TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik

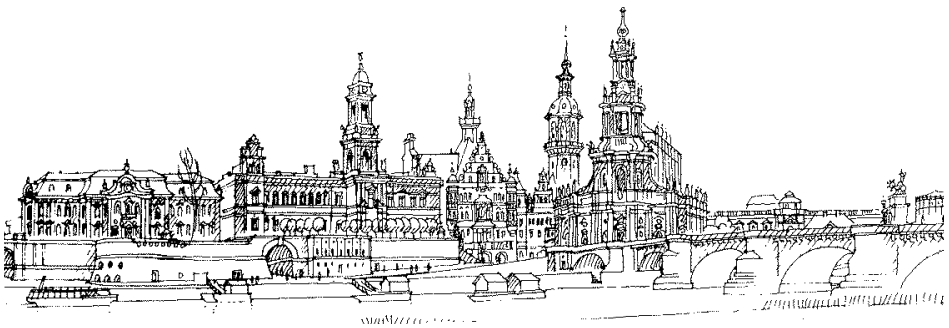
Technische Berichte
Technical Reports
ISSN 1430-211X

TUD-FI09-04-April 2009

Stefan Köpsell
Karsten Loesing (Eds.)

Institut für Systemarchitektur

Extended Abstracts of the Fourth Privacy Enhancing Technologies Convention (PET-CON 2009.1)



Technische Universität Dresden
Fakultät Informatik
D−01062 Dresden
Germany
URL: http://www.inf.tu−dresden.de/

Preface

PET-CON, the Privacy Enhancing Technologies Convention, is a forum for researchers, students, developers, and other interested people to discuss novel research, current development and techniques in the area of Privacy Enhancing Technologies. PET-CON was first conceived in June 2007 at the 7th International PET Symposium in Ottawa, Canada. The idea was to set up a bi-annual convention in or nearby Germany to be able to meet more often than only once a year at some major conference.

The First PET-CON took place on August 16, 2007 in Frankfurt (on the Main), Germany. There were participants from four German universities. The Second PET-CON was held on February 11, 2008 in Aachen, Germany. This was the first time that we accepted submissions, provided reviews, and published a booklet of Extended Abstracts. The Third PET-CON took place on September 30, 2008 in Darmstadt, Germany. There were no written submissions and no review process, but instead lively discussions of work in progress at the convention. This Fourth PET-CON will be held on March 24–25, 2009 in Dresden, Germany. Apparently, we accepted written submissions this time. Submitting a contribution was not mandatory for participating in PET-CON, and no submissions were rejected. All submissions were gratefully revised by three anonymous reviewers each.

We would like to thank all authors for submitting an Extended Abstract, all reviewers for conducting their work on really short notice, and the TU Dresden for funding this booklet of Extended Abstracts and for making the actual convention possible.

March 2009

Stefan Köpsell, Karsten Loesing
Editors

Reviewers

Lars Fischer
Lothar Fritsch, Norwegian Computing Center, Norway
Dominik Herrmann, University of Regensburg, Germany
Jens Kubieziel, Friedrich Schiller University of Jena, Germany
Karsten Loesing, The Tor Project
Nick Mathewson, The Tor Project
Sebastian Pape, University of Kassel, Germany
Andreas Pashalidis
Lexi Pimenidis, University of Siegen, Germany
Florian Scheuer, University of Regensburg, Germany

Organizers

Rainer Böhme, TU Dresden, Germany
Stefan Köpsell, TU Dresden, Germany

Contents

CHATMIX – Ein Chatsystem mit Fokus auf Senderanonymität

Manuel Breu, Christoph Gerber, Tobias Islinger, Florian Scheuer

florian.scheuer@wiwi.uni-regensburg.de
{manuel.breu, christoph.gerber, tobias.islinger}@stud.uni-regensburg.de
Lehrstuhl Management der Informationssicherheit,
Universität Regensburg, Deutschland

Abstract

Dieses Arbeitspapier stellt ein auf Chaumschen Mixen basierendes Chatsystem vor, das Benutzern die Möglichkeit eröffnet, anonym zu kommunizieren. Unser System ist genau auf diesen Anwendungsfall zugeschnitten und verfügt über effektive Maßnahmen um Angriffe abzuwehren. Zum Einsatz kommen dabei u. a. Dummytraffic, getaktete Übertragungen und ein zweistufiges Verfahren zum Schutz vor Replayangriffen.

1 Einführung

Die voranschreitende Vernetzung von Menschen weltweit führt dazu, dass ein immer größeres Angebot an Diensten im Internet zur Verfügung steht. Doch gerade in sehr sensiblen Anwendungsszenarien ist dies nicht ohne weiteres möglich: Die fehlende Anonymität im Internet erschwert die Einrichtung von virtuellen anonymen Selbsthilfegruppen oder Informantenportalen. Diese Szenarien könnten jedoch von einer anonymen Kommunikation ohne physische Anwesenheit stark profitieren und an Akzeptanz gewinnen. In diesem Arbeitspapier soll daher das Konzept eines anonymen Chatsystems vorgestellt werden.

Mixe sind ein bewährtes Mittel zur Realisierung praktikabler Anonymität im Internet (vgl. Tor[1] und JonDonym[2]). Das vorgestellte Chatsystem nutzt daher diese Technik um die Nachrichten effektiv von der Identität ihrer Absender zu trennen. Der Client verbindet sich über eine Kaskade dedizierter Mixe zu einem Server, der erhaltene Nachrichten broadcastet. Aus Nutzersicht handelt es sich dabei um einen normalen Chat mit der klassischen $1 : n$ - Kommunikationsform.

[1] http://tor.eff.org (vgl. [7]).
[2] http://www.jondonym.de; vormals AN.ON bzw. JAP (vgl. [8]).

Nach einer kurzen Betrachtung verwandter Arbeiten in Abschnitt 2 wird das Angreifermodell in Abschnitt 3 formuliert. Kapitel 4 bildet mit der Beschreibung der Architektur und des Protokolls den Kern dieses Arbeitspapiers. Im Detail werden die eingesetzte Verschlüsselung (Abschnitt 4.2), Übertragungsmechansimen (Abschnitt 4.4) und Sicherheitsfunktionalität (Abschnitte 4.3 und 4.5) dargestellt. Kapitel 5 beschreibt anschließend eine prototypische Implementierung und mit einer Diskussion in Kapitel 6 schließt diese Arbeit.

2 Verwandte Arbeiten

Es gibt mehrere Ansätze zur Realisierung senderanonymer $1 : n$ - Kommunikation. Immanuel Scholz hat 2007 die Implementierung eines Chatsystems vorgestellt, das auf David Chaums Dining Cryptographers Protokoll basiert (vgl. [13, 6]). Diese Architektur kommt mit einem einzigen, zentralen Server aus und arbeitet nach einem Mehrparteienberechnungsprotokoll (vgl. Yao's Millonairs Problem in [14]). Die Vertraulichkeit der Identität der sendenden Teilnehmer wird durch dieses System sehr gut bewahrt. Jedoch ergibt sich bei dieser Konstellation ein Problem der Verfügbarkeit, sobald sich ein Teilnehmer nicht mehr an das vorgeschriebene Protokoll hält.

Ferner ist es möglich unter Verwendung der Tor-Infrastruktur anonym zu kommunizieren. Diese Architekturform geht ebenfalls auf David Chaum zurück: Es werden Mixe benutzt um die Identität der Teilnehmer zu schützen (vgl. [1, 5]).

CHATMIX soll sich durch seine sehr einfache Konfigurier- und Bedienbarkeit auszeichnen: Im Gegensatz zu einigen anderen Systemen kann es praktisch ohne Konfigurationsaufwand in Betrieb genommen werden.

3 Angreifermodell

Die Menge an über das System veröffentlichten Nachrichten steht naturgemäß in surjektivem Verhältnis zur Menge an Urhebern. Ziel eines Angreifers ist es nun, eine eindeutige Beziehung zwischen einer Nachricht und ihrem Urheber wiederherzustellen.

Externe Angreifer werden hier nicht weiter betrachtet, da jeder an dem offenen CHATMIX-System teilnehmen und somit zum Insider werden kann. Angreifer können mehrere Clients betreiben und versuchen, beliebige Nachrichten an das System abzusetzen. Zudem können sie alle Leitungen überwachen und Traffic-Analysen durchführen. Sie sind jedoch nicht in der Lage, kryptographische Verfahren zu brechen.

Die Betreiber der eingesetzten Mixe sowie des Servers werden als semi-vertrauenswürdig ('honest-but-curious', vgl. [2, 11]) angenommen: Sie greifen ausschließlich passiv an und halten sich ansonsten streng an das Protokoll. Zudem kooperieren niemals alle Betreiber der Komponenten gleichzeitig.

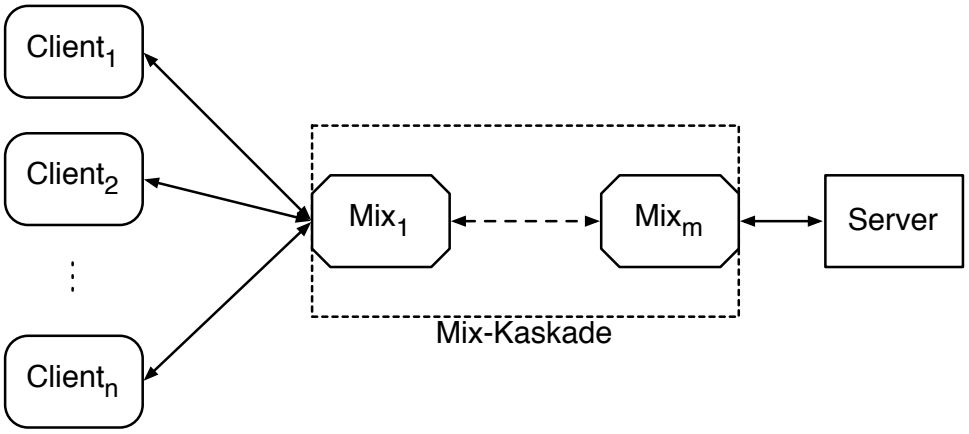


Figure 1: Die verteilten Komponenten des Chatsystems.

4 Architektur und Protokoll

4.1 Komponenten

Die Systemarchitektur besteht im wesentlichen aus drei Komponenten wie Abbildung 1 zeigt:

Server Der Server stellt die Klartexte der Nutzerbeiträge zur Verfügung. Er besitzt ein asymmetrisches Schlüsselpaar bestehend aus dem privaten Schlüssel Sec_S und dem öffentlichen zertifizierten Schlüssel Pub_S.

Mixe Die Mixe verbergen die Kommunikationsbeziehungen zwischen Clients und dem Server. Jeder Mix i ($i \epsilon [1 .. m]$) einer Kaskade verfügt ebenfalls über ein asymmetrisches und zertifiziertes Schlüsselpaar (Sec_i, Pub_i) und besitzt zudem die Zertifikate der benachbarten Mixe seiner Kaskade.

Clients n Clients verbinden sich über die Mix-Kaskade mit dem Server, um anonym mit anderen Clients kommunizieren zu können. Sie besitzen keine eigenen Schlüssel, benötigen jedoch die Zertifkate der anderen Komponenten.

4.2 Nachrichtenaufbau und Verschlüsselung

Die Nachrichten des CHATMIX sind nach dem von Mixen bekannten Zwiebelschalenprinzip aufgebaut, mit dem Unterschied, dass hybride Kryptographie zum Einsatz kommt. Wie in [9, 12] dargestellt, performt symmetrische Kryptographie um Größenordnungen besser, sie lässt jedoch den Vorteil der einfachen Schlüsselverteilung eines asymmetrischen Systems vermissen. Daher kommt hier ein hybrides System zum Einsatz: Alle Nachrichten werden mit einer symmetrischen Blockchiffre unter einem

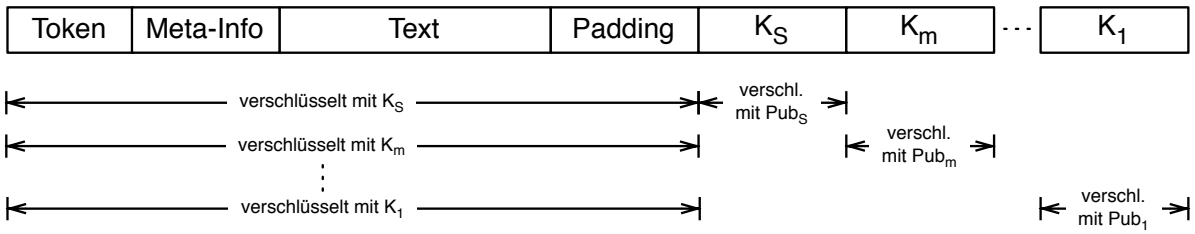


Figure 2: Aufbau eines Nachrichtenobjekts.

zufälligen Schlüssel K verschlüsselt. Dieser wird nun für den Empfänger mit dessen öffentlichen Schlüssel Pub verschlüsselt und der Nachricht beigefügt.

Nachrichten selbst beinhalten, wie in Abbildung 2 dargestellt, neben dem eigentlichen durch den Nutzer eingegebenen Text einige Meta-Informationen, ein Padding sowie ein Sicherheitstoken. Die Meta-Informationen bestehen aus Kennungen für virtuelle Chaträume sowie ein optionales Pseudonym, unter dem die Botschaft veröffentlich werden soll. Dies dient der Gruppierung von Nachrichten eines Themas sowie zur Verbesserung der Kohäsion des Textes. Das Padding und das Token dienen der Erschwerung der Verkettung von Nachrichten bzw. dem Replayschutz und werden detailliert in den Abschnitten 4.3 und 4.5 beschrieben.

4.3 Dummynachrichten und Padding

CHATMIX arbeitet getaktet. Nachrichten werden in vorgeschriebenen, diskreten Zeitintervallen Δt von Clients gesendet. Sollte ein Chatteilnehmer in einer Zeiteinheit keine Nutzdaten generiert haben, erzeugt der Client eine sog. Dummynachricht (vgl. [4]). Diese Nachricht trägt keinerlei Information mit semantischer Bedeutung in sich, wird aber wie eine sinntragende Nachricht durch den Chatclient zum Versand aufbereitet und ist von einer solchen nicht unterscheidbar. Sie dient lediglich dazu, Traffic auf der Datenleitung zu erzeugen, um einem potentiellen Angreifer, dem es möglich ist die elektronischen Verkehrswege auszuforschen, eine Analyse der Daten zu erschweren. Auf diese Weise werden Situationen vermieden, die zur Aufdeckung der Identität eines Clients führen könnte, der als einziger innerhalb eines Zeitintervalls Δt eine Nachricht sendet. Dummynachrichten werden durch den Server auf Grund eines Redundanzmerkmals, welches erst nach korrekter Entschlüsselung durch alle beteiligten Netzkomponenten lesbar wird, erkannt und ausgefiltert.

Zudem werden alle Nachrichten mit Hilfe eines Paddings auf die gleiche Länge gebracht. Nachrichten die kürzer als ein festgelegter Wert sind, werden vor der Verschlüsselung mit zufälligen Bits aufgefüllt. Abgetrennt ist das Padding durch ein Trennzeichen, damit die nicht informationstragenden Bestandteile der Nachricht auf Serverseite wieder entfernt werden können.

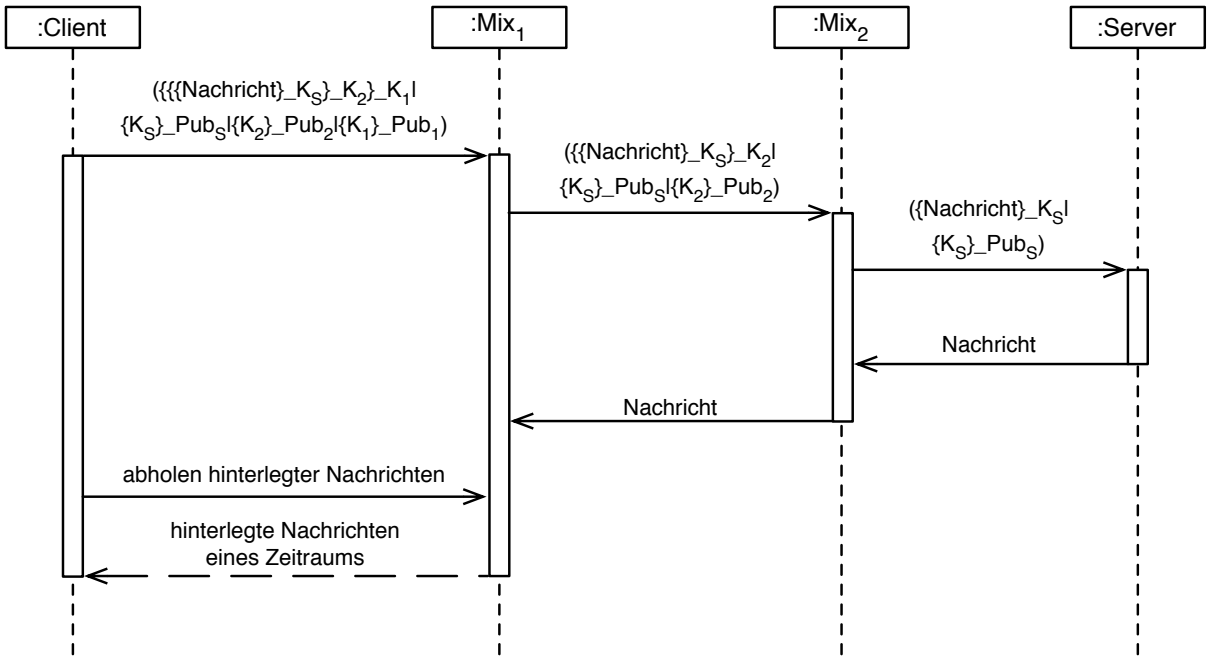


Figure 3: Sequenzdiagramm der Nachrichtenübertragung.

4.4 Kommunikation

Die Kommunikation über die Mix-Kaskade läuft nach dem üblichen Prinzip ab: Jeder Mix entschlüsselt die erhaltenen Nachrichten einmal und sendet sie anschließend umsortiert an den nächsten Mix in der Reihe weiter. An letzter Stelle steht der Server, der die erhaltenen Nachrichten ein letztes Mal entschlüsselt und nun informationstragende Nachrichten von Dummynachrichten unterscheiden kann. Erstere werden nun mit dem geheimen Schlüssel des Servers Sec_S signiert und an den letzten Mix der Kaskade zurückgeschickt. Dieser leitet sie über die Kaskade zum ersten Mix weiter. Dort werden die Klartextnachrichten zwischengespeichert und können von den Clients in einem Polling-Verfahren abgeholt werden. Abbildung 3 fasst den Ablauf exemplarisch für eine Kaskade aus zwei Mixen zusammen. $(\{Nachricht\}\_K_l)$ bezeichnet eine mit dem Schlüssel K_l verschlüsselte Nachricht; $(\{Nachricht\}\_K_l|\,\{K_l\}\_Pub_l)$ eine mit K_l verschlüsselte Nachricht, die um den mit Pub_l verschlüsselten Schlüssel K_l ergänzt wurde.

Auch der erste Mix einer Kaskaden arbeitet getaktet: Er wartet das Zeitintervall Δt ab, in dem jeder Client eine Nachricht zu schicken hat und leitet die inzwischen gesammelten Pakete an den nächsten Mix weiter.

Der aufwändigere Weg der Broadcastnachrichten über die gesamte Kaskade stellt sicher, dass dem Server und den Mixen jeweils nur der nächste Kommnikationspartner in der Kette bekannt sein muss. Ein Wechsel von Kaskaden ist somit wesentlich einfacher möglich.

4.5 Replayschutz

Bei einem Replayangriff wird von einem Angreifer ausgegangen, der die Kommunikationsbeziehungen zwischen einzelnen Clients und dem ersten Mix einer Kaskade ausforschen und manipulieren kann. Es werden Datenpakete mitgeschnitten und hierbei wird das gleiche Paket wiederholt in das System gespielt, in der Hoffung, dass gleiches Eingabeverhalten gleiches Ausgabeverhalten hervorruft. Sollte das der Fall sein, kann auf diese Weise eine Nachricht einem Sender zugeordnet werden.

Ein mögliches Angiffszenario, das auf Chaumsche Mixe abzielt, ist Folgendes: Ein Angreifer, der die Leitung zwischen Sender und Mix abhört, schneidet ein Paket mit und spielt es erneut ins System ein. Damit ein solcher Angriff erfolglos bleibt, besitzen Mixe einen Speicher, in dem bereits bearbeitete Nachrichten vorgehalten werden (vgl. [10]). Sollte sich eine Nachricht wiederholen, wird sie vom Mix einfach ignoriert und ggf. werden weitere Maßnahmen getroffen. Mit dieser Methode lassen sich Replayangriffe effektiv verhindern. Sie hat jedoch einen Nachteil: Bei langen Betriebszeiten müssten große Nachrichtenspeicher vorgehalten werden. Selbst mit effizienten Speichermethoden, wie der Verwendung von Hashwerten, wachsen die Datenmengen über die Zeit stark an. Die Daten müssen so lange gespeichert werden, so lange das gleiche Schlüsselpaar für die Netzkomponenten verwendet wird.

