion in the profile. With the utility predictor trained on the first six weeks, the efficacy of online profiling was evaluated by simulating profile construction over the seventh week, using behavior following the construction period to estimate utility.

Client-side profiles contain two portions: the profile, which matches keywords, and an additional keyword cache used to enhance the profile construction candidate pool. Both components were constrained to be no greater than 100 keywords, which ensures that profiles fit the 4KB cookie size limit. Profile construction uses the utility model that corresponds to matching the keywords for which future ad clicks are observed, which is equivalent to bid increments. Cache construction is more straightforward: it is based on least-recently-used caching, a standard approach that typically has good performance and is efficient to compute.

Utility is reported as the fraction of ad clicks in post-profile-construction behavior for which the profile matched the bidded keyword, and hence would have triggered the bid increment for advertisers who specify it. This metric, percentage of incremented clicks, can be viewed as the percentage of ad revenue that would be increased via increments.

Figure 1 illustrates this relative performance of client-side profiles (with their limited knowledge of user history) with respect to server-side profiles for different profile sizes, which correspond to server-side tracking. The figure demonstrates that maintaining a modest cache size alongside the profiles allows achieving targeting performance comparable to that of server-side profiling, but without the need to track user behavior server-side. For example, if profiles are limited to 20 keywords, utilizing a cache of the 50 most recent queries allows capturing 97% of the revenue gain achieved by server-side tracking while providing users

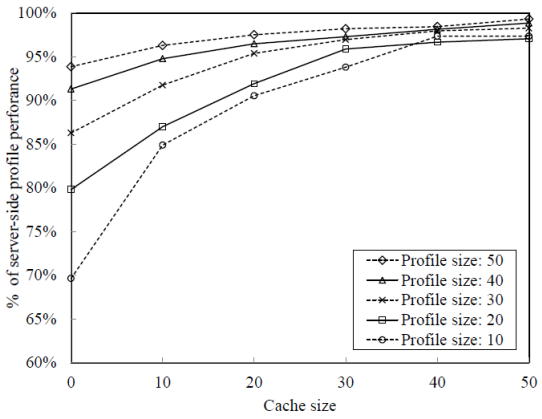


Fig. 1. Relative client-side profile utility (utility as a proportion of server-side utility for profiles of the same size)

full control of their data. These results demonstrate the practicality of allowing users to opt-out from server-side tracking with minimal revenue or performance cost. Complete description of the methodology and extended results are available in [4].

4 Discussion

These four approaches make different trade-offs in a space with dimensions of efficiency, changes from the existing infrastructure, guaranteed privacy, flexibility of targeting mechanism, and more. In Table 1, we compare the four methods, along with the existing methodology along these dimensions.

The dimensions are:

- **Profile Construction**: Is the profile constructed on the client (**C**) or server (**S**)? The user profile consists of the behavioral data stored for the user. RePriv constructs the profile on the client, but using code that can be sent from the server down to the client, indicated by C(*).
- **Profile Storage**: Is the profile stored on the client or server? Profile storage determines what party has control over the profile and the ability to manipulate or delete it.
- **Ad Targeting**: Is ad targeting performed by the client or server?
- **User Control over Profile**: Does the user maintain full control over the content of their profile?
- **Can target based on (non-contracted) third party behaviors**: Does the system allow ad networks to utilize user behaviors on web sites with which the ad network has no relationship?
- **User behaviors revealed to platform:** Which of the following user behaviors are revealed to the ad platform, tied to a particular user:

  - **Page Visits**: User visits to pages on which the platform serves ads.
  - **Profiles**: Profile information derived from raw user behavior.Derived user information, such as aggregate counts of the number of times the user has visited a page of a particular topic.
  - **Ad Impressions**: The ads viewed by the user. *agg* indicates that this information is provided in aggregate.
  - **Ad Clicks**: The ads clicked by the user. *agg* indicates this information is provided in aggregate
- **Required Changes To**: What changes would be needed in order to implement the proposed solution, by:
  - **Client**: The user's machine. "Plugin" means a Web browser plugin would have to be installed by the user.
  - **Advertiser**: The advertiser. "Slow stats" means that the advertiser will not receive real-time statistics about their advertisement budget and performance.
  - **Platform**: The advertising platform (network). Indicates a general notion of the quantity of work required for the platform to implement the system and change their targeting to work within the proposed system.
  - **Extra Parties**: Are there any additional new parties (besides the client, advertiser, and platform) necessary? "Online dlr" refers to the *dealer* required by Privad, which must be online with high availability. "Offline ttp" refers to the *trusted third party* by Adnostic, which needs only process data occasionally (they suggest once a month).
- **Requires unified topic taxonomy**: Does the system require ad networks to agree on a single unified behavioral targeting topic taxonomy?
- **Requires trust in platform**: Does the user need to trust that the company running the advertising platform is trustworthy? This is simply a reflection of the breakdown of which data is received by the advertising platform shown in the middle portion of the table.
- **Increased traffic from ads**: What (if any) is the increase in the network traffic between the user and the platform due to communicating more ads?
- **Increased traffic from profile**: What (if any) is the increase in the network traffic between the user and the platform due to communicating user profile information?

One of the primary divisions between methods is whether the ad personalization is done by the client or the server. In Privad and Adnostic, personalization is done on the client, whereas for RePriv and CoP, it is done on the server. This design decision has important implications. Methods which personalize on the client require multiple ads to be downloaded per page view, causing increased network traffic and/or slower page load times. Methods which personalize on the server necessarily must reveal to the server which ads were shown to the user, and also must pass up to the server any profile information that is used for user targeting. These latter methods rely more heavily either on providing the user controls to edit what is sent to the servers, or make assumptions about server

Table 1. Comparison of various proposals for respecting user privacy while still enabling ad targeting.

Concept	Privad	Adnostic	RePriv	CoP	Today
Profile Construction	C	C	C(*)	S	S
Profile Storage	C	C	C	C	S
Ad Targeting	C	C	S	S	S
User Control over Profile	Y	Y	Y	Y	N
Can target based on (non-contracted) third party behaviors	Y	Y	Y	N	N
User behaviors revealed to platform:					
Page Visits		Y	Y	Y	Y
Profiles			Y	Y	Y
Ad Impressions	agg.	agg.	Y	Y	Y
Ad Clicks	agg.	Y	Y	Y	Y
Required changes to:					
Client	plugin	plugin	plugin		
Advertiser		slow stats			
Platform	lots	lots	minor	very minor	
Extra Parties	online dlr	offline ttp			
Requires unified topic taxonomy	yes	yes	no	no	no
Requires trust in platform	no	some	some	yes	yes
Increased traffic from ads	download all ads	download 10x ads			
Increased traffic from profile			profile	cookie	

policies regarding retention and use of the user data in the future (in the case of CoP, the policy is that the server is not allowed to remember the profile after it has been used to personalize the ad). A primary advantage of personalization on the server is that it enables up a wider variety of personalization methods. In the case of ads, not only can the server select which ad to show, but also vary the ranking of a set of ads, as well as charge differentially for them. Further, servers may choose to change the ad copy (title, text, URL) or appearance of the ad as well. By assuming that ad personalization is equivalent to simply selecting one of N ads, Privad and Adnostic are provide a much more limited set of personalization options to the ad network.

A second significant difference between approaches is whether the ad network has the ability to develop targeting techniques. Much of the literature assumes that targeting should be done based on categorical membership. In our experience, other targeting methods can be as or more effective, such as fine-grained keyword-based targeting. Other platforms may find other attributes more beneficial, such as estimating user demographics, or geographics. Techniques which perform personalization on the client must also prescribe how that personalization is done. This means all ad platforms will be required to conform to a single personalization technique, a goal that is undesirable and untenable in our opinion. For example, Adnostic uses a single categorization system built into the client based on natural language processing heuristics. Ad networks are unlikely to want to give up control over their category schema or the ability to develop more and more advanced techniques for categorizing users into their schema.

One advantage of the plugin-based approaches is that they provide ad networks with the opportunity to target ads based on a user's entire browsing history, as opposed to just the portion of the history for which the ad network was able to observe the user. On the other hand, plugins must be downloaded and installed by end-users, increasing the difficulty of method adoption. Further, plugins solutions are less flexible, since they constitute executing modules that are shared across multiple ad networks, requiring cross-network agreement

for modification and potentially slowing the rate of innovation and progress in tracking and targeting methods. RePriv circumvents this difficulty by providing a mechanism for servers to send routines to the plugin in a secure and verified manner, but this openness comes at a cost – RePriv asks users for permission when new routines are installed, and, more importantly, each time user profile information is sent to the ad platform.

Taken as a whole, we believe the CoP approach, with its flexibility, efficiency, and similarity to the existing ad serving infrastructure, strikes the most effective balance between users' privacy, ad networks desire to personalize, and feasibility of implementation.

5 Conclusion

In this paper, we focus on the current conflation of tracking and targeting in policy discussion, which leads to the false assumption that any method that allows users to opt out of tracking must also neccessarily prevent ad targeting. This leads to ambiguity because different parties may be performing tracking and targeting, resulting in a popular misunderstanding that it is only possible to either have both tracking and targeting operating, or neither.

Distinguishing between tracking and targeting leads to a family of middle-ground methods which are required to store behavioral profiles locally on the user's machine, while allowing the ad platform to target ads. Three previously proposed approaches in this family all assume installation of client-side plugins and a topic-based representation of user profiles. The paper presented a novel approach, CoP, which also relies on client-side profile storage, but departs from prior work in not requiring additional software or a singular profile representation, making it directly compatible with existing advertising platform infrastructure. A possible implementation of CoP using a recently proposed machine learning algorithm for compact profile construction [4] is discussed. Empirical evalution on a large-scale, real-world dataset comparing CoP with traditional server-side tracking demonstrates that client-side approaches have the potential to give users control of their data without significant losses of revenue for the advertising industry.

References

1. Exploring privacy: A roundtable series. *Federal Trade Commission*, 2010.
2. Protecting consumer privacy in an era of rapid change: A proposed framework for businesses and policymakers. *Federal Trade Commission*, December 2010.
3. What does "Do Not Track" mean? a scoping proposal by the Center for Democracy & Technology. *Center for Democracy and Technology*, January 31 2011.
4. M. Bilenko and M. Richardson. Predictive client-side profiles for personalized advertising. In *Proceedings of 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD-11)*, August 2011.
5. P. Eckersley. How online tracking companies know most of what you do online. *Electronic Frontier Foundation*, September 2009.

SIMULATION OF CIRCUIT CREATION IN TOR: PRELIMINARY RESULTS

WILLIAM BOYD, NORMAN DANNER, AND DANNY KRIZANC

Department of Mathematics and Computer Science, Wesleyan University, Middletown, CT USA
e-mail address: whboyd@wesleyan.edu

Department of Mathematics and Computer Science, Wesleyan University, Middletown, CT USA
e-mail address: ndanner@wesleyan.edu

Department of Mathematics and Computer Science, Wesleyan University, Middletown, CT USA
e-mail address: dkrizanc@wesleyan.edu

Abstract. We describe a methodology for simulating Tor relay up/down behavior over time and give some preliminary results.

1. Introduction

Tor [Dingledine et al., 2004] is a low-latency anonymity network. To communicate anonymously with Bob, Alice forms a *circuit* through Tor consisting of three relays and tunnels her TCP traffic through that circuit. Layered (onion) encryption is used to ensure that each relay only knows its immediate neighbors in the circuit. In order to minimize latency, Tor does not implement techniques such as batching and re-ordering which can be used to provide very strong anonymity guarantees. As such, it is usually assumed that Tor is vulnerable to timing analysis attacks. In such an attack, Eve controls the first and last relay in a circuit and is able to identify when two streams at either end actually belong to the same communication flow by examining timing information of the data packets. Analyses of these (and other) attacks are often done by constructing ad-hoc simulations of the network. Not only are such simulations sometimes poorly justified, their ad-hoc nature makes it difficult to compare attacks and defenses, because each simulation is different. Even fairly sophisticated models of Tor, such as the one given by Feigenbaum et al. [2007] in terms of I/O automata, often fail to take into account time-dependent relay behavior, even though this sort of behavior can be crucial for attacks.

We propose the development of more principled simulations that capture emergent behavior of the network. In particular, many attacks involve manipulating circuits through compromised relays; detecting such attacks often requires a notion of what constitutes

1998 ACM Subject Classification: C.2.2 [Computer-Communication Networks] Applications—Tor; I.6.4 [Simulation and Modeling] Model Validation and Analysis; I.6.8 [Simulation and Modeling] Types of Simulation—Discrete event.

Key words and phrases: Anonymity; clustering; hidden Markov models; simulation; Tor.

"normal" relay performance and uptime. Replays of past behavior provide an accurate but necessarily limited model of behavior. In this short paper, we describe our work in progress to (i) characterize the behavior of individual Tor relays, (ii) capture this characterization in a generative model, specifically, a set of hidden Markov models, (iii) use these models to generate representative Tor relay behavior, and (iv) evaluate the "goodness-of-fit" of this generated behavior. In the longer term, we hope to have a generative model of many observable quantities of the Tor network such as relay failures, latency, throughput, etc. Our hope is that such a generative model could be used as a uniform testbed for analyzing attacks and defenses as well as proposals for changes to the Tor protocol itself. Our generative model is based on observations of the deployed network rather than explicitly simulating details of the Tor protocol, network links, etc., so it effectively adapts itself to changes in the network, without changes in the architecture of the model itself.

Our starting point is work of Danner et al. [2009], who analyze a denial-of-service attack described by Borisov et al. [2007]. In this attack, Eve compromises some number of entry and exit relays. If a circuit passes through a compromised relay but it is not the case that both the entry and exit relay are compromised, then Eve kills the circuit, causing the client to reform the circuit. In this way, Eve increases the probability that she controls both the entry and exit relays. From there she can perform data correlation attacks (see, e.g., Murdoch and Zieliński [2007]) to confirm that streams at either end of the circuit are actually the same flow. As an example of problems we wish to avoid, Borisov et al. associate a single failure rate to each relay, and then use that to analyze reliability and security of the network under this attack; but this a very poor approximation of actual relay behavior, as actual relay failure can be heavily time-dependent.

In analyzing the effectiveness of this denial-of-service attack and possible detection algorithms, we collected *lifecycle* data for relays in the deployed network. A *lifecycle* for a given relay R is a function $\ell_R : \{0, 1, \dots\} \to \{0, 1\}$. The idea is that we "probe" R some number of times. A probe consists of constructing a circuit of the form (G, R, E) and downloading a small file through the circuit, where G and E are relays that we control. Probe t *succeeds* ($\ell_R(t) = 1$) if the file is successfully downloaded and otherwise the probe *fails* ($\ell_R(t) = 0$). The probe of R may fail for many reasons; there may be transient network failures; R may refuse to allow a circuit (perhaps because of bandwidth limiting); R may not be in consensus during the probe. We consider all failure modes to be the same for this analysis, though it would be easy to treat different modes as distinct provided we can determine the reason for the failure.

Our analyses of the denial-of-service attack rely on replaying collected data of this form [Danner et al., 2009]. For example, we propose a detection algorithm that constructs circuits, records whether the circuit construction was successful, and then uses this information to detect the attack, making use of the fact that circuits that include compromised relays are more likely to fail. To simulate the execution of the algorithm, we "run" it on the replayed data under the assumption that a subset of the highly-reliable relays are actually under the control of an attacker (who would then "kill" uncontrolled circuits). Because some (replayed) relays are down at the "time" the detection algorithm is run, we get a more realistic assessment of the effectiveness of the detection algorithm in the presence of noise.

We are interested in lifecycle data because our measurements have shown that the following is a very good predictor of successful circuit creation: perform a single probe of each relay in the deployed network; then predict that a given circuit creation attempt will

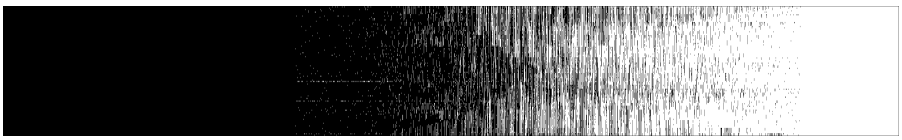


Figure 1: Visualization of lifecycle data for the deployed network. Each relay is represented as a vertical line. Each probe of the entire network is represented as a horizontal line. The pixel for relay R at probe t is white if $\ell_R(t) = 0$, black if $\ell_R(t) = 1$. Relays are sorted left-to-right in decreasing order of number of successes.

succeed if and only if each of the relays was successfully probed. Unsurprisingly, for any fixed relay R, the sequence ℓ_R is far from random; as an example, the lifecycles of the deployed Tor network as measured over a 30-hour period in mid-March 2011 (100 probes per relay) is visualized in Figure 1.

In this paper, we start with this collected lifecycle data to construct a *generative* model of relay lifecycles. The intent is that such a model displays the same emergent behavior as the collected data, while not being tied to replaying exactly the data that was collected. Our generative model is constructed as follows:

(1) Collect lifecycle data of the deployed network. We consider each relay's lifecycle to be a binary string, and all such strings have the same length. Let the set of lifecycle strings be S.
(2) Use an appropriate clustering algorithm to group the lifecycles into a comparatively small number of clusters.
(3) For each cluster C, train a hidden Markov model λ_C that produces lifecycles "similar" to the ones in C.
(4) To generate a new lifecycle, choose a λ_C with probability $|C|/|S|$ and use it to generate a lifecycle. To generate a set S' of new lifecycles, do this $|S'|$ times.

In the remainder of this note we describe the model in more detail. We also discuss an important but oft-neglected aspect of simulation: measuring how well a proposed simulation models the observed phenomenon. There is still much work to be done; in particular, we want to be able to model more interesting behavior than just lifecycles, such as latency and throughput. But the current work leads us to believe that our approach has a lot of promise.

2. Lifecycle clustering

We expect relays (as described by their lifecycles) to fall into a number of relatively well-defined categories. For example, there are many relays which exhibit perfect uptime over the course of an observation run, and some which exhibit regular uptime/downtime cycles as their administrators take them off-line at night or over weekends. A clustering algorithm partitions a set of observations (lifecycles) without an *a priori* definition of what constitutes a category. By clustering lifecycles and choosing a representative of each cluster, we obtain a simplified model of the network, in which each representative can be used to generate lifecycles comparable to those in the cluster it represents. Furthermore, we can easily modify the network model to provide insight into the behavior of the real network under various assumptions. If, for example, we wanted to observe network behavior when

```
function kmedoids(X : set of binary sequences , k : int) : list of k sets of binary sequences
begin
    Z ← [{x_1},...,{x_k}] (* x_i chosen at random from X *)
    C' ← [{},...,{}] : list k sets of binary sequences
    do
        C ← C'
        C' ← [{},...,{}] : list of k sets of binary sequences
        foreach x in X do
            i ← the j that minimizes E(x, Z(j))
            add x to C'[i]
        endf
        for i ← 0 to k do
            Z[i] ← center of C'[i]
        endf
    while C' ≠ C
    return C
end
```

Figure 2: Pseudocode for k-medoids approximation with Lloyd's algorithm. $Z[i]$ is the center of cluster i and $C[i]$ and $C'[i]$ are the old and new i-th cluster. A bit of restructuring eliminates the unnecessary last calculation of centers and also makes it easy to determine when a lifecycle has moved from one cluster to another.

high-profile relay operators are attacked or their relays blocked from access, we could reduce the proportion of several high-uptime clusters in the model.

A clustering algorithm depends upon a metric on the objects being clustered. We have considered several, but here will just concentrate on a version of edit-distance. To that end, we define $E(s, s')$ to be the number of "edits" that must be made to s to obtain s'. The edits that we allow are flipping a single bit and rotation by one position (the bit that is "pushed off the end" is tacked on at the other end). Our rationale is that the first type of edit can help minimize the effect of transient phenomena when identifying similar lifecycles. Rotation also allows us to consider as similar two relays that have similar cyclic behavior that start at slightly different times. For example, it is well-known that many relays are shut down at night (and this can be seen in the visualization in Figure 1). We end up considering such relays to be similar if "night-time" is approximately the same, but they become less similar as that time diverges.

Xu and Wunsch [2005] provide a fine survey of a wide variety of clustering algorithms. Here we just consider *k-means* clustering, which groups objects into a pre-determined number k of clusters in a way that attempts to minimize the average value of $E(x, c)$ over all objects, where c is the center of the cluster containing x. An exact solution is not feasible, but the standard approximation algorithm, Lloyd's algorithm, gives good results in time $O(nk)$ and space $O(n+k)$ where n is the number of sequences. In Lloyd's algorithm, initial centers are selected, and then items are assigned to the cluster with the nearest center. New centers are selected for each cluster, items are again assigned to the cluster with the nearest center, and the process is repeated until it converges. k-means clustering is typically used with Euclidean distance, for which it is easy to define the center of a cluster. With edit-distance, computing this new center is not so straightforward. Thus we define the center of a cluster C to be the sequence $c \in C$ that minimizes $\sum_{s \in C}(E(s, c))^2$. An algorithm that defines "center" in this way is sometimes referred to as a *k-medoids* algorithm (see [Kaufman and Rousseeuw, 1990, Ch. 2]). The k-medoids algorithm we use is shown in Figure 2.

3. Hidden Markov models

Hidden Markov models are a commonly used statistical model for temporal processes that obey the Markov property: the behavior of the process at any given point in time depends on at most a constant, finite amount of its past state. Rabiner [1989] is a standard reference, and we follow his presentation. If we have in hand a sequence of observations and want to determine a Markov model that is likely to have generated that sequence, we generate a *hidden* Markov model, which consists of the following data:

- A finite set Q of states.
- An alphabet Σ of output symbols; these correspond to our observations (so in the case at hand, $\Sigma = \{0, 1\}$).
- A state transition probability distribution function $A : Q \times Q \to [0, 1]$ giving the probability of a transition from one state to another.
- An output symbol emission probability distribution function $B : Q \times \Sigma \to [0, 1]$ giving the probability of a given symbol being observed from a given state.
- A start state probability distribution function $\pi : Q \to [0, 1]$ giving the probability that a given state is selected as the first state in a sequence.

A (non-hidden) Markov model with states Q, state transition probability distribution A, and initial state q_0 is a special case of a hidden Markov model: $\Sigma = Q$, $B(q, q') = 1$ if $q' = q$ and 0 otherwise, and $\pi(q) = 1$ if $q = q_0$ and 0 otherwise.

An HMM λ generates an observation sequence $\mathcal{O} \in \Sigma^*$ as follows:

(1) Choose an initial start state using π; call this the current state s.
(2) Repeat the following some finite number of times:
   (a) Use $B(s, \cdot)$ to choose a symbol to emit.
   (b) Use $A(s, \cdot)$ to choose the new current state s.

Given an observation sequence, we would like to determine the HMM that is most likely to have generated that sequence. The solution to this problem breaks down to solving the following three problems:

(1) Given an HMM λ and observation sequence $\mathcal{O}$ determine $\Pr[\mathcal{O} \mid \lambda]$, the probability that λ generates $\mathcal{O}$. There is a straightforward dynamic programming algorithm known as the *forward* algorithm to compute this probability.
(2) Given an HMM λ and observation sequence $\mathcal{O}$ determine the state sequence for λ that is most likely to generate $\mathcal{O}$. The *Viterbi* algorithm is a dynamic programming algorithm to compute the state sequence.
(3) Given an HMM λ and observation sequence $\mathcal{O}$, adjust the parameters of λ to obtain a new HMM λ' that is more likely to generate $\mathcal{O}$. The *Baum-Welch* algorithm performs what is essentially a hill-climbing procedure to estimate new parameters based on the solutions to the previous two problems.

For details of all three algorithms, we refer the reader to Rabiner [1989].

Iterating the Baum-Welch algorithm finds a local maximum of $\Pr[\mathcal{O} \mid \lambda]$. Because the search space is very hilly, in practice one applies the algorithm from many different randomly-chosen initial positions and chooses the best local maximum. The Baum-Welch algorithm trains an HMM on a single observation sequence, whereas we will want an HMM that is trained on a set of such sequences. A simple solution to this problem is to train the HMM on the concatenation of the sequences in the sets. Assuming we do not allow too

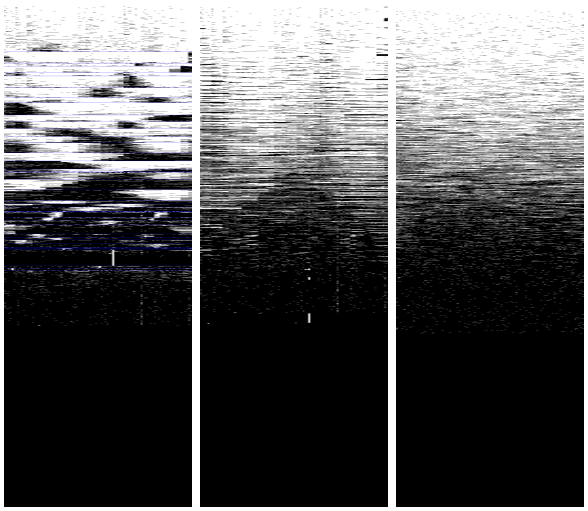


Figure 3: Collected data and our simulation. The collected data is the left and middle figures and the simulated data is the right figure. The middle figure is the same as Figure 1. The left figure shows the same data after clustering; clusters are sorted by failure rate of the center of the cluster, and lifecycles within a cluster are sorted by failure rate. Blue lines (visible in electronic version) separate the clusters. Pixel values are as in Figure 1.

many states in the resulting HMM, the spurious transitions at the concatenation boundaries should not have a significant effect on the result.

