Technische Universität Dresden
Medienzentrum

Prof. Dr. Thomas Köhler
Dr. Nina Kahnwald
(Hrsg.)

GeNeMe '12

Gemeinschaften in Neuen Medien

an der
Technischen Universität Dresden

mit Unterstützung der

BPS Bildungsportal Sachsen GmbH
Campus M21
Communardo Software GmbH
Dresden International University
Gesellschaft der Freunde und Förderer der TU Dresden e.V.
Hochschule für Telekom Leipzig
IBM Deutschland
itsax - pludoni GmbH
Kontext E GmbH
Medienzentrum, TU Dresden
Webdesign Meier
SAP AG, SAP Research
T-Systems Multimedia Solutions GmbH

am 04. und 05. Oktober 2012 in Dresden

www.geneme.de
info@geneme.de

B Konzepte, Technologien und Methoden für Virtuelle Gemeinschaften (VG) & Virtuelle Organisationen (VO)

B.1 Anonyme Kommunikation in verteilt organisierten Gitterstrukturen

Hauke Coltzau, Daniel Berg, Herwig Unger
Fernuniversität in Hagen, Lehrgebiet Kommunikationsnetze

Abstract

Es wird ein Kommunikationsschema vorgestellt, das auf Basis von regelmäßigen, dezentral organisierten Gitterstrukturen anonyme Kommunikation unter den Gitterknoten ermöglicht. Für den Empfänger einer Nachricht ist nicht ersichtlich, wer die Nachricht abgesetzt hat. Ein Angreifer, der die Kommunikation zwischen den Knoten beobachtet, kann zwar erkennen, dass Nachrichten zugesellt, vermittelt und empfangen werden, jedoch ist es ihm nicht möglich, festzustellen, welche Nachricht für welchen Teilnehmer vorgesehen ist.

1 Motivation

Das Internet bietet enorme Möglichkeiten, den Prozess der Meinungsbildung und des Austausches für Jedermann zugänglich zu machen. Dabei besteht sowohl für Privatpersonen als auch für Organisationen, die ihre Kommunikationswege mehr und mehr ins Internet verlagern, ein berechtigtes Bedürfnis nach Anonymität. Eine wesentliche Maßnahme zur Wahrung der Anonymität ist das Vermeiden zentraler Instanzen, Server also, über die Kommunikation und Prozesse abgewickelt werden, und die als Single-Point-Of-Failure leicht identifizierbar und kompromittierbar sind [17].

Um der stetig steigenden Last durch immer größeren Kommunikationsbedarf gerecht zu werden, greifen auch große Organisationen vermehrt auf Peer-To-Peer-basierte Paradigmen für ihre IT-Infrastrukturen zurück. Sie stellen in vielen Fällen effiziente Alternativen zu den klassischen Client-/Server-Modellen dar, bieten sie doch eine hohe Ausfallsicherheit und Möglichkeiten zur Lastverteilung [3, 6]. Darüber hinaus bieten gerade P2P-Netze eine gute Grundlage aber auch eine gesteigerte Notwendigkeit, anonyme Kommunikation zu ermöglichen, gerade weil es keine zentralen, kontrollierbaren Knotenpunkte gibt [7].

In diesem Artikel wird ein Kommunikationsmodell vorgestellt, das anonyme Kommunikation auf Basis von dezentral organisierten, regelmäßigen Gitter-Netzwerken ermöglicht. Im Rahmen dieses Beitrages wird unter Anonymität