Eine mögliche Lösung ist, die Schlüssel bei jedem Start der Netzkomponenten zu erzeugen. Dies macht es jedoch notwendig, Public Key Infrastrukturen zu implementieren, die eine Authentizität der weitergegebenen Public Keys gewährleisten.

Ein alternativer Ansatz ist eine von uns gewählte Mischform aus Filtern und Datenspeichern, um einerseits Replayangriffe zu verhindern und andererseits die vorzuhaltende Datenmenge zu reduzieren. Zentraler Gegenstand des Replay-Schutzes ist das Sicherheitstoken, das vom Server erzeugt und an die Clients verteilt wird. Dabei handelt es sich um eine Zufallszahl, die so groß gewählt werden muss, dass Kollisionen ein tolerierbares Maß erreichen müssen und zudem nicht erraten werden können. Der Client verschlüsselt in seiner Nachricht typischerweise ein gültiges Token (vgl. auch [3]), bevor er die Nachricht an den Server übersendet. Serverseitig durchläuft diese Nachricht nun eine Kombination aus Black- und Whitelistverfahren.

4.5.1 Blacklist

Auf Serverseite wird vor dem Entschlüsseln von jeder eingehenden Nachricht ein Hashwert gebildet und – sofern nicht bereits ein Eintrag für diesen Hashwert exisitert – in einer Liste abgespeichert. Kommt es zu einer Kollision, wird angenommen, dass es sich um eine wiedereingeschleuste Nachricht handelt und sie wird verworfen. Prinzipiell ist es somit denkbar, dass durch zufälllige Kollisionen Nachrichten fälschlicherweise als Replay-Nachrichten identifizeirt werden. In diesem Fall merkt der betroffene Benutzer, dass seine Nachricht nicht erscheint und kann sie erneut abschicken. Auf Grund der zufälligen Wahl des Paddings und des symmetrischen Schlüssels wird nun mit sehr hoher Wahrscheinlichkeit eine verschlüsselte Nachricht entstehen, welche nach Anwendung der Hashfunktion eine andere Prüfsumme als die erste, grundlos

gefilterte Nachricht hat. Somit kann einer unwahrscheinlichen Kollisionserscheinung durch vertretbare Benutzerinteraktion effektiv entgegengewirkt werden.

4.5.2 Whitelist

Die zweite Hürde, die ein eingehendes Nachrichtenobjekt nehmen muss, ist eine Whitelist mit Sicherheitstokens. Der Server erzeugt jedes Mal, wenn er Nachrichten über die Mix-Kaskade zu den Clients leitet, eine neue Zufallszahl – das Sicherheitstoken. Dieses wird von ihm zudem in eine Liste eingetragen. Bei der Erzeugung einer Nachricht wird auf Seite der Clients das zuletzt erhaltene Token eingebunden. Nachdem der Server nun eine Nachricht das letzte Mal entschlüsselt hat, kann er überprüfen, ob der Sender ein gültiges Token beigefügt hat und die Nachricht entsprechend akzeptieren oder verwerfen.

4.5.3 Zusammenspiel

Die Stärke dieser Filtermaßnahme liegt nun in ihrer Kombination. Beide Filterlisten sind als zyklische Listen gleicher Länge konzipiert. Eine vom Angreifer mitgeschnittene und unmittelbar wiedereingespielte Nachricht wird am Blacklistfilter scheitern, da der Hashwert der Replay-Nachricht bereits in ihr enthalten ist. Eine mitgeschnittene Nachricht, welche zu einem deutlich späteren Zeitpunkt wiedereingespielt wird, wird vom Server zwar entschlüsselt, scheitert jedoch daran, dass in der Whitelist kein gültiges Token für diese Nachricht hinterlegt ist. Diese Schutzfunktion beschränkt die am Server vorzuhaltende Datenmenge.

5 Prototyp

Das CHATMIX-System ist prototypisch in Java mit einer Kaskade aus zwei Mixen implementiert[3]. Hierbei gruppieren sich die Komponenten Client, Mix und Server in eigenständige Teilanwendung. Diese besitzen eine Rerefenz auf eine gemeinsame Kernanwendung. In dieser finden sich neben der Implementierung des Nachrichtenobjektes auch alle kryptografischen Verfahren (AES, RSA), sowie ein für den Replayschutz notwendiges Hashverfahren (SHA-1). Unter der Verwendung von Apache Axis2[4] und des Spring-Frameworks[5] ist CHATMIX als Webservice realisiert. Die Softwarearchitektur ist im springtypischen[6] vier-Schichten-Modell implementiert: DAO-Schicht, Service, Controller und GUI (nur Client). Alle wesentlichen Bestandteile der Projekte sind gegen Interfaces programmiert. Der Datentyp der Nutzdaten ist nicht auf *String* festgelegt, sondern muss lediglich das Java-Interface *Serializable* implementieren. Somit ist es auch denkbar, beispielsweise Dateien über das System zu versenden. Einstellungen

[3] http://www-sec.uni-regensburg.de/chatmix/
[4] Axis2: Next Generation Web Services. http://ws.apache.org/axis2/
[5] http://www.springsource.org
[6] http://static.springframework.org/docs/Spring-MVC-step-by-step/

können in den jeweiligen Projekten via Konfigurationsdatei vorgenommen werden. Eine von uns gewählte Beispielkonfiguration sieht unter anderem folgende Werte vor:

- AES-Schlüssellänge: 128 Bit
- RSA-Schlüssellänge: 1024 Bit
- Sendeintervall: 500 ms
- Nachrichtenlänge: 2702 Byte

Mit der gegebenen Konfiguration ergibt sich für jede der Systemkomponenten eine Netzlast an eingehenden Nachrichten von ca. 19 MB pro Stunde pro Client. Jeder Mix der Kaskade hat eine zusätzliche ausgehende Netzlast der selben Größe. Darüber hinaus fallen bei allen Mixen und auf Ausgangsseite des Servers weitere Kosten für den Transport der rücklaufenden Nutzdaten an.

Unter der Annahne, jeder Client sendet nur einmal in zehn Sekunden Nutzdaten (in der Basiskonfiguration auf 500 Zeichen beschränkt), ergibt sich (ohne der Signaturen im Rückkanal) bereits ein Overhead von ca. 53,5 kB pro gesendeten Klartextes[7].

6 Diskussion

Sicherheit Die vorgestellte Architektur eines über Chaumsche Mixe realisierten Chatsystems bietet guten Schutz gegen die in Abschnitt 3 beschriebenen Angreifer. Probleme ergeben sich nur für folgende Szenarien:

Der erste denkbare Fall ist, dass eine Nachricht kein gültiges Token enthält. Das Protokoll ist so entwickelt, dass die Token immer mit Klartextnachrichten ausgeliefert werden. Bei der Initialisierung des Systems schreibt der erste Client, der eine Nachricht senden will eine Nachricht ohne Token. Da es für diese Nachricht keinen Whitelist-Schutz geben kann, kann sie erneut wieder in das System eingespielt werden, sobald der Hashwert der Nachricht von der Blacklist verschwunden ist. Es gibt hierbei in unserer Anwendung kein klassisches Schutzmodell[8].

Eine mögliche Maßnahme, die an dieser Stelle implementiert werden könnte, ist die Tokenverteilung von der Verteilung der Klartextnachrichten zu trennen. Auch wäre es denkbar, dass die Serverkomponente damit beginnt, automatisch generierte Nachrichten zu versenden, wenn (noch) kein Client Nachrichten schickt.

Die zweite Einschränkung, die an dieser Stelle gilt, ist, dass ein Angreifer der die verschlüsselten Client-Datenströme belauschen kann und zugleich der Serverbetreiber des Chatmix-Systems ist, prinzipiell die Möglichkeit hat Replayangriffe vorzunehmen, da die Black- und Whitelistfilterung in seinem Schutzbereich vorgenommen wird. Solche Versuche des unerlaubten Informationsgewinns, könnten durch die Betreiber der Mixe

[7] 19 reine Dummynachrichten und einmal ein Kommunikationsoverhead von 2.202 Bytes.

[8] Eine pragmatische Lösung für dieses Problem ist, dass der erste Benutzer pseudonymlos alle Teilnehmer im Chat begrüßt.

erkannt werden, wenn sie ihrerseits Blacklisten vorhalten würden, um doppelte Nachrichten zu filtern.

Es existiert ein weiteres Szenario, in dem der Replayschutz ausgehebelt werden kann. Hierzu müsste der erste Mix aktiv in die Datenverteilung eingreifen und zusätzlich mit dem Server kollaborieren. Es wäre dann denkbar, dass er nur Nachrichten mit bestimmten Tokens gezielt an einen einzelnen Client weiterleitet, der überwacht werden soll. Im Server könnte dann an Hand der Tokens überprüft werden, welche Nachrichten von diesem einen Client stammen.

Dies kann unterbunden werden, indem man die Whitelist- / Blacklistfunktionalität auf einen anderen Mix (z. B. den letzten einer Kaskade) auslagert. Der Server erzeugt hier zwar weiterhin Sicherheitstokens und hängt sie an Plaintextnachrichten an. Da Nachrichten auf ihrem Weg zum Client erst die Mixe passieren müssen, besitzt jeder Mix stets Kenntnis über aktuelle Tokens. Diese werden nun anstelle für den Server für den entsprechenden Mix verschlüsselt. Dieser filtert dann vom Client kommende Nachrichten nach oben erklärtem Black- / Whitelist-Kriterium. Die Tokens entfernt er anschließend aus den Nachrichten und leitet sie an den Server weiter. Dies ist bislang in unserem System nicht realisiert. Grund dafür ist ein unverhältnismäßig hoher Implementierungsaufwand.

Performance Das System skaliert linear mit der Anzahl an Clients und ist daher prinzipiell auch für größere Nutzerzahlen geeignet. Allerdings wird durch das ständige Versenden von Dummytraffic ein enormer Overhead produziert, der ein Vielfaches der Nutzdaten ausmacht (vgl. 5). Mögliche Optimierungen können durch Variation von Nachrichtenlänge und Zeitintervall vorgenommen werden. Dabei sind aber möglicherweise Einschränkungen hinsichtlich der Benutzbarkeit zu erwarten.

7 Zusammenfassung

Anonymität ist teuer. Um eine vertretbare und verfügbare Senderanonymität für symmetrische $1 : n$ - Kommunikation realisieren zu können, muss ein verhältnismäßig hoher Aufwand getrieben werden. Das Ziel, eine Trennung aus Sicht des Angreifers zwischen einer Nachricht und ihrem Absender herzustellen, kann durch das vorgestellte Chatsystem erreicht werden. Mit der beschriebenen Kombination aus Mixen und einem Chatserver können Nutzer anonym Nachrichten austauschen. Die Verbindung eines Clients zum ersten Mix (und damit zum Chatsystem) ist jedoch nach wie vor beobachtbar. Sollte also ein System dediziert beispielsweise für anonyme Alkoholiker oder Kritiker eines Regimes seinen Dienst versehen, so kann natürlich allein eine Kommunikationsbeziehung zwischen Nutzer und Chatsystem sehr viel preisgeben. Daher sollte der Dienst für eine Vielzahl unterschiedlicher Themen genutzt werden, damit jeder Nutzer seine Beteiligung an sensiblen Themen bestreiten kann. Zudem ist es möglich, dem System weitere Mixkaskaden hinzuzufügen, um hier eine weitere Teilung des Vertrauens zu erreichen und die Unbeobachtbarkeit zu verbessern.

References

[1] Torchat. Messenger Application on Top of the Tor Network and its Location Hidden Services. URL `http://code.google.com/p/torchat/`.

[2] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara. Private Collaborative Forecasting and Benchmarking. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 103–114. ACM, New York, NY, USA, 2004. ISBN 1-58113-968-3.

[3] O. Berthold, H. Federrath, and S. Köpsell. Praktischer Schutz vor Flooding-Angriffen beim Chaumschen Mixen. *Kommunikationssicherheit im Zeichen des Internet*, pages 235–249, 2001.

[4] A. Beutelsbacher. *Kryptologie. Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen, Verheimlichen.* 8. aktualisierte Auflage. Vieweg Verlag, Wiesbaden, 2007.

[5] D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 4(2), 1981.

[6] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Crytology*, 1(1):65–75, 1988.

[7] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[8] H. Federrath. AN.ON – Privacy Protection on the Internet. *ERCIM News*, (49): 172–178, 2002.

[9] H. Federrath and A. Pfitzmann. Bausteine zur Realisierung mehrseitiger Sicherheit. In G. Müller and A. Pfitzmann, editors, *Mehrseitige Sicherheit in der Kommunikationstechnik*, pages 83–104. Addison-Wesley-Longman, 1997.

[10] S. Köpsell. Vergleich der Verfahren zur Verhinderung von Replay-Angriffen der Anonymisierungsdienste AN.ON und Tor. In *Sicherheit*, pages 183–187, 2006.

[11] L. E. Olson, M. J. Rosulek, and M. Winslett. Harvesting Credentials in Trust Negotiation as an Honest-but-curious Adversary. In *WPES '07: Proceedings of the 2007 ACM workshop on Privacy in electronic society*, pages 64–67. ACM, New York, NY, USA, 2007. ISBN 978-1-59593-883-1.

[12] S. Risse. Benchmarking von Kryptoalgorithmen. Diplomarbeit, Universität Regensburg, 2008.

[13] I. Scholz. Dining Cryptographers. The Protocol. *24th Chaos Communication Congress, Berlin*, 2007.

[14] A. Yao. Protocols for Secure Computations. In *Proceedings of the twenty-third annual IEEE Symposium on Foundations of Computer Science*, pages 160–164. IEEE Computer Society, 1982.

OnionCat – An Anonymous Internet Overlay
Application and Usage

Bernhard R. Fischer
St. Pölten University of Applied Sciences
Matthias Corvinus-Straße 15, 3100 St. Pölten, Austria
2048R/5C5FFD47 <*bf@abenteuerland.at*>

Abstract

OnionCat is an anonymous Internet overlay. It allows users to share any kind of IP-based services with the advantage of anonymity. This greatly improves users' privacy and defeats surveillance.

IP-based sharing of services is exactly what the Internet does – web services, email, chat rooms, and many more. But unlike the traditional Internet with OnionCat users' locations cannot be ascertained using their "IP-footprints" they leave within every logfile on the net. OnionCat gains its anonymity by using anonymizing networks like Tor or I2P[1] as its transport. It is available on various operating systems including Windows.

This paper explains how OnionCat works and gives instructions about its application and usage. With a continuously growing community the OnionCat network could evolve into a feature and information rich network like we know the standard Internet today.

1 Introduction

OnionCat [2] may be used in a wide range of different applications. It provides a kernel interface, i.e. a network device, to which an IPv6 address is assigned. Thus it provides one of the most compatible interfaces possible. Of course an IPv4 interface would be even more compatible, but it does not fulfill the requirement for the most important development goal of OnionCat.

OnionCat is based on anonymizing transport layers like Tor,[2] [8] thus it may be used by various user groups for the same reasons as they use, for example, Tor. OnionCat is an extension of anonymizers. It adds features but it also extends the group of different users and use cases.

The most important goal of OnionCat is to enable users to transport raw IP data across an anonymizing network together with automatic IP address configuration. This has two considerations.

1. It makes it easier to use.
2. It creates a single logical virtual network segment so that all users can share it. Thus they are automatically connected virtually together.

The second item is achieved by every VPN, but different from any other VPN the OnionCat network is an open network. Every user can take part without any restriction or limitation in

[1] `http://www.i2p2.de/`
[2] Currently it works just with Tor. Development of adapting to I2P is in progress.

respect to network addressing. A user can decide to change his address at any time[3] and he can also leave the network again without leaving traceable footprints. Without further requirements one can use OnionCat to achieve the following use cases.

- Usage as an open anonymous network (This is described above).
- Usage as a real VPN for a privately set up closed user group.

Both are fitted to bypass surveillance or supervised networks of any kind.

In the following sections I will discuss the concept of OnionCat as well as the setup to act as a client in the open anonymous network.

2 Behind the Scenes

OnionCat basically is a *virtual private network* (VPN) from a computer science point of view. One of the most generic definitions is found in [4]: "A Virtual Private Network is a network of virtual circuits for carrying private traffic.". I will refine and explain these terms more specifically. Virtual circuits are connections between nodes. Those connections do not exist physically, but virtually. TCP sessions, for example, could be seen as virtual circuits. While browsing the web a connection between the local computer and the webserver seems to exist but this is just a virtual connection. Of course those two are not connected physically, but still the connection carries some information, as is the case for web pages.

What Kosiur ([4]) further says is that those circuits carry *private* traffic. This does not necessarily mean that the traffic is always personal or secret to somebody. This might be the case but it is not a must. In some cases those connections could carry both types of information and very often it is only a matter of definition what is private and what is not.

Many VPNs share a similar concept, which is carrying traffic of a specific type across a network of the same type. In most cases VPN applications, like the Microsoft VPN, Cisco's VPN and OpenVPN carry IP packets. Usually they achieve the same goal, which is to access some kind of "private" network. A prime example would be a company's internal network. It's designed to work for people who have Internet access. Obviously Internet is based on the *Internet Protocol* (IP). Thus follows that those types of VPNs carry IP packets within IP packets. Expressed in a different way, IP packets get encapsulated within IP packets. Figure 1 shows a simple diagram of how VPNs fit into the OSI layer model.[4] This model defines that each upper layer depends on its lower layer and every network protocol can be assigned into a specific layer.[5]

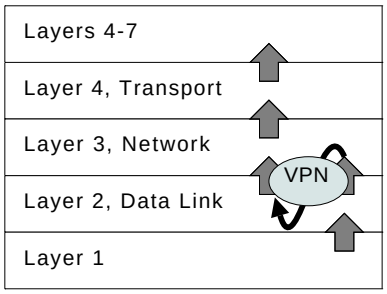


Figure 1: VPN intermediate layer.

The figure highlights three layers. Ethernet is contained within the *data link layer*. Usually we carry IP within Ethernet, hence IP is one layer above and is called *network layer*. On top of IP we have protocols such as, TCP and UDP, and categorize them into layer 4 – the *transport layer*.

If a VPN is in use, IP is encapsulated into IP and not, for example, TCP into IP as the layer model suggests. Thus, from an architectural view, it inserts a second IP layer. That is what

[3] This is the case only if the anonymizing transport allows this. But Tor does as well as I2P.

[4] The OSI model discriminates between seven different layers for classification of network protocols. It also discriminates them based on their dependencies. A detailed explanation of the model can be found in [7].

[5] That is not entirely true because all models are simplified pictures of reality, but within this context that is not of importance.

Figure 1 depicts. Above layer two the VPN layer follows (which actually also is IP). On top of that layer an IP layer follows.

If a VPN is implemented there is always some kind of VPN layer. The VPN layer creates the virtual circuits. The difference between various VPNs is where they insert the VPN layer in respect to the OSI model. Figure 1 gives just an example of encapsulating IP within IP.

2.1 OnionCat VPN

As already mentioned, OnionCat is a VPN. As has been previously explained, VPNs consist of two fundamental parts. The virtual circuits and the traffic they carry. Both can be fitted into the OSI model and both may not be of the same layer.

OnionCat does not create a completely new type of virtual circuit. It uses circuits which are created by anonymizing networks upon request. For now OnionCat supports Tor. I2P is in development.

Tor's virtual circuits, which are relevant for OnionCat, exist only within the Tor network and connect two Tor nodes. As it is the case for most virtual circuits, one end initiates the connection and the other end accepts it. The latter usually is referred to as *server*. Within Tor nomenclature this server is called *hidden service*. The circuits are based on TCP. As such they are above layer 4 in respect to the OSI model. Part of the nature of anonymizing networks calls for the capability of two nodes (connected by a virtual circuit) to open up communication channels, but not know who or where the other node is. After circuit setup Tor does not care about data carried within it. It just manages that bytes piped into it at one end and drop out at the other end and vice versa.

For all virtual circuits addressing is required to designate a connection to a specific server. For TCP sessions, addressing is achieved by an IP address[6] and a port number. Tor uses *onion-URLs* to address a specific hidden service. Onion-URLs are unique to a hidden service like IP addresses are unique for servers within the Internet. Onion-URLs are human readable 80 bits of address information based on some cryptography as described in [6].

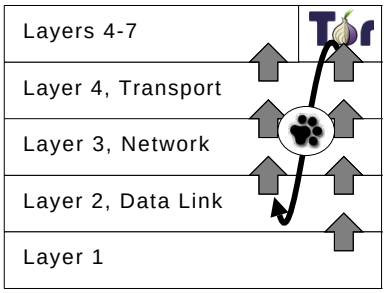


Figure 2: OnionCat in layer model.