4. Preliminary results

We cluster the lifecycle data displayed in Figure 1 using k-medoids clustering with edit-distance. We follow a rule-of-thumb that says that $\sqrt{N/2}$ is often a reasonable number of clusters when there are N items in the original dataset. We observed about 3500 relays, so we use 41 clusters. For each computed cluster C, we concatenate the sequences in C into a single sequence, then apply the Baum-Welch algorithm to train an HMM λ_C for the cluster. In our runs, the median number of states is 8 (with a maximum of 10). We then produce a simulation of the Tor network by repeating the following 3500 times:

(1) Choose an HMM λ; λ_C is chosen with probability $|C|/3500$.
(2) Generate an observation sequence using λ as described above.

In Figure 3 we display a visualization of the original data along with our simulation. As we can see, some (but not all) of the emergent behavior seems to have been captured in this simulation. Although we do not see the "peak" of successfully-probed relays (corresponding to the above-mentioned diurnal pattern that has been already observed) strongly, it is visible. And likewise we see some of the additional uptime behavior of at the edges. It looks as though there may have been some transient problem in our data collection (indicated

by the two visible vertical white lines) but that same problem does not show up in the simulated lifecycles; this might be considered to be either a feature or a bug of the model.

5. Assessment

To properly assess the quality of any simulation, we need a measure by which to do such an assessment. This is a general problem in simulation studies, and appears to be a very difficult problem. However, we believe we can make a start by proposing the following. We think that a quality measure ought to satisfy the following two properties:

(1) The best simulation of a data set is the data set itself.
(2) A simulation that cannot produce a given data set is a poor simulation.

Define a *model* $\mathcal{M}$ of lifecycle data to be any process that, given n, produces a set of n lifecycle sequences. Furthermore, for a given set S of lifecycle sequences, we require that $\Pr[S \mid \mathcal{M}]$, the probably that $\mathcal{M}$ produce S on input $|S|$, be defined. We can then define the quality of the model relative to a fixed set S_0 (such as our collected data) to be $\Pr[S_0 \mid \mathcal{M}]$. With this quality measure:

(1) The model $\mathcal{M}_0$ that just returns S_0 is the highest-quality model possible, since $\Pr[S_0 \mid \mathcal{M}_0] = 1$.
(2) A model $\mathcal{M}$ that cannot reproduce S_0 is a lowest-quality model, as $\Pr[S_0 \mid \mathcal{M}] = 0$.

With this quality measure, we compared several combinations of different clustering algorithms and distance metrics, and all resulted in seemingly-close quality. Other models that we expect to be poor in fact compare poorly to our model. An example is a model that produces $|S_0|$ sequences, each of which is a Bernoulli process that chooses failure with probability equal to the average failure rate of all relays represented in S_0.

This notion of quality is very preliminary and needs work. For example, $\mathcal{M}_0$ is the highest-quality model possible relative to S_0, but would be a lowest-quality model relative to a dataset S that is identical to S_0 except for a single change of one bit of one lifecycle. Nonetheless, we feel it is a useful starting point for discussion.

6. Conclusions

We have presented a generative model of Tor relay lifecycles based on collecting such data, clustering it, and then training hidden Markov models on the resulting clusters. Preliminary investigations indicate that this is a promising approach to simulating at least some properties of Tor. We have also presented an initial proposal of quality-measurement for any proposed model.

More work needs to be done to determine appropriate distance metrics and clustering algorithms. Showing that we can use a similar methodology to accurately simulate more sophisticated phenomena (such as inter-relay latency and throughput) is a necessary step. Finally, we need to understand how to define a generative model in general, and to compare different generative models. Little work has been done in this area, though Kleinberg [2003] and Carlsson and Memoli [2010] provide some very interesting ideas on the related question of assessing the quality of clustering algorithms.

References

N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of service or denial of security? In *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS 2007)*, pages 92–102, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-703-2. doi: 10.1145/1315245.1315258.

G. Carlsson and F. Memoli. Classifying clustering schemes. 2010. http://arxiv.org/abs/1011.5270v2.

N. Danner, D. Krizanc, and M. Liberatore. Detecting denial of service attacks in Tor. In R. Dingledine and P. Golle, editors, *Financial Cryptography and Data Security: 13th International Conference, FC 2009*, volume 5628 of *Lecture Notes in Computer Science*, pages 273–284. Springer-Verlag, 2009. doi: 10.1007/978-3-642-03549-4_17.

R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

J. Feigenbaum, A. Johnson, and P. Syverson. A model of onion routing with provable anonymity. In S. Dietrich and R. Dhamija, editors, *Financial Cryptography and Data Security: 11th International Conference, FC 2007*, volume 4886 of *Lecture Notes in Computer Science*, pages 57–71. Springer-Verlag, 2007.

L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Mathematical Statistics. Wiley, 1990.

J. Kleinberg. An impossibility theorem for clustering. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 446—453, 2003.

S. J. Murdoch and P. Zieliński. Sampled traffic analysis by Internet-exchange-level adversaries. In N. Borisov and P. Golle, editors, *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies, PET 2007, Ottawa, Canada*, volume 4776 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 2007. doi: 10.1007/978-3-540-75551-7_11.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. doi: 10.1109/5.18626.

R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005. doi: 10.1109/TNN.2005.845141.

Distributed Privacy-Aware User Counting

Florian Tschorsch Björn Scheuermann

Telematics Group
University of Würzburg, Germany
{tschorsch, scheuermann}@informatik.uni-wuerzburg.de

1 Introduction

In this work, we consider the question of how to determine—in a privacy-preserving way—the number of distinct users that have contacted an Internet service. This question is particularly challenging in a distributed setting. In order to protect the users' privacy, the service operators should obviously not exchange information about user identities. Then, however, it becomes challenging to determine the total number of distinct users in the overall system without counting users more than once.

One key motivation for our work is the anonymization network Tor [3], and the problems the Tor project is currently facing when it comes to estimating the number of Tor users [6]. Tor node operators definitely shouldn't exchange (or even record) explicit information about locally observed IP addresses of Tor users. Nevertheless, it would still be very interesting to obtain statistics about the total number of distinct users of the system.

From a more general perspective, we are looking for a way to obtain the number of distinct elements in a multiset of user IDs (e. g., in case of Tor, the IP addresses of users). These user identifiers can occur multiple times at the same service entry point, and they can occur at multiple entry points. No single point, however, will in general be able to see the "whole" multiset.

In order to tackle this challenge, we start from an existing algorithm for probabilistic counting [5]. This algorithm allows for estimating the total number of distinct elements in a sequence of user IDs without keeping track of the IDs that have already been seen. It also provides an interesting basis for obtaining such estimates in a distributed fashion at multiple observation points. What is more, this approach already mitigates the privacy problems to some extent. However, it does not fully eliminate the problems: in the worst case, it is still possible to conclude with arbitrarily high probability that a specific user was present. We thus proceed with modifications which avoid that an attacker can gain an arbitrary amount of knowledge. Here, the knowledge gain of an attacker can be considered as a privacy metric.

2 Naive Distributed Counting

In this section, we start by introducing the algorithmic basis of our approach—FM sketches—, and how they could be used to determine the total number of users of a distributed Internet service like Tor. In this first step, the application of FM sketches will be quite naive, and we will see that it comes with severe privacy problems.

2.1 FM sketches

FM sketches, introduced in [5], are an algorithmic mean to estimate the cardinality of a multiset M of n elements. Their low computational effort was the key original motivation behind them. However, as we will see, they also exhibit other very interesting properties, which make them a promising basis for solving the problem considered here. We will now outline the key ideas behind FM sketches; more details can be found in [5].

An FM sketch is based on a bit vector $S = (s_1, \ldots, s_w)$, $w \geq 1$, which is initialized to zero. We also need a hash function h_1 with geometrically distributed positive integer output, where the probability that $h_1(x) = j$ (with $j \geq 1$) for any randomly picked element x equals $P(h_1(x) = j) = 2^{-j}$. Each element $x \in M$ is hashed using h_1. The hash value is interpreted as an index in S, and the corresponding bit $s_{h_1(x)}$ is set to one. This leads to a bit pattern in S with many 1-bits on the left and many 0-bits on the right.

Flajolet and Martin found that a good estimate for the number of distinct elements can be obtained from the length of the uninterrupted, initial sequence of ones in S, i. e., from $Z := \min\left\{ j \in \mathbb{N}_0 \mid s_{j+1} = 0 \right\}$. There is a constant factor $\varphi \approx 0.77351$ such that $n \approx 2^Z/\varphi$, so that estimates can be obtained on this basis.

The accuracy can be improved by using multiple sketches in parallel. The respective technique is called Probabilistic Counting with Stochastic Averaging (PCSA) in [5]. Each element is first mapped to one of the sketches by using a uniformly distributed hash function h_2, and is then added to this (and only this) sketch. In the following, we will use this variant. If m sketches are used with PCSA, then the estimate for the total number of distinct items added is given by $C = m \cdot 2^{\sum_{i=1}^{m} Z_i/m}/\varphi$, where Z_i is the number of leading 1-bits in the i-th sketch. One can identify a PCSA set with an $m \times w$ matrix, where each row is a standard FM sketch. For a sufficiently large number of elements, PCSA yields a standard error of approximately $0.78/\sqrt{m}$ [5]. Increasing m thus results in a higher estimation accuracy.

Multiple FM sketches (and likewise PCSA matrices) can be merged to obtain the total number of distinct elements added to at least one of them by a simple bit-wise OR. Observe that combining the FM sketch with all elements of a multiset A and the FM sketch with all elements of another, possibly overlapping multiset B using bit-wise OR produces an FM sketch that is identical to the sketch of multiset $A \cup B$. Elements present in both A and B will not be counted twice, since the respective bit will always have value 1 in both sketches. This duplicate insensitivity trait allows us to perform distributed user counting.

2.2 Applying FM sketches for user counting

Now let us look at how we could apply FM sketches to distributed user counting. Each service entry point might maintain a PCSA matrix with pre-configured dimensions $m \times w$. m and w as well as the hash functions h_1, h_2 are agreed on by the service operators in advance. When a user with ID u contacts the service at one of the entry points, u is hashed into the sketch matrix and the respective bit is set locally. Clearly, if the same user contacts the entry point more than once, the user will not be counted again, since the respective bit is already set. By evaluating the sketch generated by a single entry point, we therefore obtain an estimate for the number of distinct users who contacted the respective mirror.

If the sketch matrices from multiple service entry points are collected and merged by a bit-wise logical OR, this results in a sketch for the total number of *distinct* users contacting *at least one* of the respective service entry points. The users are therefore counted in a duplicate insensitive way.

In the specific use case of Tor, the counting operation could be performed at the directory mirrors. As already argued in [6], this is a reasonable design, because each Tor user contacts at least one mirror during bootstrapping.

Since the user IDs are not exchanged directly, and typically many different user IDs are mapped to the same bit, one might expect that the sketch does not reveal much information about which specific users contacted the service. This conclusion is treacherous, though. Recall that hash function h_1, which selects the column in the sketch, is geometrically distributed. Consequently, there are relatively few user IDs mapped to bits more towards the right hand side of the sketch. If an attacker observes *such* a bit being set, then it becomes suddenly very likely that a specific user has indeed been present in the system. In the extreme case, an attacker knows for sure that only one single out of all possible user IDs maps to a specific location in the sketch; if this bit is set to one, the attacker can be sure that the user has contacted the service.

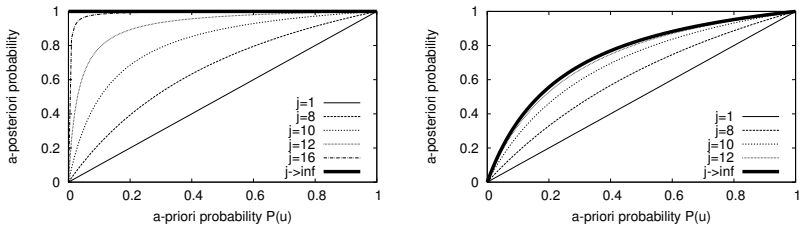
3 Privacy-Aware User Counting

In order to improve on the worst case behavior, we propose a perturbation technique. Specifically, a service entry point will proceed just as discussed above and enable bits according to the hash coordinates of the locally observed user IDs. In addition, though, each bit in the sketch matrix will be set to one with a fixed, configured probability r. While randomness has been used before in privacy-aware data bases, e. g. in [1, 4, 8], existing approaches do not allow to obtain the statistics we are interested in.

By adding these randomly switched on bits, we add an additional source of "vagueness": The attacker cannot make a definite decision whether a set bit has been set by a corresponding user or whether it has been switched on at random. The fact that we switch additional bits on uniformly at random has two important implications. First, we set bits on the right hand side in our sketch with non-negligible probability, where, as discussed above, an attacker is otherwise able to gain very substantial knowledge. Second, when it comes to extracting estimates for the number of distinct users, the uniform distribution of the randomly set bits will allow us to separate their effects from the geometrically distributed bits introduced by "real" users.

We now analyze this intuitive explanation of our approach and contrast it with the naive user counting. Subsequently, we give a short note on how accurate estimates can still be obtained. The question that we consider is how "sure" can an attacker be that a specific user has contacted the service. This is expressed by the *a-posteriori probability* of the user being there, after the information from the sketch is known. A probability of one means that the user has contacted the service for sure, a probability of, e. g., 0.5 means that it is equally likely that the user has or hasn't been there.

Clearly, this a-posteriori probability for the presence of a given user depends on the attacker's *a-priori knowledge*: how certain has the attacker been about the user being active *before* taking the information in the sketch into account? If the attacker, for



(a) Without perturbation ($r = 0$) (b) With perturbation ($r = 0.2$).

Fig. 1. Privacy evaluation: knowledge gain ($m = 8$, $n = 1000$).

whatever reason, has been 99.9% sure that the user has been active, then the a-posteriori probability will be at least 99.9%, as the attacker will not "lose" knowledge by looking at the sketch. We therefore always look at the a-posteriori knowledge *depending on the attackers a-priori knowledge*. The smaller the difference between the attacker's a-priori and a-posteriori knowledge, the less information an attacker can gain.

We are now interested in the probability $P(u \mid (i,j),r)$ that the user has used the service after the attacker has learned that bit (i,j) is set. Here, (i,j) denotes u's position in the sketch. r is our perturbation probability; consequently, $r = 0$ corresponds to the naive case without perturbation. In order to determine the a-posteriori probability, we start from the probability that a specific bit is set given that there was a total of n users and given u's a-priori probability $P(u)$ of being present. Following the definition of FM sketches in Sec. 2.1, this is $P((i,j) \mid n,r) = 1 - \left(1 - 1/(m \cdot 2^j)\right)^n \cdot (1 - P(u)) \cdot (1 - r)$. We then apply Bayes' theorem [2] and obtain

$$P(u \mid (i,j),r) = \frac{P(u)}{1 - \left(1 - \frac{1}{m \cdot 2^j}\right)^n \cdot (1 - P(u)) \cdot (1 - r)}. \tag{1}$$

One can see that the a-posteriori probability depends on several parameters, including j, n, m, r, and $P(u)$. In particular, the equation supports our intuition that bits to the right, to which a lower number of users are mapped, are more problematic: the higher the column index j, the more information an attacker gains if the bit is actually set.

Since our aim must be to protect the privacy of *all* users, we have to take the worst case into account. This can be done by taking the limit of the a-posteriori probability for $j \to \infty$. This limit is given by

$$\lim_{j \to \infty} P(u \mid (i,j),r) = \frac{P(u)}{P(u) + r - r \cdot P(u)}. \tag{2}$$

Observe that for $r = 0$ the limit turns out to be equal one for $P(u) > 0$. However, for $r > 0$ the a-posteriori probability does no longer converge to 1; there is a significant remaining uncertainty for an attacker even for large j. Consequently, we achieved our aim of mitigating the worst case. This relation is shown in Figure 1 with and without perturbation for varying j.

Now, the question arises how to still calculate accurate results despite the randomly added bits. Using the standard FM sketch evaluation formula above would lead to mas-

sive estimation errors, since the additional bits may increase the length of the initial sequence of ones in a sketch.

In [7], a related problem occurred. Even though the reasons for the perturbations and the problem setting are in fact quite different, the problem can be tackled with similar means. In short, the constant φ in Flajolet and Martin's estimation formula can be adapted according to the probability r. Along similar lines as in [7], the new constant φ_r can be obtained from $\varphi_r = \lim_{n\to\infty}(2^{E[Z|n,r]}/n)$, where $E[Z \mid n,r]$ is the (easily derived) expected value of Z for n distinct elements and additional bits switched on uniformly at random with probability r. If φ_r is used in the role of φ, the estimation error is compensated. For a deeper understanding we refer to [7].

4 Conclusion

We presented a methodology to count users based on their observed user IDs in a distributed and privacy-preserving manner. The algorithmic properties of FM sketches provide a way to deal with duplicate occurrences of user IDs, due to users contacting multiple times the same and/or different service entry points.

However this naive approach has severe shortcomings with respect to user privacy, at least in the worst case. We showed that this can be overcome by a perturbation technique that sets additional bits to one uniformly at random. In order to still calculate decent estimations by our modified FM sketches, we showed how the evaluation methodology can be adjusted. With respect to the specific use case of Tor user counting, our proposed method could be applied to estimate the number of Tor users at high accuracy, without compromising anonymity.

Acknowledgments

The authors thank the German Research Foundation (DFG) for the support.

References

1. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: SIGMOD '00. pp. 439–450 (May 2000)
2. Bayes, T.: An essay towards solving a problem in the doctrine of chances. Phil. Trans. of the Royal Soc. of London 53, 370–418 (1763)
3. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: USENIX Security '04. pp. 303–320 (Aug 2004)
4. Du, W., Zhan, Z.: Using randomized response techniques for privacy-preserving data mining. In: KDD '03. pp. 505–510 (2003)
5. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. Journal of Computer and System Sciences 31(2), 182–209 (Oct 1985)
6. Hahn, S., Loesing, K.: Privacy-preserving ways to estimate the number of tor users. Tech. rep. (Nov 2010)
7. Lieven, P., Scheuermann, B.: High-speed per-flow traffic measurement with probabilistic multiplicity counting. In: INFOCOM '10 (Mar 2010)
8. Mishra, N., Sandler, M.: Privacy via pseudorandom sketches. In: PODS '06. pp. 143–152 (Jun 2006)

P3: A Privacy Preserving Personalization Middleware for recommendation-based services

Animesh Nandi, Armen Aghasaryan, Makram Bouzid
{Animesh.Nandi, Armen.Aghasaryan, Makram.Bouzid}@alcatel-lucent.com

Bell Labs Research, Alcatel-Lucent

Abstract

We propose the design of a privacy-preserving-personalization middleware that enables the end-user to avail of personalized services without disclosing sensitive profile information to the content/service-provider or any third party for that matter. Our solution relies on a distributed infrastructure comprising local clients running on end-user devices and a set of middleware nodes that could be collaboratively donated by few end-users or hosted by multiple non-colluding third parties.

The key idea is to locally compute the user's profile on the device, locally determine the interest group of the user wherein an interest-group will comprise users with similar interest, and anonymously aggregate the collective behaviour of the members of the interest group at some middleware node to generate recommendations for the group members. In addition, our system is also open for third party content and recommendation injection without leaking the users privacy.

1 Introduction

A broad class of applications such as StumbleUpon [25] (or iGoogle [18]) URL recommendations, Foursquare [14] check-in recommendations, Netflix [21] movie recommendations, or IPTV content recommender systems suffer from the dilemma of having the user disclose sensitive profile information in order to benefit from personalized content/services. As of today, users have no option but to trust the content/service-provider with their sensitive profile information in return of the personalized content/services they seek.

Typical centralized recommender systems rely on one of the two main types of recommendations: content-based and collaborative filtering. The general content-based technique exploits content metadata (categories and tags) and based on the user's content consumption history builds a profile of the user in terms of weights associated with different categories and tags. Having built the profile of a user, items whose content metadata matches the tags and categories that have high weights are recommended. The general collaborative filtering procedure for content recommendation can be described as follows: 1/ compute clusters of similar users based on their history, 2/ compute the popularity curve of items within each cluster, and 3/ recommend to each user the most popular items within his cluster that are not present in his own history. As compared to content-based recommendations, collaborative filtering enable users to discover new types of content they may like and that they never consumed before nor expressed any explicit interest in them. However these approaches require having access to the users' profiles to take advantage from each other's experiences.

We would ideally like to have a privacy preserving personalization system that enables the end-user to benefit from a personalization/recommendation service without disclosing their preferences (i.e. user profile) to the content/service provider. Although there has been

some prior work towards this goal, prior systems suffer from several limitations (as detailed in Section4) which prevent them from being deployed globally and adopted widely by the end-users.

We will now describe some desired properties of an ideal privacy-preserving-personalization system.

- **Generic middleware for wide range of applications**: The solution should support wide gamut of potential applications varying from Telco-based IPTV content personalization, Web-based personalization like iGoogle URL recommendations, and mobile-based LBS (Locality-based services) personalization like Foursquare check-in recommendations etc. Thereby, the design should be in the form of a middleware, where different recommendation applications potentially running on different types of end-user devices (e.g. mobile phones, PCs, or Set-Top-Box) can plug-in.

- **Hybrid recommender system**: Given the obvious advantages of both content-based and collaborative filtering recommendations, we target a hybrid recommender system which benefits from both content semantics/metadata (content-based approach) and from the experience of other users (collaborative filtering approach).

- **Seamless Operation of Content-providers**: The design should enable 'seamless' (i.e. without requiring explicit cooperation) interaction with external content/service providers to receive appropriate recommendations generated by proprietary recommendation algorithms that have be developed and fine-tuned by content/service-providers like for example Google. Thereby, the ideal system should not require the content-provider to change fundamentally its APIs/interfaces, and additionally can work without having to invest in replicating sophisticated state-of-art recommendation algorithms developed by content-providers.

- **Profile Anonymization with 'Unlinkability'**: The design should obviously break the association of the user from his profile. In addition, even if the profile is anonymized, no one node should be able to see the complete profile, which we claim is prone to sophisticated linkability attacks wherein one can infer the mapping of pseudonym to user based on the profile details. Therefore we strive for unlinkability as well, which is achieved by ensuring that any single node can see only a small slice of the entire profile.

- **Trust No One**: Our design is aimed at ensuring that we do no trust any single-entity with all our sensitive information. Secondly, our system should be designed to be able to work under small scale collusion attacks.

- **Comparable Performance**: Last but not the least, the system should provide recommendation quality that is comparable or only slightly inferior to that of centralized recommender systems, at the cost of slight increase in overheads like communication costs or infrastructure costs.

We will next describe the design of our proposed P3 (i.e. acronym for privacy-preserving-personalization) system which is intended to meet the design goals mentioned above.

The rest of the paper is outlined as follows - Section 2 describes the design of the P3 system, Section 3 describes the detailed realization of the design, Section 4 contrasts our P3 system with prior works, and finally we conclude in the Section 5.

2 Design of the P3 Architecture

In this section we will highlight the key contribution of the work in terms of showing how different *functional blocks* can be interconnected to meet the goals of the privacy-preserving-personalization architecture.

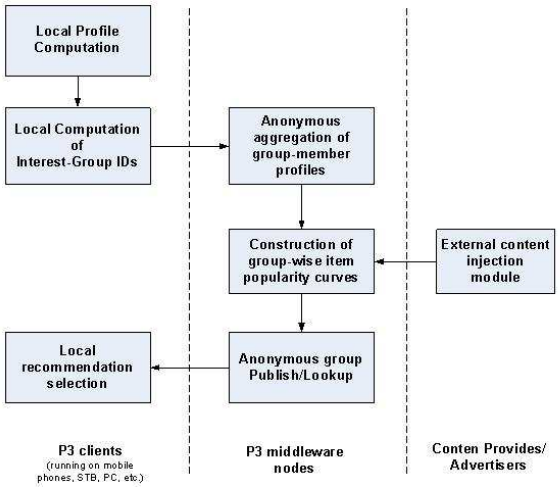


Figure 1: Functional Blocks of our Privacy-Preserving-Personalization Middleware

In order to protect the user privacy we propose to execute the steps of a centralized recommender system in a privacy-preserving way by using a distributed infrastructure comprising of *P3 local clients* running on end-user devices and a *set of P3 middleware nodes* hosted by multiple non-colluding entities. Models like that of Virtual Individual Servers (VIS) [6] or TOR [13] which employ end-users running nodes locally or in the cloud computing platforms (e.g. Amazon) could be envisioned. Note that we only require a small fraction of users to act as volunteers to host these nodes, and incentive mechanisms can be provided to such users. Alternatively, one could also imagine multiple third-parties like different cloud computing providers (e.g. Amazon, Google, Microsoft) together hosting the P3 middleware nodes.