verstanden, dass die Identität des Senders einer Nachricht für die Vermittler, den Empfänger und alle anderen Teilnehmer unerkannt bleibt. Auch der Empfänger einer Nachricht ist ausschliesslich dem Sender bekannt! Vollständige Anonymität erfordert weiterhin, dass bereitsgestelle Informationen nicht mit ihren Anbietern assoziierbar sind. Dies wird Teil weiterer Arbeiten sein. Zur Gewährleistung der Sender-Anonymität werden im Wesentlichen zwei Eigenschaften von Gitterstrukturen ausgenutzt: Sie bieten stets mehrere Wege von einer Quelle zu einem Ziel, so dass schon der Kommunikationsweg von Sender und Empfänger nicht vorhersagbar ist. Darüber hinaus lassen sich im Gitter Ziele *relativ* adressieren. Das heißt, dass eine Weiterleitungsinformation nur für denjenigen Knoten interessant ist, für den sie bestimmt ist. Für alle anderen wäre diese Information wertlos und ließe keinerlei Rückschlüsse auf Sender, Empfänger oder Vermittler einer Nachricht zu.
Zunächst werden in Abschnitt 3 einige verwandte Anonymisierungs- und Verschlüsselungstechniken vorgestellt. Dann werden in Abschnitt 4 kurz dezentrale Gitterstrukturen, auf denen das vorgestellte Kommunikationsschema operiert, vorgestellt. In Abschnitt 5 wird schließlich das Schema im Detail beschrieben. Es folgt in Abschnitt 6 eine kurze Zusammenfassung, bevor der Beitrag mit einem Ausblick auf zukünftige Arbeiten in Abschnitt 7 abschließt.

2 Verwandte Arbeiten

Proxies

Proxies dienen als Vermittlungsstellen, die Anfragen von Rechnern unter „eigenen Namen“ (d.h. mit der eigenen IP als Absender) stellvertretend weiterleiten. Spezielle Proxies, so genannte Anonymisierer, gehen dabei so vor, dass der weitergeleiteten Anfrage nicht anzusehen ist, dass sie von einem Proxy stammt. Die Kommunikation zwischen zu schützenden Rechnern und dem Anonymisierer wird verschlüsselt. Der wahre Initiator der Anfrage bleibt damit nach außen hin unbekannt. Solange der Proxy Server bzw. sein Betreiber nicht kompromittiert werden, ist ohne Weiteres nicht nachvollziehbar, wer konkret mit Hilfe des Proxies welche Information angefordert hat. Es kann aber zum Einen sehr wohl in Erfahrung gebracht werden, *welche* Information übertragen wurde. Zum Anderen ist der Informationsanbieter durch einen Proxy nicht geschützt.

Onion Routing und Mix Kaskaden

Um zumindest nicht vom Vertrauen eines einzigen Anbieters eines Anonymisierers abhängig zu sein, schaltet man mehrere Anonymisierer hintereinander und erhält damit eine so genannte Mix Kaskade. Ein Mix ist ein dedizierter Proxy, ein so genannter Onion Router. Bevor eine Nachricht über die Mix Kaskade geschickt wird, wird sie mehrfach verschlüsselt, und zwar so, dass jeder Mix nur gerade die Teilinformation aus der Nachricht extrahieren kann, die notwendig ist, um sie an den nächsten Mix zu schicken (siehe Abbildung 1).

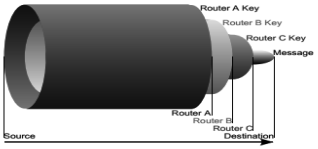

Abbildung 1: Der mehrfache Verschlüsselung einer Nachricht beim Onion Routing (aus de.wikipedia.org/wiki/Onion_Routing, Autor: Harison Neal)

Der letzte Mix der gewählten Mix Kaskade hat die Klartextnachricht und sendet sie an das eigentliche für sie bestimmte Ziel. Dieser kennt nun zwar die Nachricht, weiß aber nicht von wem sie ursprünglich stammt. Zwar stehen die verwendeten Mixes im Idealfall unter unterschiedlicher administrativer Kontrolle, sie sind jedoch nach wie vor für jeden Nutzer identifizierbar und damit theoretisch auch leicht kompromittierbar. Im Rahmen dieses Beitrages wird dieses Prinzip in ähnlicher Weise wieder aufgegriffen - mit dem wesentlichen Unterschied, dass es keine *dedizierten* Proxies mehr geben wird, sondern dass jeder Teilnehmer diese Aufgabe übernehmen kann.