OnionCat requests Tor to build such virtual circuits and sends raw IP data across. In this application the virtual circuits carry IP data as it is true for most VPNs. Figure 2 shows how OnionCat fits into the architecture of network protocols as defined by the OSI model. In the upper right corner it shows Tor's virtual hidden service circuits. They are based on TCP, hence, they are located above transport layer within the model. OnionCat (the cat's paw) inserts the VPN layer, it's not just an insertion as it was shown in example Figure 1. OnionCat actually makes a bridge from above the transport layer down to the network layer.

2.2 OnionCat Addressing

A typical configuration for most kinds of VPNs is that they are setup in a static way. An example of this would be how their virtual circuits are addressed. It is common for organizations to run a centralized VPN entrance point to which all VPN participants connect. This setup is easy and usually matches all requirements for such a private VPN. But it is not suitable for an open anonymous network for several reasons.

[6] Actually IP addresses are property of IP and not TCP but that makes no differnce within this context.

1. The person or organization that runs the entrance point probably will not stay anonymous. Even if they never appear in the public, such an entrance point might be revealed due to the fact that it is a traffic sink.
2. A centralized service is always a single point of failure.
3. The service provider might enjoy unlimited trust of its users which obviously would never be the case in todays world.
4. That kind of service could attract certain interest of various organizations like intelligence services.

Hence, the approach is to distribute it. To connect a Tor client to a hidden service, an example being establishing a virtual circuit within Tor, it is required to use Tor's addressing method of choice for hidden services. As explained above in Section 2.1, Tor uses onion-URLs which are 80 bit long addresses. If we assume that every client runs his own hidden service, then all of them also get a unique hidden service address – an onion-URL. This leads to the interdependency that every client can connect to every other client since every client now also is a uniquely identifiable server.[7]

The difficulty now arises from layer discrepancy. OnionCat lies between Tor on one end and the operating system on the other end. In respect to the layer model (see Figure 2) Tor (the hidden service) operates above layer 4. The other end of the VPN layer which OnionCat creates is at layer 3, which is the IP layer. Every layer has its own addressing method. Hidden services use the 80 bit long onion-URL and the IP layer obviously use IP addresses. A static configuration, one example is a configuration file, would solve that problem but not in respect to the requirement from above of *not* being static.

We looked for a complete dynamic solution which does automatically exclude some kind of "configuration file update service". The solution lies within the IPv6 protocol. IPv6 uses 128 bit long addresses. This is a huge address space and obviously greater than 80 bits. Because OnionCat should act as a private network with public access, we chose a network prefix of the *unique local IPv6 unicast addresses* according to [3]. It perfectly meets the requirements for OnionCat. The smallest possible prefix length as defined in the standard document is 48 bits which leaves another 80 bits for addressing hosts. Using this configuration, OnionCat can translate IPv6 addresses to onion-URLs and vice versa. If an IPv6 packet arrives from the operating system OnionCat extracts the lowest 80 bits from the packet's destination IPv6 address, translates it into an onion-URL, and requests Tor to open a virtual circuit to the desired destination. After the connection is setup OnionCat starts forwarding all packets through this virtual circuit. On the other end of the virtual circuit, OnionCat receives the packets from Tor and forwards them to the operating system. The operating system then in turn does with IP packets what it always does. From the operation system's point of view, there is no difference if a packet arrived on a physical Ethernet interface or from OnionCat's virtual tunnel interface. More details about OnionCat's addressing and forwarding mechanism can be found in [1].

This method perfectly distributes the VPN entrance point. In this configuration every client is an entrance point. Summarized for network users, all one needs to know about is the destination IP address.

[7] Specifically for Tor it is true that running a hidden service does not require it to be a transit node which eliminates the headache of attracting huge amounts of traffic. This is of high importance for users with low bandwidth Internet connectivity.

3 Installation And Configuration

The default application for OnionCat is to create an anonymous VPN which is publicly accessible. It enables users to take part in the anonymous network. Once being a participant one could either use the network's services or provide ones own services or both. A prerequisite is to have a network address. The addressing method basically was discussed in 2.2. Thus, we first need to install and configure the anonymizer and run a hidden service. The following explanations refer to Tor as an anonymizing transport network, because OnionCat was originally developed for Tor and it is known to run stable with it.

3.1 Install and Configure Tor

To install Tor there are basically two ways: install it with a package manager or compile and install it from source. The first solution is probably the easiest way and usually suits most users' requirements. To build from source gives a little bit more flexibility in fine tuning several build options. Details on package installations and package mirrors should be looked up on the operating system's or distribution's main sites.

For Windows and MacOS X you should follow the links on the download page of the Tor project: http://www.torproject.org/easy-download.html.en. For both OSes *Vidalia* (www.vidalia-project.org) is installed together with Tor. Vidalia is a configuration and control GUI for Tor which is also available for Linux.

After successful installation we need to add a *hidden service*. This is done by either editing the Tor configuration file torrc, or by adding it with Vidalia. The latter results in Vidalia editing the configuration file of Tor. The configuration file is usually located in /etc/tor/torrc or /usr/local/etc/tor/torrc. On Windows it is located in C:\Documents and Settings\<user>\Vidalia\torrc Add the following two lines to the configuration file:

```
HiddenServiceDir /var/lib/tor/hidden_service/
HiddenServicePort 8060 127.0.0.1:8060
```

On Windows the full path may be omitted. The directory will be created in C:\Documents and Settings\<user>. The `HiddenServiceDir` directive specifies the directory where to locate the private key for the hidden service. `HiddenServicePort` specifies that all TCP connections, which are dedicated to virtual destination port 8060 from within Tor, are forwarded to the local host (127.0.0.1) on TCP port 8060. This is the port that OnionCat listens to by default. The port numbers should not be changed unless you know exactly what you're doing.

Now start Tor but, **make sure that the system clock is correct** beforehand. Tor will then create a directory at the location specified by `HiddenServiceDir`, as well as put two files into it: private_key and hostname. The first one contains the private key associated with the local hidden service. If running a service for other users, a web service for example, it is a good idea to backup this key to a safe place. It is with this key that a specific hidden service is uniquely identified. If the machine crashes and all data is lost, the hidden service can be recovered by copying the backed up key to the hidden service directory on the new machine.

The file hostname contains the hostname which is used by the Tor network to lookup and connect to this hidden service. It is the onion-URL. Look into the file. It contains a string like a5ccbdkubbr2jlcp.onion. Vidalia will display the hostname in "Provided Hidden Services" field in the "Services" settings window. You will need this hostname for OnionCat setup as explained below. Have a look at the log file to see if Tor is working. If using Vidalia, then just click on "Message Log".

If Tor works correctly it will say "Tor has successfully opened a circuit. Looks like client functionality is working.". Note that Tor may need some time (a few minutes) to boot.

3.2 Installing OnionCat

Now, after successful installation of Tor, we can run OnionCat. As long as there are no packages[8] OnionCat must be built from source. The steps are

1. Prepare build environment.
2. Build and install OnionCat.
3. Configure and test OnionCat.

On Unix-like OSes step 1 is not very difficult and it is most likely to be already setup. All that OnionCat needs is a C compiler, usually GNU gcc and the GNU make utility. On Windows we also need those two programs. Before this can be done, it is necessary to create a POSIX-like environment. This is done with Cygwin (www.cygwin.com). Using Cygwin is more difficult, hence, I will explain it in a separate Section 3.3. If you are going to install OnionCat on Windows read Section 3.3 before.

Download an OnionCat source tarball from www.cypherpunk.at/ocat/download/. Untar it. Change to the directory and configure it as described on that page. Now you should be able to run OnionCat by typing ocat.

3.3 Installing Cygwin on Windows

To run OnionCat on Windows create a POSIX-like environment. Go to www.cygwin.com and download and run the Cygwin installer. It will ask some questions, but click continue until you reach the package selection menu and select "gcc: C-Compiler" and "make: GNU 'make' utility". Both are found in the "devel" section of the package selection window. Continue with the installation process until finished.

OnionCat is IPv6-based but Cygwin unfortunately does not support IPv6 at the current stage of development. But luckily there is an IPv6 patch available at win6.jp/Cygwin/ [9] by Jun-ya Kato. Download and install it as described on that page.

The next step is to install the TAP driver. This is a virtual network interface, usually called a *tunnel device*. This is the virtual layer 3 interface for Windows. Go to www.openvpn.org and download the OpenVPN Windows installer. To run OnionCat, OpenVPN itself is not necessary but the installer contains the TAP driver which was developed by the OpenVPN project. It is licensed under GPL version 2 with some additions. After downloading, execute the installer. It will display an options menu where you can un-select everything except the TAP driver. Continue with installation.

After successful installation click on the Cygwin icon on the desktop and continue reading at Section 3.2.

3.4 Configuring OnionCat

To configure OnionCat for its primary intention as client for the open anonymous network, we need the onion-URL which is located in the hostname file in the hidden service's directory

[8] The package building process can be very time consuming. There are already packages in preparation for FreeBSD, OpenBSD, Debian and Ubuntu Linux and MacOS X.

(see Section 3.1). When running OnionCat the first time you probably should run OnionCat in foreground to make sure that everything works correctly.[9]

In the shell, run OnionCat as root with the command ocat -B <your_onion_url>. OnionCat will produce some output. There might be errors like "select encountered error: "Interrupted system call", restarting". As long as this just happens during startup it can safely be ignored. There might also be the warning "can't get information for user "tor": "user not found", defaulting to uid 65534" which can also be ignored.

Now check if everything is configured correctly. Issue the command ifconfig. It lists several stanzas. There should be a stanza for every registered network device. One should read tun0 and have an IPv6 address assign. If the tun0 stanza exists but has no IPv6 address assigned OnionCat may have failed assigning the address. In some very rare cases this may happen for a currently unknown reason. In that case assign the address manually. In order to do this, issue the command ocat -i <your_onion_url>. It should return the IPv6 address associated with your onion-URL. Now configure the address with ifconfig. Lookup the correct syntax of ifconfig in the appropriate man page.

Now check the IPv6 routing table: netstat -nr6.[10] It lists all entries of the kernel's IPv6 routing table and it should contain at least one entry: the OnionCat IPv6 prefix fd87:d87e:eb43::/48 pointing to the tunnel device. If there is no such entry, which could happen in some very rare cases, add the route manually. Lookup the correct syntax in route man page.

4 Using The Global Anonymous Network

If everything is setup correctly as described in the previous Sections you should now be able to use the OnionCat global anonymous network. First try to ping one of the existing hidden OnionCat services. Currently there are a few services known to be permanently online.

- dot.aio[11] (fd87:d87e:eb43:f683:64ac:73f9:61ac:9a00) is a web-based service registration directory. It's intended to let OnionCat service providers register their service in order to be found by others. It's not required for a service to be registered but it enhances usability for new users. They can browse this page and lookup existing OnionCat services.
- irc.onion.aio (fd87:d87e:eb43:2243:5f84:5b12:7bb5:bbc2) basically is a *Internet Relay Chat* (IRC) server. IRC is based on the protocol definition of RFC1459 [5]. For a quick introduction have a look at Wikipedia at en.wikipedia.org/wiki/Internet_Relay_Chat. There is also a web-based audio stream ("OnionCat Radio") available on port 1337 at this same address and a new, web-based community plattform called "*Whose Space?*".
- ping.onion.aio (fd87:d87e:eb43:f947:ad24:ec81:8abe:753e) currently does nothing than just respond to echo requests.
- mail.onion.aio (fd87:d87e:eb43:744:208d:5408:63a4:ac4f) is a combined SMTP/POP3 server. It accepts mails on port 25 for recipients of domain onion.aio (e.g. eagle@onion.aio which is my email address). Users can fetch mail using the POP3 protocol on port 110. Mailboxes need to be registered in advance. Unfortunately there is currently no automatic registration service available. Post an email to onionmail@onion.aio on this server in order to get an account. Note that this is completely anonymous as long as you don't send personal information across with your email.

[9] At the time of writing this document (March 8, 2009) OnionCat will fail on Windows if not run in the foreground.
[10] The digit '6' might be omitted on some OSes.
[11] The term "aio" refers to *anonymous Internet overlay*.

Try to ping one of those hosts by issuing the ping6 command which is ping for IPv6. After some time it will respond and list the *round trip time* (RTT) in the right most column. Be patient, Tor may need up to one minute for the first time it connects to a hidden service. After the connection is setup the RTT will be between 0.5 and 10 seconds. Currently there are many efforts within the Tor project to improve connection setup time and RTT in respect to hidden services.

If everything worked until now, you can start using the network as you do with Internet with the sole exception that there is no DNS. It requires the use of plain IP addresses instead of domain names. As long as there's no feasible DNS solution, the host names can be registered locally. On all Unix-like OSes this is easily done by just putting IP address hostname pairs into the file /etc/hosts. This is possible even on Windows. The file is usually located at C:\WINDOWS\SYSTEM32\drivers\etc\hosts.

5 Conclusion

OnionCat is an add-on for anonymizing networks like Tor. It interfaces with the IP routing process of the kernel and creates a VPN on top of anonymizing networks. Within this document I promoted the primary development idea of OnionCat, described the basic concepts, and gave a brief installation, configuration, and usage guide for OnionCat. Additionally there are already some services available which were presented.

The OnionCat software is still in heavy development and may not work on every system without further intervention, but we have managed to port it to major operating systems like Windows XP and MacOS X.

One problem still not solved sufficiently is the DNS problem, specifically how to resolve hostnames to OnionCat IPv6 addresses. As a matter of course they could be stored within Internet DNS but this most likely will leak information. With that in mind it is not a good idea. A feasible solution might be to setup a private DNS within OnionCat. That is basically no problem but it would require a user to have some kind of Split-DNS service running locally. This scenario would lead to additional installation effort.

References

[1] Bernhard R. Fischer. OnionCat - A Tor-based Anonymous VPN. In *Proceedings of the 25th Chaos Communication Congress*. Chaos Computer Club, December 2008.

[2] Bernhard R. Fischer. OnionCat Project Site. http://www.cypherpunk.at/onioncat/, 2009.

[3] R. Hinden and B. Haberman. Unique local IPv6 unicast addresses. RFC 4193, October 2005.

[4] Dave Kosiur. *Virtual Private Networks*. John Wiley & Sons, Inc., 1998.

[5] J. Oikarinen and D. Reed. RFC 1459: Internet Relay Chat Protocol, May 1993. Status: EXPERIMENTAL.

[6] The Tor Project. Tor Rendezvous Specification. http://www.torproject.org/svn/-trunk/doc/spec/rend-spec.txt, 2008.

[7] Andrew S. Tanenbaum. *Computer Networks, Fourth Edition*. Prentice Hall PTR, August 2002.

[8] The Tor Project. http://www.torproject.org/.

[9] Jun ya Kato. Cygwin IPv6 Extension Patch. http://win6.jp/Cygwin/, 2008.

On Relations between Anonymity and Unlinkability

Lars Fischer
inj4n@chaos-darmstadt.de

March 18, 2009

Abstract

Anonymity and *unlinkability* are two distinct privacy problems. In this work a formal construction is introduced that shows that anonymity formally is a sub-problem of unlinkability. Anonymity is described as inability of an attacker to choose the correct matching. Unlinkability is modelled as the inability to choose the correct partition. In this paper it is shown that anonymity problems are sub-problems of unlinkability. A formalisation allows to find a mapping between anonymity and unlinkability.

1 Introduction

From an abstract point of view anonymity problems are sub-problems of unlinkability problems. This means that every anonymity problem can be modelled as an restricted unlinkability problem with additional context information. This relation has already been mentioned in the known terminology by Pfitzmann/Hansen, [PH08] only in this paper this is formalised using context information classes from [FMP07].

An unlinkability-attacker is not able to, or interested in, identification of subjects. Its objective is to discover the equivalence relation on *items of interest* IOI that links IOI with equal (unidentified) subject/role. The question in anonymity is *who* is the sender/receiver of *this* IOI. The question in unlinkability is *which* IOIs are in the same equivalence class. In most examples, as in the PKI scenario used herein, the equivalence class will be a same-sender relation. The graphs in Section 2.3 and Section 3.1 will clarify this view on the relation between unlinkability and anonymity.

The main difference between anonymity and unlinkability problems is, that anonymity problems generally consider the linkability of *actions* and *subjects*, while unlinkability problems consider only relations between *actions*. Anonymity is concerned with matchings in bipartite graphs while unlinkability is concerned with set partitions.

This paper is structured as follows in Section 2 anonymity problems are defined. A definition of unlinkability problems is found in Section 3. In Section 4 a

Fourth Privacy Enhancing Technologies Convention (PET-CON 2009.1), pp. 19–25, 2009

mapping between anonymity and unlinkability is introduced. The paper is then concluded in Section 5

2 Anonymity Problems

Anonymity is probably the most often used term in privacy related works. Although anonymity is not equivalent to privacy but privacy is only possible if a person is able to chose to be anonymous with regard to his actions. The graph model in Section 2.3 provides a more formal view on anonymity that visualises the connection between unlinkability and anonymity.

A *unique identity* of a *subject*, from the privacy viewpoint, is the most important IOI as identities stand in m-to-n relation to subjects. Control of personal information starts at the point where a subject is enabled to decide to not disclose its unique identity.

Definition 1 (Anonymity)
"Anonymity of a subject from an attacker's perspective means that the attacker cannot sufficiently identify the subject within a set of subjects, the anonymity set." [PH08]

From the viewpoint of an attacker, anonymity is the inability of the attacker to distinguish between matchings of a bipartite graph, and know the matching that corresponds to the real mapping between identification anchors and items of interest. We put the emphasis in this definition on relation between IOI and subjects, respectively identification anchors, which is the feature distinguishing anonymity from unlinkability defined below.

2.1 Anonymity Metrics

The set of identification anchors that might be related to an IOI is commonly denoted *anonymity set*. This term has been introduced by Chaum in [Cha88]. The cardinality of an anonymity set provides the basic anonymity metric. The anonymity set is probably the most often used estimation of anonymity. Size of anonymity sets and advanced anonymity related metrics are summarised in [KRGB08].

In [SD02] Serjantov and Danezis defined the anonymity problem as follows. Given a set Ψ of subjects and roles $R = \{sender, receiver, none\}$. Let $r \in R$ be the role of a user $u \in \Psi$ with respect to a IOI $m \in M$. The objective of an anonymity-attacker is to determine the role of u with respect to m.

2.2 Anonymity Scenarios

The classical problem given where a single item of interest, e. g.,an e-mail, is related to one subject from the anonymity set. The relation between the IOI and the subjects is obscured to the attacker by the (anonymous) communication system. The image depicts the objective of the attacker to correctly relate the single IOI to a subject despite of the anonymity protection of the communication system. In general a single IOI scenario is not very favourable for anonymity as it is very difficult to hide the relation from the standard *global passive adversary*.

Given a scenario with multiple IOI and a set of subjects, the attacker's objective is to find a the real matching between subjects and IOI, which is obscured by the

(anonymous) communication system. (See [TH04] for a formal model of anonymisation system, the PROB-Channel.) The number of IOI might exceed the number of subjects, but not the other way round. Unless, that is, the set of subjects includes subjects that are not communicating, i. e., related to any IOI. Generally we assume that those subjects are not included in the scenario.

Consider for example a scenario where the mapping between IOIs "sent into an anonymity preserving network", and IOIs "leaving this network" are hidden from the attacker. This is an anonymity problem insofar, as both sets are disjoint, one could be interpreted as identification anchors, and the attacker seeks to find the mapping. A similar interpretation exists with the mapping between disjoint sets identification anchors. *Receive* and *send* are, though, not the only possible distinction classes, but probably the most common. For simplicity reasons we may assume, that in these scenarios both sets have equal cardinality and a bijective matching exists.

In other works, e. g., [Mal08] this type of scenarios has been denoted as unlinkability problem. We refrain from this terminology here, because the main distinction of anonymity problems herein is that they can be modelled as matching in bipartite graphs. The reasons for this different terminology will become more clear in the remainder of this chapter.

In the previous scenarios the choice which set defined the *identification anchors* naturally was the set of subjects. In these two scenarios the choice is arbitrary as long as it can be assumed that the chosen identification anchors are unique within their subset.

2.3 PKI Graph Model

A *horizontal linkability graph*, describes anonymity in a PKI scenario with certification authorities S, certificate using devices D, and messages M which are signed using keys related to the certificates.

$$\begin{aligned} \mathcal{G}_h &= (\mathcal{V}_h, \mathcal{E}_h, \mathcal{P}_h) && (1)\\ \mathcal{V}_h &= S \cup D \cup C \cup M && (2)\\ \mathcal{E}_h &\subseteq (S \times D) \cup (D \times C) \cup (C \times M) && (3)\\ \mathcal{P}_h &: \mathcal{E}_h \to [0,1]. && (4) \end{aligned}$$

$\mathcal{G}_h$ is a graph consisting of vertexes $\mathcal{V}_h$, edges $\mathcal{E}_h$, and a weighting function $\mathcal{P}$. $\mathcal{G}$ is, in this construction, a reduced 4-partite graph with edges only between classes. The subset $S \times D$ of $\mathcal{E}_h$ describes the relations between owners and devices. Subsequently the edge subsets $D \times C$ and $C \times M$ denote the relations between devices and certificates, as well as between certificates and devices.