The key idea is to build users' profiles locally on their personal devices, to identify the interest-group that a user is associated with locally wherein an interest-group will comprise users with similar interest, and anonymously aggregate the collective behaviour of the members of the interest-group to do both types of recommendation - collaborative filtering as well as content-based.

The core contribution of the work revolves around the idea of how the different functional blocks described in Figure 1 can be combined in order to realize the goals of our privacy-preserving-personalization middleware. Although our proposed realizations of each functional block as described in Section 3 exist in some adapted form in prior state-of-art, the key contribution of our work lies in how the interconnection of the functional blocks enables us to realize our desired goal. While referring to Figure 1, we will now describe the role of each functional block and how they interconnect.

STEP 1 ('Local Profile Computation' block in Figure 1): First, each user client collects the local traces of users' activities in the targeted service domain. These traces are analyzed and compacted in such a way that they represent the user profile, i.e. the user preferences in terms of items or in terms of item categories. Such a local profile can for example be represented as a set of <key, value> pairs, as it is described in Section 3. The P3 middleware can provide the personalization service to any external application for which such a local profile is available.

STEP 2 ('Local Computation of Interest Group IDs' block in Figure 1): Then, each local client determines its user's interest group by considering only the local profile data and some globally available information (e.g. a concept taxonomy, a term vocabulary, or seeds for generating random vectors). Note that this can be done without sending out the local profile to any external entity by using techniques like LSH (Local Sensitivity Hashing) [16] as described in the Section 3.

STEP 3 ('Anonymous aggregation of group member profiles' block in Figure 1): At the next step, we aggregate anonymously all the group member profiles corresponding to a particular interest group at a dedicated middleware node which we call the *Group-wise aggregator*. All the data related to a given group are collected in a single Group-wise aggregator.

STEP 4 ('Construction of group-wise item popularity curves' in Figure 1): Using collected group members' profiles, the Group-wise aggregator will generate a set of potential recommendations for the interest group based on the aggregate statistics of the already con-

sumed content by the group members. In particular, the Group-wise aggregator can compute the top-K popular items of the group and regard these as the collaborative-filtering recommendations for the group.

STEP 5 ('External content injection module' in Figure 1): The Group-wise aggregator can communicate the categories/tags corresponding to the top-K items to the external content provider to get content-based recommendations corresponding to the interests of the group. In fact, it can also seamlessly interact with the external content provider by posing as an end-user who consumes the top-K popular items of the interest group and get recommendations generated as per the proprietary recommendation algorithm running at the content provider.

STEP 6 ('Anonymous Publish/Lookup of group recommendations' in Figure 1): The generated recommendations at the Group-wise aggregator, which are a combination of the internal collaborative-filtering recommendations and the external recommendations, can be looked up anonymously by the group members or alternatively published to the group members anonymously.

STEP 7 ('Local recommendation selection' in Figure 1): Finally, the list of recommendations is filtered out on each user device, by removing already consumed content, and used to make recommendations for the end user.

3 Realization of the P3 Functional Blocks

Bellow we give more details on the realization of the functional blocks introduced in the previous section. We map these functions on different types of components of a distributed system which host the P3 middleware. Figure 2 describes one possible realization of the functional blocks.

3.1 Local profile computation

Component type: P3 local client running on end-user's device

We consider two possible realizations for this phase; of closed and open systems (in terms of consumed content space) respectively. We describe how these types of systems or applications allow the construction of local profiles in terms of <key, value> pairs, where keys are either items (item references) or item categories (tags, taxonomy concepts, etc.) while the values represent the interest level (e.g. on a 0 to 1 scale).

(a) Example of <key, value> construction in a closed-system: content provider portal, e.g. VoD portal. Each content item is explicitly associated with its metadata provided by the content provider. These metadata include the title and/or the artists and/or genres and/or keywords/tags, etc. The user consumptions are then mapped to these metadata terms and reflect user interests; each user consumption itself is a set of <key, value> elements where key is the metadata term and value is the interest towards that term. The aggregation of these consumption sets over the time allows to infer user interests in the form of <key, value> elements [1]. In this case, no additional global information is necessary for local profiling other than the metadata provided by the portal.

(b) Example of <key, value> construction in an open system: web browsing. In this case, the content items are web pages and their unique identifiers are the URLs. In addition, each web page visited by the user can be processed to find additional metadata characterizing its content. Such metadata can be either specified explicitly under the HTML 'title' tag or meta tags 'keyword' and 'description', or discovered implicitly, by parsing the source text of the web page content. In any case, after the extraction of these metadata an additional normalization is needed to incorporate some readjustments wherein common tags are given lower rates akin to IDF (Inverse Document Frequency) analogy. The normalization procedure can be based on some global tag popularity that can be retrieved for instance from Google Analytics.

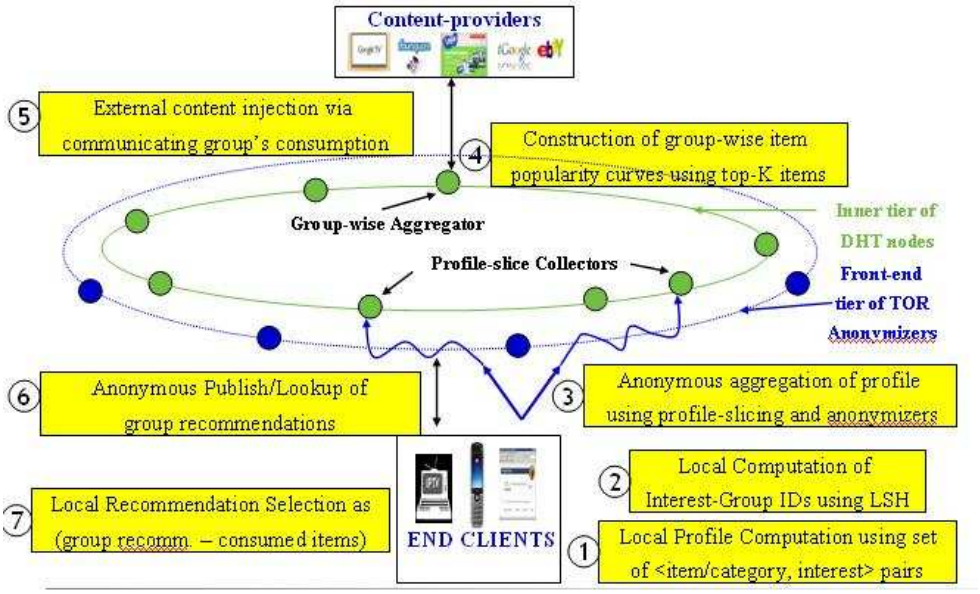


Figure 2: A realization of the functional blocks on different components of the P3 Middleware

Note that in both examples keys can represent unique identifiers such as the content title as well as other types of metadata such as genres or tags which characterize more than one content. The advantage of mixing both unique identifiers and categories is that the framework allows applying hybrid types of recommendation algorithms using both collaborative filtering (exploiting only the unique item identifiers) and content-based recommendations (exploiting the content categories). Optionally, to avoid mixing different types of keys, the <key, value> data structure can be replaced by <item-category, item-list, value> where item-category stands for keys that represent categories and item-list stands for key that represent unique identifiers content items the consumption of which contributed to inferring user interest in the given item-category.

Note that in both cases of closed and open systems keys that represent item categories can be mapped onto a common ontology, a taxonomy or flat vocabulary terms. This will also ensure semantic uniformity between different local profiles and also between different applications requesting the personalization service from the P3 middleware [2]. The mapping dictionaries then should be available locally at each local client. Note that such a semantic uniformity facilitates the local clustering described in the next session.

3.2 Local computation of global Interest-Group IDs

Component type: P3 local client running on end-user's device

The objective of this phase is the local computation of user interest groups (clusters), i.e. each local client determines its user's interest group by considering only the local profile data and some globally available information (e.g. a concept taxonomy, a term vocabulary, or seeds for generating identical random vectors). The global information is shared among all the clients independently of their local profile content; by mapping the local profiles onto this global information space one can determine the interest-groups as clusters characterized by commonly shared terms. So, two distinct user profiles that map (locally) to the same cluster discriminator will make part of the same interest-group without the need for explicit pair wise comparison between these two local data entities.

Two examples of such local interest-group computation can be given.

(a) Example using Local Sensitivity Hashing (LSH) techniques: an approach to compute all the similarities between all users in a computationally efficient way is given by LSH techniques [16]. This approach is commonly used to solve nearest neighbor and clustering problems in case of high dimensional spaces, where the 'Curse of dimensionality' makes an exhaustive search unfeasible [1]. The basic idea behind LSH is that two similar objects will hash to the same value with high probability, the value outputted by the hash functions could be used as the 'label' of a cluster (or cluster_id) to which the two objects belong. The output of several hash functions could be concatenated in order to reduce the probability of errors.

Google is using MapReduce and LSH to run a collaborative filtering algorithm that generates recommendations for users of Google News [12]. Here, we suggest to apply a similar approach in order to compute user similarities by using only local processing at each end-user client.

Each user client executes the appropriate LSH algorithm depending on the chosen distance metrics, e.g. Charikar offers a solution to implement cosine similarity using LSH [8], and similarly can use the MinHash algorithm [9, 10] to implement set similarity using the Jaccard coefficient.

The LSH code obtained from the local profile data gives the cluster identifier(s), cluster_id, which are statistically identical for similar users. One can generate several LSH codes and concatenate them for the same user to reduce false positives. At the same time, in order to catch several aspects of user's consumption patterns we can hash each user to a number of clusters (as suggested in [12]).

Finally, note that in an open system described in the previous section, the LSH will be computed by using only the keys representing item-categories. Note that privacy is not compromised in this process as all the computation is done locally.

(b) Example using semantics-based clustering: Each local client can identify the top-K item-categories within the local profile. This list will be considered as cluster descriptor of the interest group to which the given user belongs to. As an extension to this approach, one can also consider different subsets of top-K categories of size m, $m < K$, so that the given user is affiliated to more than one cluster.

[1] http://en.wikipedia.org/wiki/Curse_of_dimensionality

3.3 Anonymous aggregation of group-member profiles

Component type: P3 Middleware nodes

To compute the item popularity curve within each interest group, one needs to collect in a single place all the local profiles belonging to the same cluster. Doing this directly would however violate the user privacy because the local profile of each member user could be distinguished during the data transfer. To allow anonymous aggregation of profiles, we divide this process in two steps: (1) collecting partial profile data via an anonymization network like TOR [13] (i.e. Onion Routers [24]) at arbitrarily chosen P3 middleware nodes (called Profile-slice collectors) that participate in a DHT [11] (i.e. Distributed-Hash-Table), and (2) aggregating the information on a per cluster basis at the P3 middleware node (called Group-wise aggregator) which is the DHT node responsible for storing/computing information related to the corresponding cluster. Such a DHT node is chosen using the DHTs key-based-routing (KBR) primitive with the cluster_id as the key.

The role of the anonymization network is to hide the identity of the end-users from the Profile-slice collectors. The goal of slicing the profile into small segments is to ensure 'unlinkability' such that even after anonymization, no middleware node sees enough of the profile to be able to intelligently infer the identity of the profile owner from the profile contents.

Below we provide more details on the two-step aggregation mechanism.

(1) Profile slice collection: At this step, each user client slices the local profile into segments $s_1, , s_n$ composed of one or more <key, value> elements each. The profile slicing mechanism should be intelligent to ensure that no profile-segment by itself contains enough profile content

items that can be pieced together to infer the identity of the individual even after anonymization. By slicing to small-enough segments (with the limit being single item per segment), the probability of intelligent inference attacks on a profile segment reduces. However, one can design a profile slicing mechanism that minimizes the number of segments to reduce communication overhead while still ensuring safeguard from intelligent inference attacks.

Then, each segment together with the users cluster identifier(s), i.e. $< s_i$, cluster_id$>$ are sent via an anonymization network like TOR to different profile slice collectors. A typical 3-hop onion-routing path [24] can be established with the end-user encrypting the $< s_i$, cluster_id$>$ information with the public-key of the exit node of the onion-routing path, and the exit-node decrypting the information and relaying it to the profile slice collector. The end-user periodically chooses a different random set of DHT nodes to act as its set of Profile-slice collectors. Thus, different parts of the local profile will be delivered to different profile slice collectors, and none of them will have a full view of the local profile of any user.

Its worth noting that in scenarios where the end-user clients device is well-provisioned (e.g. IPTV set-top-box) to be a member of the DHT itself, the TOR-based anonymization network could be optionally replaced by a DHT-based anonymization mechanism like AP3 [20]. In this case, the client nodes generate a new hash code H_i for each pair $< s_i$, cluster_id$>$ and select a DHT node as a profile slice collector using H_i as the DHTs KBR key, and send the profile-slice anonymously using AP3 to this profile slice collector.

(2) Group-wise aggregation: At this step, each profile slice collector forwards a given data element $< s_i$, cluster_id$>$ to a Group-wise aggregator selected with a DHTs KBR mechanism by using the cluster_id as the hash code or key. As a result, all the data elements related to a given cluster will be collected in the same physical node. Note that each DHT node could be responsible for several clusters; this will depend on the number of clusters generated and the total number of DHT nodes.

Its worth mentioning here that one can explore other complex alternatives for anonymous aggregation like Anonygator [23] which in addition to some of the desired properties mentioned above can give additional protocol properties like resistance to data pollution by malicious end-users and more scalable aggregation using multi-trees. Similarly, a system like that proposed by Applebaum et al [3] doing cryptographic-based aggregation of <key, value> pairs, can be used for stronger privacy guarantees under collusion attacks as well as additional properties like 'keyword privacy' where we can ensure that the Group-wise aggregator can only know the top-K items of the interest group instead of the entire group's consumption.

3.4 Construction of group-wise item popularity curves

Component type: Group-wise aggregator

The group aggregator responsible for the given cluster concocts all the <key, value> elements belonging to its members. This allows computing the item popularity curve of each cluster, or in other words, each recommendation peer will select the top-K items for the given interest group. The top-K items could either be items, categories or tags. These top-K items serve as the group recommendations generated by the collaborative filtering approach. Additionally, they also reflect the primary interests of the group members in terms of items, categories or tags.

3.5 External content injection

Component type: External content injection module

This module is optional; it connects group-wise aggregators to external content and ad providers to push external content sources like targeted ads or third-party content recommendations. To achieve this functionality, the group-wise aggregators can either explicitly pull recommendations on behalf of its group; or alternatively can interact 'seamlessly' with the external

providers without requiring any special cooperation from the external provider.

In the first type of interaction (i.e. explicit), the group-wise aggregators can communicate the primary interest of the group in terms of categories and tags corresponding to the top-K items of the interest group to the external content provider to get content-based recommendations. The latter can return a list of contents/ads to be injected in the respective Top-K list. This external module will make its suggestions in correspondence with the interest group.

In the second type of interaction (i.e. seamless), the group-wise aggregator can seamlessly interact with the external content provider by posing as an end-user who consumes the top-K popular items of the interest group or for that matter the entire consumption list of its group members as its own consumption. The external content provider profiles the group-wise aggregator just as it profiles any other end-user, and generates recommendations (by any type of specific recommendation algorithm) for the group-wise aggregator, which in reality represents the interests of the group members.

3.6 Anonymous group Publish and Lookup of group recommendations

Component type: P3 Middleware nodes

The recommendations generated at the group-wise aggregator can be made available to the group members either via a PULL-based approach (i.e. lookup) or via a PUSH-based approach (i.e. publish). As in the aggregation mechanism, such a lookup or publish must be anonymous as well. We will describe two realizations below via the anonymous lookup and the anonymous publish-subscribe mechanism.

(a) Anonymous Lookup: In this operation, the end-user's P3 client issues a DHT-lookup [11] by using the cluster_id as the hash code or key. However such a DHT lookup is done over a typical 3-hop onion-routing path [24], with the cluster_id encrypted with the public-key of the exit node of the onion-routing path, and the exit-node decrypting the cluster_id, issuing a DHT-lookup with cluster_id as the DHTs KBR key, encrypting the returned results of the DHT-lookup (i.e. the recommendations) with the symmetric key that is provided by the end-user. The encrypted recommendations are sent back on the reverse path and the end-user's P3 client finally decrypting the recommendations. Note that privacy is preserved in the lookup mechanism, because the only node that knows the identity of the end-user is the entry-node of the onion-routing path, which however can neither know the end-user's group association (i.e. due to encrypted cluster_id) or the group's recommendations (encrypted recommendations). Its worth noting that in scenarios where the end-user client's device is well-provisioned (e.g. IPTV set-top-box) to be a member of the DHT itself, the TOR-based anonymization network could be optionally replaced by DHT-based anonymization system using AP3 [20].

(b) Anonymous Publish: Alternative to the above PULL-based anonymous lookup mechanism, one can employ a more complex PUSH-based anonymous publish-subscribe mechanism, wherein newly generated recommendations at the group-wise aggregator are published to the group members anonymously, i.e. ensuring that no P3 middleware node including the group-wise aggregator knows the identity of the group members. This can be done using anonymous channels that allow an end-user to specify a kind of mailbox-address for its intended messages as the channel address without divulging their identity. When a node wishes to construct an anonymous channel, it first picks a random id, as the address of the channel. Messages sent to this channel id are then forwarded anonymously back to the receiver, and nodes who send messages to the channel are unaware who is the actual recipient. Thus, if an end-user wishes to join the publish-subscribe group corresponding to cluster_id, it first creates an anonymous channel and then includes the address of the channel in the anonymously routed group-subscription request to the group-wise aggregator. The channel ids corresponding to different group members that the group-wise aggregator receives is maintained as the group state. Generated recommendations at

the group-wise aggregator can then be sent to the channel ids corresponding to its group members. Both TOR [13] as well as AP3 [20] allow the creation of anonymous channels. As in the case of anonymous lookups, depending on whether the end-user's P3 client is provisioned enough to run a DHT node locally, we use either an AP3 or TOR based realization of anonymous channels.

3.7 Local recommendation selection

Component type: P3 local client running on end-user's device

On the reception of group recommendations corresponding to the user's interest group(s), the user's client compares them with the local profile; by making a simple extraction operation one can filter out the items already consumed by the user in the past and obtain the recommendation list from merging the remaining items in the top-K lists provided by each cluster the user in affiliated to. In the case, where targeted ads have been injected in top-K lists a specific behaviour can be adopted by the user client depending on the user's preferences. On two extremes, all the ads could be filtered out or all the ads could be kept in the list which will allow repeated visualization of the same ad over the time).

4 Related Work

In this section, we will categorize the related work according to high level functionalities provided by prior state-of-art, and contrast their functionality with that of our P3 system.

Privacy preserving local content-based recommender systems: A few works like Adnostic [26] and Privad [17] employ a local profiling algorithm that categorizes the user into audience segment(s) based on the observed browsing and content consumption history. The content-provider broadcasts all ads/content to each user, together with specification of the intended audience-segment corresponding to each ad/content. The local content-filtering algorithm however only displays ads/content corresponding to the user's audience-segment.

To reduce the bandwidth costs resulting from broadcast of all content/ads, systems like Privad [17] anonymously subscribe the user to high-level interest categories via an anonymizing proxy that is assumed to not collude with the content-provider, receive content/ads for this high level category, and do fine-filtering of relevant content/ads locally.

Such systems however face the challenge of being able to develop an algorithm that can run on end-user's device with recommendation quality comparable to proprietary recommendation algorithms developed by centralized content-providers like Google etc. In addition, these systems only provide content-based recommendations and are unable to provide collaborative filtering recommendations since there is no way to know other top-rated items liked by users with similar interests.

In contrast, P3 enables both content-based as well as collaborative filtering recommendations, is more scalable, has greater resistance against collusion attacks, and also does not have to rely on replicating proprietary personalization algorithms on the local device.

Another very recent work Re-Priv [15] relies on doing local profiling in the browser. However, after having done the local profiling it relies on users to give explicit permissions to service/content providers to utilize the meta-profile computed locally to push personalized content. Their privacy benefits come from the fact that the browsed urls need not be revealed but only the synthesised meta interests can be provided to the content provider. In contrary, our approach is aimed to even hide the high level interests from the service/content provider.

Privacy-preserving collaborative filtering: Collaborative Filtering (CF) approaches present big privacy issues since they need to gather all users consumptions on a centralized server. Anonymizing users' profile (i.e via assigning pseudonyms) before sending them to a centralized server is prone to 'linkability' attacks where the real identity of the pseudonym can be inferred based on complete profile details (e.g. AOL scandal in 2006).

Solutions to preserve user's privacy in a CF system, propose to compute local aggregations of individual profiles for/by a community of users. The computed 'aggregate' is public and doesnt expose individual users profiles (e.g. Canny et al [7]). The aggregate is calculated iteratively by only adding vectors of user profile. Homomorphic encryption is used to allow sums of encrypted vectors to be computed and decrypted without exposing individual data. This work is however focused on a peer-to-peer framework, supposes that the majority of users are honest and states that the community should ideally know each other!

Another solution proposed by Polat et al [22] is to use randomized perturbation techniques to disguise the real user's consumption before sending them to the server. The server applies then a CF algorithm (SVD-based) on the disguised profiles and builds a matrix. To get a prediction for an item, the user sends a query to the server that computes a certain scalar product using the matrix and sends it back to the user. The latter computes the real prediction of the user likes/dislikes of the item using a particular formula involving the real private user profile. Authors indicate that the accuracy of their system is good but could be improved if more aggregate information about the user is disclosed along with the disguised data (i.e. knowing the real user preferences allows to generate more accurate recommendations).

Other approaches like that of Berkvosky et al [4] try to combine the obfuscation of a part of the user profile with the distribution of users profiles between multiple repositories to offer a CF system preserving user privacy while not hampering system accuracy. They studied what information of the user profile should not be hidden to continue having accurate recommendations. However, even though the generated recommendations were accurate, their results showed that a CF system working with real users' profiles (i.e. without any obfuscation) always provide better accuracy.

In summary, prior privacy-preserving collaborative filtering solutions either rely on heavyweight computations employing cryptographic operations or rely on degradation of recommendation quality by introducing perturbations in the users actual profile. In contrast, the collaborative-filtering approach of P3 is much more lightweight and works without requiring random perturbations to the user profile.

Collaborative personalized applications via anonymous peering with like-minded peers: In addition to collaborative-filtering recommendations, some Web2.0 applications like personalized web-search [19] could benefit from connecting the end-user with like-minded users with similar interests. The Gossple [5] system enables every end-user to be associated with a network of anonymous 'like-minded' acquaintances, and shows how applications like the personalized query-expansion can be built using Gossple. To achieve anonymization, Gossple uses a gossip-on-behalf approach where each node n is associated with a proxy P that gossips profile information on its behalf. The end-user's identity is hidden from P, by using an encrypted two-hop communication akin to onion-routing [24] wherein the end-user relays its profile information to P via an intermediate proxy that cannot decrypt the profile information. Although one could imagine using the Gossple substrate for designing a privacy preserving collaborative filtering application, the drawback of the Gossple's anonymous peering approach is that in order to know which two users are similar the complete set of consumed items (i.e. complete profile) consumed by the other user (although pseudonymized) is known to the other user. This is open to linkability attacks, wherein the pseudonym to user mapping can be inferred by sophisticated attacks that can intelligently infer the possible user based on the overall set of items consumed by him.

Scalable Collaborative Filtering: The Google-News personalization system [12] investigates how the scalability can be achieved in a recommender system that needs to deal with high item churn and large set of users. Their first technique to get scalability is the use of model-based algorithms for collaborative filter-

ing wherein the user is mapped to one (or more) cluster(s) of 'like-minded' users using techniques like MinHash [9, 10] (an LSH [16] scheme for set-similarity using the Jaccard coefficient) and PLSI, and only the item-ratings of cluster members (instead of the entire set of users) is used to calculate the recommendation. Their second technique to get scalability is to develop MapReduce-friendly versions of the MinHash and the PLSI algorithms. In contrast to our P3 system, their system is a centralized system which assumes access to all the user's consumption history, and uses a MapReduce framework to scalably generate the recommendations. Although their PLSI algorithm cannot be trivially applied to our P3 setting because of the need of several global item consumption statistics, we observed that the MinHash algorithm can however be used to do local computation of cluster_ids as proposed in the realization of Step 2 of our P3 system.

5 Conclusion

We have proposed the design and realization of P3 system, a privacy-preserving-personalization middleware that enables the end-users to participate in a wide range of recommendation-based services, without privacy concerns of revelation of their sensitive profile information. We are currently in the process of evaluating P3's performance in terms of recommendation quality and overheads, and contrasting it with that of centralized recommender systems.

References

[1] A. Aghasaryan, S. Betge-Brezetz, C. Senot, and Y. Toms. A profiling engine for converged service delivery platforms. In *In Bell Labs Technical Journal (BLTJ), Volume 13, Issue 2, pages 93-103*, 2008.

[2] A. Aghasaryan, M. Kodialam, S. Mukherjee, Y. Toms, C. Senot, S. Betge-Brezetz, T. Lakshman, and L. Wang. Personalized application enablement by web session analysis and multisource user profiling. In *In Bell Labs Technical Journal (BLTJ), Volume 15, Issue 1, pages 67-76*, 2010.

