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.

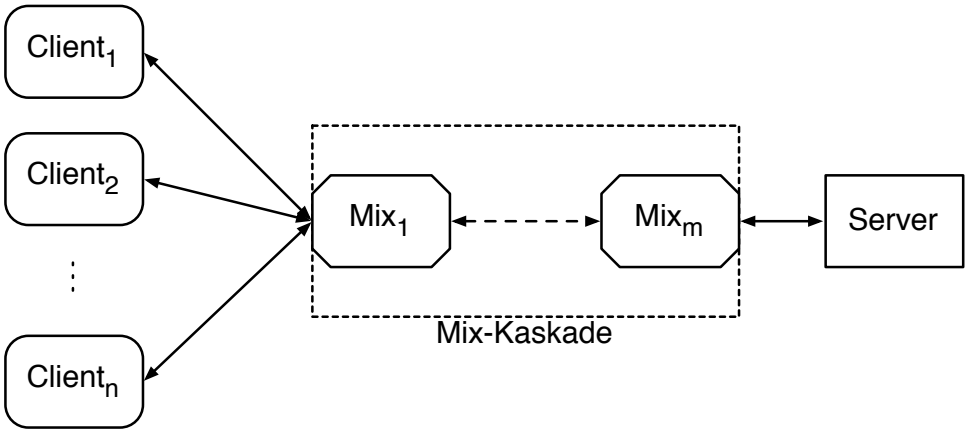

Figure 1: Die verteilten Komponenten des Chatsystems.

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

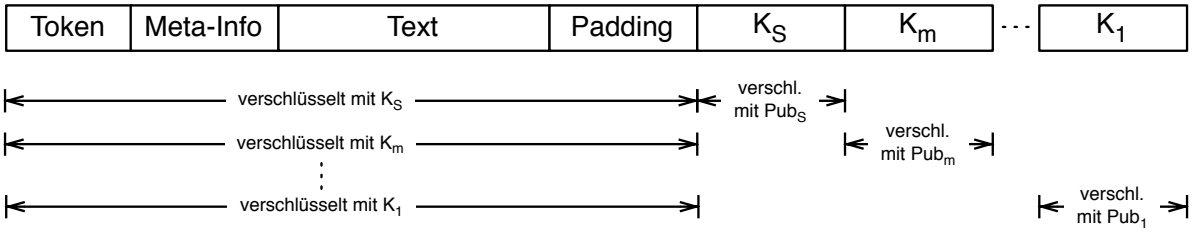

Figure 2: Aufbau eines Nachrichtenobjekts.

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.

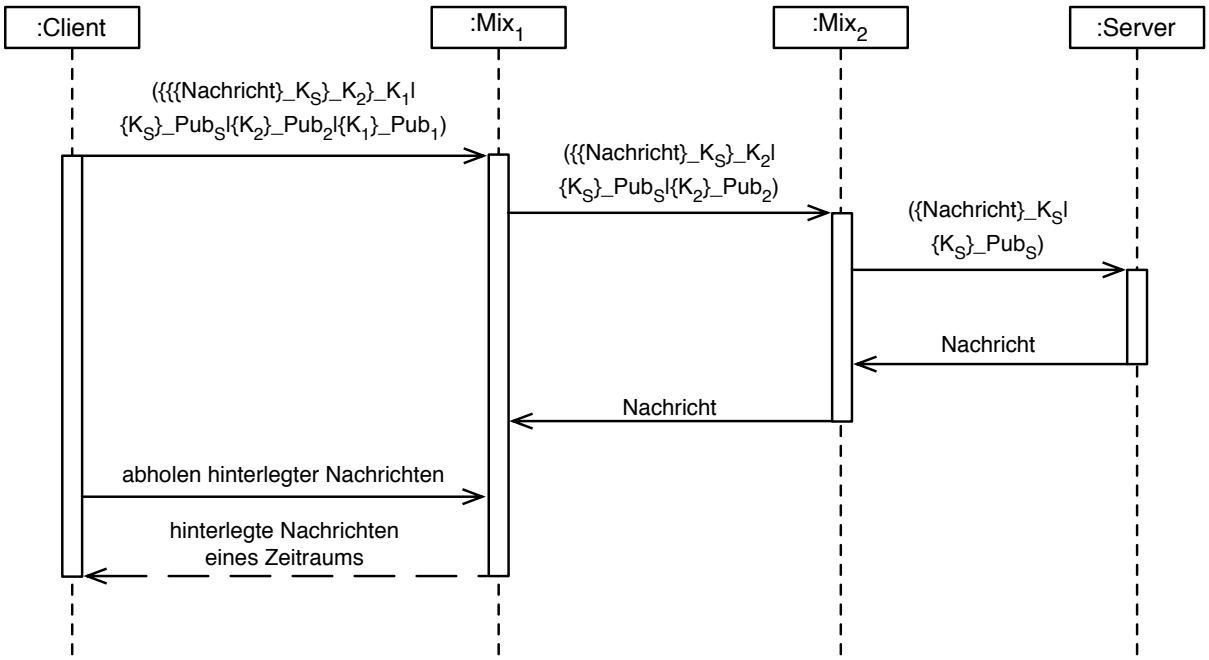

Figure 3: Sequenzdiagramm der Nachrichtenübertragung.

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.