An edge-weight $\mathcal{P}_h$ of zero denotes that the edge's endpoints are not related. In that way $\mathcal{P}_h$ may be used to denote the belief of an onlooker that a certain edge is part of the real graph. For example the world view of an attacker might be expressed in that way.

Applying Definition 1 of anonymity to $\mathcal{G}_h$ one may observe that $\mathcal{G}_h$ represents three distinct layers of anonymity problems, anonymity with *identity anchors* IA as subjects and IOI as devices, between IA as devices and IOI as certificates, as well as between IA certificates and IOI messages. In Figure 1 an example graph is

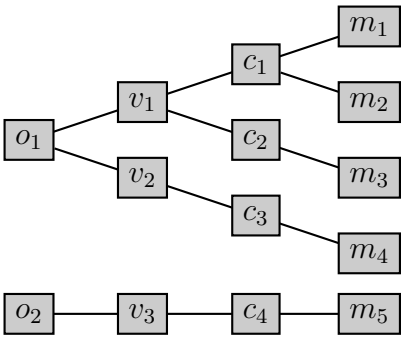


Figure 1: Horizontal Linkability Graph

shown. Observing the picture it becomes intuitively clear that any other combination of a "left" and a "right" set as IA and IOI may be derived from the graph by combination of edges (and weights) on shortest paths.

An anonymity problem thus is the correct mapping of IA to IOI. The hypotheses space is the set of all matchings between the set of IA and the set of IOI. Every layer in $\mathcal{G}_h$ defines an anonymity problem.

3 Unlinkability Problems

In the extremal case every subject may enjoy perfect anonymity, e. g., using a different pseudonym for every action or better, no identification at all. From the anonymity point of view this subject enjoys perfect privacy, but alas, it is not, for an attacker might be able to relate different actions of a subject to each other. In an unlinkability problem identification anchors are not distinguished from IOI, i. e., all considered objects are IOI without special function.

Due to the lack of explicit identification anchors *anonymity* is an insufficient description of these problems. The notion used to describe that an attacker can not (correctly) relate IOI is *unlinkability* [PH08, SK03, FMP07].

Definition 2 (Unlinkability)
"Unlinkability of two or more items of interest (IOIs, e.g., subjects, actions, ...) from an attacker's perspective means that within the system (comprising these and possibly other items), the attacker cannot [...] distinguish whether these IOIs are related or not." [PH08]

We take the viewpoint of an attacker here, defining unlinkability as the inability to find a clustering of a set of IOI that corresponds to a true relation between IOI. Relations in this sense normally denote *related with respect to sender equivalence*, but in general any (unknown) attribute of actions can be observed. The term *sender equivalence* denotes that IOI are in the same equivalence class if they have been originated by a subject under the same identification anchor.

Unlinkability describes problems where no set of identification anchors is known and only IOI are concerned. The opposite of unlinkability is *linkability*, which is also sometimes used in this work. Linkability describes the ability of an attacker to correctly relate IOI to each other.

Global unlinkability problems can be formally defined as finding the correct partition π^* from the set of all partitions Π_M of a set of items of interest M.

3.1 PKI Graph Model

A *vertical linkability graph* in the PKI scenario, representing unlinkability problems, is a collection of complete graphs as the equality relation modelled is transitive. A vertical graph is constructed as follows

$$\begin{aligned} \mathcal{G}_v &= (\mathcal{V}_v, \mathcal{E}_v, \mathcal{P}_v) && (5)\\ \mathcal{V}_v &= S \cup D \cup C \cup M && (6)\\ \mathcal{E}_v &\subseteq (S \times S) \cup (D \times D) \cup (C \times C) \cup (M \times M) && (7)\\ \mathcal{P}_v &: \mathcal{E}_v \to [0,1]. && (8) \end{aligned}$$

Each class of nodes in $\mathcal{G}_v$ provides an individual unlinkability problem,e. g., (un-)linkability of devices with respect to equal subjects, unlinkability of messages with respect to equal device, etc.

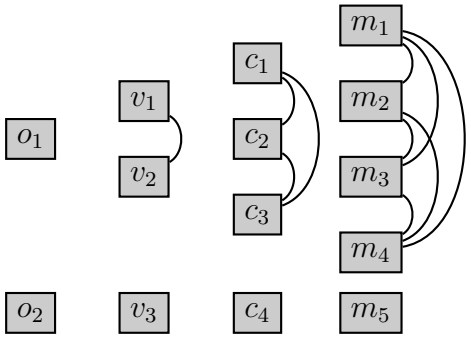


Figure 2: Vertical Linkability Graph

4 Mapping Problems

These horizontal and vertical graphs provide an representation that clearly shows the relation between anonymity and unlinkability. The connection between $\mathcal{G}_h$ and $\mathcal{G}_v$ is the node set which contains the same elements, i. e., subjects, devices, certificates, and messages. One may observe the duality between the edge sets of the graphs. Given $\mathcal{E}_h$ and $\mathcal{P}_h$ corresponding $\mathcal{E}_v$ and $\mathcal{P}_v$ can be inferred. This observation leads to the conjecture that it must be possible to express anonymity problems as special unlinkability problems.

Anonymity problems can be mapped onto unlinkability problems by using the hint-class "breach of unlinkability" from [FMP07]. This class describes the situation where an unlinkability-attacker gets to know a set of elements that are in different equivalence classes.

Given our known set of IOI M and an additional set of subject identifiers U that contains identification anchors[1]. The disjoint union $M_U := M \cup^* U$ provides a new set of items of interest. TWe denote $\Pi_M(\mathcal{H}_U)$ the set of set partitions of set M, conditioned by the hint $\mathcal{H}_U$. The hint $\mathcal{H}_U$ denotes, that no two elements of the set U are in the same cluster. The hypotheses space then is defined in [FMP07] by

$$\Pi_M(\mathcal{H}_U) := \{\pi \in \Pi_{M_U} : \forall\{m, m'\} \subseteq U \Rightarrow m \nsim_\pi m'\}.$$

[1] assume the attacker stumbled upon a list of identifiers

Where $m \nsim_\pi m'$ denotes that messages m, m' are not in the same equivalence class as defined by partition π of M_U and Π_{M_U} denotes the set of set partitions of set M_U.

Knowing the subject identifiers U, we can further use the hint $\mathcal{H}_{|U|}$ which defines that the number of clusters is equal to $|U|$. This reflects the common global anonymity scenario where all subject identifiers are known and each item of interest has to be related to exactly one subject. This can be modelled along the lines of the first class of hints in [FMP07]: "number of equivalence classes". Combining both hints, we can compute a restricted unlinkability hypotheses space of set partitions as

$$\Pi_M(\mathcal{H}_U, \mathcal{H}_{|U|}) = \{\pi \in \Pi_{M_U} : |\pi| = |U| \text{ and } \forall\{m, m'\} \subseteq U \Rightarrow m \nsim_\pi m'\}. \quad (9)$$

Where $|\pi|$ denotes the number of clusters, i. e., subject equivalence classes, in a set partition π.

By construction we have shown that any anonymity problem can be mapped onto an unlinkability problem. Obviously the other direction is not easily possible because, as above construction shows, the set of hypotheses in anonymity is but a subset of the unlinkability hypotheses set.

5 Conclusion

In this paper anonymity and unlinkability problems have been formalised as graphs. Using unlinkability hint classes it could be shown by construction that anonymity is a sub-problem of unlinkability.

Acknowledgements

This work has been notably improved by the comments from two of the anonymous reviewers.

References

[Cha88] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1(1):65–75, January 1988.

[FMP07] Matthias Franz, Bernd Meyer und Andreas Pashalidis. Attacking Unlinkability: The Importance of Context. In *Proceedings of the Privacy Enhancing Technologies 2007*, 2007. correction needed.

[KRGB08] Douglas J. Kelly, Richard A. Raines, Michael R. Grimaila und Rusty O. Baldwin. A Survey of State-of-the-Art in Anonymity Metrics. In *Proceedings of the 1st ACM workshop on Network Data Anonymization*, Seiten 31–40, 2008.

[Mal08] Bradley Malin. k-Unlinkability: A privacy protection model for distributed data. *Data Knowl. Eng.*, 64(1):294–311, 2008.

[PH08] Andreas Pfitzmann und Marit Hansen. Anonymity, Unlinkability, Unobservability, Pseudonymity, and Identity Managment - A Consolidated Proposal for Terminology. Bericht v0.31, TU-Dresden, February 2008.

[SD02] Andrei Serjantov und George Danezis. Towards an Information Theoretic Metric for Anonymity. In *Workshop on Privacy Enhancing Technologies*, LNCS 2482, April 2002.

[SK03] Sandra Steinbrecher und Stefan Köpsell. Modelling Unlinkability. In Roger Dingledine, Hrsg., *Proceedings of Privacy Enhancing Technologies Workshop, PET 2003, Dresden, Germany, March 26–28*, Jgg. 2760 of *LNCS*, Seiten 32–47. Springer, March 2003.

[TH04] Gergely Tóth und Zoltán Hornák. Measuring Anonymity in a Non-adaptive, Real-time System. In *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, Jgg. 3424 of *Springer-Verlag, LNCS*, Seiten 226–241, 2004.

Usages of Steganography for Protecting Privacy

Ádám Máté Földes
Dept. of Telecommunications,
Budapest University of Technology and Economics
fa606@hszk.bme.hu

Abstract

Steganography, the art of data hiding is a rapidly developing discipline. Many kinds of data hiding algorithms exist, but concepts that incorporate them into a framework are also needed for them to be useful. In this paper I analyse some existing possibilities for the application of steganography in the domain of protection of privacy through covert storage of information. A novel software, the 'stegodrive' is also outlined. The goal of this concept is to show steganographic capacity of a group of files as a contiguous, randomly accessible space – in other words, a steganographic file system. This way the user may hide and retrieve arbitrary type of data, provided that it fits in the 'expensive' steganographic space. In contrast to the stegodrive, the concept of most existing steganographic file system implementations is to use free space of a filesystem as cover. This kind of storage inherently carries the danger of accidental overwriting of data by the operating system – the stegodrive is, however, based on already existing files, which means that it is entirely the user's responsibility to take care about the integrity of the steganographic objects rather than that of the steganography-oblivious allocation strategy of a file system driver.

1 Introduction to steganography

Steganography [PH03] is the discipline of information hiding. Algorithms that belong to this field hide data in a *cover medium* (e.g. an image), producing the *stego medium*. The goal is that the latter does not differ greatly from the former – in other words, it cannot be decided with certainty if information is hidden in a given medium. Good steganographic algorithms use a *stego key* as a secret parameter, similarly to cryptographic keys.

1.1 The level of protection provided by steganography

The aspects of information that can be protected are described as four levels in [Go00]. The first is the 'null' protection, i.e. when nothing is protected, e.g. an e-mail in plaintext. The second level of protection targets the *contents* of the information – this is

Fourth Privacy Enhancing Technologies Convention (PET-CON 2009.1), pp. 26–33, 2009

where cryptography comes into play. The third level is the protection of *metadata* (e.g. the sender of an e-mail). Privacy Enhancing Technologies (PETs) are meant to provide such protection.

The fourth level is hiding the *existence* of information. This is the protection that steganography can provide. The purpose of the use of steganography for protecting privacy is to grant *plausible deniability* [MK00] to the user, which means that an adversary cannot prove or disprove a statement from the user that a given medium is an 'empty' cover object. This way the user can repudiate having partaken in any covert storage or transmission of information, making steganography inherently useful in the field of protection of privacy.

1.2 Classification of steganographic algorithms

There are several classifications of steganographic algorithms. Such a grouping is useful for separating algorithms that were meant to fulfil different needs. Two such classes[1] are *simple data hiding* and *watermarks*. The former aims to hide arbitrary data in the cover, while the latter is used to mark the cover medium (i.e. they do not make sense without it). While *fragile watermarks* are meant to suffer noticable damage when the stego medium is transformed, *robust watermarks* are designed to survive transformations to such a degree that the cover becomes useless for its original purpose.

It can be seen that simple data hiding is the most suited for protecting the outlined purpose, although a fragile watermark system can also be used to provide integrity protection of the hidden information.[2]

1.3 Steganographic efficiency

Steganographic algorithms can be measured by several metrics[PH03] [Cv04]. Four of them are the following:

Capacity: The percentage of the cover medium that can be used to hide information. In certain cases it is dependent on the content to be hidden.

Security: The undetectability of the hidden information. Note that in many cases it is not important for an adversary to be able to read whatever was stored in the cover – it is enough to prove that hidden information exists. The concept of the security can be approached from many directions, most notably from those of human sensory, probabilistic [CM03] and information-theoretic models [HLA02].

Robustness: The durability of the hidden information against the transformation of the stego medium. It is important mostly when evaluating a robust watermark concept.

[1] http://qosip.tmit.bme.hu/twiki/pub/Main/InfoSzolgBizt/11-Adatrejtes_kepekben

[2] A simple checksum may also be enough in many situations, but watermarks may offer some benefits. For instance, some watermarks can be used to tell with a given probability where the damage to the stego medium took place.

Complexity: The needs in memory and time to execute the algorithm.

1.4 Examples of simple data hiding algorithms

As it has already been stated, many algorithms for covert storage target images and sound files. I hereby shortly describe some examples.

For uncompressed files the Least Significant Bit method is a common – but statistically vulnerable – concept: the LSBs of the cover are replaced according to the information to be hidden, bit by bit. For bitmap images this means altering the LSB for each colour component of a pixel, while in a waveform the samples are shifted.

For a compressed file (which are more likely to appear on the average user's hard drive than uncompressed ones) the LSB concept cannot be used directly, since it is not known in general if the lossy recompression will erase a certain bit. Instead, it may be a good idea to alter the quantisation process of the lossy compression algorithm, as seen in JSteg[3] and MP3Stego[4].

Furthermore, I would like to propose a concept for uncompressed images[5] that can be considered an improvement of the LSB method. Its goal is to decrease detectability of steganography for a human observer. For this the algorithm divides the image into 5 by 5 blocks of pixels (or 3 by 3, depending on capacity needs), and uses only the pixel in the center for hiding. If, however, the deviation of the entire block is smaller than a certain value, it is rejected as steganographic space. This can be described as in formula 1 where $LSBi$ is a function that maps a real number x and a set I of pixel colours to a boolean value.

$$LSBi(x, I) = \begin{cases} \text{TRUE} & \text{if } \sigma I \geq x \\ \text{FALSE} & \text{else} \end{cases} \tag{1}$$

The result is a decrease in capacity in contrast to simple LSB, but detectability is also decreased since no hiding occurs in a block if it is found too 'smooth', i.e. when it consists of too similar pixels where even the slightest modification might be detected when looking carefully enough.

It must be noted that formal evaluation is still in progress for this algorithm. From a statistical point of view it is likely to be vulnerable to the same steganalytical attacks as the normal LSB method. However, for an average human observer with no statistical software at his disposal, even the change of several low-order bits can go unnoticed when one avoids the smooth surfaces during information hiding. Therefore, a future improvement of the algorithm could be a model where the number of cover bits in a certain pixel are defined by the deviance of its proximity.

[3] http://www.computing.surrey.ac.uk/teaching/2006-07/csm25/Chapter6/jsteg-h.pdf

[4] http://www.petitcolas.net/fabien/steganography/mp3stego/index.html

[5] A similar concept may be suitable for uncompressed sound files, but research has not yet been done for that type of cover with this concept.

2 Steganographic file systems

One of the applications of steganography is to use it as a base for implementing a file system. In such a structure a certain type of steganographic space is used to store data in such a way as normal file systems do – allowing files and folders to be hidden. Many of those file systems are 'real' in the sense that they can be mounted under the operating system as traditional ones. I would like to describe two implementations from the surprisingly few: StegFS [MK00] and the TrueCrypt Hidden Volume[6].

2.1 StegFS

StegFS is a driver that allows the user to hide data in unused space of an Ext2 file system. It has 15 security levels – steganographic 'storage bins' – which can be incrementally unlocked by their corresponding passphrases, allowing the user to hand over to the adversary a stego key that corresponds to a low level while repudiating the existence of data in higher levels. Several mechanisms make it difficult for the adversary to prove that the additional security levels also contain data, which greatly enhances the *security* of data hidden with StegFS.

It must be noted that the place where StegFS hides data is seen as empty space by the operating system, which means that if write operations occur on the file system, covertly stored data is likely to be lost.[7] To counter this effect, the driver can be set to replicate stored data in the specified number of instances. It is up to the user to define the compromise between *capacity* and *robustness*. The *complexity* introduced by steganographic hiding manifests in very high performance drop (sometimes 99%), especially when writing to a StegFS file system with a high replication factor.

2.2 TrueCrypt Hidden Volume

TrueCrypt is a cryptographic software and its approach to steganography is different from the classical concept. TrueCrypt hides data inside a file system contained within the free space of an already encrypted file system. The two file systems have different passphrases associated with them – thus, the user can choose between mounting the main file system and the hidden one by entering the corresponding passphrase. If the user gives the passphrase for the normal file system to the adversary, the latter cannot prove that there is a hidden one.

TrueCrypt can be instructed to protect the hidden file system when mounting the normal one by entering both passphrases in the corresponding text fields. If the protection is not activated, the hidden data is prone to be overwritten if the user writes to the normal file system.

[6] http://www.truecrypt.org/docs/hidden-volume.php

[7] This does not happen if the steganographic file system is mounted when the write operations take place.

When *capacity* and *robustness* are considered, TrueCrypt is fairly similar to a StegFS file system with a replication factor of 1: the more data is stored on the hidden drive, the less 'real' free space the cover file system has, therefore, the probability of the damage of hidden data increases when the cover file system is used for writing. *Undetectability* relies on the randomness of the output of the cryptographic algorithms in use. If the used cryptosystem produces an output whose entropy is almost the same as that of random data that occupies 'real' free space, detecting the presence of the hidden volume through statistical analysis becomes very hard. *Complexity* also depends on the cryptographic algorithms in use, most notably the number of them. In a worst case scenario a cascade of 3 encryption algorithms protects the hidden data which can cause a serious performance drop.

3 The 'stegodrive'-concept

My proposal, the 'stegodrive' also addresses the problem of covert storage of information. The following goals motivated its design:

- devising a concept that, when impelemented, can be used for protecting privacy through covert storage of arbitrary type of data
- making steganographic capacity of a group of cover media visible as a randomly accessible, contiguous space
- allowing the user to customise the compromise between capacity and security
- designing the system so that remaining free space can be estimated

The model that aims to address these needs is described in subsection 3.1. It may undergo minor refinements in the future.

3.1 Structure of the stegodrive

The stegodrive has three parts: the database, the storage part and the graphical interface (see figure 1). The *database* is responsible for storing the metadata for hidden files, i.e. their name and size. It is exported into a regular file as a sequence of SQL queries when hiding some data into the cover files. This list of queries is loaded when 'mounting' the stegodrive, and the HSQLDB driver is used to create a database in memory from them. The user needs to hide this file through other means (e.g. taking it with himself on a flash drive that is not accessible to adversaries or storing it with StegFS with a very high replication factor). In later implementations both the place and the storage format of the database is likely to change.

The database is designed to support hiding a file into multiple covers if it does not fit in one in its entirety.It is the responsibility of the other parts to assemble the hidden files from their respective covers upon request.

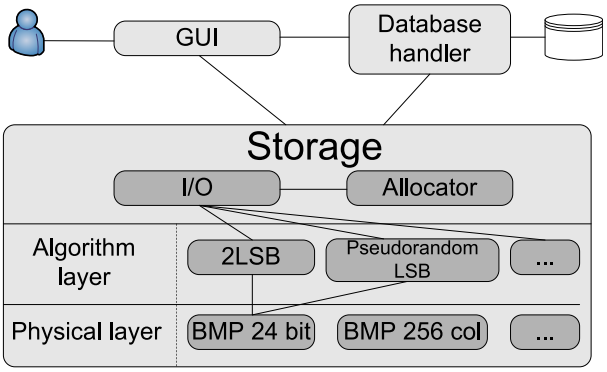


Figure 1: Structure of the stegodrive

The *storage part* has basically four elements. First, the physical layer is responsible for knowing the properties of a certain cover file, i.e. how the parts of the cover that can be exploited for data hiding can be accessed. There can be physical layer objects for 24 bit BMP files, 256 colour BMP files, WAV files, HTML files, etc.

Second, the algorithm layer is responsible for implementing a specific algorithm for a certain type of cover medium. For instance, a 24 bit BMP file can have an algorithm that hides data in the 2 LSBs of a colour component, one that uses only the LSB for each colour components and one that uses only the LSB but the pixel to operate on is chosen in a pseudorandom fashion. It is the responsibility of the algorithm layer to hide the inner structure of the cover for the upper layers, and make it seem as a continuous space that is randomly accessible.