[3] B. Applebaum, H. Ringberg, M. Freedman, M. Caesar, and J. Rexford. Collaborative, privacy-preserving data aggregation at scale. In *Proceedings of Privacy Enhancing Technologies Symposium (PETs)*, 2010.

[4] S. Berkovsky, Y. Eytani, T. Kuflik, and F. Ricci. Enhancing privacy and preserving accuracy of distributed collaborative filtering. In *Proceedings of ACM Conference on Recommender Systems (RecSys)*, 2007.

[5] M. Bertier, D. Frey, R. Guerraoui, A. Kermarrec, and V. Leroy. The gossple anonymous social network. In *Proceedings of ACM/IFIP/USENIX International Middleware Conference (Middleware)*, 2010.

[6] R. Caceres, L. Cox, H. Lim, A. Shamikov, and A. Varshavsky. Virtual individual servers as privacy -preserving proxies for mobile devices. In *Proceedings of ACM SIGCOMM Workshop on Networking, Systems, Applications on Mobile Handhelds (Mobiheld)*, 2009.

[7] J. Canny. Collaborative filtering with privacy. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, 2002.

[8] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, 2002.

[9] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. In *Journal of Computer and System Sciences 55*, 1997.

[10] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang. Finding interesting associations without support pruning. In *Proceedings of International Conference on Data Engineering (ICDE)*, 2000.

[11] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common api for structured peer-to-peer overlays. In *Proceedings of International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[12] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of International World Wide Web Conference (WWW)*, 2007.

[13] R. Dingledine, N. Mathewson, and P. Syverson. TOR: The second generation onion router. also www.torproject.org. In *Proceedings of USENIX Security Symposium (Usenix Security)*, 2004.

[14] Foursquare. http://www.foursquare.com.

[15] M. Fredrikson and B. Livshits. RePriv: Reimagining content personalization and in-browser privacy. In *Proceedings of IEEE Symposium on Security and Privacy (SP)*, 2011.

[16] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, 1999.

[17] S. Guha, B. Cheng, and P. Francis. Privad: Practical privacy in online advertising. In *Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2011.

[18] iGoogle. http://www.google.com/ig.

[19] A. Mislove, K. Gummadi, and P. Druschel. Exploiting social networks for internet search. In *Proceedings of Workshop on Hot Topics in Networks (HotNets)*, 2006.

[20] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. Wallach. AP3: Cooperative, decentralized anonymous communication. In *Proceedings of ACM SIGOPS European Workshop (SIGOPS)*, 2004.

[21] Netflix. http://www.netflix.com.

[22] H. Polat and W. Du. Svd-based collaborative filtering with privacy. In *Proceedings of ACM Symposium on Applied Computing (SAC)*, 2005.

[23] K. Puttaswamy, R. Bhagwan, and V. Padmanabhan. Anonygator: Privacy and integrity preserving data aggregation. In *Proceedings of ACM/IFIP/USENIX International Middleware Conference (Middleware)*, 2010.

[24] M. Reed, P. Syverson, and D. Goldschlag. Anonymous connections and onion routing. In *IEEE Journal on Selected Areas in Communications (JSAC)*, 1998.

[25] Stumbleupon. http://www.stumbleupon.com/.

[26] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Adnostic: Privacy preserving targeted advertising. In *Proceedings of Network and Distributed Systems Security Symposium (NDSS)*, 2010.

Demographic Profiling from MMOG Gameplay

Peter Likarish[1], Oliver Brdiczka[2], Nicholas Yee[2], Nicholas Ducheneaut[2], and Les Nelson[2]

[1] Dept of Computer Science
The University of Iowa
Iowa City, IA 52245
[2] Palo Alto Research Corporation
3333 Coyote Hill Rd
Palo Alto, CA 94304

Abstract. This paper examines profiling basic demographic information (gender and age) from the gameplay of 1040 World of Warcraft (WoW) players. The authors develop two monitoring systems to track the players, one based on in-game observation and the other on a data source provided by the operators of the game. We describe and extract four feature sets, each from different assumptions regarding the type and amount of data available to an adversary: 1) a one-time snapshot of each character, 2) a series of snapshots from which we extract features for character progression, 3) a mapping of players to characters that allows us to extract higher level features over all the characters belonging to a player and 4) a superset of the previous three sets.
We show that one can predict gender and age (within ± 5 years) for 53% of players using machine learning and one can predict gender and age (within ± 1 year) for over 11% of participants solely based on the features monitored by our systems.

1 Introduction

Video games continue to increase in popularity, evolving from a niche hobby into a massively popular activity pursued by millions. In 2009, marketing survey group NPD found that 63% of their survey respondents had played a video game in the last six months while only 53% had been to the movies, laying to rest any doubt that video games have achieved widespread appeal[1]. Massively Multiplayer Online Games (MMOG) are one of the fastest growing segments of the video game market. These games allow millions of people to simultaneously play the same game over an internet connection.

This paper examines whether one can profile online gameplayers solely based on how they choose to play a game. Online profiling of this sort has a variety of applications. Knowing a player's demographic characteristics could allow a company to display advertisements that are more likely to be meaningful or interest to an individual. Knowing demographic details about a player may even enable companies to personalize the game world to that player, making the experience more engaging. Profiling is of particular interest to "social gaming"

companies whose tend to have a high churn rate and who may not know anything about their players aside from their style of play.

To determine if it is possible to extract demographic characteristics from gameplay, we observe 1040 individuals on the world's most popular MMOG, World of Warcraft [2]. We examine how well one can predict a person's real world (RW) demographic characteristics based on features extracted from their in-game behavior. We show that one can reliably predict a player's gender and age based on the features extracted in this paper, finding that one can predict gender and age (within ± 1 year) for over 11% of our players. With a wider age range (± 5 years), one can predict gender and age for nearly 53% of players.

As a second contribution, we investigate whether knowing the mapping between players and characters (which player plays each character) improves demographic prediction. Many MMOG players choose to play multiple characters, one "main" character and several "alts". Determining the mapping from players to characters is difficult. We use the ground truth mapping from our players and extract additional features that treat all the characters played by a single player as one entity. Statistical testing confirms that the mapping improves predictions of gender but does not improve predictions of age by a statistically significant amount.

We extract hundreds of features from the players' combat, exploration, achievement and social gameplay, divide the features into sets based on the type of observation (one-time character-based, continual progression-based, player-based or a combined superset of the previous three) required to generate each set, and show that one can accurately predict demographic characteristics for the majority of characters using classifiers or regression models from gameplay data. Models trained on our feature sets predict gender with an F-Measure and ROC AUC up to 0.9. SVM-based regression models trained on our feature sets predict age within ± 5.0-5.5 years (Mean Absolute Error from actual age). Before proceeding, we describe Blizzard's World of Warcraft.

1.1 The World of Warcraft

Our participants play what is currently the most popular MMOG in the world, Blizzard's World of Warcraft (WoW).[3] Due to its popularity, we assume most readers are familiar with basic MMOG mechanics and limit the length of our description. WoW has an active subscriber base of at least 11.5 million users (the last time Blizzard acknowledged a subscription figure in 2008[3]. Current subscriber numbers are estimated at 14-15 million users.). WoW is set in the fictional land of Azeroth, where various races battle for survival. Each WoW player creates one or more in-game alter-egos known as a character. The player selects a race aligned with one of two factions, the Horde or the Alliance, each made up of different races (e.g. elves or orcs).

[3] An expansion to World of Warcraft, Cataclysm, will be released after the publication of this paper and make portions of our description outdated.

The character must also select their class: Druid, Hunter, Mage, Paladin, Priest, Rogue, Shaman, Warlock or Warrior. This decision is the first step toward determining if the character is a Tank, Healer, melee DPS, or ranged DPS. Players often create more than one character: one "main" character and several "alts". These "alt" characters allow the player an alternative gaming experience (via race, class or role).

The players earn money and experience by completing quests and killing mobs (non-player controlled characters). When the player gains a set amount of experience, they level up. Player levels are currently capped at 80. Money is used to purchase equipment to improve the character's skills. One can play most of the game's content by oneself but to access the best equipment and most challenging game content, players need to form groups with others. These groups are formalized as guilds. Guilds also provide an in-game social network for players and can range in size from 1 to several hundred members.

In addition to combating mobs, a player may also fight other players (PvP). PvP can happen in a variety of settings, from large-scale fights during raids on the opposing faction or in battlegrounds, to duels and arena combat. Certain servers are designated PvP servers, allowing a player to attack opposing faction members at any time. Other servers are specified as Player vs Environment (PvE), imposing restrictions on PvP.

The rest of the paper is organized as follows. First, we review previous research, followed by a description of our participants and a detailed discussion of the systems that monitor their play. We then discuss extracted features. The features are described in more detail in Section 4. This is followed by an evaluation of how well we predict demographic characteristics from these features. We discuss implications in Section 7 before concluding with future work.

2 Related work

This paper focuses on extracting demographic variables via gameplay profiling and so we highlight related work that involves this sort of prediction. Hu et al attempted to predict demographics such as gender and age by using a Bayesian framework based on webpage click-through data [4]. In a very large study, Singla and Richardson observe that associates in social networks (even friends of friends) tend to have similar interests and personal characteristics and the strength of that relationship is correlated with their level of similarity [5]. ItemSpider, by Tsukamoto et al, is a social network centered around books. The authors found that people with similar characteristics were interested in similar types of books [6].

In 2007, Jones et al studied anonymized query logs and showed that with a series of classifiers one could map queries to gender, age and location of the user. They had a real-world acquaintance of a target user attempt to identify the target in an anonymized data set and found that personal information often enabled identification [7].

Another subset of this work has focused on predicting demographic data based on linguistic features in electronic media, including Herring's study of gender in electronic communication [8] and Koppel's work on determining the gender (and age) of a text's author or of bloggers [9, 10]. Linguistic features were not available to us in this study but could be incorporated to improve performance.

While there has been a large amount of social science and HCI research on online games, there is a dearth of research on demographic profiling from online gameplay. Examples of this type of work include studies by Ducheneaut et al [11], Yee et al[12, 13], Bessiere et al [14], Nardi and Harris [15] and Williams et al [16], among others. In a related vein, Grimes and Bartolacci examined the potential for using Second Life as a platform to teach profiling online behavior [17].

Finally, Nokelainen et al provides an example of how the demographic predictions can be used to personalize their experience by building a Bayesian model of a user based on a questionnaire they fill out [18].

3 Methodology

We established the ground truth for our predictions of gender and age by recruiting a large group of World of Warcraft players and monitoring their gameplay for a period of 6 months from April 5^{th}, 2010 through October 5^{th}, 2010. We now describe our participants and the recruitment process before giving an overview of the two tools developed to monitor their play. Then, Section 4 describes the various features we were able to extract based on our monitoring tools.

3.1 Participant details

We recruited 1,040 WoW participants, 533 participants from the United States, 512 from Hong Kong or Taiwan. The participants were recruited on forums dedicated to WoW, by publishing an article on the proposed study on popular gaming sites (e.g., WoW.com), through word-of-mouth, and via mailing lists collected during previous studies of online gamers. Participants completed a basic demographic survey as well as listing up to 6 WoW characters they were actively playing. Hong Kong and Taiwanese recruitment was done by scholars residing in these countries, with recruitment materials and surveys translated into appropriate dialects of Chinese. 26.25% of the participants were female. Our participants ranged in age from 18-65. The average age of our sample was 27.04 years old with a standard deviation of 8.22 years. Hong Kong/Taiwan participants were more concentrated in their early twenties, with fewer players over thirty. In contrast, over 41.70% of US players were 31 years of age or older.

The 1,040 participants played 3,862 characters during the course of our study, an average of 3.73 characters per participant (stdev=2.15). 2,034 (52.66%) of these characters were aligned with the Alliance. 1,988 (51.48%) of the characters were female. As one would expect from a mature game (WoW is over 6 years old), a large number of the characters in our study, 69%, have reached the highest level possible. Figure 1 plots the age and level distributions.

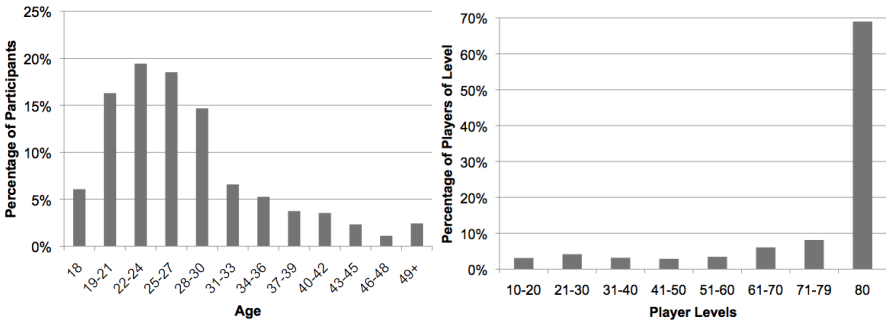


Fig. 1. The age distribution for our participants is plotted on the left and the level distribution for their characters on the right.

3.2 Monitoring participants

The data collection system monitored two sources of data. We conducted in-game monitoring based on a system described by Ducheneaut et al [11]. This system tracked characters on any of the 249 US servers or 31 Hong Kong/Taiwan servers. The software managed 12 WoW robots, each running in a separate virtual machine on one of two Quad Core Mac Pros. The robots log into the game, issued a /who query for the characters they were currently tracking and noted if they were online, collecting in-game location data for each character as well as for their online guildmates. This enables us track with whom our participants are playing, similarly to [12]. The robots cycled through the characters in roughly 45 minute intervals.

A second data collection system consists of a web scraper to gather character information from the WoW Armory [19] in the form of large XML files. The Armory is a Blizzard-provided service that supplies detailed information for all WoW characters over level 10. The Armory includes everything from generic information about a character's race and class to minutiae such as the number of monsters killed, the number and type of deaths and kills, the achievements the character has earned and information about the equipment currently in use. A character's armory entry is updated once per day if that character was active the previous day.

We process the output of both monitoring tools and extract the features described in the next section into a SQL database. A negligible percentage of the data was discarded due to network transfer errors. Heavy load on specific servers had a negligible impact on our in-game collection as well.

4 Feature extraction

Given the data sources detailed in the previous section, we extract a total of 435 features. We divide the features into four levels based on the information required to extract them.

- *Character-based features* are features one could extract given a broad knowledge of a character's gameplay at a single point in time.
- *Progression-based features* are features one could extract given temporal knowledge, that is, a series of character snapshots over a period of time.
- *Player-based features* are features one could extract given character-based features and a mapping between characters and participants/players to enable the extraction of "higher level" features for each participant.
- *Combined features* are a superset of the other three feature sets.

The following sections provide detailed descriptions of the features included in each set. We extract too many features to provide a detailed description of all features in this paper. A list of all our features is available at:
http://www.cs.uiowa.edu/ plikaris/WoWfeats.txt.

4.1 Character-based Features

We extract a set of 246 features for each character from a one-time snapshot of the most recent data for each character taken on October 6^{th}, 2010. We divide the features into categories and summarize the features in each category below.

General features A character's race, class, gender, guild, level, faction, base stats (such as strength or spirit), their professions skill and how they choose to allocate their talent points (after leveling, players can allocate talent points to increase their abilities).

This category also includes miscellaneous information such as how many mounts (rideable NPCs) and pets the character owns, their reputation with various non-player factions and how often they roll greed or need (when a group of characters encounter a valuable object, they "roll" to determine who receives it. The player with the highest roll keeps the object. If a character needs an object, they choose "need", otherwise they select "greed". Need rolls are always higher than "greed" rolls). (total features: 75).

Achievement features WoW grants achievements for completing certain game objectives, such as exploring areas or defeating certain bosses. Achievements categories include: General, Dungeons, Exploring, Feats, Professions, PvP, Quests, Reputation and World Events. We track the number of achievements each character has completed in each of the above categories. We also create a binary complete/incomplete feature for difficult-to-complete specific achievements (total features: 79.

Combat features Combat features include combat-related statistics such as the biggest hit received or dealt by each character. This section also includes the number of deaths the player has experienced (e.g. the number deaths from other players, from NPCs, from falling, from fire, etc) as well as the number of monsters killed. We also track the number of other players killed, and how

often the character uses PvP-specific features of the game, such as the arena or battlegrounds. The final piece of information in this category is the value of the character's equipment for each piece of gear they had equipped at the time (total features: 86).

Emotive features We observe the number of times a player hugs, waves, cheers, lols, facepalms, or violins in game. Other emotes are not tracked in the Armory (total features: 6).

4.2 Progression-based features

156 features are extracted from observations of the character's progression over the 6 month period of observation. For each character-based feature that changes over the period of observation, we tracked its rate of change. For instance, the average number of deaths per session played, the increase in equipment value per session played, or the number of hugs per session played. Included in this data set are features such as the percentage of time the character plays on each week day and during which part of the day the character is most active. We omit character-based binary features because completion is a yes/no proposition. An example would be achievements. Progression-based includes the rate of achievement completion in each category but not completion of specific achievements (total features: 114).

The social network data is inherently temporal because it requires multiple observations to establish the network. We calculate a large variety of standard social network analysis metrics including network size, transitivity, centrality, betweenness and clustering metrics for each character. This category also includes information about the variance in racial, class and level balances in the participant's social network (total features: 42).

4.3 Player-based Features

We extracted 33 features at the participant level by analyzing all characters played by a single participant as a group. These are features that are unavailable unless one knows a mapping from players to characters. They include: the percentage of characters of a given gender for each participant and the percentage of characters belonging to each faction for each participant. Other features include the amount of time a participant spends playing each role (melee DPS, ranged DPS, tank and healer).

We also compare the participant's focus in the game relative to other characters. That is, does the player spend more time on PvP or exploring? How much questing do they do relative to other participants? To answer these questions, we divide the participant's into quintiles depending on how many achievements they have completed in comparison to other participants.

4.4 Combined Features

This feature set is a combination of the other three feature sets. We include it to estimate the maximal predictive performance given all information available in this study. To generate Combined, we attached the progression-based features to the character-based features and added the corresponding player-based features for each character.

5 Predicting Real-World Demographics

This section evaluates the accuracy with which one can predict demographic characteristics (gender or age) based on the four feature sets described in the previous section (character-based, progression-based, player-based and Combined).

With the character-based and progression-based feature sets, we make a prediction for each of the 3,826 characters for which we have both character-based and progression-based features. With the player-based feature set, the underlying assumption is that we know the mapping from characters to participants and thus, we make our prediction for each of the 1,040 participants. Combined, a super-set composed of all three feature sets makes a prediction for each of the characters.

We adopt different data mining strategies for predicting gender (a discrete variable) and age (a continuous variable). For gender, we repeatedly trained a C4.5 classifier on each feature set with randomly selected training data. We experimented with other classifiers, including an SVM (Platt's Sequential Minimal Optimization (SMO) [20], an Radial Basis Function (RBF) kernel, varied complexity and gamma), but no classifier substantially outperformed the others. Some previous research binned people into age groups [4]. Instead, we opt to treat age as a continuous variable and to train a regression model to predict age directly. We experimented with several regression models including: linear regression, Partial Least Squares and a Multilayer Perceptron and regression from an SVM model. None outperformed the SVM regression model. We elected to use an SVM regression model [21] to directly estimate each character's (or participant's) age. We used the classifiers as implemented in the Machine Learning Toolkit, Weka [22]

5.1 Description of experiments

We develop two research questions that correspond to the contributions claimed in the introduction:

1. How reliably can we predict gender and age for each character or participant? (RQ1)
2. Which feature set yields the best performance for each demographic characteristic? Does player-based outperform character-based? (RQ2)

To addresses these questions, we carry out two experiments.

Experiment 1: training/test split To address RQ1, we examine the average performance of a model given varying percentages of data reserved for training. We reserve a set amount of data for training a model to predict either gender or age. The amount varies from 10% to 90%. We repeat this evaluation at each percentage split multiple times, randomly selecting the training data to improve robustness. We evaluate gender using F-measure and Receiver Operating Characteristic Area Under the Curve (ROC AUC) and age using Pearson's correlation coefficient, Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). This experiment addresses RQ1 by reporting performance metrics of how well we can predict each characteristic given set amounts of training data.

Experiment 2: differences in feature set performance To investigate RQ2, we select a set amount of training data (50% for gender, 80% for age) and increase the number of repetitions with randomly selected training data. We report the differences in performance and perform an ANOVA in order to confirm that there are statistically significant differences in performance. We note that we are able to perform sufficient repetitions that statistically significant differences become the norm rather than the exception.

5.2 Predicting Gender

This section investigates how reliably we can predict gender for each of our feature sets. We address RQ1 (how reliably we can predict gender) and RQ2 (differences in performance between feature sets) using a C4.5 classifier and the two experiment described in the previous section.

Experiment 1: training/testing splits We computed the F-measure and AUC for a C4.5 classifier trained on increasing amounts of data (from 10% of all instances up to 90%). At each percentage split, we randomly selected the training instances each time and using the remaining data to train the classifier, repeating the evaluation twenty times to reduce the effect of random selection on performance. The *a priori* class distribution was preserved in the training and test sets.

Figure 2 plots the results of this experiment. Combined outperformed the individual feature sets when at least 30% of the data was reserved for training. Progression-based underperformed the other feature sets. Character-based and player-based performed roughly similarly according to AUC but player-based dominates character-based when one considers only F-measure.

Classifiers trained on character-based, progression-based and player-based feature sets show limited improvement as the amount of training data increases. This suggests one can predict gender based on the ground truth for a relatively small number of characters. The stable performance with small amounts of training data alleviates concerns that we are overfitting the data.

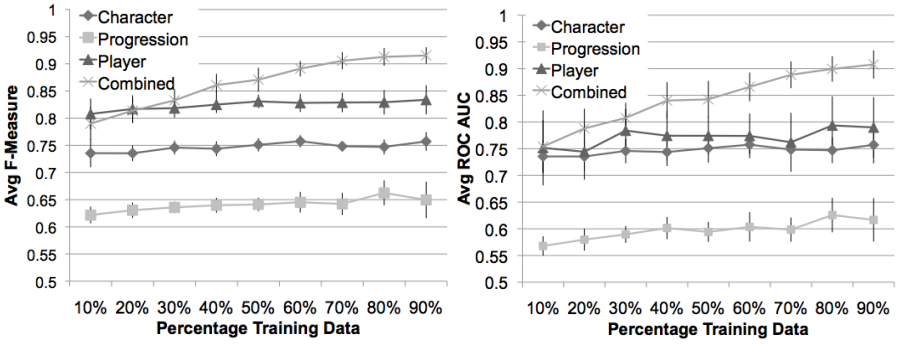


Fig. 2. Average F-measure and AUC of gender for feature sets with varying percentages of data used to train a C4.5 classifier. Training data was randomly selected for each split and the process repeated 20 times.

Experiment 2: differences in feature set performance To ensure we are looking at meaningful differences in performance between feature sets (RQ2), We repeated our random selection of 50% of instances as training data 100 times to be sure of the ordering among feature sets. We preserved the class distribution in both the training and test sets. Table 1 presents the average precision, recall (F-measure is the harmonic mean of precision and recall) and AUC for a C4.5 classifier trained using 50% training data for each of the feature sets and the Combined superset.

The player-based feature set performed better than the character-based and progression-based feature sets. We note that the player-based and character-based AUCs were within approximately standard deviation of one another while the difference in F1 Measure was more pronounced. The progression-based feature set performed poorly, suggesting that in isolation this feature set is the least useful for predicting gender. Of course, we only observed character progression over a six month period and the majority of our characters were mature characters.

Feature set	Precision	Recall	ROC AUC
Combined	0.871 (0.016)	0.871 (0.016)	0.843 (0.028)
Participant	0.826 (0.015)	0.829 (0.014)	0.775 (0.045)
Character	0.775 (0.011)	0.767 (0.011)	0.746 (0.026)
Progression	0.646 (0.015)	0.647 (0.017)	0.602 (0.022)

Table 1. C4.5 results for gender based on 50% training data. Feature sets ordered by decreasing AUC. Standard deviation in parenthesis.

We conducted ANOVAs using either AUC or F-measure as the dependent variable and the feature set type as the factor. The authors emphasize that, due

to the large number of observations, statistically significant differences between means of the feature set types are expected rather than remarkable. For AUC, the ANOVA confirmed that the main effect, feature set type, was significant ($F[3, 396] = 1038.074$, $p < 0.001$). Post hoc tests using Tukey HSD showed that the feature sets all differed from one another (means and standard deviations in Table 1, $p < 0.001$). These results suggest that the ordering observed in Table 1 is robust. An ANOVA conducted on F-measure produced very similar results ($F[3, 396] = 4583.297$, $p < 0.001$).

We also examined the most effective features using a traditional feature evaluation metric, Information Gain, to calculate the average rank of features relative to one another across a 10-fold cross validation. Appendix A presents the rank of the top 50 gender features, as well as the set to which each feature belongs. 46% of the top-ranked features belong to the player-based feature set. The difference is even greater for the top 20 features, with 90% belonging to the player-based set. This explains why the player-based set provids better predictions of gender than the character-based set. 42% of the top-ranked features belong to the character-based set but tend to be of lower rank than the player-based features. Only 12% of the features belong to the progression-based set.

5.3 Predicting Age

In this section we use an SVM-based regression model to predict ages. We evaluate RQ1 and RQ2 similarly to gender except that we use a different model (SVM and regression) and metrics used to do so.

We trained an SVM regression model with varying amounts of training data. Figure 3 plots the results of the training/testing split. all four feature sets continued to improve as the amount of training data increased. With regard to RQ1, similarly to our previous results, the progression-based feature set underperformed in comparison to the other sets. The Combined and character-based feature sets both produced correlation coefficients exceeding 0.5, our baseline for a moderately strong correlation.

With 80% of data reserved for training, Combined is able to predict age within ± 5 years in terms of MAE (± 6.5 RMSE), with the character-based feature set able to predict age within ± 5.5 (± 7.5 RMSE) years. Regression models trained with any of the four feature sets perform better than the baseline (the average standard deviation about the mean, depicted as the dotted line in Figure 3) except for progression-based when measured by RMSE. The standard deviation from the average MAE and RMSE varied by over a year. The usefulness of predictions as wide as ± 5 years is discussed in the next section.

Experiment 2: differences in feature set performance Predicting age via regression is difficult (even for with high levels of Combined feature set reserved for training, the MAE of our predictions is ± 5 years from actual ages) and so we reserved 80% of data for training, and used 20% for testing. Table 2 presents the MAE, RMSE or Pearson's correlation coefficient of an SVM regression model

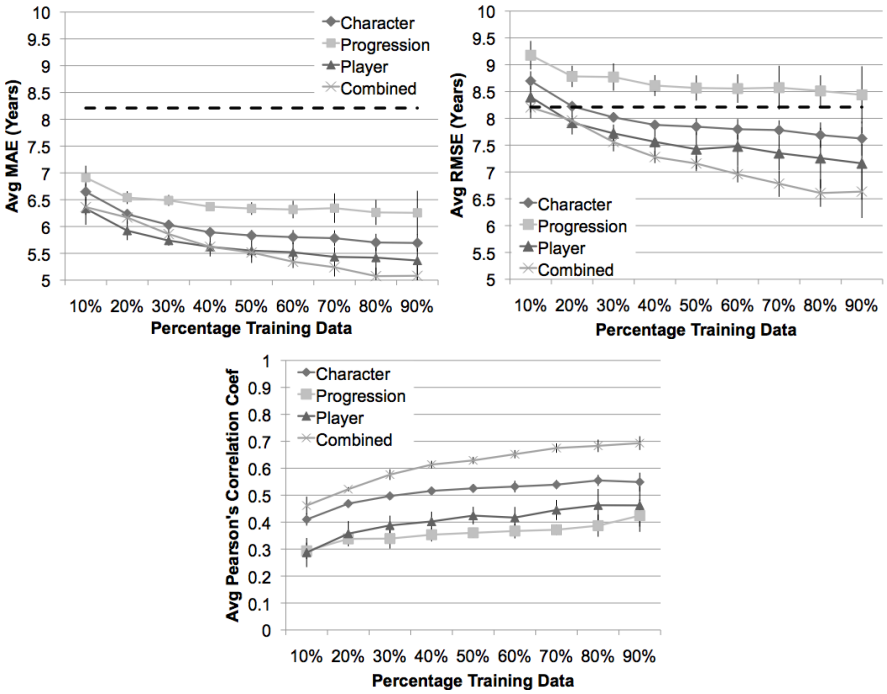


Fig. 3. From top-left, clockwise: MAE, RMSE and Pearson's correlation coefficient of predicted age for each feature set with varying percentages of data used to train an SVM regression model. Lower values represent an improvement in MAE and RMSE. The dotted line is the standard deviation of the distribution about the mean.

trained on each of the feature sets and the Combined superset. We repeated this analysis 25 times.

With regard to RQ2, as was the case with gender, Combined performed substantially better than the other data sets and progression-based substantially worse. The difference in performance between player-based and character-based was well within one standard deviation of both means. While the difference in performance between character-based and player-based was small, the deviation in performance was substantially larger for player-based, with a standard deviation of 2.85 years in terms of RMSE.

We conducted ANOVAs with RMSE and MAE as the dependent variables and feature set type as the factor. The main effect of feature set type was significant for RMSE ($F[3, 96] = 105.592$, $p =< 0.001$) and MAE ($F[3, 96] = 96.674$, $p < 0.001$). Post-hoc testing using Tukey HSD found that with regard to both RMSE or MAE, there was no significant difference between character-based and player-based (RMSE: $p = 0.110$, MAE: $p = 0.101$). Combined's mean was significantly higher than the other three sets ($p < 0.001$) and progression-based was significantly lower ($p < 0.001$). Means and standard deviations are in Table 2.

Feature set	Pearson's	RMSE	MAE
Combined	0.691 (0.009)	6.49 (1.15)	5.01 (0.65)
Participant	0.456 (0.026)	7.35 (2.85)	5.48 (1.64)
Character	0.559 (0.010)	7.60 (1.15)	5.64 (0.751)
Progression	0.388 (0.017)	8.46 (2.02)	6.22 (1.59)

Table 2. Regression model results for age based on 80% training data. Standard deviation in parenthesis. Feature sets ordered by MAE performance.

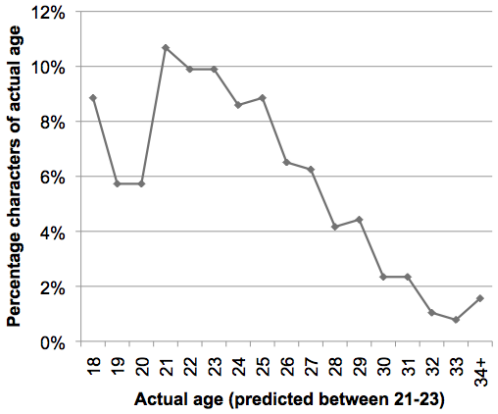


Fig. 4. Probability distribution character is of age given model predicts character is between 21-23 years old. For future predictions within 21-23 age range, we can use this distribution to calculate probability character is of certain age.

Depending on the application, being able to predict a person's age within ± 5 years (on average) may or may not be acceptable. We note that in related work, some age bins, particularly for older participants, have been 10 years wide and even larger [4]. It is also possible to treat the prediction generated by a model as a probability distribution rather than an exact value, at least given sufficient amounts of data from previous predictions and their ground truth for a large population of varied ages.

Figure 4 illustrates what such a distribution looks like given that the model has predicted a character is 22 ± 1. The figure was generated from data produced by a model trained on the Combined feature set. We extracted the actual ages of participants for which the model predicted the character was between 21 to 23 years old. The distribution was generated by calculating the fraction of characters that were of each actual age. With this distribution, given that the model predicted a new character is 22, one can determine there is a 48% chance the individual's actual age is 20-24 and only a 7% chance the person is 30+.

As with gender, we ranked the top 50 most predictive features for age using Information Gain (Appendix B). To use the Information Gain algorithm available, we discretized age into three bins. Again, the feature ranking helps to

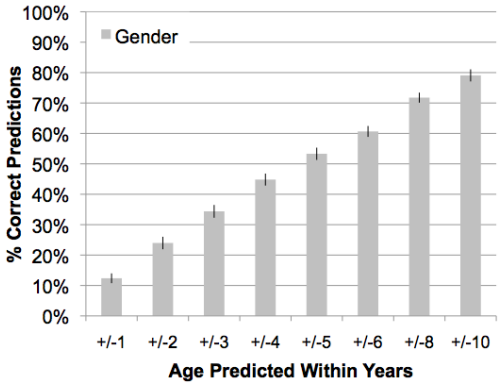


Fig. 5. Percentage of characters with gender correct and age predicted within $\pm x$ years of actual.

explain the relative performance of the feature sets. Unlike with gender (in which player-based outperformed character-based), with age player-based does not outperform character-based. 22% of the top 50 features belonged to the player-based set (25% in top 20). 46% of the top 50 features belonged to character-based and 32% to progression-based.

6 Predicting Multiple Demographic Characteristics

The previous section treated the prediction of gender and age in isolation. However, it could be beneficial to predict demographic characteristics in combination with one another. Making predictions in concert, however, can potentially compound the overall error rates. The purpose of this section is to estimate the percentage of characters for which we can determine both gender and age. Since age is continuous, we calculate the percentage of characters for whom the predicted value falls within a range of $\pm x$ years, where x ranges from one to ten.

We use the Combined feature set and randomly reserve 80% of the instances for training. We then predict a character's age (via SVM regression) and gender (via C4.5 classifier) for the remaining 20% of the data. We repeat this procedure 25 times. The results of this evaluation are presented in Figure 5. For over 11% of the participants, we can predict their gender and age within ± 1 year. For 53% of participants, our models predict gender and age within ± 5 years.

7 Discussion

People typically flock to free services over pay services online. This has limited the business models for companies who operate entirely in a digital space. One popular model is the use of predictive analytics/ad supported services. The techniques in this paper could potentially improve the effectiveness of this business

model. Further, if a game developer can determine information about a player based on how they play the game, it may be possible to tailor the game world to better suit that individual, alerting them of gaming events that are likely to be of interest or even individualizing the in-game experience to produce a more engaging experience. Additionally, recommender systems that monitor social gaming and utilize homophily between subjects[4] to improve recommendations could leverage the models generated in this paper to improve recommendations, increasing user satisfaction.

As researchers, we should also explore the implications of rampant monitoring and mining of online activities in order to establish what it is possible to determine from this data. Gaming is no different than any other activity we carry out online: our digital presence constantly leaks information about who we are in the real world. Although not the goal of this paper, profiling can be used to identify who we are even in the absence of personally identifying information. The furor over Blizzard's attempt to tie gaming profiles to their people's "real IDs" reveal that many MMOG players are uncomfortable with efforts to link them to their gaming personas [23].

8 Conclusion

This paper monitors 1,040 online game players and extracts a large number of features based on how the participants play the popular MMOG, World of Warcraft. The high levels of accuracy with which we can predict gender and age from gameplay alone suggest that profiling otherwise anonymous players may allow companies to tailor the gaming experience to individuals. Predictions generated for gender produce an F-measure of 0.75-0.85 with 50% of data reserved for training. A regression model for age with 80% of data reserved for training predicts actual age with MAE of 5.0-5.7 years. We correctly predict gender and age ($\pm$5 years) for 53% of our participants. Features extracted with knowledge of the character to player mapping does improve predictions of gender over character-based features alone. This is not the case for age. One can also restate this finding: a small number of features extracted from a player to character mapping (33 features) produces the same level of predictive performance as a substantially more detailed set of character-based features (289 features).

Future Work We intend to explore the prediction of less obvious demographic variables such as level of education, income or even personality. Finally, one could investigate how difficult it would be to estimate each feature in-game rather than relying on Blizzard's WoW armory. It would also be interesting to develop and study similar feature sets for a different game to explore if these findings are generalizable across games and genres.

[4] Roughly defined as the tendency for individuals to associate with individuals who are similar to them.

References

1. NPD: Entertainment trends in america. http://npd.com/lps/Entertainment_Trends2009 (2009)
2. Blizzard Entertainment: World of Warcraft. (http://www.wow.com)
3. Blizzard Entertainment: World of warcraft subscriber base reaches 11.5 million worldwide. (http://us.blizzard.com/en-us/company/press/pressreleases.html?081121)
4. Hu, J., Zeng, H., Li, H., Niu, C., Chen, Z.: Demographic prediction based on user's browsing behavior. Proceedings of the 16th international conference on World Wide Web (2007)
5. Singla, P., Richardson, M.: Yes, there is a correlation:-from social networks to personal behavior on the web. Proceeding of the 17th international conference on World Wide Web (2008)
6. Tsukamoto, T., Nishiyama, H., Ohwada, H.: Itemspider: Social networking service that extracts personal character from individual's book information. LNCS KNOWLEDGE ACQUISITION (2009)
7. Jones, R., Kumar, R., Pang, B., Tomkins, A.: I know what you did last summer: query logs and user privacy. Proceedings of the sixteenth ACM conference on information and knowledge management (2007)
8. Herring, S.: Two variants of an electronic message schema. Pragmatics & beyond. New series (1996)
9. Koppel, M., Argamon, S., Shimoni, A.: Automatically categorizing written texts by author gender. Literary and Linguistic Computing (2002)
10. Schler, J., Koppel, M., Argamon, S.: Effects of age and gender on blogging. 2006 AAAI Spring Symposium (2006)
11. Ducheneaut, N., Yee, N., Nickell, E.: Alone together?: exploring the social dynamics of massively multiplayer online games. In SIGCHI06 (2006)
12. Yee, N., Ducheneaut, N., Nickell, E.: The life and death of online gaming communities: a look at guilds in world of warcraft. In SIGCHI07 (2007)
13. Yee, N., Ducheneaut, N., Nelson, L., Likarish, P.: Introverted elves and conscientious gnomes: The expression of personality in world of warcraft. In SIGCHI11 (2011)
14. Bessiere, K., Seay, A., Kiesler, S.: The ideal elf: Identity exploration in world of warcraft. Cyber Psychology and Behavior (2007) 530–535
15. Nardi, B., Harris, J.: Strangers and friends: Collaborative play in world of warcraft. CSCW06 (2006)
16. Williams, D., Yee, N., Caplan, S.: Who plays, how much, and why? debunking the stereotypical gamer profile. Journal of Computer-Mediated Communication (2008)
17. Grimes, G., Bartolacci, M.: Second life: A virtual world platform for profiling online behavior for network and information security education? an initial investigation. International Journal of Interdisciplinary Telecommunications and Networking (2010) 60–64
18. Nokelainen, P., Tirri, H., Miettinen, M., Silander, T., Kurhila, J.: Optimizing and profiling users online with bayesian probabilistic modeling. Proceedings of the International Networked Learning Conference of Natural and Artifical Intelligence Systems Organization (2002)
19. Blizzard Entertainment: WoW Armory. (http://www.wowarmory.com)
20. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. Advances in kernel methods: support vector learning (1999) 185–208

21. Smola, A., Schoelkopf, B.: A tutorial on support vector regression. NeuroCOLT2 Technical Report NC2-TR-1998-030 (1998)
22. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The weka data mining software: An update. SIGKDD Explorations **11** (2009)
23. M.G.: A blizzard of protest over privacy. (http://www.economist.com/blogs/babbage/2010/07/online_gaming)

A Top 50 features for gender

Rank	Feature	Set	Rank	Feature	Set
1	perc_male_chars	Player	26	defenses_armor	Player
2	fem_male_diff	Player	27	stat_armor	Player
3	num_male	Player	28	isolates	Progression
4	num_female	Player	29	melee_expertise	Player
5	char_gender	Character	30	heal_recd	Character
6	perc_tank	Player	31	arena_played	Player
7	perc_alliance	Player	32	blows_bg	Character
8	fewest_achievs	Player	33	prime_role	Player
9	rel_hugs	Player	34	hon_kills_world	Character
10	melee_main_dps	Character	35	hon_kills_total	Character
11	perc_range_dps	Player	36	total_dmg_dealt	Character
12	respecs	Character	37	duels_won	Character
13	perc_melee_dps	Player	38	equip_epic_items	Character
14	rel_pvp	Player	39	hon_kills_pvp	Character
15	hord_alli_diff	Player	40	central_close	Progression
16	stat_str	Character	41	char_class	Character
17	hit_recd	Player	42	num_hugs	Character
18	achiev_sum_pvp	Player	43	duels_lost	Character
19	total_dmg_recd	Player	44	stat_spi	Character
20	total_heal_recd	Player	45	melee_off_dps	Character
21	most_achievs	Player	46	deaths_raiddung	Character
22	melee_power_base	Character	47	transitivity	Progression
23	t_total_heal_recd	Progression	48	defenses_dodge	Character
24	t_total_dmg_recd	Progression	49	achiev_tab_pvp_total	Character
25	stat_stamina	Player	50	t_duels_won	Progression

Table 3. Most useful features for predictions of gender, as ranked by Information Gain, average rank across 10-fold cross validation.

B Top 50 features for age

Rank	Feature	Set	Rank	Feature	Set
1	perc_melee_dps	Player	26	achiev_tab_pvp_total	Character
2	duels_lost	Character	27	arenas_played	Character
3	duels_won	Character	28	t_dung_lk_25_bosses	Progression
4	perc_bg_wins	Character	29	dung_5play_entered	Character
5	t_duels_won	Progression	30	rel_pvp	Player
6	t_duels_lost	Progression	31	arenas_won	Character
7	need_rolls	Character	32	t_hon_kills_pvp	Progression
8	perc_need	Character	33	arenas_played	Character
9	respecs	Character	34	achiev_sum_pvp	Character
10	t_summons	Progression	35	summons	Character
11	most_achievs	Player	36	avg_lvl	Player
12	num_daysPlayed_month	Player	37	deaths_other_players	Character
13	t_need_rolls	Progression	38	t_blows_bg	Progression
14	fewest_achievs	Player	39	emblems_valor	Character
15	t_hon_kills_total	Progression	40	t_deaths_total	Progression
16	CallCrusade25_PlayerRaid	Character	41	LKDungeon	Character
17	CallCrusade10_PlayerRaid	Character	42	t_dung_lich_25done	Progression
18	prop_80	Player	43	t_deaths_falling	Progression
19	t_total_kills	Progression	44	dung_lich_10play_done	Character
20	greed_need_ratio	Character	45	blows_arena	Character
21	fem_male_diff	Player	46	achiev_tab_pvp_Arena	Character
22	t_total_heal_recd	Progression	47	t_total_dmg_recd	Progression
23	num_female	Player	48	deaths_warsong	Character
24	lvl_var	Player	49	t_lich_25_bosses_killed	Progression
25	t_deaths_other_players	Progression	50	rel_dungeons	Player

Table 4. Most useful features for predictions of age, as ranked by Information Gain, average rank across 10-fold cross validation.

Privacy: Gone with the Typing! Identifying Web Users by Their Typing Patterns

Prima Chairunnanda, Nam Pham and Urs Hengartner

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada N2L 3G1
{pchairun, npham, uhengart}@cs.uwaterloo.ca

Abstract. The lack of privacy protection for Internet users has been identified as a major problem in modern web browsers. Despite potentially high risk of identification by typing patterns, this topic has received little attention in both the research and general community. In this paper we present a simple but efficient statistical detection model for constructing users' identity from their typing patterns. Extensive experiments are conducted to justify the accuracy of our model. Using this model, online adversaries could uncover the identity of Web users even if they are using anonymizing services. Our goal is to raise awareness of this privacy risk to general Internet users and encourage countermeasures in future implementations of anonymous browsing techniques.

Keywords: Typing Pattern, Biometric, Privacy, Identity

1 Introduction

Among all the information, the personal information of users is being sought by several third parties, such as advertisers and identity thieves. Even the simple act of browsing inadvertently leaks a variety of information to the Internet. First, HTTP cookies stored on the user's computer can provide unique identifying information about the user. Fortunately, most browsers allow user to delete cookies, and a privacy-conscious user might choose to do so after every browsing session. Second, the user's IP can be used to infer user's location up to several hundred-kilometer accuracy [5]. A web proxy can provide a simple anonymizing service, but to defend against more sophisticated network surveillance or traffic analysis, the Tor network[1] needs to be utilized.

In spite of all these mitigation strategies, Eckersley was still able to uniquely identify the majority of visitors to his website [3]. This technique, called "Browser Fingerprinting", uses HTTP header values and other information made available to Javascript. One possible workaround suggested is to activate the private browsing mode offered by most modern browsers, effectively blocking some information from being sent out. Recent research by Aggarwal et al. [1], however, showed several shortcomings of private browsing mode implementations on

[1] http://www.torproject.org

four major browsers. Furthermore, Fioravanti [4] demonstrated that even if the browser spoofed the User-Agent and DOM information, intricate details about a browser's scripting environment were already sufficient to identify a user. Some browser vendors have taken positive steps to address this issue; Mozilla has announced plans to close some information leakage to prevent fingerprinting [10]. It has to be noted that at least in the case of Mozilla, some types of information leakage will not be closed due to its potential to provide an enhanced user experience (e.g. timezone).

The above works focused on the information leakage coming from technology and its implementation, but we would like to explore leakage coming from the user him/herself. This can be compared to employing social engineering tactics to attack a cryptosystem instead of directly attacking its implementation. We foresee the rise of this kind of attacks in the near future, as attackers try to find other ways to de-anonymize a user. Also, it has the advantage of being harder to prevent: browser vendors could easily fix their browsers to close loopholes, but they cannot "fix" their users. Some studies, such as [11], have indicated that user education might be the most effective way to address social engineering.

To be more specific, we are looking at users' typing pattern as the side-channel information we want to exploit. Internet activities require users to do a significant amount of typing, from entering his/her username and password in a login screen, typing to answer chat messages, to editing an article on Wikipedia. To improve user experience, many Web 2.0 websites immediately transmit any key strokes entered by the user directly to the website (e.g., Google Instant). Alternatively, a website could use Javascript to measure a user's keystroke timings while the user is filling in a form. These measurements will be sent to the website at the same time when the form data is submitted. We hypothesize that typing pattern analysis can reveal a user's identity to some extent. Combined with information gained from other leakage sources, it can potentially de-anonymize a user completely, regardless of the usage of private browsing mode, web proxies, or other anonymizing networks. In this study, we verify that such weakness exists, hence bringing attention to the security and privacy community.

1.1 Related Work

Using typing patterns to detect a unique user has been studied for a long time. The Allies in World War II developed a technique based on Morse code to distinguish their real operators from those pretending to be on their side [13]. This kind of techniques have been more formally researched since the 1980s, as mentioned in the comprehensive survey article by Peacock et al. [12]. Monrose and Rubin [9], for example, studied the keystroke dynamics, or typing rhythm pattern, as a biometric for authentication. Their classifiers were able to authenticate a user with an accuracy of 83.22% to 92.14%. Clarke and Furnell [2] further extended authentication using keystroke analysis to mobile devices. Though, mobile devices have inherently different usage patterns from normal computer keyboards; for example, phone numbers are frequently typed in, and some devices (including

the one used in their experiment) require users to press a key several times to produce a character.

In another study, Song et al. [14] were able to develop a Hidden Markov Chain Model based on inter-keystroke timings of users' typing during an SSH session to recover a significant amount of information on their passwords. Leggett and Williams [8] also performed an experimental evaluation of keystroke authentication utilizing various kinds of filters. However, most of the above researches require the user to type several pairs of characters repeatedly (30-40 times in [14]). In [8], users were asked to enter the same prose twice, the first one for profile creation, and the second one for testing. This approach might work in the case of authentication as the user must type the same pair of username and password every time, but it is unrealistic in the context of general Internet usage. To post the same paragraph on two different forums, for instance, a user could just post a URL referring to the first forum, or even simpler, copy-paste the whole paragraph content.

Various aspects of a user's typing pattern can be used for analysis, such as: digraph, trigraph, and tetragraph latencies (interval between two, three, and four successive characters) [8], overall typing speed, error frequency, and key-hold time [7]. We primarily use digraph latency information in our model, however it can be extended to consider the other aspects mentioned earlier. The problem of how best to aggregate information from different aspects is out of the scope of this paper, and we leave it for future work.

1.2 Contributions

Compared to previous studies, which focused on authentication, our study puts typing pattern analysis in a different light: to identify a user in the otherwise anonymous context of the Internet. The problems of authentication and identification are closely related, but they are not the same. In the authentication setting, the system typically knows which user is trying to authenticate, for example by means of a username. Also, the user is actively trying to pass the authentication in order to get some privileges reserved only for authenticated users. Contrast this with our identification scenario: the users are not aware that they are being profiled, and the system is trying to uncover who they are by searching over a set of known typing patterns.

With this paper, we make the following contributions: first, we show that keystroke dynamics can be easily harvested on webpages without the user being any wiser about such activity. Next, we devise a model in order to demonstrate that analysis of some keystroke dynamic features can identify a user with high accuracy rate. Finally, by analysing the method by which we collected the typing patterns, we are also able to outline several countermeasures to mitigate the problem.

This paper is organized as following: we will briefly describe the threat model, followed by an introduction to our statistical model for analyzing and detecting the typing dynamics of different users in Sect. 2. Our experiments and results

are presented in Sect. 3. We propose several mitigation strategies to the problem in Sect. 4. Finally, in Sect. 5 we give our conclusions and future directions.

2 Our Model

To verify our hypothesis, we started by exploring techniques for detecting typing patterns based on the time difference between two keystrokes (digraph latency).

2.1 Threat Model

In our setting, the typing pattern of each user is the main asset of interest to attackers. However, constructing a unique pattern for each individual requires a significant amount of sample data. In order to achieve it, the attacker could be a single powerful attacker who has frequent access to users typing on the Internet, such as a search engine or an email service provider. A collaboration effort among a group of attackers is also possible if each individual attacker knows the identity of the user. For example, advertisers might be willing to buy partial typing patterns of users from different sources and then combine them to use against anonymous browsing sessions.