Third, the I/O layer integrates the space chunks provided by the algorithm layer objects as a topmost object. It provides an interface to write to and read from the stegodrive an entire hidden file. It can also provide services such as lossless compression of hidden files to save space and encryption for additional security. The I/O layer uses an allocator, an ancillary layer to define the place where the parts of the next file are to be hidden. Several allocation strategies can exist, e.g. linear and 'load balance'.

The *graphical user interface* allows the user to select the files to be used as covers, supply a passphrase to derive the stegokey, define the compromise between steganographic capacity and security with a slider, and hide-extract files of arbitrary type to be protected. The position of the slider defines the algorithm to be used for a cover medium type for which several algorithms have been implemented, and – if applicable – the security parameter of the chosen algorithm.

3.2 Comparison with other implementations

The stegodrive is fundamentally different from the other implementations mentioned: it does not work on the filesystem level. Consequently, the danger of overwriting the hidden information is somewhat lower than for a steganographic file system whose cover file system is in regular use – the user is unlikely to delete the cover files if he knows that they contain his hidden information, which is impossible for a free space contained file system because it is up to the operating system to decide where to write,

depriving the user of the influence on the integrity of his hidden data.

The database file is the most vulnerable point in the system: if it is stolen by an adversary, the purpose of steganography is thwarted. It is possible that in future implementations the database will be stored steganographically. The obstacles mainly derive from the fact that the entire database is unlikely to fit into a single cover file if many files are hidden. Furthermore, if the user has too few cover files, the database can occupy a considerable part in their valuable steganographic capacity. However, steganographic storage of the database seems to be the only feasible solution from the point of view of deniability on the long run.

3.3 Evaluation of the stegodrive

The major merits of the stegodrive are:

Controllability: the user has the choice to pick where he wants his data to be hidden, without the fear that the OS will overwrite it.

Versatility: the concept inherently supports adding almost arbitrary cover types and algorithms.

Flexibility: the user can pick somewhat less secure steganographic algorithms if he wants to win some space while sacrificing some degree of security.

On the other hand however, the stegodrive cannot be considered a real file system in its current state, because it cannot be mounted under an operating system as such – even support for directories is missing. The implementation of these features would require considerable additional work, but it is a good way forward.

From the point of view of steganographic efficiency of the stegodrive, the parameters are dependent on the actual implemented algorithms. However, *robustness* may be a merit of the concept itself if the user has and exercises control over the placement and integrity of cover files. There can be further improvements if chunk replication is implemented.

4 Conclusion

In this short description I have analysed current implementations, and proposed a solution for applying steganographic algorithms in a novel framework which is suitable for use in the domain of protection of broadly interpreted privacy. However, there are some aspects that need further research, most notably the most convenient and secure placement of the metadata of hidden files. Futhermore, implementation of more secure steganographic algorithms is also desirable in the future, since the current methods can be detected by statistical steganalysis.

5 Acknowledgements

This paper was made in the frame of Mobile Innovation Centre's integrated project Nr. 2.3 supported by the National Office for Research and Technology (Mobile 01/2004 contract). I would also like to thank Gábor Gulyás, PhD student and Győző Gódor, assistant professor at the Department of Telecommunications at BUTE for supporting my research.

References

[PH03] Niels Provos, Peter Honeyman, *Hide and Seek: An Introduction to Steganography*, IEEE SECURITY & PRIVACY, 2003

[Cv04] Cvejic, *Algorithms for Audio Watermarking and Steganography*, Oulu University Press, Oulu, 2004

[Go00] Ian Avrum Goldberg, *A Pseudonymous Communications Infrastructure for the Internet*, UC Berkeley, 2000

[MK00] Andrew D. McDonald, Markus G. Kuhn, 'StegFS: A Steganographic File System for Linux', in A. Pfitzmann (Ed.): IH'99, LNCS 1768, pp. 463-477, 2000.

[Pa05] Papapanagiotou, Kellinis, Marias, Georgiadis, *Alternatives for Multimedia Messaging System Steganography*, in Y. Hao et al. (Eds.): CIS 2005, Part II, LNAI 3802, pp. 589 – 596, 2005

[HLA02] Hopper, Langford, Ahn, *Provable Secure Steganography*, School of Computer Science, Carnegie Mellon University, 2002

[CM03] Chandramouli, Memon, *Steganography Capacity: A Steganalysis Perspective*, Department of E.C.E., Stevens Institute of Technology, Department of Computer Science, Polytechnic University, 2003

Design of an Anonymous Instant Messaging Service

Gábor György Gulyás
Dept. of Telecommunications
Budapest University of Technology and Economics
H-1117 Budapest, XI. Magyar tudósok körútja 2. Room: I.B.113.
gulyasg@hit.bme.hu

Abstract

Instant messaging is in its renaissance; there are hundreds of millions of users worldwide. However, as we are using these services at home and work, and even on the way between, several privacy issues arise. In this paper I formalize requirements for privacy friendly messaging services and propose a novel anonymous instant messaging service that fulfils these requirements and allows anonymity as well. The suggested solution applies the technique called Role-Based Privacy by organizing profiles in a tree hierarchy. I also provide the analysis of total anonymity and unlinkable pseudonymity in the service and highlight interesting research objectives for extending the model presented in the paper.

1 Introduction

Today, in the digital age the Internet is getting more integrated with everyday life and so do social services including Instant Messaging (IM). Age-groups ranging from teenagers to adults use these kinds of services in their everyday life at multiple locations including work, home or even use mobile messaging software while being on the move.

The authors in [SP04] interviewed several subjects and mention several privacy related issues regarding IM services: privacy from non-contacts, privacy regarding availability, and privacy of the communication. However, these problems are generic and concerning communication committed in chat services as well (previous work in [GG06]).

Fourth Privacy Enhancing Technologies Convention (PET-CON 2009.1), pp. 34–40, 2009

In this paper I propose a messaging service model that allows anonymity. The model has both the characteristics of instant messaging and the chat services: should have a contact list and allow conferences (instant messaging) and rooms that may be explored separately (difference between conferences and rooms are conferred later). For achieving anonymity and enhanced privacy settings on visibility I propose Role-Based Privacy (RBP) [RL08].

2 Requirements

The motivation of this work is to propose a privacy enhancing messaging service. The goals that such a service should accomplish are enlisted below and based on the proposal of [SP04] but refined and derived from previous work in [GG06]:

- user privacy should be protected from non-contacts by flexible protection options
- privacy should be strengthened regarding availability
- anonymity should be achievable in some contexts
- in other contexts unlinkable pseudonymity should be available for managing multiple personae
- the privacy of communication should be protected on the network
- flexible and coherent privacy settings should help users

The threat model assumed by this paper is simple: service and service level operator users are trusted (in future work this assumption may be revised), and regular users are not. Hence, user privacy should be primarily protected against other users.

3 Anonymous Messaging Service

There are different aspects and architectures for messaging services. However, due to the nature of instant messaging and chat services I propose the use of a centralized service (harmonizing with the concept of a trusted service), but different architecture types may also be applicable. For the other requirements enlisted in the previous section I propose the use of identity management based on the technique of Role-Based Privacy (see Section 3.2).

3.1 Network Architecture

For achieving anonymity user privacy should be strengthened separately on the network and application level as well. Besides protecting the confidentiality, integrity of network level communication application level protocols, such as identity management should be designed with privacy in mind (this concept is presented in previous work for the web in [GG08]).

For achieving network level anonymity an anonymizing service should be used, however, as the architecture is centralized, a MIX type service should be used to access the central servers [DC81], such as TOR, JAP or I2P[1]. Peer-to-peer connections (for file transfers, private conversations, etc.) may be anonymized or protected by other means, such as traffic analysis protected Transport Layer Security (TLS) channels.

3.2 Identity Management with Role-Based Privacy

Role-based profile management is the core concern of the service. In everyday life we share information with others according to the transactions we commit (e.g. shopping in the grocery), the role we play (e.g. co-worker, a chess club or a family member), or we conform to other criteria. If necessary these roles could be represented with unlinkability, meaning other participants are unable to link profiles realizing different roles. This concept should be implemented in messaging services to offer enhanced features on availability and anonymity as well.

Accordingly, I suggest that in services model the user should be able to manage her identities by setting up different profiles for different places, such as rooms, conferences. Profiles are structured data sets including many kinds of descriptive information on the identity such as screen name, contact information, status (visible, unavailable, busy, etc.), and in some cases a globally unique identifier that was selected during the registration process.

The most prominent difference between conferences and rooms is how users identify themselves: with their globally unique pseudonym in conferences, and with locally unique identifiers within rooms for allowing anonymity. In the latter case anonymity is possible (unlinkable pseudonymity) as profiles may be changed any time without the presence of any trivially linkable information (such as identifiers). However, identity changes should be carried out carefully (e.g. no messages or timings should also indicate the link between profiles). In rooms total anonymity (no identifiers) should also be an option.

There should be other ways for contacting other users, like dialogues and contact lists (also called buddy lists in some IM services). A dialogue should be represented as a conference; however, global identifiers may not be revealed as it may not be known. Contact lists would be useless without global identifiers; therefore contacts (ordered in contact groups) should be identified under any circumstances.

For providing flexible and easily perspicuous identity management a profile hierarchy should be used: profiles should be ordered within a tree-like hierarchy for providing inheritance of profiles. Unset profiles should inherit their settings from the nearest ancestor that is set. The concept of using a profile hierarchy is similar to the concept introduced by the authors in [MH08] on how pseudonyms should be used

[1] http://www.torproject.org, http://www.i2p2.de, http://anon.inf.tu-dresden.de/index_en.html

for managing linkability. The lower profiles are set with the more distinct values the higher unlinkability is achieved against other users.

However, total unlinkability of profiles is only achievable in rooms where no global identifiers are attached. In other contexts only separable visibility may be achieved, which is also an important privacy protecting feature [SP04]. The profile hierarchy is illustrated on Figure 1.

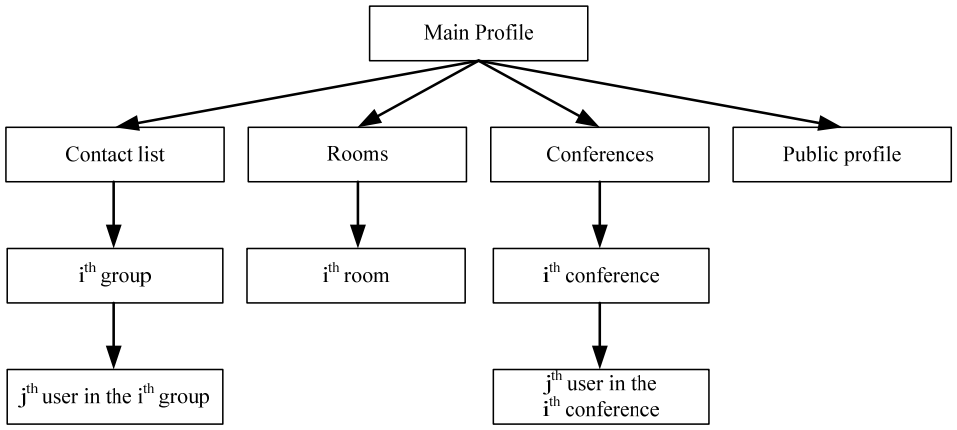


Figure 1. Profile hierarchy.

I elaborate the usage of profile hierarchy with a simple example. For instance a user is known as 'John Doe' within all rooms. However, he joins the room called 'Flea Market' with an alias 'Bob'. Since other users only see these information these identities may not be linked, although the user may remove the profile set for the flea market any time to reveal a higher level profile know as 'John Doe'. In conferences the pseudonym acquired would also be visible in the profile providing linkability.

In messaging services today similar operations are used as below for managing privacy; however, the proposed operations are sound with the RBP model presented and defined to manage profiles in the hierarchy by realizing privacy and status settings at the selected node of the profile hierarchy:

- *Ignore* (⇔ *enable*): an ignored user (identity) will see the ignorer user's profile, however, will not be able to send messages to her.
- *Ban* (⇔ *enable*): similar to the ignore operation, but the banned user sees the banner's offline profile (or which is for unknown users). Banning would allow hidden surveillance in rooms, thus it should not be allowed (allowing anonymous reading would realize the same effect).
- *Identity change* ⇔ *reveal identity*: in rooms this operation means an identity change (introducing unlinkable identities), in other places it is a simple profile changing operation.

The need for privacy protection against non contact users should be handled by introducing anonymous credentials [JC01]. For instance malicious actions need to be

stored in a special passport which is only accessible for the service, but users may declare restriction towards it.

For instance if the service detects that a user sent a SPAM message, it increases the SPAM counter in the passport, or a service operator should be able to add tags to it (e.g. "virus" indicating that the user's computer was infected with some kind of a virus). These entries can not be accessed by other users, but should be able to define constraints for specific actions regarding these settings, such as limiting the access to their public profile for user without spamming activities.

3.3 Further Privacy-Related Enhancements

Profile management settings need to be harmonized with privacy protection to provide a better protection against distraction caused by alerts. For instance at work contacts in the friends group should not be able to distract the user, however, events originated by contacts in the co-workers group should alert the user. This feature provides a better and more flexible privacy protection.

Further properties may be introduced for the rooms and conferences to strengthen privacy (and for other contact places also), such as limitations (file transfer, file size, message per sec, etc.), password or key requirements, proper credentials required for join, anonymous comments. In rooms anonymous observation should also be an option.

Another possible extension would be enabling modular event sources, such as a user defined time-table for modifying profile settings, or adding location based modules (based on WiFi Access Point information or GPS location). By using these modules the user may disable co-workers after 17h, or only allow access to her office profile while she is in the company's building.

4 Analysis of the Anonymous Messaging Service Model

The anonymity criteria presented in [GG08] can be adapted to the conferred messaging scenario, in which anonymity is still a core concern, but additionally the unlinkability of identities is another one, and should be treated equally to anonymity: introducing unlinkable profiles allows a different level of anonymity (unlinkability is regarded against other users).

As the presented service model relies on a network level anonymizer (which eliminates threats regarding the privacy of the communication) only users within the service should be considered as potential attackers. Requirements regarding availability issues, anonymity and unlinkable pseudonymity are achieved by RBP within rooms, conferences and contact lists.

In conferences and on the contact list anonymity is not an option, although the proposed technique provides flexible and easily manageable privacy protection by allowing proper profile management. These parts of the service provide pseudonymous identification instead of anonymity (this may be recognized as a certain level of anonymity). Anonymity achieved in rooms by introducing unlinkable identities is always possible and total anonymity may also be enabled.

In some contexts the presence of too few individuals may reduce the chance of unlinkability. Hence, allowing the presence of some bots in the room may also help, especially if it is possible for the user to take over the control on one of them. The bot should be exiting when the user does, for avoiding confusing situations (e.g. two different users take over the same bot within an uninterrupted session).

5 Conclusion and Future Work

The proposed technique, Role-Based Privacy, is a possible solution for offering better privacy management and anonymity if implemented properly. In my opinion the suggestions presented in this paper should be generalized furthermore and extended for services based on social networks (some privacy vulnerabilities addressed in [MC08]), which are getting more and more widespread. Related work to this topic has already been submitted and accepted [GG09]. Further analysis of anonymity is also a research objective in the future within the generalized RBP model for social networking services; for instance examining analytically the unlinkability of separated identities is an interesting problem.

6 Acknowledgements

I wish to thank my supervisors, Dr. Sándor Imre and Róbert Schulcz, for supervising and supporting my research, and for funding my travel expenses.

This paper was made in the frame of Mobile Innovation Centre's integrated project Nr. 2.3 supported by the National Office for Research and Technology (Mobile 01/2004 contract).

7 References

[DC81] D. Chaum, “Untraceable electronic mail, return addresses, and digital pseudonyms”, The Communications of the ACM 24, February, 1981, pp. 84-88.

[GG06] G. Gulyás, “Analysis of anonymity and privacy in instant messaging and chat services”, In: Dr. Ferenc Kiss (ed.), Tanulmányok az információ- és tudásfolya-

matokról 11. (Alma Mater Series), pp. 137-157., BME GTK ITM, October 2006. ISSN: 1587-2386, ISBN: 963-421-429-0. (in Hungarian)
Online: http://pet-portal.eu/files/articles/2006/10/im_privacy.pdf

[GG08] G. Gulyás, R. Schulcz, and S. Imre, “Comprehensive Analysis of Web Privacy and Anonymous Web Browsers: Are Next Generation Services Based on Collaborative Filtering?”, Joint SPACE and TIME International Workshops 2008, Trondheim, Norway, 17/06/2008.

[GG09] G. Gulyás, R. Schulcz, and S. Imre, “Modeling Role-Based Privacy in Social Networking Services”, SECURWARE 2009, Athens, Greece, 18-23/06/2009. (accepted)

[JC01] J. Camenisch, A. Lysyanskaya, “An Efficient System for Non-transferable Anonymous credentials with Optional Anonymity Revocation”, In the Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '01), London, UK, 2001, pages 93-118.

[MC08] M. Chew, D. Balfanz, and B. Laurie, “(Under)mining Privacy in Social Networks”, W2SP 2008, Oakland, California, USA, 22/05/2008.

[MH08] M. Hansen, A. Schwartz, and A. Cooper, "Privacy and Identity Management", IEEE Security and Privacy, vol. 6, no. 2, Mar/Apr, 2008, pp. 38-45.

[RL08] R. Leenes, J. Schallaböck, and M. Hansen, “PRIME white paper (V3)”, 15/05/2008.

[SP04] S. Patil, A. Kobsa, “Instant Messaging and Privacy”, In Proceedings of HCI 2004, Leeds, U.K., pp. 85-88.

Effectivity of Various Data Retention Schemes for Single-Hop Proxy Servers

Dominik Herrmann
University of Regensburg, Germany

Rolf Wendolsky
JonDos GmbH, Regensburg, Germany

Abstract

Recently, member states of the European Union have legislated new data retention policies. Anonymisation services and proxy servers undermine such data retention efforts, as they allow users to masquerade their IP addresses. Providers of such services have to implement effective data retention mechanisms allowing for traceability while at the same time preserving users' privacy as far as possible. In this paper we analyse the effectivity of four data retention schemes for single-hop proxy servers which use information already stored in logs today. We assess their effectivity by applying them to the historic logs of a mid-range proxy server. According to our evaluation it is insufficient to record data on session-level. Users can only be unambiguously identified with high probability if access time and source address of each request are stored together with the destination address. This result indicates that effective data retention based on currently available identifiers comes at a high cost for users' privacy.

1 Introduction

In 2006, the European Union issued the Data Retention Directive [4]. For the purpose of law enforcement, Internet Service Providers (ISPs) may thereby be required to uncover the identity of a user, given an IP address and a timestamp. The Directive has to be implemented by member states until March 15th, 2009. The German implementation for Internet access has gone into force on January 1st, 2009. Since then, providers of telecommunication services have to retain transformation data for a period of six months.

While the implementation of data retention measures is rather straightforward for ISPs, interesting questions arise when data retention is applied to proxy servers and anonymisation services. There is still considerable uncertainty about which types of

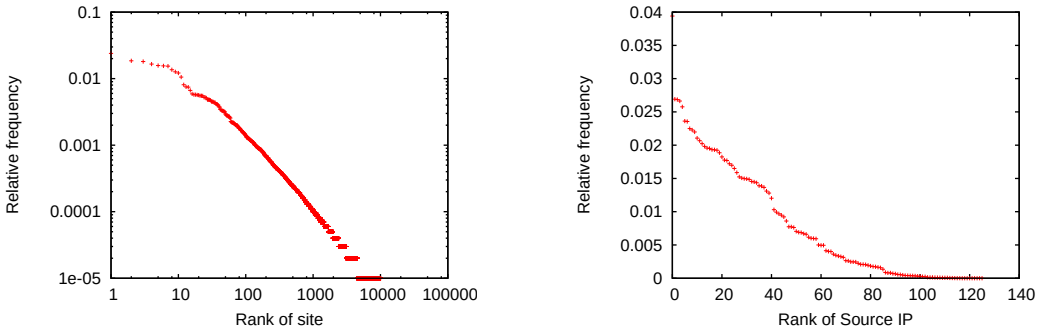
Fourth Privacy Enhancing Technologies Convention (PET-CON 2009.1), pp. 41–48, 2009

services are affected by the new data retention regulations. Although those legal discussions are of high practical relevance, they often neglect the implications regarding the involved technologies and the impact on users' privacy. The factual evidence presented in this paper is intended to foster a more technology-aware discussion.

In the context of anonymisation services and proxies, data retention measures have to allow for *traceability*, i. e., uncovering the IP address of a user from whom a suspicious connection originated (cf. Richard Clayton's PhD thesis for more information on that topic [3]). While some people consider traceability of Internet users fundamentally necessary to enable crime detection and prevention, it is criticised by others for unduly infringing users' privacy. Moreover, ISPs complain that implementing and operating a data retention infrastructure is a costly undertaking. Law enforcement agencies (LEAs) or related governmental organisations have not specified technical requirements regarding data retention on proxy servers and anonymisation services so far. Devising effective data retention mechanisms allowing for traceability while at the same time preserving users' privacy is the challenge at hand.