2.2 System Model

The threat that we demonstrate in this paper is unlikely to affect all Internet users. In particular, it is unlikely to affect people who care about privacy to a great degree and who already, for example, use Tor for all their browsing or who have Javascript disabled in their browser. Instead, the threat mainly concerns the (probably larger) set of people who care about their privacy while browsing the web, but are willing to trade off privacy for usability and performance. For example, these users have their browsers accept cookies and have Javascript enabled since otherwise many web sites simply would not work. On the other hand, these users have their browser delete all cookies at the end of a browsing session. Also, they may use Tor in cases where it is really important to remain anonymous (e.g., while entering a particularly sensitive Google search query), but they do not use Tor for their everyday web surfing due to performance reasons. They also use their browser's private browsing mode (or an extension of it since current implementations have been shown to be vulnerable, cite paper that we talked about) to prevent a website from learning their detailed browser/OS fingerprint. Finally, they use an ISP that employs NAT to assign the same IP address to multiple users, which makes it difficult for a website to track a particular user based on IP address. (If NAT is not available, the users could use a simple proxy instead.) Due to these measures, in particular due to the last one, a user can arguably expect to remain anonymous among dozens or maybe even hundreds of users, for example, while making a Google query.[2] Unfortunately, as we demonstrate in this paper, this expectation is overly optimistic.

[2] Of course, this assumes that the user does not log in to Google. Otherwise, all bets are off.

2.3 Notations and Definitions

We introduce several notations in our model:

- X - a set of observations of a user's typing timestamps
- $T_X[a][b]$ - the list of observed digraph latencies, or the time difference in milliseconds, between two characters a and b.
  For example, T_X['e']['r'] $=< 69, 75, 85, 90, 94 >$ means that in observation X, the user typed the pair ('e','r') five times.
- $D_X = \{(a, b)|length(T_X[a][b]) \geq \beta\}$ - the dimension, or set of character pairs in observation X with no less than β data points. The value β is the *Minimum Data Points (MDP)* parameter of our model, which controls the minimum number of observations required for a character pair in order to be included in our calculation. We set $\beta = 5$ for now, and we will discuss the effect of this parameter in Sect. 3.5.
  For example, given T_X['e']['r'] $=< 69, 75, 85, 90, 94 >$,
  T_X['a']['n'] $=< 94, 94, 101, 132, 146 >$, T_X['i']['t'] $=< 87, 89, 101, 134 >$,
  T_X['e']['s'] $=< 75, 79, 83, 89, 93, 99 >$, then $D_X = \{$('a','n'), ('e','r'), ('e','s')$\}$
- $D_{X,Y} = D_x \cap D_y$ - The shared dimension between observation X and observation Y, where the observations might be made on the same or a different person.
  For example, given $D_X = \{$('a','n'), ('e','r'), ('e','s')$\}$ and
  $D_Y = \{$('a','r'), ('e','r'), ('e','s'), ('o','n')$\}$, then $D_{X,Y} = \{$('e','r'), ('e','s')$\}$
- $\|P\|$ - The size of a collection P (can be a set or a list)

2.4 The Detection Model

In our model, we rely on the Kolmogorov-Smirnov test (K-S test) to determine if two observations X and Y belong to a same user. We define $KS(P, Q)$ to be the probability that P and Q are samples from the same distribution based on the Kolmogorov-Smirnov test. In our setting, we make an assumption that the latency between every character pair is independent of those of other pairs. This assumption allows us to approximate the likelihood that X and Y are sample data from one user as:

$$\prod_{(a,b)\in D_{X,Y}} KS(T_X[a][b], T_Y[a][b]) \tag{1}$$

However, this score has several problems. First of all, notice that $KS(P, Q)$ is a probability, thus its value has the range [0,1]. It is therefore theoretically possible for the value to be 0, especially if $\|T_X[a][b]\|$ or $\|T_Y[a][b]\|$ for some pair (a, b) is relatively small. Unfortunately, the overall product will then be 0, regardless of the values of the $KS()$ function for other pairs in $D_{X,Y}$. We need to limit the effect of a mismatch, so that it does not penalize our likelihood function too much. We do this by introducing the *Minimum Sampling Correlation (MSC)* parameter α. Hence our likelihood function $L(X, Y)$ is defined as follows:

$$L(X, Y) = \prod_{(a,b)\in D_{X,Y}} max(KS(T_X[a][b], T_Y[a][b]), \alpha) \tag{2}$$

where $0 \leq \alpha \leq 1$. While α can be as high as 1, it is worth mentioning that normally we do not want α to be too high such that it is too forgiving on matching failures. In our experiments, we set $\alpha = 0.001$ unless noted otherwise.

The second problem stems from the fact that the sizes of X and Y are not equal and different pairs of (X, Y) potentially have different shared dimensions. Since $KS(T_X[a][b], T_Y[a][b])$ is in the range $[0, 1]$, the more terms you multiply together, the smaller the value becomes. Consequently, $L(X, Y)$ then becomes predominated by $\|D_{X,Y}\|$ rather than the amount of information they share. In order to solve these problems, we introduce a new improved similarity score function:

$$score(X, Y) = \frac{log(L(X, Y))}{\|D_{X,Y}\|} \tag{3}$$

The range of the similarity score is heavily influenced by the $KS(P, Q)$ values. The highest possible score is 0, which occurs when all $KS(P, Q)$ return 1 (i.e. P and Q are sampled from the same distribution based on the K-S test). On the other end, the lowest possible score occurs when all $KS(P, Q)$ return 0 (i.e. P and Q have no chance of being sampled from the same distribution based on the K-S test). When this happens, $L(X, Y)$ will evaluate to $\alpha^{\|D_{X,Y}\|}$, and in consequence, $score(X, Y) = log(\alpha)$.

We can then conclude that the value of $score(X, Y)$ has the range of $[log(\alpha), 0]$ (from 'totally different' to 'exactly the same'). Note that when $\alpha = 0$, the range of the similarity score will be $(-\infty, 0]$.

3 Experiment

We seek to answer the following questions:
1. Can we accurately match each user's typing pattern to his/her profile?
2. Can we accurately identify a new user whose typing pattern does not exist in our set of profiles?
3. Can we accurately match typing patterns of a user collected at different times?
4. What are the effects of parameters *Minimum Sampling Correlation (MSC)* and *Minimum Data Points (MDP)* on our model?

3.1 Experiment Setup

We recruited 36 participants through mailing lists and personal invitations. All participants are frequent Internet users. The participants were asked to visit a website we set up to collect typing patterns[3]. All experiments were conducted exclusively through this online website, and participants could view full information about the study before agreeing to take part.

In the experiment, each participant was asked to retype a randomly selected sequence of English words from a database of 1340 words. These words were

[3] This study has received approval from the Office of Research Ethics at University of Waterloo (ORE# 16814)

chosen from the top 1000 Internet search terms as ranked in [6]. We used Internet search terms as our basis because we felt that those words would be more likely to be familiar to the participants. Some words that we found to be unfamiliar or inappropriate were also filtered out. While typing, the timestamp of each keystroke was collected by means of Javascript. Once the Enter key was pressed to signify completion, the timing data was sent to our website together with the actual text typed by the participant.

This setup mimics exactly how we imagine our attacker to be. The user types some texts in an innocent-looking textbox, and without his/her knowledge, the typing pattern is collected and sent out as part of the form submission data. There is no additional round of communication involved to avoid rousing users' suspicion, although with AJAX becoming more popular nowadays, users might be led to believe the extra communication is part of the interactivity system to provide better customer experience. Ironically, in our putative scenario the extra communication is actually for a completely opposite purpose.

There were two different phases in our study. There was no difference in the conduct of the experiment between the two phases; only the amount of data gathered differed. For the first phase, each participant was asked to do 20 trials, each consisted of entering 50 characters on average. 28 participants took part in the first phase.

We invited 13 participants to take part in the second phase, 5 of whom took part in the first phase. This phase was conducted several days after the first phase on a different website URL. The task given was exactly the same as in the first phase, but there were 50 stages, consisting of entering 100 characters each. Therefore, for each participant, we collected approximately five times as much data from the second phase as from the first one.

For clarity, we will use the following convention to refer to our participants. Participants who took part in the first phase are each assigned a unique number from 1-28. Those taking part in the second phase are each assigned a unique character from A-L. For the five participants who did both phases, we will use numbers to refer to their typing pattern collected during the first phase, and characters to refer to their typing patterns collected from the second phase. The number-to-character pairings for those five are: {(22, A), (19, B), (18, C), (9, D), (2, J)}.

3.2 Model Accuracy

We would like to answer the first question posed regarding the accuracy of our model. We created a custom-made Java application to digest the typing timing information and compute the similarity score between two typing timing patterns. The Java Statistical Classes (JSC) library[4] is used to conduct the Kolmogorov-Smirnov test. We then conducted 5-fold validation on the data obtained from the second phase of the experiment. Unfortunately, we found that data from one participant was corrupted, therefore we could only work with the

[4] http://www.jsc.nildram.co.uk/

remaining 12. The validation showed that our model gave the highest similarity score to the participant from whom the training set was built. All 5 validation runs return similar results, and one of them is depicted in Fig. 1.

In Fig. 1, the triangles denote the best similarity score for each participant's data, while the bars represent the range of other similarity scores. It can be seen from the figure that there is a significant difference in the similarity score between the best match (μ=-0.85, σ=0.17) and the other candidates (μ=-3.90, σ=1.09). The difference is still significant even when we compare it only against the next best match (μ=-2.19, σ=0.45). The lowest similarity score achieved for a correct match is -1.06 for participant G, but the next best match scored nearly thrice at -3.10. The highest similarity score for an incorrect match is -1.63 between participant B and D, but then the correct match for participant B scored only half of it at -0.83. Overall across the board, our model can easily distinguish a correct match from the incorrect one. We realize that the difference in score might be less significant on a bigger scale involving thousands of participants, resulting in several close matches. Other traditional detection methodologies (e.g. IP address, browser's User-Agent, etc) can then be used to refine the results to reveal the actual user.

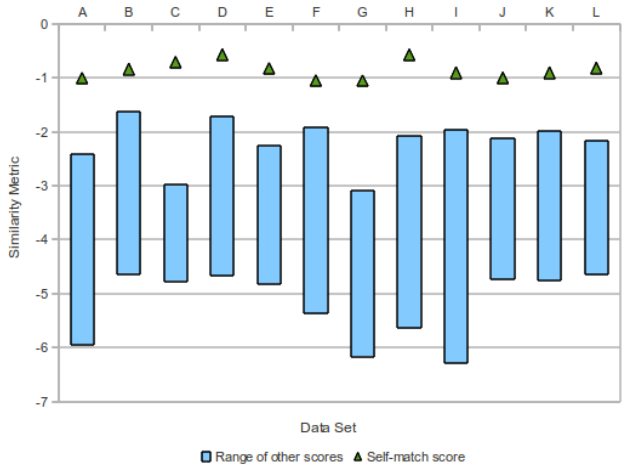


Fig. 1. Similarity scores for 12 second phase participants. A score closer to 0 means a better match. The triangles indicate the best score, while the bars represent the range of other scores for each participant. The top of the bar, therefore, can be seen as the next-best score. We can see a significant difference between the best and the next-best score across all participants.

3.3 New User Identification

We are then interested in knowing if our model could identify a new user whose typing pattern has never been profiled before. We cannot simply take the highest similarity score as our "correct" match: this could be an entirely new user. In reality the importance of this capability really depends on the adversary and his/her goal for user identification in the first place. An adversary whose goal is to track the activity of a group of anonymous users in a forum, for example, needs to be able to accurately distinguish those users from other website visitors. However, if the goal is just to show advertisements tailored for a specific user, it will not matter as much. Any advertisement can be shown to the new user, so it does not make a difference whether it was tailored or picked at random.

The problem of new user identification can be seen as a classification problem with two classes: *existing users*, and *new users*. For simplicity, we used the similarity score as the only feature considered, and performed the classification based only on whether the score exceeds a threshold value τ. For the purpose of our experiment, we selected the threshold value by empirical observation from the previous experiment. We chose the mid-point between the lowest similarity score for a correct match (-1.06) and the highest similarity score for an incorrect match (-1.63), yielding a threshold value τ = -1.345. We admit that this may not be the best threshold selection method, but given the limited data points we have, we choose to leave threshold selection analysis as future work.

We started by creating profiles for the 12 participants of the second phase study, using random selection of 4000 out of 5000 character data. The reason for this is because we wanted to make the similarity score comparable to the ones in Fig. 1. Recall that in each iteration of 5-fold cross validation, four-fifths of the data set is used as training set to validate the remaining one-fifth.

Afterwards, the typing patterns collected from the first phase were compared against the 12 profiles. Note that only 23 out of 28 typing patterns were used for this experiment, because the remaining 5 were also participants of the second phase, thus they are not *new users*. The best similarity score for 10 of the 23 *new users* are shown in Table 1 ordered by the scores. The remaining 13 have worse best score than the lowest shown in the table, and we purposely omit them for clarity. As can be seen from the table, not even the largest score passes the threshold value τ and our model correctly identifies all new users. In other words, there were no false-positives during our experiment.

3.4 Matching Typing Pattern Collected at Different Times

Another point which comes to mind is to see if a user's typing pattern is identifiable over time. We acknowledge that there will be variance when a user uses different input methods (e.g. keyboard vs. stylus) or even when different keyboard layouts are used. These aspects warrant a further detailed study on its own, but for now we would like to focus on something much more fundamental. That is, given a set of user profiles obtained from an observation, can we still correlate a user's typing pattern observed at a different time to his/her profile?

Table 1. Best similarity scores for 10 *new users*

Participant ID	Best Match	Similarity Score
21	D	-1.62
12	J	-1.66
5	F	-1.76
27	D	-1.83
14	E	-1.90
16	L	-1.93
15	G	-2.08
25	H	-2.14
4	D	-2.17
7	J	-2.18

To answer this question, we used the data of five users who did both phases of the experiment. We started with the 12 profiles we had created for the previous experiment. Recall that we also had data of those five users from the first phase, and this data was never considered in the creation of the profiles. We emphasize that there were several days of gap between the first and the second phase. From there, we wanted to see if our model could match those five users to their corresponding profiles.

We found that we could successfully match 4 out of 5 users to their profile, while the remaining one was detected to be a *new user* based on the threshold τ we set earlier. This result is shown in Table 2. One participant specifically told us that she did not think we could correlate her typing pattern because she was very tired and made a lot of mistakes in the second phase, but it turned out we could still match her correctly.

We also investigated the only participant whose typing pattern failed to be matched to his profile. His typing pattern similarity score of -1.655 was quite good but still below the threshold. We manually checked the data we gathered from this participant, and we were surprised to find that there were noticeably a lot of typing errors made in his second phase experiment. This could be because of the length of the experiment, or perhaps the participant tried to induce some noise into the data. Nevertheless, we are encouraged by the fact that our model could identify 4 out of the 5 participants, and even the one with a large number of errors still got relatively good similarity score albeit it was below the threshold.

3.5 Effect of Model Parameters

Our similarity score model depends on two important parameters: the Minimum Sampling Correlation α, and the Minimum Data Points β. We initially set each parameter to a value that we estimated to be "sufficiently useful" to fulfill its intended purpose. In this subsection, we strive to explore the effect of these values on the accuracy of our model. We refer a discussion of the threshold parameter τ to future work. Recall that the threshold τ is used by our simple classifier to differentiate between new users and existing users. A more elaborate classifier,

Table 2. Classification results for the 5 users who did both phases, using their first phase data against their second phase data

Participant ID	Best Match	Similarity Score	New User/ Existing?	Correct?
9	D	-0.699	Existing	Yes
22	A	-1.182	Existing	Yes
19	B	-1.190	Existing	Yes
18	C	-1.206	Existing	Yes
2	J	-1.655	New	No

though, may take more features into account, such as the dimensionality used to produce the score, or the history of the similarity score from previous matches.

As we have discussed in Sect. 2.4, we introduce the MSC parameter α to limit the effect of a mismatch on one dimension to the other dimensions. Also, α has the side effect of providing the lower limit of the similarity score function. When α is 0, $score(X, Y)$ may evaluate to $-\infty$ which may not always be suitable for plotting or further processing. Table 3 shows the similarity score values between selected pairs when α is 0.001 and 0. Introducing MSC also allows us to deduce some information which would have been lost otherwise. Had we set $\alpha = 0$, we could only say that participant 28's typing pattern does not match profile E and G. However, setting $\alpha = 0.001$ reveals that participant 28's typing pattern is significantly closer to E than to G.

Table 3. Effects of MSC value α on similarity scores between selected pairs

Participant ID	Profile ID	Similarity Score	
		$\alpha = 0.001$	$\alpha = 0$
3	E	-4.45	$-\infty$
	F	-5.71	$-\infty$
	G	-3.10	-3.64
8	E	-3.66	-4.69
	F	-3.30	-4.05
	G	-4.91	$-\infty$
28	E	-3.84	$-\infty$
	F	-2.18	-2.35
	G	-5.50	$-\infty$

Our model also depends on the MDP parameter value β, which filters out dimensions containing too few data to be useful. Some pairs of characters occur very infrequently in words, and inevitably they cannot be considered for processing. In particular, the Kolmogorov-Smirnov test may not be meaningful if there are too few data to compare. We now would like to see if varying β will have an impact on accuracy. For the purpose of this experiment, we use data

from the 12 participants of the second phase. We performed 5-fold validation for different values of β, and the result is depicted in Fig. 2.

With the profiles from 12 participants, we can see that the precision starts to drop between β values of 12 and 15. It deteriorates rapidly with only around 75% accuracy for β=20, and 35% for β=25. We then look at the number of dimensions being used to calculate the similarity score. We can clearly see that the number of dimensions being considered starts with 250 for β=1, then drops nearly half when we increase β to 2, and only around 20 for β=10. This shows that a big portion of the dimensions have only few data points. In spite of this, the distribution of samples from other dimensions is still enough to accurately identify a user's typing pattern. Only when there are too few dimensions to explore does the model start to be inaccurate (there are only 12 dimensions considered for β=12).

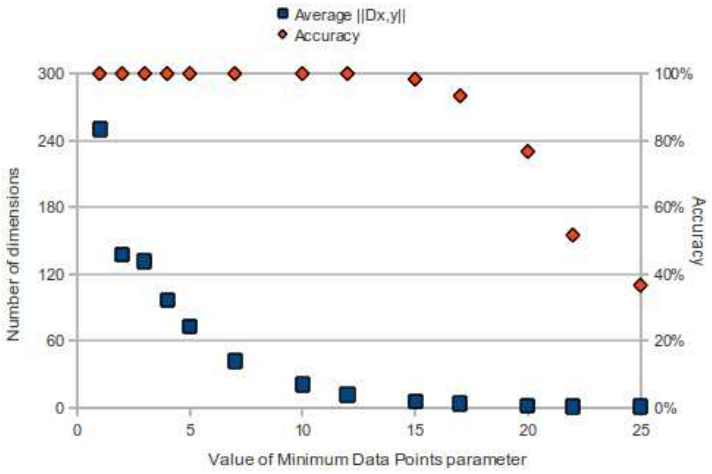


Fig. 2. The effect of MDP parameter β on average $\|D_{X,Y}\|$ and overall accuracy. Increasing β past a certain point brings down accuracy considerably.

4 Countermeasures

We have shown through experiments that users' typing pattern can be efficiently collected and used to identify the user again. Fortunately, the user is not completely defenseless. We have identified several ways using which the risk can be mitigated.

First, a user might simply alter his/her typing pattern, or introduce random noise as one participant in the second phase might have done. That particular participant was able to marginally avoid our detection, but more noise is required to completely avoid detection. This may be difficult to do in all circumstances,

but the user only needs to do this when he/she needs anonymity with respect to the content of the text. Another variation of this is to combine keystrokes with cursor movements. For example, a user can write a word omitting several middle characters, then use the mouse and/or arrow keys to go to the middle of the word and fill in the remaining characters. This method, however, is very time consuming and may not be very applicable if the user needs to write a whole lot of texts.

Disabling Javascript can certainly beat our data collection system, but websites have grown from simple content providers to become a platform. AJAX plays a big role for increased interactivity on websites, and more and more applications are delivered over the Internet as a webpage. Therefore, we felt that this approach is not feasible.

The simplest and perhaps the most effective way is for the user to write the texts with a separate program, such as a text editor, and then paste the text into the browser window. Assuming that there is no keylogger or spyware on the user's computer collecting the typing pattern, this method is foolproof. However, this requires that the user knows that he wants to protect his anonymity before he even started writing. If the user has written half-way in the browser, then writes the other remaining half in a text editor, the collected timing information might still reveal the user's identity.

Modern browsers becoming more open and offering more ways for customization provides another countermeasure. We could develop a Firefox extension that automatically scrambles the timing of keystrokes as user enters the text. The copy-and-paste approach can also be employed – when the user clicks on a text box, the extension could overlay another textbox on top of it. Once focus moves to another element, the text is pasted into the actual textbox below, and the overlay is removed. This way, it will be almost seamless to the user.

5 Conclusions

Broadly, our paper made the following contributions: First, we analyzed the feasibility of identifying web users by their typing patterns. Based on the Kolmogorov-Smirnov test, we constructed a new statistical detection model, and used it to measure the similarity between the keystroke dynamics of two web users.

Additionally, we demonstrated the accuracy of our detection model by conducting an extensive set of experiments with real typing statistics of 35 participants. The participants were asked to type several English sentences on our website at their normal typing speed. Our detection model was able to accurately identify both *existing users* and *new users*. The results confirmed our initial hypothesis that we could use typing pattern as a mean to de-anonymize a user. These findings further suggested the existence of yet another side-channel attack to privacy on the Internet. Several countermeasures to this problem were also proposed and discussed.

6 Future Work

At the current stage, we have done a pilot test for our model with a small number of participants. We are conducting a larger-scale experiment with hundreds of participants. We are also examining other aspects of typing patterns, especially the trigraph latency and key-hold time. While we used digraph latency for our experiments, a combination with other aspects might produce higher identification accuracy.

There are also several directions for future works. First of all, in the current implementation, we use the Kolmogorov-Smirnov test to calculate the probability that two typing patterns belong to a same person. It would be interesting to apply other methods, such as Hidden Markov Models or Support Vector Machines to calculate the similarity score. Second, even though our detection model is able to detect accurately a user whose typing profile has been collected before, it still has to rely on a good choice of threshold τ to detect a new user. In our experiments, we were able to heuristically pick a value for τ based on empirical observations and use it to separate *new users* from *existing users*. However, we feel it is desirable to have a more structured and elaborate approach to distinguish the two.

It would be interesting to use our detection model in practical applications, such as authentication or theft prevention. For example, it could be developed as a security software running on background. If an unauthorized user tries to use the computer, his typing patterns could reveal his identity, hence logging out the session. It could also be developed into an optional plugin to email accounts, where illegal access by a third party could be prevented.

References

1. G. Aggarwal, E. Bursztein, C. Jackson, and D. Boneh. An analysis of private browsing modes in modern browsers. SYYM '10, Washington, D.C., USA, 2010.
2. N. L. Clarke and S. M. Furnell. Authenticating mobile phone users using keystroke analysis. *Int. J. Inf. Secur.*, 6, December 2006.
3. P. Eckersley. How unique is your web browser? Technical report, Electronics Frontier Foundation, 2009.
4. M. Fioravanti. Client fingerprinting via analysis of browser scripting environment. Technical report, 2010.
5. F. Holzhauer. IP geolocation. Technical report, TU Berlin, 2007.
6. HowRank. Top 1000 internet search terms. `http://www.howrank.com/top-1000-internet-search-terms.php`.
7. J. Ilonen. Keystroke dynamics. *Advanced Topics in Information Processing*, 2003.
8. J. Leggett and G. Williams. Verifying identity via keystroke characteristics. *Int. J. Man-Mach. Stud.*, 28, January 1988.
9. F. Monrose and A. D. Rubin. Authentication via keystroke dynamics. CCS '97, Zurich, Switzerland, 1997.
10. Mozilla. Fingerprinting. Technical report, 2010.
11. G. L. Orgill, G. W. Romney, M. G. Bailey, and P. M. Orgill. The urgency for effective user privacy-education to counter social engineering attacks on secure computer systems. CITC5 '04, Salt Lake City, UT, USA, 2004.

12. A. Peacock, X. Ke, and M. Wilkerson. Typing patterns: A key to user identification. *IEEE Security and Privacy*, 2:40–47, 2004.
13. J. Richards. Security at your fingertips. Technical report, 2007.
14. D. X. Song, D. Wagner, and X. Tian. Timing analysis of keystrokes and timing attacks on SSH. SSYM '01, Washington, D.C., USA, 2001.

Psychic Routing: Upper Bounds on Routing in Private DTNs

Jonathan Anderson and Frank Stajano
University of Cambridge
firstname.lastname@cl.cam.ac.uk

May 30, 2011

Abstract

We present early work investigating a one-way delay-tolerant communications channel which affords its users perfect unobservability at the price of a limited bitrate. We suggest an unrealizable protocol, *Psychic Routing*, against which we can compare the performance of concrete Delay-Tolerant Networking routing schemes. We then use Psychic Routing to evaluate the performance of routing in our perfectly unobservable channel.

1 Introduction

Today's social networking sites provide users with fast, convenient access to shared user experiences, but they also share private information more widely that users may intend [3, 4]. We have previously described the Footlights social networking system, albiet under a different name [1], which seeks to provide users with a service whose availability and performance are even higher than today's services, but which does not blatantly violate user privacy.