In this context Kesdogan et al. [7] have researched the effectivity of various intersection attacks from the literature using the log files of a proxy server. Berthold et al. [1] have evaluated the effectivity of intersection attacks on the AN.ON/JonDonym anonymisation service, i. e., whether the provider of the anonymisation service can unambiguously reconstruct the source IP address of an offender, given a number of events when the designated offender was using the service. The authors find that the size of the anonymity group decreases rapidly with an increasing number of events available for building the intersection. According to their results another means to improve traceability is increasing the accuracy of the timestamps used by LEAs. Intersection attacks have one drawback, though: they rely on the fact that LEAs are able to identify multiple requests from the same offender, all of them coming from the source IP address. Köpsell et al. [8] propose a request-level data retention scheme specifically designed for distributed anonymisation services. It is based on threshold group signatures to allow for the revocation of the anonymity of offending users while preserving the privacy of all other users. Köpsell et al. do not define which kind of information is stored to identify offending users, though. The schemes in this paper are possible realisations for their proposal.

The debate regarding to what extent providers of proxy servers and anonymisation services will have to implement data retention has not settled yet. In this paper we will analyse various conceivable retention schemes which only utilise data already available today to the providers of such services. The evaluated schemes do not rely on intersection attacks and could be implemented easily. Based on an empircial study using the log files of a medium-range proxy server we find that data retention schemes utilising currently available data is only effective if information about the requested destination addresses is stored, which is not satisfactory from a user's perspective. Therefore, our paper motivates further research in this field in order to find better data retention schemes which address the security and privacy requirements of all involved parties.



(a) Popularity of web sites (b) Activity of source IP addresses

Figure 1: Distribution of request numbers for the evaluated sample

2 Evaluation Methodology

In the interest of conciseness we limit our analysis to HTTP traffic which is relayed by single-hop web proxies. In order to get comparable results we implemented various data retention schemes and applied all of them to a common log file of a proxy server. As providers of anonymisation services refrain from keeping log files containing information to the necessary extent, for this preliminary study we used Squid log files of a local school with about 1,000 students and about 100 staff members. The log files contained the pseudonymised requests of six months (August 2008 to February 2009).

The combined log file contains 9,074,962 requests in total originating from 126 distinct (local) source IP addresses. The users requested objects from 33,258 destination IP addresses which have been accessed via 51,746 different host names. The plot in Figure 1a shows the relative access frequencies of the host names ordered by their popularity (based on the number of total requests per host name, most active first), which indicates that in our sample the retrieved web sites follow a Zipf-like or power law distribution [10]. This feature has been observed in several earlier studies for web requests from a homogenous community of users (cf. [2, 5]). According to the histogram in Figure 1b the user group consists of both, power users and less active ones.

These characteristics have to be kept in mind when interpreting the results of our study, i. e., they only apply to systems which serve a rather small and homogenuous user group and probably cannot be easily generalised to large-scale anonymisation services. The absolute values of the results are certainly affected by the specific composition of our user group and its behaviour in a school setting.[1] Nevertheless, we believe our proposed methodology may be used to assess the effectivity of data retention schemes on such systems.

For the evaluation we created stripped-down versions of the Squid log file containing only the information which would be available for the examined data retention schemes. We then analysed the effectivity as expressed by the ratio of requests which could have been unambiguously attributed to the correct source IP address for the var-

[1] Some pages containing unsuitable content for students are filtered at the proxy level. This may add to the bias in our sample.

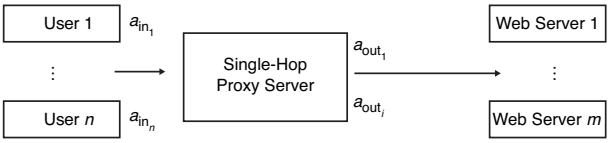


Figure 2: Model of the evaluated single-hop proxy scenario

ious schemes. The ratio was calculated by complete enumeration, i. e., we created LEA queries for each request contained in the log file, every time recording the number of potentially matching requests. For maximum effectivity the result set would have to contain only one request for each query.

3 Data Retention Schemes for Single-Hop Systems

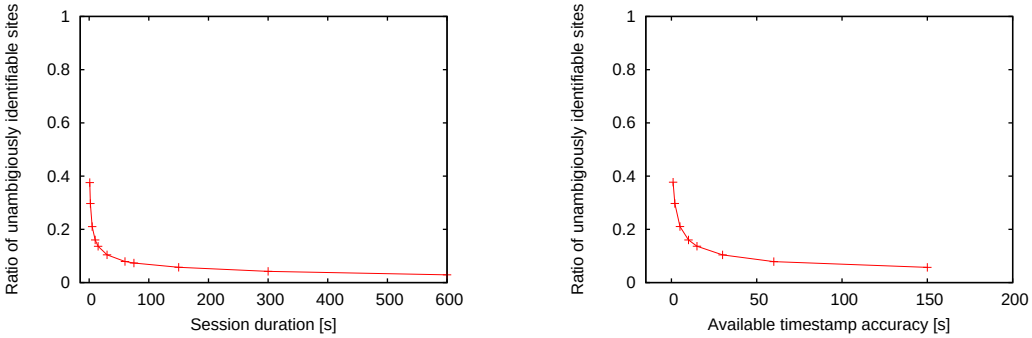
Figure 2 illustrates the single-hop setup. The proxy is used by n users with IP addresses $a_{\text{in}_i} \in A_{\text{in}}$. From the viewpoint of the destination server, the request originates from an IP address $a_{\text{out}_j} \in A_{\text{out}}$. Note that $|A_{\text{in}}| > |A_{\text{out}}|$ in most cases, i. e., the number of unique input addresses exceeds the number of IP addresses of the proxy. For our proxy $|A_{\text{out}}| = 1$. We will present four different retention schemes in the following sections.

3.1 Recording Input Addresses on Session-Level

Session-oriented services like VPN-based anonymisation services could record the relevant session-level information. If t_{start} and t_{end} denote begin and end timestamps of a user's session with the anonymisation service, the provider would store the tuple $(t_{\text{start}}, t_{\text{end}}, a_{\text{in}}, a_{\text{out}})$ for each session. Note that individual HTTP requests, which are relayed during a session, are not considered. From a privacy point of view this solution is the most desirable form of data retention. Only a bare minimum of information is recorded. Personal information – apart from the usage time – is not stored.

Traceability cannot be guaranteed at all times with this approach. Faced with a LEA query $q = (t^{(q)}, a^{(q)}_{\text{out}}, a^{(q)}_{\text{dest}})$ for some timestamp $t^{(q)}$, one of the proxy's output addresses $a^{(q)}_{\text{out}} \in A_{\text{out}}$ and the destination address $a^{(q)}_{\text{dest}}$, e. g., *q*=(2008-10-10 9:43am GMT, 132.199.2.111, 66.249.93.104), the service provider may not be able to uniquely identify one of its users as requested. He can only provide all source IP addresses a_{in_i} of all sessions established at $t^{(q)}$ and relayed over a_{out}. Note that the destination address $a^{(q)}_{\text{dest}}$ does not help to reduce the anonymity group because the service provider is not storing any destination addresses in this scheme.

With this scheme even inactive users contribute to the anonymity group. Intuitively, tracing a request back to its originator is only possible if there is only a single session at $t^{(q)}$, which is very unlikely for popular proxies. If multiple requests from different sessions could be attributed to the same user, LEAs could intersect the result sets to decrease the size of the anonymity group.



(a) Varying session durations (b) Varying timestamp accuracy

Figure 3: Data retention effectivity for session- and request-based services

Obviously, traceability largely depends on the duration of the individual user sessions. We analysed the influence of the session length on the effectivity by grouping consecutive requests from an individual IP within the simulated session duration into one contiguous session. As shown in Figure 3a the effectivity of this scheme is dropping extremely fast with increasing session durations. Even for rather short sessions of only 300 seconds, less than 5 % of requests can be identified unambiguously. For busier proxies with thousands of users this figure is expected to approach zero.

Using a regression analysis we found that the plotted data closely fits a power function ($y \approx 0.3921x^{-0.3932}$ with a residual sum of squares $\mathrm{rss} \approx 3.018 \cdot 10^{-4}$).

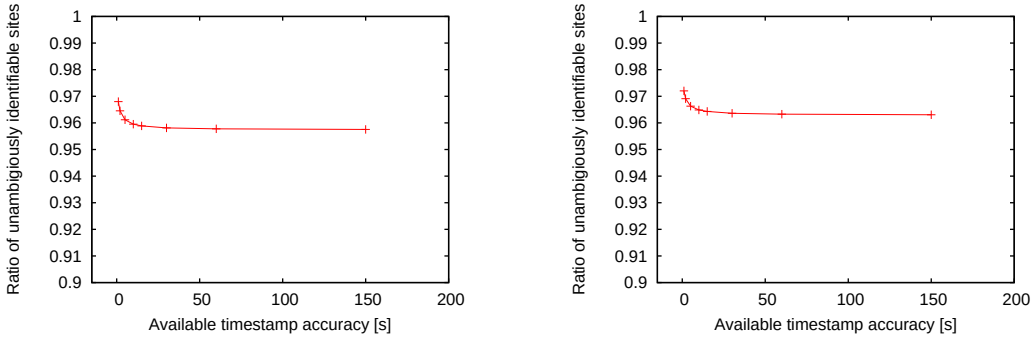
3.2 Recording Input Addresses on Request-Level

Common web proxy servers, e. g., the Squid cache proxy or many form-based CGI proxies, operate on individual HTTP requests. They could store the tuple $(t_{\mathrm{transform}}, a_{\mathrm{in}}, a_{\mathrm{out}})$, where $t_{\mathrm{transform}}$ is the point in time when the input address was transformed into the output address.[2] Anonymity groups become considerably smaller as inactive users are not included in the result set any more. Traceability cannot be guaranteed when multiple users issue requests at the same time, though.

Figure 3b depicts the effectivity of this scheme. Although the plot looks similar to the session-based case, request-based data retention is more effective: the effectivity depends only on the accuracy of the timestamps used in the log files and the LEA query. The accuracy will be degraded if the clocks of the service provider and the destination site are not synchronized or if non-deterministic network latencies cause unforeseen delays.

In comparison to the session-based data retention scheme, logging data on the request level offers potentially higher effectivity because of a much more precise time resolution. Given a hypothetical timestamp accuracy of 60 seconds, all requests within a time window of 30 seconds around the point in time specified in the LEA query are part of

[2] Of course, this scheme is not limited to services operating on a request level, i. e., session-based services like VPNs could store request-level data, too.



(a) Recording IP addresses (b) Recording host names

Figure 4: Impact of storing destination addresses on data retention effectivity

the result set. For a hypothetical timestamp accuracy of 60 seconds about 7.9 % of requests can be unambiguously identified in our sample. This ratio climbs up to 39 % if timestamp accuracy would be increased to one second. Again, we expect these figures to decrease tremendously on busy proxies.

We presume that realistic values of the timestamp accuracy for Internet hosts lie in the range between one and 60 seconds. To the best of our knowledge the available timestamp accuracy has not been analysed so far. Further research is necessary.

3.3 Recording Destination IP Addresses

In the previous section we have excluded destination addresses from data retention. For increased traceability, anonymisation services might be forced to store the IP addresses of the destination servers. In this case they would store $(t_{\text{transform}}, a_{\text{in}}, a_{\text{out}}, a_{\text{dest}})$ for each request. This approach reduces the size of the anonymity group considerably. Now, only IP addresses of users requesting an object from $a_{\text{dest}}^{(q)}$ at time $t^{(q)}$ are included. So, again, the effectivity of this scheme depends on the available timestamp accuracy (cf. Figure 4a) . Given a timestamp accuracy of 60 seconds 95.8 % of the requests in our sample can be unambiguously attributed to a single user with this scheme (96.8 % given an accuracy of one second). Effectivity is still not perfect, though, as there is still a (relatively small) possibility that several users are accessing different objects on the same destination server within the requested time window, which may happen for example when various web sites are (virtually) hosted on the same physical server.

From a privacy viewpoint storing destination IP addresses is not desirable, though, as they may reveal information about the interests of users to the service provider for the whole retention time span.

3.4 Recording Destination Host Names

The last scheme we present in this paper is based on the previous one. Instead of recording destination IP addresses, DNS host names are stored in order to further re-

duce the size of the result set. The result set will then only contain source IP addresses of users who have accessed the same (virtual) host at a given point in time, thus allowing for an exact match in most cases.

As expected our results show only small increases in effectivity when host names are stored (cf. Figure 4b). Given the timestamp accuracy of 60 seconds, for 96.3 % of the requests the originator can be identified. Apparently, the set of simultaneously retrieved pages which are co-located on the same host is rather small in our sample. Note that effectivity could still be improved slightly if – instead of host names – the complete URLs including HTTP query parameters would be stored. Even then, traceability could not be guaranteed for encrpyted web sites (HTTPS), though, because the proxy could only log host name and port of them. And of course multiple users might still coincidentally request the same URL. As the expected benefits of this scheme are rather low for our sample, we have not implemented it so far.

The effectivity of this approach comes at a high cost. While host names may disclose the personal interests and habits of users, URLs may even contain personal or sensitive information (e. g., search engine queries, session IDs, and unencrypted credentials). Storing information of this kind on a proxy server over a period of six months causes considerable privacy and security issues and therefore seems disproportionate.

4 Conclusion

This paper examined four data retention schemes in terms of their effectivity. The presented schemes only rely on data easily available to providers of proxy and anonymisation services, i. e., they are straghtforward to implement based on already existing logging facilities. Effective data retention schemes have to offer traceability of – ideally – all requests which are handled by such services to law enforcement agencies.

According to our empirical study, none of the examined schemes can guarantee traceability for all requests. Namely, we found that storing *session-level data* is not sufficient because the anonymity groups become too large even on our little-frequented proxy for typical session lengths. Logging on a *request-level basis* seems more promising, but only if the *destination address* of each request is recorded – which infringes users' privacy. None of the evaluated data retention schemes provides effective traceability while respecting users' privacy. Although we have utilised a synthetic sample, we believe that our methodology is of general value and it could be applied to many kinds of anonymisation services, e. g., CGI-based proxies using HTML forms or VPN solutions (as provided by anonymizer.com), mix cascades (provided by JonDonym [6]) and Onion Routing (cf. the Tor project [9]).

In future work we plan to repeat the evaluation with log files from a proxy server with a higher load and a more diverse user base or even a real-world anonymisation service. This will allow us to rule out any bias caused by the data source chosen for this preliminary study. Within this future story we will also be able to examine the efficacy of intersection attacks, i. e., under which circumstances they reduce the size of the anonymity groups over time.

Furthermore, we plan to evaluate what timestamp accuracy can be achieved in a practical environment in order to quantify the actual size of the anonymity groups for the various schemes. Another promising field for future research activities is the design of more advanced data retention techniques, e. g., by introducing dedicated retention identifiers which preserve the privacy of users, while at the same time offering improved traceability.

References

[1] Stefan Berthold, Rainer Böhme, Stefan Köpsell. Data Retention and Anonymity Services. In: Proceedings of IFIP/FIDIS Summer School 2008, Brno, Czech Republic, http://www.buslab.org/SummerSchool2008/slides/Stefan_Koepsell.pdf.

[2] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In: INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, New York, USA, 1999.

[3] Richard Clayton. Anonymity and traceability in cyberspace. Technical Report, based on dissertation submitted August 2005. University of Cambridge, 2005.

[4] European Parliament & Council. Directive on the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks (Directive 2006/24/EC), March 15, 2006.

[5] Steven Glassman. A Caching Relay for the World Wide Web. In First International Conference on the World-Wide Web, CERN, Geneva, Switzerland, May 1994.

[6] JonDonym. http://www.jondos.de/

[7] Dogan Kesdogan, Lexi Pimenidis, and Tobias Köllsch. Intersection Attacks on Web-Mixes: Bringing the Theory into Praxis. In: Proceedings of First Workshop on Quality of Protection, Milan, Italy, 2005, http://www.freehaven.net/anonbib/cache/KesdoganPK06.pdf.

[8] Stefan Köpsell, Rolf Wendolsky, and Hannes Federrath. Revocable Anonymity. In: Emerging Trends in Information and Communication Security. Lecture Notes in Computer Science, 3995. Springer, Berlin, pp. 206-220, 2006.

[9] The Tor Project. http://www.torproject.org/

[10] George Kingsley Zipf. Relative frequency as a determinant of phonetic change. Reprinted from the Harvard Studies in Claasical Philology, Volume XL, 1929.

Anonymity Techniques – Usability Tests of Major Anonymity Networks

Jens Schomburg
IT-Security Group
University of Siegen, Germany
jens.schomburg@student.uni-siegen.de

Abstract

The anonymity provided by different anonymity networks also depends on the amount of users participating in these networks. As one cannot expect that a typical user is advanced in installing, configuring and using anonymity software, it becomes clear that the usability of these networks is important. This paper evaluates the usability of the four major anonymity networks Tor, Jap/JonDo, I2P and Mixmaster/Quicksilver adopting the cognitive walk-through method. The results indicate that Jap/JonDo and Tor are easier to install, configure and use than I2P or Mixmaster/Quicksilver. It was also found out that Mixmaster can not be installed by a novice user.

1 Introduction

There is academic coverage on the major software implementations of anonymizing network techniques. Most of these works cover the technical issues linked to anonymizing networks, which constitutes an important aspect. Anonymizing networks can only offer a certain degree of anonymity if the amount of participating users is high enough[1]. The developer and provider should try to acquire as many users as possible. Additionally, one cannot assume that all users are computer experts or advanced users thus the importance of usability becomes clearer [11]. Novice users typically rate their security not by hard facts rather their subjective feelings.
Because there are a lot of software implementations of anonymous communication techniques, this work focuses on the most popular ones [10] which are the ones with the highest amount of users. Consequently, Mixmaster (Email Messaging and Usenet) and a few low latency networks (Tor, I2P, JonDo) were examined. At this point it has to be mentioned that it was impossible to install and configure a pure Mixmaster implementation. An alternative Mixmaster client had to be chosen (Quicksilver). This will

[1] In general it can be said the more users those networks have, the bigger is the offered anonymity.

Fourth Privacy Enhancing Technologies Convention (PET-CON 2009.1), pp. 49–58, 2009

be explained in detail in section 3.4. Typical application areas of the Internet are web-browsing, sending e-mails and file sharing. This research is limited to the use cases web-browsing and e-mailing because additional examination on file sharing is out of scope.
As an unconditional prerequisite to using the software it is necessary to install and configure it first [2]. At this point many novice users fail and loose interest in something if this task is not easy to achieve. Usability, thus is an important aspect that should not be neglected and will therefore be focused on in this paper. The tests are done with Windows Vista Home Premium as this is the operating systems of many end-users.

2 Related works

As mentioned in the introduction, no academic paper compared the four major anonymity network implementations considering usability. In detail this is the installation, configuration and usage of the software. There are several overviews [10] and [8], characteristic comparisons (technical ones regarding the anonymity provided) but just one paper about usability which focuses on different configurations of Tor. Underlying the current paper are the following works:
In his PhD theses [10], chapter three Lexi Pimenidis gives an overview over anonymizing network techniques. He also separates the deployed anonymity networks in major and minor ones, based on the amount of users. This classification was also employed here.
Jeremy Clark, P. C. van Oorschot, and Carlisle Adams examine the usability and deployability of different Tor interfaces [7]. They developed several guidelines which are employed in this paper as well (in section 3).
Claudia Diaz, Len Sassaman, and Evelyne Dewitte evaluate the anonymity provided by two mix implementations [9], namely Mixmaster and Reliable. When discussing the provided anonymity by Mixmaster their findings will be drawn on.
George Danezis and Claudia Diaz give an overview of anonymous communication channels in this paper [8]. When comparing security benefits, their research will be alluded to.

3 Installation and configuration test

In the current evaluation of usability of the different anonymisation services the same evaluation methodology as in [7] is chosen. They used a cognitive walk-through. The core tasks designated in the latter paper are also reused:

- **CT-1** Successfully install the anonymisation software and the components in question
- **CT-2** Successfully configure the browser (E-mail client in Mixmaster/Quicksilver case) to work with the anonymisation software

[2] There applications available that do not have to be installed, but they are omited here.

- **CT-3** Confirm that the web-traffic/E-mail is being anonymised
- **CT-4** Successfully disable the anonymisation software and return to a direct connection

The set of guidelines that are also used:

- **G1** Users should be aware of the steps they have to perform to complete a core task.
- **G2** Users should be able to determine how to perform these steps.
- **G3** Users should know when they have successfully completed a core task.
- **G4** Users should be able to recognize, diagnose, and recover from non-critical errors.
- **G5** Users should not make dangerous errors from which they cannot recover.
- **G6** Users should be comfortable with the terminology used in any interface dialogues or documentation.
- **G7** Users should be sufficiently comfortable with the interface to continue using it.
- **G8** Users should be aware of the application's status at all times.

For a detailed explanation of the guidelines see [7].

3.1 Tor

The Tor website [1] as a starting point to achieve anonymity in the internet (completing CT-1 – CT-3) is clearly layouted. The language of the site is held simple and natural so it fits to G6.
The user can go on in order to complete CT-1 by clicking on "Download Tor" in the "Summary" navigation on the right side. This is G1 and G2 compliant. Afterwards one can choose between two Windows installation bundles and one for OS X. There is no hint as to an installation manual as it is on the download page to which one get by choosing the "Download" link in the navigation bar on top. This is inconsistent and violates G1 and G2.
It is also not clear why the downloaded file is named Vidalia-bundle. This partly conflicts with G2. Users might expect a file named "Tor" for example. The circumstance that also Vidalia (a GUI for Tor), Privoxy (a filtering web proxy) and Torbutton (a Firefox extension) have to be installed additionally to Tor, is explained in the manual and in the first installation dialogue. In the next dialogue is a description box where the component description text is shown if the user moves the mouse over the component. This factor advanced since the research done in [7] and fulfils G6. The installation is straight forward and supports the user ideally to achieve CT-1. This is concordant

with G1 and G2. An installation progress bar, which is used to show the progress of unpacking the program files, is used. This supports G8. As the bar reaches 100% the Firefox standard dialogue for installing extensions pops up. It recommends to install Add-Ons from trusted sources. After confirming to install this extension, the installation of the Vidalia Bundle is completed. This confirmation screen signals the user that CT-1 is completed (G3).
Afterwards the Vidalia control panel window opens and builds up a connection to the Tor network. It takes about two minutes until a connection is established. Meanwhile the user is not aware of the application status. This violates G8. In Firefox there is a cue added to the status bar that states "Tor deactivated". By clicking the cue switches to "Tor activated" and it is indicated to the user that her traffic is being anonymised (G8). The recommended settings were chosen and it worked immediately, so CT-2 is also completed.
The user is not automatically led to a test website where she can see if CT-3 is fulfilled but there is a check website at the tor-project[3].

3.2 I2P

The Website [2] is clearly arranged and welcomes the visitor with an introduction about what I2P is, which applications are supported except for web-browsing. This is contradictory to G1 and G2. The language is too technical for a novice user (G6). This holds for the picture as well which is meant to explain the function of I2P. In order to complete CT-1 it can be assumed that a user opens the download website where she is confronted with three different downloadable versions: graphical installer, headless install and source install. The descriptions given for each version might bring novice users to download the graphical version. This is consistent with G2. G6 and G2 are violated as the precondition for installation (Sun Java 1.5 or higher, or equivalent JRE) is stated without a link or explanation. It remains to mention that G6 is disregarded several times.
After downloading the graphical installer file and executing it, the user gets to the installation directory-selection dialogue. Confirming a folder as an installation path the software is being installed and a progress bar is shown. By completing this task the user sees a dialogue which signals him that the installation is finished. This installation procedure is straight forward and complies with G1, G2, G3 and G6; CT-1 is reached. When selecting default options, three icons are installed on the desktop: "Start I2P (no window)", "Start I2P (restartable)" and "I2P router console". In order to complete CT-2 and regarding the manual, the instruction to the user is that she should simply click on the "Run I2P" button which will bring up the router console with further instructions. Because there is no button or shortcut named "Run I2P" (see above which shortcuts have been created) the user does not know how to proceed (G1, G2). It also cannot be assumed that a novice user knows what a router console is, so G6 is disregarded. Because of the router console mentioned in the instruction, a user might execute the "I2P router console" shortcut. On the test-system the Firefox Browser is opened with the

[3] `http://check.torproject.org/`

URL `http://localhost:7657/index.jsp` which displays a connection fail error (G4).
Because no further documentation is available, the "Start I2P (no window)" shortcut is chosen and, as a result, the Internet Explorer (and not the default browser Firefox) opens with the URL `http://localhost:7657/index.jsp`. This page states: "Congratulations on getting I2P installed", fulfilling G3.
The opened page is very complex. On the left side there are several parameters displayed, such as Peers, Bandwidth, Tunnels and Congestion. On the top there is a navigation bar where links to Susimail, SusiDNS, I2PSnark, etc. . . are placed. The use of technical terms and applications the user does not know or even might not be interested in violates G6. G7 is also disregarded because there are so many options available that a user cannot use it comfortably.
The content area welcomes the user with further instructions on how to proceed. This fits G3 and CT-1 is completed. The instructions may fulfil G1 and G2 in order to perform CT-2 but the whole presentation of that page is too complex and uses too much technical language (G6). The instructions in detail are adjust bandwidth settings and open up port 8887 on the firewall and further enable inbound TCP on the linked configuration page. This is clearly not addressed to novice users (G6) and may result in errors (G5). After the tasks are completed the user gets no feedback if she has reached CT-2 (G3).
In contrast to the download website there are more configuration hints on the local page presented when starting I2P. These are separately from those mentioned above. The effect is, that it looks confusing and does not support the user in achieving CT-2, so G1 and G2 are disobeyed. Amongst other hints one finds "browse the web" which states that one should "simply" set the browsers proxy to use an HTTP "outproxy" of I2P. This description is not suited for a novice user because it uses technical language (G6) and the user might not know how to complete this task (G2). The same circumstance was examined for Tor in [7].
Because there is no application offered to check whether the traffic is being anonymised (CT-3) a website has to be chosen which displays the actual IP-address.
CT-4 can be performed by clicking a "shutdown" link in the user interface. But as this just turns off I2P, the user additionally has to reverse the configuration in the Browser, too. Due to the fact that the initial configuration step violated G2 and G6, it is clear that the reverse action does the same.

3.3 Jap/JonDo

A download button to obtain JonDo is placed clearly visible on the left-hand side, so the user is aware of the next steps she has to perform (G1 and G2). With clicking on it, one can choose between JonDo versions for Windows, Linux and MacOS X. By choosing the Windows version, the user is taken to another page where she can choose between the JonDo desktop installation or the portable version. The downloadable file is named `japsetup.exe`. The installation starts with a dialogue where the components to be installed can be chosen. Namely, these are Jap, Swing and Java 1.3; Jap is preselected. The fact that the name Jap instead of JonDo is presented five times in this dialogue could irritate a user and is thus conflicting with G6 because the user

does not necessarily know that Jap is a different name for the same application. Indirectly it might also violate G2. It would be more comprehensible if one name would be used continuously. After installing the application, a wizard starts up and prompts to configure Jap/JonDo in every browser. A list of different browsers is given. It is explained how to use the Jap/JonDo proxy settings for each of these browsers. This is done in a non-technical language (G6) which meets G2 because it is a straight forward description of the single steps. The next step is to check if a warning is displayed when Jap/JonDo is switched off and tried to open a website. The expected warning comes and the user can proceed with the next step of the configurating.
The next dialogue in the wizard is a manual to deactivate Java, JavaScript, ActiveX, Flash and others in several browsers. Subsequently, a dialogue with the option to run JonDo in either a simple or extended view is presented. After that, a link to the JonDo FAQ and ultimately a confirmation screen that Jap/JonDo is successfully configured is shown. From this last dialogue the user knows that CT-2 is achieved (meeting G3).
The step-by-step wizard is a good way to prevent users from making errors in configuring JonDo(G4 and G5) and is deemed understandable for novice users (G6). It also can be restarted from the JonDo Application and thereby follows G4.
On the website an anonymity test is available [4] which shows diverse information transmitted by the visiting system (G3 for CT-3).
To achieve CT-4 the user can click the "anonymity off" switch. A message that JonDo does not support any protection further on is displayed. This procedure fits to G1, G2 and G3.

3.4 Mixmaster / Quicksilver

This research yielded that it is impossible for a novice user to install Mixmaster correctly. There is no manual available and the installer files have to be compiled by the user. Because this is seen as unacceptable for novice users, the research on this implementation is aborted at this point. Alternatively, the client implementation Quicksilver [6] is chosen for further tests. The website comes in a very simple style without graphics etc. It is written in non technical language (G6). There is also a download link, together with the hint to read a welcome.txt for more information, which meets G2.
When the downloaded file is executed the user gets to a screen where she is requested to enter an e-mail address and an SMTP Host. Additionally there is a text saying that an actual email address and mail server from which the User normally sends messages is needed in order to create a default message and a USENET article. Although there are examples and a text given this is not comprehensible for a User and thus violates G6. On the basis of the given examples the user cannot determine how to complete this step (G2)
The last dialogue before completing the wizard-style installation is an overview of the configuration options chosen by the user. An affirmation that the installation is complete is shown and the user knows that she completed CT-1 and G3 is fulfilled.
When starting Quicksilver a prompt comes up and informs the user that Mixmaster

[4] `https://www.jondos.de/de/anontest`

is not installed but needed to send encrypted messages. By using a non technical language (G6) and providing a button "Get Mixmaster" the User can comprehend what she has to do next (G2). By choosing the button "Get Mixmaster" the "QuickSilver Update Express" is started. It asks the user to pick an FTP site from a drop-down list. It can be expected that a novice user does not know what ftp and a proxy is, thus it marks a violation against G6. By selecting the default site the program retrieves a list of available updates, including Mixmaster (Mix29b39.zip). For a novice user it might be unknowable which file to choose and what to do so G2 and G6 are violated.
The file is downloaded and afterwards the Mixmaster setup can be run out of the QuickSilver update wizard. The Mixmaster setup is a wizard too. G6 is violated by using technical terms like "computer's path statement" and mentioning "mix.cfg", "mixlib.dll" and "libeay32.dll" without explanation. In the next dialogue the user is requested to write her name with the mouse, for the purpose to gather some random data to initialize the Mix random pool. When proceeding, the Mixmaster setup is completed which is then shown by a confirmation screen (G3).
The QuickSilver interface opens and it seems to be ready to use. Because the interface is new to a novice user and the function is not evident, G7 is violated. In the help-system there is a Quickstart section which is very long and therefore does not deserve the name "Quick". The chapter "I-8 Anonymous Messages" describes how to use QuickSilver in order to send anonymous messages. It explains the interface "New Message". The dialogue "New Message" is basically a text field with predefined values and the user can edit or add some parameters on her own. This design is not supportive to achieve CT-2[5] because it is not as intuitive to handle as usual and does not prevent the user from making erroneous inputs (G7 and G5)
After composing a test message and clicking the "Send" button a dialogue appears that says there are Mixmaster Remailer Documents missing. So CT-2 can be seen as unfulfilled which was unnoticeable for the user (G1). There is a "Get documents" button presented to lead the user through this configuration step (G2). By choosing this button a dialogue with several possible options is presented. Here a user can specify URLs to find certain files. It is linked to a help topic for a brief explanation which files the user needs so it meets G2. After reading the Help, executing the instructions (check mlist.txt and rlist.txt) and clicking update, the application fetches new remailers and keys. This whole procedure is complicated and intransparent for a novice user who has not read the complete manual.
Afterwards, it is tried again to send a message, but the process aborts with the error message "No reliable remailers!". After searching the internet for new sources of remailer stats and keys, they are found. This practice is also not practicable for a novice user and violates G2 and G6. The whole configuration process is definitely too complex for a novice user as it violates G1, G2 and G6 several times. The interface cannot be handled very comfortably which is also a drawback. It is dispensable to test whether CT-4 is achieved because there is no need to disable Quicksilver.

[5]CT-2 in this context means that the user should be able to configure the application the way that anonymous e-mail can be sent.

3.5 Summary of the installation and configuration test

The amount of guideline violations on every Core Task during the cognitive walk-through are summed up and displayed in table 1. It is visible that all of the tested implementations have flaws at the installation (CT-1). While there are less flaws at Tor and JonDo, I2P and Quicksilver have more and thus are more difficult to install for novice users. The configuration (CT-2) of I2P and Quicksilver is very difficult and as a result the chance that novice loose interest at that point is seen as probable. In contrast Tor and JonDo are very easy to configure. Core task three and four are achievable for novice users with the three low latency network implementations as there are no (Tor, JonDo) or just a few guideline violations (I2P) found. The latter mentioned core tasks are not applicable on Quicksilver.

Table 1: guideline violations while trying to achieve a core task

Anonymity network	CT-1	CT-2	CT-3	CT-4
Tor	3	2	0	0
I2P	5	9	1	2
Jap/JonDo	3	0	0	0
Mixmaster	nr	nr	nr	nr
Quicksilver	6	9	nm	nm

nr = core task not reachable; nm = metric not measurable

4 Comparison regarding subjective security effect

The following section consist of a subjective opinion of the security provided by the tested implementations. The subjectivity aspect is crucial as there is no unified measurement of anonymity yet and users typically rate their security by subjective feelings instead of consulting hard facts. A certain faith in software is necessary because it is nearly impossible for most users to evaluate the entire functionality of the product.
At first, the different websites are considered because they are the first contact point for many users of an anonymity networks. The Tor project web presentation [1] inspires confidence, as it is clearly structured and written in a non-technical language. On the one hand the first good impression may be reduced by reading the part about possible attacks on the Tor network but on the other hand it may increase the trust in the developers for being honest. The same applies for the I2P website [2]. On the Jap/JonDo homepage [3] instead, the risks are not described that clearly. It is rated negatively that many technical terms are used on the I2P website. Also, some of the graphics do not make a professional impression[6].
The websites of Mixmaster and Quicksilver are designed in a sparse look as it is found often in the open source community. This design is not practical to gain novice users.

[6] They are presented in a comic like look.

The next facet that shall be discussed are the persons who develop the software or run the service. The Tor, Mixmaster and Jap/JonDo developers are known or can at least be linked to an identity unlike the I2P developers. The revelation of the developers names is seen as a benefit because it supports trust in a project. A special case is JonDo as there is a company (jondos) involved in selling access to the premium services of the network and paying Mix operators for relaying traffic.
The applications themselves also differ in several ways: The Tor-Bundle is easy to install and configure with a few trivia. It gives the user a feeling of control because the status of the application is visible at most times. This is enhanced by the Jap/JonDo client since more information is presented (amount of Users, "Anonymity-meter", activity). In contrast to Tor and Jap/JonDo stands I2P because the user is overstrained with all the possible configuration options. This reduces the trust in the application because a user cannot handle and control it intuitively. The Quicksilver application has deficits here as well even though the control aspect is different because it just handles e-mails instead of streams or packets. It is seen critically that one has to entrust the account of the usual mail-server to the application even though one could open up a free-mail account just for this purpose.
Another important fact when it comes to trust in software is the amount of academic research work. It supports the user in trusting an anonymity software if she knows that, through a survey flaws might be identified and, in consequence, be corrected. This, again is a deficit of I2P as there is not much academic coverage yet.

5 Conclusion

The remaining question is that how much effort one can expect from the user to gain knowledge about anonymisation and the examined implementations of the different anonymity networks. It is more difficult to install and use I2P and Mixmaster/Quicksilver than Tor or JonDo. The latter require a shorter period of vocational adjustment. On every of the examined implementations, flaws can be fixed so that they are better to handle for novice users.

References

[1] Tor 0.2.0.31. `http://www.torproject.org`.

[2] I2p 0.6.4. `http://www.i2p2.de`.

[3] Jondos. `http://www.jondos.de`.

[4] An.on. `http://anon.inf.tu-dresden.de`.

[5] Mixmaster. `http://mixmaster.sourceforge.net/`.

[6] Quicksilver. `http://www.quicksilvermail.net/`.

[7] Jeremy Clark, P. C. van Oorschot, and Carlisle Adams. Usability of anonymous web browsing: an examination of Tor interfaces and deployability. In *Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS '07)*, pages 41–51, New York, NY, USA, July 2007. ACM.

[8] George Danezis and Claudia Diaz. A survey of anonymous communication channels. Technical Report MSR-TR-2008-35, Microsoft Research, January 2008.

[9] Claudia Diaz, Len Sassaman, and Evelyne Dewitte. Comparison between two practical mix designs. In *Proceedings of ESORICS 2004*, LNCS, France, September 2004.

[10] Lexi Pimenidis. *Holistic Confidentiality in Open Networks*. PhD thesis, University RWTH Aachen, 2009.

[11] Alma Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *8th USENIX Security Symposium*, 1999.

Peer Profiling and Selection in the I2P Anonymous Network

zzz, Lars Schimmer

The I2P Project

http://www.i2p2.de/

zzz@i2pmail.org

echelon@i2pmail.org

Abstract

I2P is an anonymous communication network that routes data through tunnels. It selects subsets of peers for tunnels using peer profiling mechanisms, including opportunistic measurement of per-tunnel bandwidth and the number of accepted tunnels. The profiling and selection system was implemented in early 2004 and its effectiveness has been conclusively demonstrated. It optimizes bandwidth while maintaining the anonymity of its clients. In this paper we document I2P's peer profiling and selection algorithms, discuss their strengths and weaknesses and describe directions for future work. We hope that it will be useful to guide improvements in I2P and other anonymous networks with similar features.

1 I2P Overview

I2P is an anonymous overlay network based on unidirectional encrypted tunnels. The project evolved from the Invisible Internet Project (IIP), which was a specialized IRC server transmitting data through a mix network. The project was formed to support the efforts of those trying to build a more free society by offering them an uncensorable, anonymous, and secure communication system able to operate successfully in arbitrarily hostile environments [Jr03].

Starting from a low-latency transport protocol for The Freenet Project, it grew into an independent project. I2P is primarily public domain software, but also includes code under GPL, BSD, MIT, and Apache licenses. The project is undergoing rapid development, with seven releases in 2008. The I2P software suite consists of the core router and several software packages (web server, BitTorrent clients etc.).

The routing method is called *garlic routing*. It is similar to the onion routing used by the Tor project [Tor] in that it transmits data through multiple peers, thus concealing the true IP address of the sender from the recipient and vice versa. Garlic routing allows multiple messages or *cloves* inside a single container. I2P employs a VPN-like approach that is designed primarily for communication within the network (hidden services in Tor terms). In I2P, only a few *outproxies* (exit nodes in Tor terms) operate as gateways to the standard Internet.

Fourth Privacy Enhancing Technologies Convention (PET-CON 2009.1), pp. 59–70, 2009

At the time of writing the I2P network consists of approximately 830 active peers. This number varies over time, as 50-100 new nodes join the network every day and others leave. Usually on weekends the network peaks at 1000 active nodes and on weekdays the minimum is about 700 active routers. The network has a stable base of approximately 550 active clients or *destinations*. Only one HTTP outproxy is publicly advertised and accessible (it is possible there may be private outproxies as well). There is also an IRC *inproxy* (gateway from the standard Internet) for I2P development discussion, and an email gateway. The current bandwidth used by all peers is roughly 11 MByte/sec with peaks at 16 MByte/sec and the peers build ~40,000 tunnels to route this traffic. In the last 6 months the network doubled in terms of running routers, bandwidth and tunnels. We project continued stable growth for the network.

1.1 Tunnel Overview

I2P transmits its payload through tunnels. A tunnel is a unidirectional encrypted connection through zero or more peers. Every router and destination has some incoming and outgoing tunnels. Tunnels are created for each router and destination upon initialization. Message transport may use one or more tunnels, and an application-level TCP connection is not correlated with a particular tunnel. We distinguish between *exploratory* and *client* tunnels.

1.2 Exploratory tunnels

Exploratory tunnels are generally low bandwidth, and are used by the router itself. These tunnels are used to test other tunnels, send network database (*netDb*) queries, and build client tunnels. The paragraph **Peer Selection** describes the process of tunnel building in more detail.

1.3 Client tunnels

Client tunnels transmit the payload for protocols like HTTP, IRC and others through I2P. These tunnels can be high bandwidth and reach up to 200 KByte/sec.

Each service in I2P has a *destination* as a representation of the location to reach the service. Examples are servers like eepsites (the I2P internal webpages), monotone server, IRC server or audio streaming servers, and the clients have a destination as a return address, e.g. IRC clients, bittorrent clients, monotone clients or else. Servers have stable destinations while clients destination are created new on every router restart.

Each destination has an associated set of tunnels to transport the data. The application and/or user can control the specifications of the tunnels:

- Length: The number of peers or *hops* in the tunnel; typically 2 or 3
- Length Variance: A range to be randomly added to the tunnel length at creation; typically [0-1]

- Pool Quantity: The number of tunnels in the *tunnel pool*; messages are distributed across tunnels in the pool
- Backup Tunnels: Additional tunnels may be specified in case the original tunnels die unexpectedly

The route from a client to server is divided into two tunnels: the *outgoing tunnel*, controlled by the client, and the *incoming tunnel*, controlled by the server. The connection between these two tunnels is direct; the *outbound endpoint* of the outgoing tunnel connects directly to the *inbound gateway* of the incoming tunnel. Tunnels provide anonymity by separating the destination's inbound gateway and outbound endpoint from the destination's router. By setting the length of his tunnels, every participant may control his anonymity level. Longer tunnels provide more anonymity at the cost of lower performance and reliability.