However, whatever the good intentions of the authors, such a system provides an opportunity for service providers to link users to encrypted data and, from there, to each other. In order to subvert such detection, going beyond privacy and into the realm of providing anonymity properties, Footlights will also provide users with a low-bitrate communications channel that is *perfectly unobservable*: even a global adversary is unable to determine whether or not the channel is being used, despite detecting all communications in the network.

This system relies on users constructing a Delay Tolerant Network (DTN) [7] and forwarding a certain amount of traffic on others' behalf. Routing in such a network is very difficult, so we have attempted to define the criteria of success via the concept of *Psychic Routing*, an unrealizable protocol which provides us with something that we believe is currently missing from the literature: an unattainable upper limit, *à la* Shannon limit, with which we can compare new protocols.

In this early work, we explore the concept of Psychic Routing and its relevance to routing in private DTNs.

2 Footlights and Perfect Unobservability

In order to put users in control of their private information, we have proposed an architecture for a privacy-enabling social networking system [1]. This system, called Footlights[1], provides confidentiality and integrity properties through cryptography, while relying on centralised infrastructure such as Content Delivery Networks for availability.

In this system, private data is stored as fixed-length encrypted blocks in a centralised, highly-available store. Blocks are content-addressed—therefore immutable—as in the Venti archival store [16], and are organized in a directed acyclic graph, as in the Git version control system [15]. Explicit linkages between blocks are revealed only by plaintext, so the presence of ubiquitous encryption will prevent services from intentionally and explicitly revealing user data to data miners [4] or the world at large [3]. We have mitigated the most obvious privacy attacks that affect real-world systems today but, in anonymity terms, our use of a centralised block store has introduced a global adversary [5] into the system.

We pessimistically assume that our block-oriented communications substrate—which in practice will be based on a CDN and backing store from a service provider like Amazon—is able to uniquely identify every user of the system by the IP address that they connect from. Thus, the operator of the block store can identify who has uploaded any particular encrypted block as well as who is downloading it. Given this assumption, we can safely model our system as a set of one-way messages: if Alice stores a 4 kB block that is later read by Bob, that is effectively the same as Alice sending Bob a 4 kB message through a medium that is being observed by a global adversary; this duality is illustrated in Figure 1. If Bob then stores a block of his own on the server, which is later read by Carol, the adversary can observe that Alice has talked to Bob and Bob has talked to Carol, but the contents of those messages are only known to the communicants.

It is worth noting that, in such a system, the graph of users (nodes) and messages (edges) may be disjointed: Alice, Bob and Carol may form a clique that does not communicate with the rest of the user population. The system can observe such communication patterns, but the presence or absence of cliques does not change any of the analysis that follows.

For the sake of plausible deniability[2], we use blocks which can take on a

[1] So named because, like a strip of theatrical footlights, it helps users to define the interfaces between themselves and their diverse audiences, *à la* Goffman [10]. The name "Footlights" is also traditionally associated with a Cambridge comedy troupe, although our project has no affiliation with said ensemble.

[2] It is important for users to be able to hide very small amounts of data in blocks that allow the same block to be interpreted differently by different recipients. If all blocks are of a pre-specified size, there is a place for random padding, which may be purely random, or may in fact be ciphertext saying, "there's more to read over here." Further details are available in

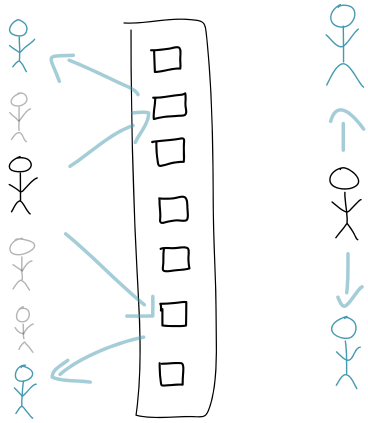


Figure 1: Footlights communication model.

limited number of sizes, perhaps even one fixed size of e.g. 4 kB. Because of this, we expect to find "spare" bits in these communications which could be used to carry covert traffic. This spare capacity can be regarded as a Delay Tolerant Network (DTN) [7], which could be used to route data around the network via intermediaries such that the global adversary cannot observe the hidden communications. In the above example, Alice might use the fact that she is sending Bob a message to also say, "Bob, the next time you talk to Carol, please tell her something for me." This message routing is *perfectly unobservable*—from the adversary's perspective, Alice has sent Bob one message, and Bob has sent Carol another message, but the number, size and timing of messages is unchanged whether the cover traffic is included or not. Only the content of the messages will change, so assuming that we use a good cipher, *zero bits of information* are conveyed by the presence or absence of covert traffic.

Having developed a DTN substrate upon which covert communications can occur, we must consider another problem: how to route Alice's message to Carol.

3 Psychic Routing

Delay Tolerant Networking is an umbrella term which describes a heterogeneous collection of network types, ranging from interplanetary networks to opportunistic Bluetooth contacts [7], and many routing schemes have been proposed for these various types of DTN [19]. When communications opportunities are fully deterministic, as in the case of the interplanetary network, models can be built and deterministic routes selected [8, 11]. In stochastic networks, two protocols are commonly used as benchmarks: Epidemic Routing, a form of controlled network flooding [18] and PRoPHET, a probabilistic scheme which keeps track of the likelihood that a node will have contact with other nodes [14].

Anderson et al. [1].

The trouble is, when developing a routing protocol for our one-way opportunistic network, it is unclear whether "delivers 15% more data than PRoPHET" is really very good; perhaps the existing protocols perform miserably on our network because their assumptions (e.g. near-instantaneous two-way sharing of routing probabilities) are not met, and thus we have no cause to celebrate being slightly better than them. What we need is something like a Shannon limit [17], a theoretical maximum that can never be reached, but which progressively better routing protocols can approach more and more closely. Such a limit would provide a real sense of both how far we've come *and* how much further we might yet go.

We propose that such a limit can be expressed via *Psychic Routing*: the most efficient routing of data that could be done today if one had full knowledge of future events and a protocol which imposes no communication overhead. Clearly, such a scheme is impossible to implement in practice, but a protocol with good probabilistic estimation of future events, based on statistics of past events, could begin to approach the upper bound imposed by Psychic Routing.

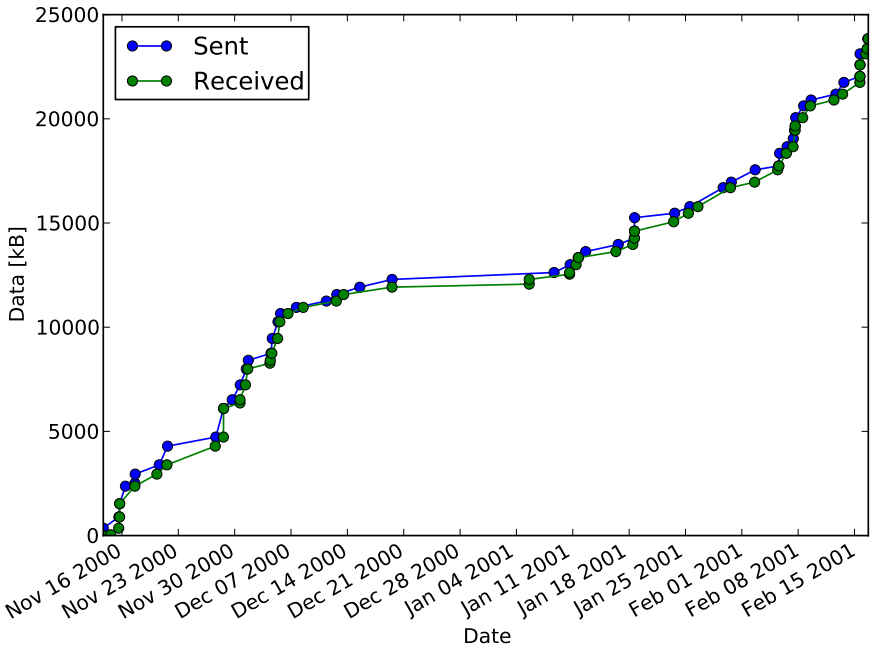


Figure 2: An example of maximal data transfer between two nodes in a DTN.

Psychic Routing does not provide general, closed-form equations like the Shannon limit in an AWGN channel; rather, it provides an upper bound on the pairwise performance of a routing protocol in a particular context, with particular parameters. For instance, Figure 2 shows the psychic limit for hypothetical communication using the Footlights system from Section 2 as a communications

substrate, between two people in the Enron e-mail corpus [12] who have been selected at random. This is an absolute maximum: no routing scheme will be able to transfer more data in the same time, or the same data in less time.

Psychic Routing only considers the maximum flow of data from one source node to one sink, it only provides pairwise maxima; it is not a global network-wide optimisation. The model can assume that e.g. some of the channel's capacity has been consumed by other traffic, but global optimisation for network-wide properties such as "fairness" is firmly in the realm of future work.

3.1 Calculation

Figure 2 is generated by walking "backwards" from the destination node (here denoted Bob), applying a mark to all e-mails that Bob received in the Enron corpus. We then walk backwards from the senders of *these* messages, marking all messages that could have "influenced" a message sent to Bob. We continue walking backwards, recursively, building a set of possible routes to Bob, stopping when we discover cycles in the graph. We then discard all routes that do not contain Alice, and throw away any portion of a route the precedes Alice.

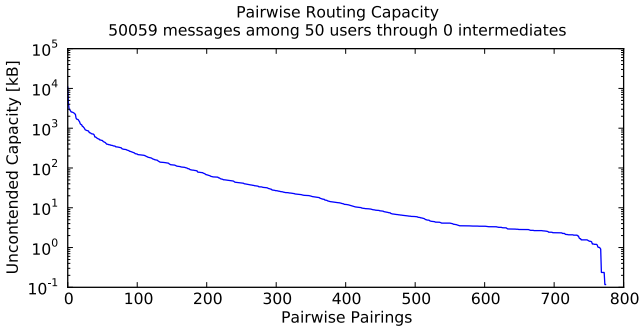
We then calculate a capacity of "spare" bytes—the number of bytes required to pad the message to a fixed block size, in this case, 4 kiB—for each message in a potential route. Each of these point-to-point messages, and associated capacity, can be seen as an edge in a *flow network*; Psychic Routing corresponds to pushing the *maximum flow* [9] from source Alice to sink Bob.

More precisely, for each instant x on the X axis, there will be a specific flow network, formed by the messages that emanated from Alice and its descendants up to that time; and the y value for that x will be the solution of the maximum flow problem for that flow network. The meaning of that (x, y) data point is that, if every participant had cooperated and used the spare capacities of the available messages in the most favourable way towards that goal, the maximum amount of data that could have been transferred from Alice to Bob by time x (without altering the observable pattern of messages that were to be sent, regardless of this covert communication) would be y.

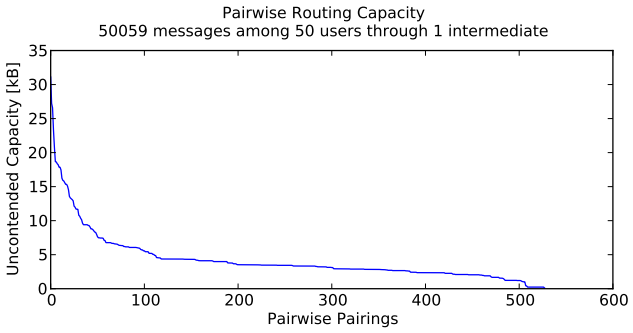
3.2 Related Work

The use of unattainable maxima as comparison points is well-established in the communications and computer science literature. The oft-cited Shannon limit—the most popular of which, applied to an Additive White Gaussian Noise (AWGN) channel, is sometimes treated as synonymous with "Shannon limit"—specifies the most error-free information transmission that is possible over a noisy channel [17]. Real communications systems cannot reach the Shannon limit, but it provides a useful absolute comparison point: we can say that an error correction code comes within 0.3 dB of the Shannon limit, rather than "10% better than code X."

Similarly, Belady described an unimplementable page replacement algorithm that real paging algorithms can be compared against [2]. Belady's MIN algo-



(a) Direct messaging.



(b) Routing through exactly one intermediary (data laundering).

Figure 3: Routing capacity over two years (no contention).

rithm, much like Psychic Routing, relies on full advance knowledge of what Virtual Memory (VM) pages will be required by the system in order to make optimal decisions about which pages to swap out of memory. A realizable algorithm such as Least-Recently-Used (LRU) that approaches the performance of Balaly's MIN algorithm in a variety of VM workloads can be deemed appropriate for concrete systems.

4 Measurement

In order to test the applicability of Footlights' covert DTN to real-world traffic, we have driven a Footlights model with data from the Enron e-mail corpus [12]. We first took a subset of the most "interesting" e-mails in the corpus[3]: 50,059 messages among the 50 users that communicated the most in the period 1999-

[3]This reduction was performed in the interest of computation time: it would simply take too long to process all 300k e-mails in the corpus, many of which are single messages to or from e-mail addresses outside of Enron (e.g. an invitation to one Enron employee to attend a University graduating class reunion).

2002. Each e-mail in the corpus was then translated into a Footlights message with a spare capacity of $4096 - l \bmod 4096$, where l is the length of the e-mail, including SMTP headers.

Figure 3 shows the routing capacity of the network; it is able able to support some limited communication. In these graphs, we see how much data Alice can send to Bob—for all possible values of Alice and Bob—either by sending it directly to Bob (Figure 3a) or via some intermediary (Figure 3b). The x axis represents ranked pairings of Footlights users: pairing 0 is the "couple" with the best routes from one to the other, pairing 1 the next best, etc., and the y axis is how much data can be transmitted covertly between 1999 and 2002. Clearly, the ability to communicate kilobytes of data over a period of years does not make for a general-purpose communication system, but as a means of sharing keys, perhaps in order to establish other channels, it could be quite useful.

Implicit in Figure 3 is the assumption that there is no contention for any of the network resources that Alice wishes to use. This is clearly an inaccurate assumption, but nonetheless we can see that *the network is capable of bearing some traffic*—the question is how much.

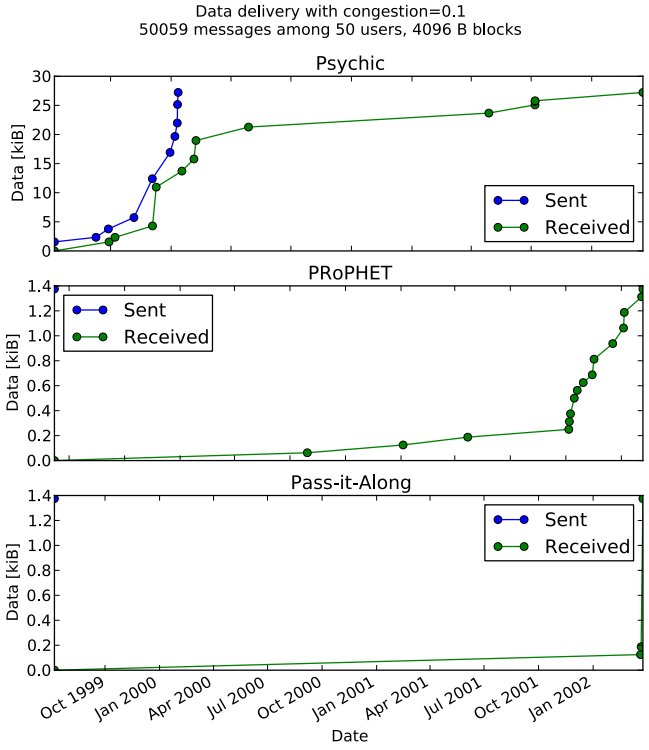


Figure 4: Routing messages via Footlights in the Enron e-mail corpus.

This question is answered in Figure 4, which shows a comparison of simulated routing performance by several routing algorithms—Psychic Routing, PRoPHET and the "Pass-it-Along" strawman—routing data through one intermediary, for pairing 0 in Figure 3b.

4.1 Psychic Routing

Psychic Routing, at the top of Figure 4, is clearly the best scheme. Using knowledge of future communications among all participants in the network, Alice is able to send approximately 27 kiB of data to Bob over the course of approximately 30 months. Such a scheme is, of course, unrealizable, but it sets our upper bound, our analog to the Shannon limit.

4.2 PRoPHET

The next graph represents the performance of the popular PRoPHET algorithm, with some tunable parameters selected from an IETF draft [13] and some chosen arbitrarily. We can see that almost 1.4 kiB of data does propagate from Alice to Bob via at least one intermediary, but unlike the Psychic Routing case, most of the transfer occurs at the end of the period of interest, rather than the beginning. Unlike Psychic Routing, a realizable scheme like PRoPHET must expend time and communications bits in order to propagate routing information—in this case, probabilistic estimates of how soon each node expects to communicate with other nodes.

In the absence of other information, we might suppose that PRoPHET's performance here is reasonable, and invest significant effort in tuning parameters to improve it incrementally. Compared with Psychic Routing, however, we can see how much further we still have to go; our time and effort might be better used in searching for a different routing algorithm entirely which better reflects the realities of the medium (e.g. does not assume that nodes are able to exchange routing information in a two-way exchange).

From a privacy perspective, we might also wish to invest in a scheme which does not require reporting accurate contact information to all peers.

4.3 Pass-it-Along

The final routing scheme depicted in Figure 4 is a straw man called "Pass-it-Along routing." As the name implies, when a node engaged in pass-it-along routing receives a packets of data, it stores the data and sends it out again attached to the next message to go out. No consideration is taken as to the suitability of the next node to finally deliver the packet, so this strawman protocol can be regarded as a constrained form of network flooding.

The performance of this routing scheme is, as might be expected, rather poor. It is only by chance that a fortuitous message succeeds in delivering over 1 kiB at the last possible moment; were it not for this one message, less than 200 B would have been delivered. Nonetheless, the comparison between Pass-it-Along

Routing and PRoPHET is telling: if we were to only compare the two schemes, we would say that Pass-it-Along is slower in delivering data, but under some circumstances, it actually delivers *more* data, since PRoPHET must consume some of the available channel in order to propagate routing information. It is only by comparing to Psychic Routing, our objective upper bound, that we see how truly atrocious the performance of Pass-it-Along Routing really is.

An interesting property of Pass-it-Along routing is that, like other flooding protocols, no source or destination addresses need be visible to routing nodes: they simply act as data mules, assuming that the intented recipient is able to recognize the packets intended for her (e.g. by decrypting them with a pre-shared key). Thus, its performance is quite poor, but it has useful privacy properties and, in this particular case, its performance is not severely worse than a scheme which requires all nodes to broadcast who they talk to and how often.

5 Future Work

There is an obvious trade-off between routing efficiency and the privacy of routing nodes: in order to improve efficiency beyond network flooding, nodes need to know about each others' communication patterns. In source-routed systems such as Tor [6], the sender of a packet does not reveal to whom she is speaking, but such a system can only function because routers have been published in a directory, and their communication graph (fully connected, over the Internet) is implicit. IP requires that destination addresses be visible to all routers, which themselves broadcast messages saying, "if you want to send packets to any of the following networks, give them to me; I am connected to them." In delay-tolerant networks with *late binding*, destination addresses may not even be fixed: routers may say, "I understand the mapping from a high-level name to a low-level one, and will forward traffic accordingly."

Having introduced Psychic Routing, an upper limit on the effectiveness of DTN routing, we now wish to study this information-efficiency trade-off, and establish a *lower* bound on how efficiently traffic can be routed in an unobservable DTN, given a certain amount of information about the communication graph the various nodes are willing to reveal to each other.

Furthermore, Psychic Routing—as it currently stands—only considers pairwise optimisation of network flows. Future work might consider global optimisations, and the incentives that would encourage all participants to "play by the rules." Even more useful might be a consideration of how local incentives—such as sender-pays vs. a *quid pro quo* arrangment of packet handling—lead to global network properties such as congestion and packet loss.

6 Conclusion

We have introduced a new feature for the Footlights social networking system: while providing high-speed, high-availability access to shared user data, Footlights will also provide users with access to a low-bitrate communications channel that is *perfectly unobservable*—even a global adversary will be unable to distinguish between users conducting normal conversation and users "piggybacking" covert traffic on their normal messages. Such a capability allows users to communicate indirectly, via other users, in a Delay Tolerant Network (DTN).

In order to write high-performance routing protocols for this DTN, we have introduced the concept of *Psychic Routing*, a routing scheme which relies on full knowledge of future communications in order to make routing decisions that maximize pairwise throughput. We have shown how this scheme could serve the role of a Shannon limit for DTN routing, rendering obsolete existing comparisons such as "15% better than an existing scheme whose properties are themselves understood relative to other schemes."

We have compared the performance of the popular PRoPHET protocol with both a straw man protocol and Psychic Routing, and have qualitatively observed an inverse relationship between pairwise routing performance and the amount of information about the network used by the protocol. In the future, we hope to explore this relationship further in order to establish a useful lower bound on privacy-preserving DTNs, so that users will be able to select routing schemes that maximize performance while respecting user-specified limits on the disclosure of contact information.

References

[1] Anderson, J., Diaz, C., Bonneau, J., and Stajano, F. Privacy-Enabling Social Networking Over Untrusted Networks. In *the Second ACM SIGCOMM Workshop on Social Network Systems (WOSN '09)* (May 2009), pp. 1–6.

[2] Belady, L. A. A study of replacement algorithms for a virtual-storage computer. *IBM Systems Journal 5*, 2 (1966), 78–101.

[3] Bonneau, J., Anderson, J., Anderson, R., and Stajano, F. Eight Friends Are Enough: Social Graph Approximation via Public Listings. In *the Second ACM EuroSys Workshop on Social Network Systems (SNS '09)* (2009).

[4] Bonneau, J., Anderson, J., and Danezis, G. Prying Data Out of a Social Network. In *the 2009 International Conference on Advances in Social Network Analysis and Mining* (2009).

[5] Diaz, C., Seys, S., Claessens, J., and Preneel, B. Towards measuring anonymity. In *Privacy Enhancing Technologies* (2002), pp. 184–188.

[6] Dingledine, R., and Mathewson, N. Tor: The second-generation onion router. In *the 13th USENIX Security Symposium* (2004).

[7] Fall, K., and Farrell, S. DTN: an architectural retrospective. *IEEE Journal on Selected Areas in Communications 26*, 5 (2008), 828–836.

[8] Ferreira, A. Building a reference combinatorial model for MANETs. *IEEE Network 18*, 5 (Sep 2004), 24–29.

[9] Ford, L. R., and Fulkerson, D. R. Flows in Networks. Tech. Rep. R-375-PR, RAND Corporation, Aug 1962.

[10] Goffman, E. *The Presentation of Self in Everyday Life*. Anchor Books, Jan 1959.

[11] Handorean, R., and Gill, C. Accommodating Transient Connectivity in Ad Hoc and Mobile Settings. In *Pervasive Computing (PERVASIVE)* (2004), pp. 305–322.

[12] Klimt, B., and Yang, Y. Introducing the Enron Corpus. In *the First Conference on Email and Anti-Spam (CEAS)* (Carnegie Mellon University, 2004), Carnegie Mellon University. `http://www.cs.cmu.edu/~enron/`.

[13] Lindgren, A., Doria, A., Davies, E., and Grasic, S. Probabilistic Routing Protocol for Intermittently Connected Networks. Internet draft (work in progress). Retrieved from `http://tools.ietf.org/html/draft-irtf-dtnrg-prophet-09`.

[14] Lindgren, A., Doria, A., and Schelén, O. Probabilistic Routing in Intermittently Connected Networks. In *Service Assurance with Partial and Intermittent Resources (SAPIR)* (Fortaleza, Brazil, 2004), Springer Berlin Heidelberg, pp. 239–254.

[15] Loeliger, J. *Version control with Git*. O'Reilly, Jun 2009.

[16] Quinlan, S., and Dorward, S. Venti: a new approach to archival storage. In *the FAST 2002 Conference on File and Storage Technologies* (Monterey, California, Jan 2002), Bell Labs, Lucent Technologies.

[17] Shannon, C. A Mathematical Theory of Communication. *Bell System Technical Journal* (1948).

[18] Vahdat, A., and Becker, D. Epidemic Routing for Partially-Connected Ad Hoc Networks. Tech. Rep. CS-2000-06, Duke University, Apr 2000.

[19] Zhang, Z. Routing in intermittently connected mobile ad hoc networks and delay tolerant networks: overview and challenges. *Communications Surveys & Tutorials, IEEE*.

Sleeping dogs lie on a bed of onions but wake when mixed

Paul Syverson

Naval Research Laboratory, USA.

Abstract. We introduce and investigate *sleeper attacks* and explore them in the context of anonymous communication, especially mix networks. Sleeper attacks can make use of the interference inherent to mix protocols. Simply by leaving his own messages in a mix network an adversary can learn about the communication of network users. Sleeper attacks can be combined with epistemic attacks, statistical disclosure, or other attacks to be made even more effective. We use sleeper attacks to disprove the common belief that mix networks are necessarily more secure than onion routing networks. Finally we use our results to disprove another commonly held belief about computer security in general, that it is always conservative to prove security against the strongest possible adversary.