Each router selects the peers with which it builds the tunnel based on the profile of its capabilities described below. It also determines the order of peers within a tunnel, using the XOR distance of a peer's router ID from a random key, to inhibit predecessor attacks [Wr04, Wr08]. Also, two or more I2P peers within the same /16 subnet cannot be used within the same tunnel, to frustrate simple forms of collusion.

Each tunnel in I2P has a fixed lifetime of 10 minutes and will be discarded after this time. This enhances anonymity by having all tunnels look the same, to reduce the chance of correlation with a router or destination. A new tunnel will be created and added to the pool before the existing one times out. Data will be sent on one or more of the tunnels of the pool associated with the specific destination.

Peers can reject or drop tunnel build requests sent by other peers for a number of reasons (overload, shutdown in progress, limit reached, out of sync). Even within the lifetime of a tunnel these can be discarded due to overload or unexpected shutdown of one of the peers in the tunnel. To detect tunnel failure, each tunnel is tested on a periodic basis. The test sends a single 1 Kbyte message per minute, which adds a small overhead to the router traffic that is inconsequential for all but the lowest-bandwidth routers. Test messages are designed to verify tunnel connectivity and not to test tunnel bandwidth.

The tunnels on a peer are sorted into different classes:

- Exploratory tunnels: Tunnels used by the router itself
- Client tunnels: the tunnel starts or ends in this peer, bound to a server or client on the peer
- Participating tunnels: this peer is a member of one of the tunnels and the participating tunnels come in from a peer and continue through to another peer.

Three cases are possible: outbound endpoint of an outgoing tunnel, entrance or "inbound gateway" of an incoming tunnel, and an inner peer of a participating tunnel.

1.4 Network Database Overview

The NetDB is a collection of information stored about each peer and destination. It contains a *RouterInfo* structure for each peer and a *LeaseSet* (similar to the hidden service descriptor in Tor) for each known destination. LeaseSets will not be described here, for more information on this topic see the I2P website [I2P].

The RouterInfo is a small collection of all vital information describing a peer. The IP address, port, peer ID, I2P stable version number, network version, transport capabilities and some statistical data are included. The statistical data are for network diagnostics and are not trusted by routers. Each peer is set by default to 48KBps input and 24KBps output bandwidth with 90% share percentage (90% of maximum bandwidth is allowed to be used up by participating tunnels).

Each peer is sorted into a bandwidth class based on the user-configured shared bandwidth: these classes are named K, L, M, N and O. Limits are <=12 KBps for K, to >=128 KBps for O. This classification is just for network diagnostics, except that peers of class K are not used at all for routing participating tunnels. Statistics show only a small number of K routers (96 % are class L-O), so almost all peers are configured to route tunnels for others.

1.5 Tunnel Building

Each router in I2P selects a subset of peers for its tunnels. Ideally, the routers should select the fastest peers available. A simple implementation would be to allocate tunnels in proportion to the self-reported bandwidth values for each peer. This allows a simple low-resource attack where malicious nodes can report a high bandwidth so that a larger fraction of tunnels are routed through them [SB08]. As such an attack can easily attract a large number of tunnels and thus compromise anonymity [Ba07], I2P implements *peer profiling*.

2 Peer Profiling and Tiers

Peer profiling was proposed for the I2P network in December 2003 by jrandom [Jr03a]. It was introduced in the 0.3 release in March 2004 [Jr04].

Peer selection within I2P is the process of selecting the path, or sequence of other routers, for locally generated messages and their replies. This path is an ordered set of peers in a tunnel. The router maintains a database of each peer's performance, called the *profile*. The router uses that data to estimate the bandwidth of a peer, how often a peer will accept tunnel build requests, and whether a peer seems to be overloaded or otherwise unable to reliably perform. The profiling system includes mechanisms similar to the "opportunistic bandwidth measurement algorithm" proposed for Tor [SB08]. It does not require "active bandwidth probing" or generation of special-purpose traffic for measurement. Direct measurement, such as transport layer latency or congestion, is not used as part of the profile as it can be

manipulated and associated with the measuring router, exposing the router to trivial attacks.

The profile contains several statistics and is continuously updated. For each statistic, averages, event counts, and maximums for several periods (for example 1 minute, 10 minute, 1 hour, and 24 hour) are available. Example statistics are: how long it takes for the peer to reply to a network database query, how often a tunnel through the peer fails, and how many new router references the peer sends. The profiles are also stored persistently on disk, so the statistics are not reset at router initialization. Also, the profile stores several timestamps, including the last time a peer was heard from, the last time it accepted a tunnel request, and the last communication error.

Much of the profile data is unused in the current software. It remains a "reserve" defense that can be easily used to enhance the router's resistance to theoretical or actual attacks, such as denial-of-service attempts, selective packet drops, network database (floodfill) disruption, and others. The profiles are a router's unique assessment of each peer's capabilities, and are not distributed to other peers or published, for this would easily compromise anonymity.

The profiles are periodically coalesced and sorted into tiers of peers that are used for various functions, as described further below.

2.1 Speed

The speed calculation simply estimates how much data can be sent or received on a single tunnel through the peer in a minute based on past performance (this may also be termed bandwidth or capacity, but we use the I2P terminology here). Specifically, it is the average of the bandwidth of the fastest three tunnels, measured over a one-minute period, through that peer. Previous algorithms used a longer measurement time and weighed recent data more heavily. Another previous calculation used total (rather than per-tunnel) bandwidth, but it was decided that this method overrated slow, high-capacity peers (that is, slow in bandwidth but high-capacity in number of tunnels). Even earlier methods were much more complex. The current speed calculation has remained unchanged since early 2006.

Only a router's locally generated and received traffic is used for these measurements - transit or "participating" traffic is not used. As it is not known if the peer before or after the router in a tunnel is a true participant or the originator or destination, that data would not be valid. Also, this could be exploited to get a router to rank some peer of their choosing as quite fast. Having a remote peer influence the rankings in this way could be dangerous to anonymity.

2.2 Capacity

An estimate of peer tunnel capacity, defined as the *number of successful tunnel builds* through the peer in a time period, is crucial to the smooth operation of an I2P router. As tunnel building is expensive, it is important to rate peers based on their willingness to accept tunnels. Peers may reject or drop tunnel requests for any

number of reasons. Generally the rejection or drop is caused by a bandwidth limit (from participating tunnels or locally generated traffic), a processing (CPU) limit which manifests itself as large request processing delay, or an absolute limit on participating tunnel count.

The capacity calculation simply estimates how many tunnels the peer would agree to participate in over the next hour. The actual capacity rating, before adjustments (see below), is as follows: Let r(t) be the successful builds per hour, over a certain time period *t*:

$$R = 4*r(10m) + 3*r(30m) + 2*r(1h) + r(1d)$$

If there are no successful builds in a given time period, the value is zero. Build requests are never sent for the sole purpose of constructing the rating. The rating also includes a 'growth factor' that adds a small amount each time, so that builds through new peers are periodically attempted.

2.3 Manual Adjustments and Unused Statistics

For both the speed and capacity metrics, *bonuses* may be used to manually adjust preferences for individual peers.

There are several other statistics available in the profile that are not currently used for peer selection. These include latency for client messages, tunnel build request response time, communication error frequency, and network database lookup success rate and response time. The potential for improving peer selection based on these statistics is a topic for further research.

The router also maintains an "integration" metric reflecting the number of other peer references received from that peer. The integration metric is used to qualify a peer as a floodfill router (directory authority in Tor terms) but is not used for peer selection.

2.4 Capacity: Crime, Blame, and Punishment

Raw tunnel build capacity is not a sufficient measurement - it is essential to decrement measured capacity as a form of "punishment" for bad behavior, because tunnel build attempts are expensive, and malicious peers must be avoided. A tunnel build request is about 4KB, and the reply is identically sized. Generation of the build request also consumes significant CPU and entropy to create the cryptographic keys. As an example of the factors that must be considered:

- A peer that accepts 10 out of 10 requests is better than one that accepts 10 out of 100.
- A peer that explicitly rejects a request is better than one that drops it.
- A peer that accepts a request but later drops data through the tunnel should be avoided.

Ideally, capacity should be decremented for build request rejections, build request timeouts, and tunnel test failures. Unfortunately, a router does not know which of

the tunnel peers to blame when a request or tunnel test message is dropped. Tunnel builds requests are handed off from peer to peer along the path, and since tunnels are unidirectional, a tunnel cannot be tested in isolation. What should the router do with such incomplete information?

- Naive solution: Do not blame any peer.
- Better solution: Blame each peer with equal weight.
- Best solution: Blame each peer, but use a weighted value if there is partial information available on the probability of a particular peer's fault.

The naïve solution was used in I2P for many years. However in mid-2008, we implemented the weighted blame system, as it became apparent that recognizing and avoiding unreliable and unreachable peers is critically important.

As an example, assume a tunnel build request (4 outbound hops through peers A-B-C-D) has expired. The reply was due back through the inbound exploratory tunnel (2 hops E-F). The following options are among the possibilities:

1. Any of the peers could be at fault, so blame no one.
2. Blame each of the 6 peers equally with weight 1/6.
3. Weight each tunnel equally, and distribute the blame equally in each tunnel, so blame outbound peers A-D with weight 1/8, and blame inbound peers E-F with weight 1/4.
4. Knowing that the usual failure point in I2P is an unreachable inbound gateway[1] (E in this case), blame E with weight 1/2 and the other peers with weight 1/10.

I2P now uses option 3 for build request timeouts, and option 4 for tunnel test failures in most cases. The effectiveness of these changes has been demonstrated by significant improvement in tunnel build success rates and network bandwidths in the latter half of 2008. The system works because consistently "bad" peers are discovered and avoided fairly quickly, while peers that are falsely blamed are blamed with roughly equal frequency, which does not hurt their relative ranking.

Finally, we multiply the test failures by 4 to increase the punishment for agreeing to a tunnel but then dropping data. The current calculation for capacity, r(t) for each time *t*, is:

$$R(t) = \text{accepts} - \text{rejects} - \text{weighted timeouts} - 4*\text{weighted failures} + \text{growth factor}$$

[1] Consider a test of previously built outbound tunnel A-B-C-D and inbound tunnel E-F build by router X. Tunnel build requests follow the path of the tunnel itself, therefore the act of router X building these tunnels establishes and verifies the transport connections X-A, A-B, B-C, C-D, E-F, and F-X. In addition, due to the 10-minute tunnel lifetime, and transport idle timeouts that are generally longer than that, those connections will probably remain up for the lifetime of the tunnel. The only connection that is not necessarily established in advance is the connection between the outbound endpoint (D) and the inbound gateway (E). Therefore, if a tunnel test fails, it is usually due to configuration or firewall issues at the inbound gateway.

2.5 Sorting Profiles Into Tiers

To review, exploratory tunnels are generally low-bandwidth, and are used for router operations, including building and testing other tunnels. Client tunnels are used for all user client and server traffic, including accessing internal I2P network destinations or "hidden services" such as *eepsites*, connection to external gateways (*inproxies* and *outproxies*) and other uses.

Every 30 seconds, all the profiles are sorted into three tiers:

- The *Not Failing* tier contains all peers with whom communication was attempted in the last few hours, including the following two tiers. Typical size is 300-500 peers.
- The *High-Capacity* tier includes peers with above average rating for accepting tunnel build requests, and the following tier. Typical size is 10-30 peers.
- The *Fast* tier includes peers from the High-Capacity tier whose speed rating (i.e. peak bandwidth per tunnel) is above average for all peers in the Not Failing tier. Typical size is 8-15 peers.

Both the speed and capacity metrics are skewed, with a small number of high ratings and a "long tail". By using the average and not the median, we select a small number of peers for the top two tiers.

2.6 Peer Selection

Candidate peers for tunnel builds are selected as follows:

- Client tunnels are built from peers in the Fast tier.
- Exploratory tunnels are built from peers either in the Not Failing tier or the High Capacity tier.

For exploratory tunnels, the tier selected is chosen on a per-build basis, using a weighted random function. The proportion of builds using the High Capacity tier is

(client success rate – exploratory success rate) / client success rate

As exploratory build success declines, the router builds more tunnels from the high capacity tier, to limit the amount of effort spent on the expensive tunnel build request operation. Therefore the selection maintains a balance between minimizing tunnel build requests and the need to explore peers.

It may seem inadvisable to use the Not Failing tier (generally the lowest-bandwidth, lowest-capacity peers) for exploratory tunnels, since these tunnels are required to function for a router to build client tunnels. However, failing exploratory tunnels are recognized quickly, so this is not a significant limitation. Using all peers for exploratory tunnels provides I2P a system of opportunistic bandwidth and capacity measurement.

Peers are selected with equal weight within each tier. If a sufficient number of peers for a tunnel are not found within a given tier, the peer selection moves on to the next-lower tier.

3 Performance Evaluation and Further Work

As shown in available network statistics, the recent improvements to the I2P profiling and peer selection system have significantly improved bandwidth and other performance metrics [Stats]. The core profiling system, including the "opportunistic bandwidth measurement algorithm" proposed in [SB08], has been in place since early 2004.

I2P does not use claimed bandwidth, which eliminates a class of low-resource attacks. While we have not included here experimental data demonstrating that the selection system is effective, it is readily apparent to the authors that in a broad range of experimental conditions, the members of the Fast and High Capacity tiers are those peers with high claimed bandwidth (class O). In our opinion the current bandwidth constraints within I2P are not caused by poor peer selection, but lie elsewhere[2]. The basic peer selection method has been in place for five years, and has been tuned only modestly in the last two years.

The algorithm is stable; the Fast and High Capacity tier members do not change rapidly. When a router uses a peer for tunnels, it tends to increase that peer's speed and capacity metric, thus keeping that peer in the High Capacity tier. This is desirable for anonymity, as using a large or rapidly varying set of peers for tunnels would increase vulnerability to predecessor attacks by increasing the odds that an attacker will eventually be a participant in a tunnel [Wr04, Wr08].

The tier system reacts quickly to individual peer failure or overload, and to increased local demand for bandwidth. The speed and capacity metrics are strongly weighted for recent performance. When a peer starts to drop test messages, or fails to respond to tunnel build requests, it will quickly be demoted out of the high-capacity pool. As bandwidth demand increases, the speed metric for individual peers will rapidly increase, and the fast tier will quickly become reorganized to include the newly recognized fast peers.

The tier system tends to use the highest-bandwidth peers when the network is not congested. As congestion increases, the total network traffic "spreads out" to lower-capacity peers. From an overall network perspective, this is optimal as it maintains a similar level of service for all routers.

The profiling system does not over-optimize. The router uses its own, normally generated traffic for peer profiling. No high-bandwidth test messages are required or used. When a router does not require high bandwidth or a high number of tunnels,

---

[2] Lock contention, memory usage, issues in the internal TCP implementation (*streaming library*), network database inefficiencies, protocol overhead, message fragmentation, message dropping strategies, UDP transport issues, and others – some of which may be topics for future papers.

the metrics for each peer are correspondingly lower. Therefore, even a low-bandwidth peer may be classified as "fast" and used for tunnels. This tends to, at least partially, spread low-bandwidth tunnels broadly across the network, and leaves the high-capacity peers for high-speed traffic. However, even very-low-bandwidth routers tend to accurately find a few fast peers and thus are well prepared when higher bandwidth is demanded.

Also, there is no need for a complete global optimum. I2P routers know only a subset of the active peers in the network, generally 20% to 80%. Through exploratory tunnel building and other peer discovery mechanisms[3], routers have no difficulty finding a sufficient portion of peers, and preventing network partitioning. As the I2P network grows the percentage of peers known to each router will decline as we implement additional mechanisms to limit memory usage, TCP and UDP connections, and other resources in the router. This poses no threat to the profiling system.

The profiling system is persistent across restarts, and maintains measurements for the last 24 hours. This allows a recently started router to quickly re-integrate to the network, whether the router was just restarted or has been down for some time.

The network performance is sensitive to adjustments of the parameters, weighting, and calculations. It is difficult to test and debug in a distributed network, and may be impossible to fully optimize. The I2P router contains a framework for local network simulation and testing; however, we have not used this framework for profiling and selection testing. As described above, the routers include bandwidth and tunnel build success statistics in the network database entry they publish to the floodfill routers. While this information is not trusted or used in the router, it is gathered by the stats.i2p website [Stats]. On that website, several network performance graphs are presented, and the I2P developers rely on this facility to monitor the network and judge the effectiveness of software changes in each release.

The basic measurements have been greatly simplified in the process of development. The speed calculation, for example, was at one time over 400 lines of code, and it is now only a few lines.

The punishment for bad behavior keeps the network running well, but also is an area for further research. How heavily a router punishes determines how fast the load spreads out across the network as the load increases, and how quickly an overloaded peer is avoided. The implementation contains an implicit estimate of the cost of a tunnel build request, as it rates the relative performance of a rejected requests and a dropped request. It also weighs the costs of an accepted request vs. a request not made at all. One possibility is to establish a baseline of a peer that has never been asked to participate in a tunnel, then consider percentage (or absolute number) of

---

[3] A newly installed router downloads RouterInfo structures out-of-network through a process called *reseeding*. A router accepting tunnel build requests learns about the previous and next peers in the tunnel. A router acting as an outbound endpoint learns about the inbound gateway when it must route a message to that gateway, and vice versa. A router also periodically queries the floodfill peers for a random key, and the reply will contain routers close to that key, a process called *exploration*.

request rejections, dropped requests, and test failures is required to drop the capacity rating below the baseline.

If peers are punished too heavily, the network will tend to congestion collapse as most peers are driven to negative capacity ratings, tunnel load spreads quickly throughout the network, and routers attempt to route tunnels through very-low-capacity peers. If peers are punished too lightly, routers will be slow to react to overloaded peers, and maintain the same high capacity peers for too long by accepting poor performance from a peer even when better peers may be available.

We recommend that those wishing to implement a profiling and selection system start with relatively simple algorithms, and add complexity later if necessary. I2P's development has sometimes taken the reverse path; for example, I2P's speed calculation used to be several pages of code, now it is quite simple.

The evaluation of a distributed anonymous network's performance is difficult but not impossible. A more formal measurement of I2P's peer selection algorithms, either on the real network, an isolated test network, or simulation, would be a valuable extension to the analysis in this paper.

4 Conclusions

I2P routers accurately discover fast peers for tunnel routing without trusting claimed bandwidth or generating large amounts of traffic for testing. When a router requires little bandwidth, the precision of its peer selection is unimportant. When the router does require more bandwidth, the selection will be correspondingly better. Even very-low-bandwidth routers tend to accurately find fast peers and thus are well prepared when higher bandwidth is demanded.

To use the terms of [SB08], I2P's peer profiling and selection system is an opportunistic bandwidth measurement algorithm that is sensitive to network load and client demand. It does not use self-reported values. However it does not provide a "tunable" mechanism for users to trade off anonymity and performance. I2P provides alternate method (tunnel length configuration) for the user to make that adjustment. Not only is active bandwidth probing (i.e. generating large amounts of special-purpose data for testing is not practical, as [SB08] states, it is not necessary. In addition to the bandwidth measurements proposed in [SB08], I2P measures tunnel build acceptance rate, with adjustments for various bad behavior by peers. I2P's profiling and selection system has been in continuous use for approximately five years.

While the system works well, several improvements are possible. The authors will continue to research, evaluate, and tune I2P's peer profiling and selection system in the coming months.

References

[Ba07] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker: Low-resource routing attacks against anonymous systems, Proceedings of the 2007 Workshop on Privacy in the Electronic Society (WPES), 2007.

[I2P] This paper includes material from the I2P website http://www.i2p2.de/ by jrandom and others.

[IIP] http://invisibleip.sourceforge.net/iip/, http://en.wikipedia.org/wiki/Invisible_IRC_Project

[Jr03] jrandom: Invisible Internet Project (I2P) Project Overview, August 28, 2003 http://www.i2p2.de/_static/pdf/i2p_philosophy.pdf

[Jr03a] jrandom: I2P Development Meeting 68, December 9, 2003 http://www.i2p2.de/meeting68

[Jr04] jrandom: I2P Development Meeting 82, March 23, 2004 http://www.i2p2.de/meeting82

[SB08] Snader, R.; Borisov, N.: A Tune-up for Tor: Improving Security and Performance in the Tor Network, Proceedings of the Network and Distributed Security Symposium - NDSS '08, February 2008.

[Stats] http://stats.i2p/ or http://stats.i2p.to/

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

[Wr04] Wright, M.; Adler, M.; Levine, B.N.; Shields, C.: The Predecessor Attack: An Analysis of a Threat to Anonymous Communications. In ACM Transactions on Information and System Security (TISSEC) **4**(7), November 2004, pages 489-522.

[Wr08] Wright, M.; Adler, M.; Levine, B.N.; Shields, C.: Passive-Logging Attacks Against Anonymous Communications Systems.