1 Introduction

Suppose Alice_1 and Alice_2 are known to be the two possible correspondents to Bob, and suppose each sends a message into two basic threshold mixes, mix_1 and mix_2 respectively. For purposes of this example it does not matter what the firing threshold of the mixes are.[1] Assume the adversary Dorm can see Alices send messages and can see Bob receive messages, but he cannot generally see any mixes send or receive messages or see the mixes internal workings. Under these circumstances, if Dorm later sees Bob receive a message from mix_3 he cannot tell which Alice sent it.

Suppose, however, that Dorm has previously left his own messages, S_1 and S_2 (each to and from himself) in mix_1 and mix_2, respectively. If he receives S_1 but not S_2, then, absent other considerations, he knows that mix_1 has fired while mix_2 has not. He thus knows that Alice_1 sent the message to Bob. The information leak in this toy example is not strictly speaking a passive channel

[1] For the unfamiliar reader, a *threshold mix* receives messages until it reaches a given *threshold* at which point it fires, forwarding all of the received messages to their next destination, which might be the ultimate receiver, a bulletin board, or another mix. Messages are transformed by the mix and the batch of messages permuted by the mix so that it is not feasible to match which honest messages going into the mix match which honest messages coming out, as long as there are at least two honest (not adversary controlled) messages. This paper assumes general familiarity with anonymous communications research. See [13, 6] for a background survey.

since Dorm had to place his own messages in the mixes in order for it to work. It is not, however, a typical active attack. As long as his messages are left in the mix, it does not matter when he put them there. His active component can be any time before, possibly long before, Alice sends her message. For this reason we call this a *sleeper attack*.

In this paper we will explore sleeper attacks on anonymous communication. We begin by describing sleeper attacks in the contexts of various types of mixes. We also note that a weaker adversary can be more effective using sleeper attacks in combination with other attacks than simply using those attacks by themselves.

Next we consider sleeper attacks on onion routing networks. A commonly held belief amongst anonymous communication researchers and practitioners is that mix networks are more secure than onion routing networks. On the other hand onion routing networks are far more practical and usable for most users and applications. Thus, these design alternatives are generally presented as making a trade-off between security and practicality. For example, the original Tor design paper [11] says that, "relay-based anonymity designs have diverged in two main directions. Systems like Babel [15], Mixmaster [23], and Mixminion [7] have tried to maximize anonymity at the cost of introducing comparatively large and variable latencies. Because of this decision, these high-latency networks resist strong global adversaries, but introduce too much lag for interactive tasks like web browsing, Internet chat, or SSH connections." We show that characterizing mixes versus onion routers as only a security versus practicality trade-off is misconstruing security: mix networks are in important ways less secure than onion routing networks, even if they are more secure in other ways.

In previous work [28], we examined the dependence of this characterization on unfounded trust-uniformity assumptions, on ignoring usability implications, and on unrealistic adversary models. Herein, we go a step further. There are certainly realistic configurations, environments, and adversaries for which mix networks are more secure than onion routing networks. We present examples where the opposite is true. Our examples have moderate and realistic adversaries. The networks consist of the same number of nodes in the same configuration, just one composed of mixes and the other composed of onion routers. Both networks have the same number of users. By their natures usage of the networks cannot be identical, but we will make them as comparable as possible. The capabilities of the adversaries and their deployment is the same in both networks. We will show that there exist such circumstances in which the onion routing network is more secure than the comparable mix network.

Another even more broadly held belief is that it is always conservative to assume the strongest possible adversary. This is a notion from many areas of computer security and cryptography not just anonymous communication. We show (with the same realistic systems and realistic adversaries) an example of two systems in which one system is more secure against the stronger adversary but the other is more secure against the weaker adversary.

2 Mixing sleepers awake

"It is nought good a sleping hound to wake."

Geoffrey Chaucer — Troilus and Criseyde

Mixes derive their security from altering the order of messages they receive to obscure the relation of inbound to outbound messages. This is true for mix designs from simple threshold mixes to timed dynamic pool (Cottrell) mixes to binomial mixes. Put differently, mixes create interference between messages. This interference puts bounds on the information leaked or rate of information leaked to an observer of the mix [24, 25, 29, 2] But it also puts a lower bound on information leaked to an observer. In a threshold mix with batch size n, an adversary observing a single input to and single output from the mix has uncertainty about whether they match that is bounded by n. A sleeper attack can take advantage of this.

Consider a layered network of threshold mixes with a sleeper in each mix, where there is one layer of mixes receiving inputs from senders forwarding to a second layer of mixes that forward messages to their ultimate recipients. Suppose the adversary observes just one message being sent into some mixes and sees just one message being received. Suppose he learns from his sleepers which layer1 mixes fires and which do not, and then learns which of the layer2 mixes fires and that the others do not. From this he knows the received message could not come from the sender into any layer1 mix that did not fire. Assuming the threshold and the distributions of messages are known, then for the observed input messages that could match the observed received message at all he also can attach a significantly higher probability to their matching the received message than he could without the sleepers. This could also be combined with knowledge about sending rates and thus the likely number of messages in a layer2 mix based on time since last firing to infer still more.

There are three basic categories of interference that mixes can have, based on the type of the mix. Mixes that require a number of messages to be received to fire have mandatory interference between a message sent by the mix and previous messages received by the mix. (This includes messages sent but not received by the mix, for example, dummy messages.) This also applies whether they are simple threshold mixes or use some sort of pool or other function that relates the probability of sending a message to previously received messages.

Mixes may also be purely timed: they randomly order the messages that they have received during a given interval and forward (some of) them (along with any messages from the mix itself) at the end of the interval regardless of what messages if any have been received in that interval. These mixes have contingent interference. Messages that are available for mixing will interfere, but if no messages are available, there is no interference with received messages. If the mix itself generates messages, then there is interference with those.

If a mix requires both a minimum interval of time and a minimum of received messages in order to fire, then it still has mandatory interference between the messages sent and previously received messages.

Stop-and-go (sg) mixes [19] forward a message that was sent at a time designated in the message regardless of other messages in the mix. Messages sent from stop-and-go (sg) mixes have no interference at all. (We restrict interference to that inherent in the protocol and treat as out of scope any interference from processing time for necessary computation or transmission of messages. This scope will also apply to onion routing networks, to which we will return below.) What this shows us is that stop-and-go mixes are mixes in name only. Any mixing they provide is virtual rather than inherent to their operation. For the remainder of the paper, we will restrict usage of 'mix' to systems that base the ordering of output messages at least partially on other messages the mix outputs, whether they were received or generated by the mix.

Sleeper attacks cannot reveal anything in an sg mix network or a purely timed mix network. For sg mixes, other messages can simply not interfere with a sleeper. For purely time mixes, there can be interference, but anything a sleeper attack could reveal is already known to the adversary from the mix protocol.

If a mix has any kind of pool or other function that makes the forwarding of a message held by a mix probabilistic when the mix fires, then a sleeper cannot determine with certainty when a message of interest was sent by the mix. Nonetheless, as long as the adversary can keep an adequate representative sample of sleepers in the mix, the he can learn from when sleepers are sent by the mix the same probabilistic information about received messages in sent batches as he could if he could observe the batches themselves emerging from the mix.

2.1 Combining the sleeper with epistemic and other attacks

"*He sees you when you're sleeping. He knows when you're awake.*"

John Frederick Coots and Haven Gillespie — "Santa Claus is coming to town"

Epistemic attacks on anonymity were first introduced by Danezis and Clayton [5]. They described a *route fingerprinting* attack in which an adversary knows which nodes in an anonymity network are known to which possible senders. Using this information, an adversary observing a message on even part of a route can use which senders would know how to construct that route to narrow down the set of possible senders. Danezis and Syverson [9] later described *route bridging*, which makes use of what senders do *not* know about the network nodes to determine which routes it would be impossible for some senders to construct and again narrow down the possible senders. These attacks use an adversary's observation of all messages entering or leaving a mix in a single batch. By knowing which senders could know about all the possible combinations of the three mixes involved in every possible route through the observed mix for that batch he can narrow down the number of senders. If the adversary can make use of

sleepers and patterns of mix firings, then he can conduct such attacks without having to observe as much of the actual network. For example, if a message is received that would require going through a mix that did not fire if it were sent by one of the senders who otherwise match what is known about a route and received message, then that sender is eliminated without having to be able to directly observe the mix.

Disclosure attacks [18] and statistical disclosure attacks [4, 8] are long-term intersection attacks to determine who is talking to whom by observing when potential senders and receivers are present together and when they are not. The statistical version provides answers with high probability rather than with certainty, but it is also much more efficient. The original versions, both statistical and not, required a global passive observer. Later, Dingledine and Mathewson [20] showed how effective the attack could be when only part of the network was observed by the adversary. Sleeper attacks can be added to eliminate or support possible communication patterns by knowing which mixes fired and in which order, again making it possible to have an equally effective attack with a weaker adversary.

3 Sleepers and Onions

As already illustrated, there are settings where an adversary can learn information from a sleeper attack on a mix network. To make our initial toy example into something more real and concrete, suppose that an adversary (Dorm) is following a blog (Bob) and has some candidate posters to that blog (Alices) under observation. For simplicity we will assume two Alices. Suppose the Alices are known to use various anonymous communication systems, but for jurisdictional, legal, or resource reasons, none of these are observable by the adversary. All he can do is passively watch when either Alice sends or receives messages and he can see when the blog updates with new posts. Suppose the Alices are unknown to each other but both are relatively paranoid and relatively up on the anonymous communications literature. They thus each choose to use a high latency mix network for sensitive communications. Bob only updates twice a day. But given the high latency of the mix network, when Bob is observed to be updating with an item of interest, Dorm is able to discern from the sending activity of the Alices, and the pattern of mix firings he observed from his sleeper attacks that one of the Alices could not have sent the information of interest but the other could.

If we combine the pure sleeper attack with other information, Dorm may be able to conduct this attack even if he cannot directly observe the Alices. For example, if he is aware that they only know about different parts of the mix network, then he can use this for an epistemic attack together with a sleeper attack to make the same inference just described even if the only thing he can observe is Bob's public blog updates. Similarly if Dorm is aware that the Alices trust some parts of the network more than others and are thus inclined to prefer those parts [17], he can use this in conjunction with a sleeper attack and simply

seeing Bob's updates to indicate which Alice is the likely poster, or to increase his confidence in previous suspicions.

Contrast this with onion routing networks. Assume the exact same situation as the above except that instead of using mix networks the Alices are using onion routing networks. In other words, assume the situation is exactly as above, except that the network nodes are onion routers rather than mixes. And assume that the Alices are communicating with Bob (or whoever) via an onion routing protocol rather than a mix protocol.

If Alice_1 did not use the onion routing network at all during the relevant period until after Bob's update appeared and Alice_2 did, then Dorm would be able to discern that Alice_1 could not have posted the information. But this is also true of a mix network for the same period, although the mix network can obscure the period that either Alice's messages could arrive at Bob. If we limit to situations where both Alices used the network in the period before Bob's update, then the sleeper attack will provide no information about which Alice posted the message in the case of an onion routing network and can determine which Alice made the post in the case of a mix network. And, this remains true if neither Alice is observed at all and a sleeper attack is combined with epistemic or trust-based attack.

In this section we have looked at sleeper adversaries that are relatively trivial to implement and deploy, either as capable of only sleeper attacks or in combination with other simple attacks. We have examined such adversaries applied to realistic communication settings involving anonymity networks. To compare mix networks to onion routing networks in these environments we kept the adversary capabilities exactly the same, and we kept the communicants, their use of the networks and the network configurations virtually the same. The only change was to have the network nodes run either an onion routing protocol or a mixing protocol. In these identical settings the mix protocol can leak significant information but the onion routing protocol leaks no information. We have thus shown unequivocally that it wrong to say mix networks are more secure than onion routing networks.

There are interference attacks that have been run against Tor [26, 14]. They are not actually attacks on onion routing at the protocol level. Rather they are attacks on implementations taking advantage of the time it takes to actually process and send communication through the Tor network. They are thus outside the scope of this paper. But we will consider them briefly. The attacks found by Danezis and Murdoch [26] have been shown by Evans et al. [14] to simply not work against the current Tor network, which is much much larger than the Tor network at the time of [26]. Evans et al. went on to explore extensions of Danezis and Murdoch's attacks that were feasible. They are feasible however, only in combination with several other attacks that, while plausible, will work only with certain types of application communication used in a particular way rather than against Tor communication in general [14]. And they still require sending a significant amount of traffic into the network at a constant rate, good clocks relatively well synchronized, and numerous other assumptions. Thus even

if we considered implementation rather than just protocol-level attacks, compared with sleepers they require far more resources and assumptions, as well as specialized settings and specific types of application communications.

4 Anonymity networks and their adversaries

Onion routing networks are similar to mix networks in some respects, but they primarily derive their security from the unpredictability of routes and the inability of an adversary to watch enough of the network to be likely to see both ends of a connection. Though they have been combined with mixes for research purposes (so that some form of mixing was done at onion routers) this is not typical and currently considered to serve no useful purpose.

Security for mix networks is typically evaluated assuming a global passive adversary (GPA). For a large distributed network a global adversary is very strong, perhaps even unrealistically so. On the other hand, for a publicly accessible network that does not require registered users, it is also unrealistic to assume the adversary is not able to generate his own messages. Similarly, the original motivation for having mix networks rather than communicating via single mixes was that some mixes might be compromised [3]. It is also unrealistic to think that an adversary compromising a mix might not try to add, drop, or alter messages in his control if he can get away with doing so. For these reasons, many security analyses add to the GPA the ability to send messages into the network and the ability to create and/or manipulate messages at a compromised subset of the mixes.

This is the adversary model against which Mixminion was designed and evaluated and the one we initially adopt. Against this adversary onion routing is completely broken. Just the global passive element is enough to break onion routing. It has been long understood and experimentally verified on the Tor network, that a passive adversary can virtually always confirm the association of inbound connections to and outbound connections from the network by the timing and volume of traffic [27]. Indeed, it has been shown in simulation that simply creating connections is enough, the correlation can be confirmed even without sending any data [1]. For these reasons, we have said since inventing onion routing that it guards against traffic analysis, not traffic confirmation.

Observations such as we have just made are the reason that people have generally held that onion routing networks are less secure than mix networks. And against the above adversary they are. However, many have noted that when taking usability and performance into account, the size of both the network and user base for onion routing networks is much larger than for mix networks [10]. The public Tor network is orders of magnitude bigger and has orders of magnitude more users than the largest public mix networks that have existed. And this is one of several reasons that onion routing networks may be more secure than mix networks: it is much harder to have a realistic global adversary against the much larger Tor network than against a Mixmaster or Mixminion network. It is also easier for an adversary to apply all of its available resources to the few

hundreds of Mixmaster users it detects than against the hundreds of thousands of Tor users, if it could even observe them all [28]. Still these are somewhat apples-and-oranges comparisons. Even if it is more realistic to do so, these observations are based on comparing very different network sizes and different sizes of user base.

In the previous section, however, we have shown that (for the same network configurations and size, with the same senders and receivers, and against the same adversary) mix networks can leak important information when onion routing networks do not. But this does not imply that onion routing networks are in general more secure than mix networks. As already noted, against an adversary that includes a global observer, onion routing is completely broken. The source and destination of every connection are exposed. A mix network is not completely broken against the above described adversary. Exactly what protection it provides is complicated and cannot be determined without at least parameterizing the number of senders, receivers, and network nodes—also what fraction of the network is compromised, what exactly the adversary can do at what rate, the rate and distribution of sending messages, whether observations are one-time or extended, and if so the dynamics of all the above, etc. Nonetheless, it is clear that, even with all this, for many reasonable choices of parameters mix networks provide some protection. One of the reasons a GPA is often chosen for analysis is that it simplifies such analysis to a tractable level. But as already noted, such a model is so unrealistic that it is not clear what we learn from using it. Though one could perhaps create believable settings where the GPA makes sense, for no application for anonymous communication yet published has it been plausible to assume such an adversary. This adversary would need to be able to watch the entire network, regardless of size, and yet cannot attempt to even slightly delay messages, send messages of its own, or corrupt some users to send (or not) at useful times.

The sleeper attack by itself requires a very weak adversary. He does not do anything to the messages of other users in any way that plays a role in the attack. (He cannot help affecting the firing rate and message ordering of mixes by placing messages in them, and he could conduct a 1 attack—the complement of an $n-1$ attack, but in a pure sleeper attack we ignore these.) He corrupts no nodes in the network. He can only observe sending and receiving behavior at a few points. He needs to generate messages at a relatively low rate. (Call a sleeper attack *complete* if the adversary always learns when a mix fires. For a complete sleeper attack in a network of threshold mixes, he must send at least one message per mix firing. For other types of mixes he can only achieve high likelihood of a complete attack.) He does not need a clock at all. He only needs to tell the ordering of mix firings relative to each other and any of the few transmissions he observes. This is thus an attack that even a quite low-resource adversary should be able to conduct easily. It is thus much more realistic than a global passive adversary. Like the GPA, however, once some other parameter settings are given, it should also prove tractable to analyze, although we do not explore that in this paper. We thus have two adversaries that are subadversaries of the most powerful

adversary we have described above. One, however, refines the powerful adversary to something plausible, while the other refines it to something unrealistic.

We have uncovered something else in the above, however, besides a lesson about useful versus inherently impractical adversary models. A generally accepted truth of computer security is that it is conservative to assume the most powerful adversary. This has strong intuitive plausibility. If a system is secure against a more powerful adversary it should remain secure against an adversary with fewer capabilities or diminished capabilities. As has been shown in the multilevel security literature, however, intuitions can be deceptive.

There are theoretical examples of multilevel-secure systems that are effectively secure against a strong adversary but leak information against an adversary with fewer capabilities [21, 16, 22]. The examples we have shown are *monotonic*: they do not show better security of a given system in a given environment against a more powerful adversary than against a strictly weaker adversary. An adversary that is both globally observing and able to mount a sleeper attack can learn more against a mix network than an adversary that can only mount a sleeper attack. Similarly an adversary that is globally observing and can mount a sleeper attack is able to learn more against an onion routing network than one that can only mount a sleeper attack. Nonetheless, in the settings we have described, when the adversary is both globally observing and lays sleepers the mix network is stronger, whereas when the adversary is only able to lay sleepers and can only make the smaller set of observations described above, the onion routing network is stronger.

This result undermines a fundamentally held belief about computer security. We typically assume the strongest possible adversary for at least two reasons. One is that we assume that if the system is secure in that setting it will be secure in weaker settings. Previous literature has explored what is needed to make that assumption correct. The other reason for evaluating against the strongest possible adversary is that we assume that if one system is more secure than another against the stronger adversary, it will also be more secure than the other against the weaker adversary. Our example shows that this is not necessarily true. And this is not simply a point about differences in implementation that can create different vulnerabilities in the different systems. The crossover occurs with the protocols at the same level of abstraction; only the capabilities of the adversary are diminished. And, if what makes the strongest possible adversary stronger is something that is unrealistic, following the standard reasoning may lead us to choose the system that is less secure against a more realistic adversary.

In a fuller analysis we intend to explore this with more mathematical detail and rigor. We also intend to more fully explore the relationship between different types of networks and sleeper adversaries, both alone and combined with other adversaries. In [12] we discussed the security of combining messages with different latency and security needs in what we called alpha mixing. As we have seen, any kind of actual mixing (in other words, interference between received or generated messages in a mix) can be vulnerable to sleeper attacks. But one of the variants described was called timed alpha mixing, which was effectively a less restrictive

variant of sg mixing. Like sg mixing, it is not actually mixing at all. Like basic onion routing, neither of these forms of 'mixing' is vulnerable to sleeper attacks. It will be interesting to explore the use of sg mixes or timed alpha mixes in combination with onion routing to examine the interplay of security it provides.

References

1. Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource routing attacks against Tor. In Ting Yu, editor, *WPES'07: Proceedings of the 2007 ACM Workshop on Privacy in the Electronic Society*, pages 11–20. ACM Press, October 2007.
2. Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Prakash Panangaden. Anonymity protocols as noisy channels. *Information and Computation/Information and Control*, 206(2–4):378–401, 2008.
3. David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2):84–88, February 1981.
4. George Danezis. Statistical disclosure attacks. In Gritzalis, Vimercati, Samarati, and Katsikas, editors, *Security and Privacy in the Age of Uncertainty, (SEC2003)*, pages 421–426, Athens, May 2003. IFIP TC11, Kluwer.
5. George Danezis and Richard Clayton. Route fingerprinting in anonymous communications. In *Sixth IEEE International Conference on Peer-to-Peer Computing, P2P 2006*, pages 69–72. IEEE Computer Society Press, 2006.
6. George Danezis, Claudia Diaz, and Paul Syverson. Anonymous communication. In Burton Rosenberg, editor, *Handbook of Financial Cryptography*. CRC Press, 2010.
7. George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *Proceedings, 2003 IEEE Symposium on Security and Privacy*, pages 2–15, Berkeley, CA, May 2003. IEEE Computer Society.
8. George Danezis and Andrei Serjantov. Statistical disclosure or intersection attacks on anonymity systems. In Jessica Fridrich, editor, *Information Hiding: 6th International Workshop, IH 2004*, pages 293–308. Springer-Verlag, LNCS 3200, May 2004.
9. George Danezis and Paul Syverson. Bridging and fingerprinting: Epistemic attacks on route selection. In Nikita Borisov and Ian Goldberg, editors, *Privacy Enhancing Technologies: Eighth International Symposium, PETS 2008*, pages 151–166. Springer-Verlag, LNCS 5134, July 2008.
10. Roger Dingledine and Nick Mathewson. Anonymity loves company: Usability and the network effect. In Ross Anderson, editor, *Fifth Workshop on the Economics of Information Security (WEIS 2006)*, June 2006.
11. Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, pages 303–319. USENIX Association, August 2004.
12. Roger Dingledine, Andrei Serjantov, and Paul Syverson. Blending different latency traffic with alpha-mixing. In George Danezis and Philippe Golle, editors, *Privacy Enhancing Technologies: 6th International Workshop, PET 2006*, pages 245–257. Springer-Verlag, LNCS 4258, 2006.
13. Matthew Edman and Bülent Yener. On anonymity in an electronic society: A survey of anonymous communication systems. *ACM Computing Surveys*, 42(1), 2010.

14. Nathan S. Evans, Roger Dingledine, and Christian Grothoff. A practical congestion attack on Tor using long paths. In *Proceedings of the 18th USENIX Security Symposium*, pages 33–50, Montreal, Canada, August 2009. USENIX Association.
15. Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Proceedings of the Symposium on Network and Distributed Security Symposium - NDSS '96*, pages 2–16. IEEE, February 1996.
16. Jeremy Jacob. On the derivation of secure components. In *IEEE Symposium on Research in Security and Privacy*. IEEE CS, May 1989.
17. Aaron Johnson and Paul Syverson. More anonymous onion routing through trust. In *22nd IEEE Computer Security Foundations Symposium, CSF 2009*, pages 3–12, Port Jefferson, New York, USA, July 2009. IEEE Computer Society.
18. Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In Fabien A. P. Petitcolas, editor, *Information Hiding: 5th International Workshop, IH 2002*, pages 53–69, Noordwijkerhout, The Netherlands, October 2002. Springer-Verlag, LNCS 2578.
19. Dogan Kesdogan, Jan Egner, and Roland Büschkes. Stop-and-Go MIXes: Providing probabilistic anonymity in an open system. In David Aucsmith, editor, *Information Hiding: Second International Workshop, IH 1998*, pages 83–98, Portland, Oregon, USA, April 1998. Springer-Verlag, LNCS 1525.
20. Nick Mathewson and Roger Dingledine. Practical traffic analysis: Extending and resisting statistical disclosure. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies: 4th International Workshop, PET 2004*. Springer-Verlag, LNCS 3424, 2005.
21. Daryl McCullough. Specification for multi-level security and a hook-up property. In *Proceedings, 1987 IEEE Symposium on Security and Privacy*. IEEE, May 1987.
22. John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *IEEE Symposium on Research in Security and Privacy*. IEEE CS, May 1994.
23. Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster protocol - version 3. IETF Internet Draft, 2003.
24. Ira S. Moskowitz, Richard E. Newman, Daniel P. Crepeau, and Allen R. Miller. Covert channels and anonymizing networks. In Pierangela Samarati and Paul Syverson, editors, *WPES'03: Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, pages 79–88, October, 2003. ACM Press.
25. Ira S. Moskowitz, Richard E. Newman, and Paul Syverson. Quasi-anonymous channels. In *IASTED International Conference on Communication, Network, and Information Security (CNIS 2003)*, pages 126–131. ACTA Press, December 2003.
26. Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy,(IEEE S&P 2005) Proceedings*, pages 183–195. IEEE CS, May 2005.
27. Lasse Øverlier and Paul Syverson. Locating hidden servers. In *2006 IEEE Symposium on Security and Privacy (S& P 2006), Proceedings*, pages 100–114. IEEE CS, May 2006.
28. Paul Syverson. Why I'm not an entropist. In *Seventeenth International Workshop on Security Protocols*. Springer-Verlag, LNCS, 2009. Forthcoming.
29. Ye Zhu and Riccardo Bettati. Anonymity vs. information leakage in anonymity systems. In *25th International Conference on Distributed Computing Systems (ICDCS 2005)*, pages 514–524. IEEE CS, June 2005.