Freenet

Das Ziel von Freenet [8] ist es, eine Infrastruktur zu schaffen, in der sowohl Konsumenten als auch Produzenten von Informationen anonym bleiben. Dazu implementiert es ein dezentral organisiertes, verteiltes Dateisystem auf Basis einer Distributed Hashtable (DHT) [1, 19], in der Daten anonym abgelegt und abgerufen werden können. Jeder Freenet-Teilnehmer stellt einen Teil seiner Festplattenressourcen für das verteilte Dateisystem bereit. Durch den Einsatz von asymmetrischen Verschlüsselungstechniken und Einweg-Hashfunktionen hat ein Teilnehmer und niemand anderes Kenntnis darüber, auf welchem Rechner des Freenet-Netzwerkes konkret Daten gespeichert werden, die er veröffentlicht. Freenet ist völlig dezentral organisiert. Das Freenet-Netz wird dynamisch so strukturiert, dass häufig angefragte Inhalte mit höherer Wahrscheinlichkeit gefunden werden können. Wie etwa bei Gnutella [16] gibt es keinerlei Garantien, eine gewünschte Information wirklich aufzufinden - selbst, wenn sie sicher im Netzwerk vorliegt. Strukturierte Netzwerke, wie etwa CAN oder Chord [18, 21] und die hier zugrunde gelegten Gitterstrukturen verfügen über eine feste Topologie, die das Abspeichern und Abrufen von Resourcen in deterministischer Weise realisieren, so dass eine zielgerichtete, inhaltsbasierte Vermittlung möglich ist.

Geheimnisverteilung, Secret Sharing

Secret Sharing bezeichnet ein kryptografisches Verfahren, mit dem ein Geheimnis (shared secret), etwa ein Schlüssel, in *n* Teilgeheimnisse aufgeteilt wird [20]. Diese Teilgeheimnisse können auf *n* Instanzen (etwa Personen) verteilt werden. Erst, wenn ein gewisser Teil dieser Instanzen ihre Teilgeheimnisse zusammenbringen, lässt sich daraus das zu schützende Geheimnis wiederherstellen. In der naiven Form sind *alle* Teilgeheimnisse zur Rekonstruktion erforderlich, man spricht von einem n-aus-n-Schwellwert-Schema; ein Geheimnis X kann etwa durch *n* Summanden x_i dargestellt werden, von denen alle bekannt sein müssen, um X wieder zu rekonstruieren: $X = \sum_{i=1}^{n} x_i$.

Im Rahmen dieses Beitrages wird der Weg vom Sender zum Empfänger einer Nachricht als Shared Secret aufgefasst und so über die Knoten des Gitters verteilt, dass kein Knoten Kenntnis über den kompletten Kommunikationsweg erhält.

3 Architektur der dezentralen Gitterstrukturen

Gitterartige bzw. torusartige Verbindungsstrukturen haben sich bei der Entwicklung von Parallelrechnern, wie etwa den Cray-Systemen oder IBMs Blue Gene [11, 14] bewährt. Dabei werden Recheneinheiten (Knoten) in einer regelmäßigen, orthogonalen Gitterstruktur organisiert, wobei jeder Knoten mit Kommunikationsverbindungen zu seinen Nachbarknoten ausgestattet ist. Auf Basis solcher Strukturen lassen sich etwa zweidimensionale, parallelisierbare Probleme, wie Matrixmultiplikationen effizient auf die Recheneinheiten verteilen und parallel berechnen. Gitter bzw. Tori bieten aber noch weitere Vorteile. So existieren immer mehrere Kommunikationspfade von einem Knoten zu einem anderen. So lassen sich stets Alternativrouten finden, falls ein oder mehrere Knoten eines gewählten Kommunikationspfades ausgefallen sind oder Überlast droht. Genaue Untersuchungen darüber, wie das konkret aussehen kann, liefert [15]. Die gitterartige Struktur erlaubt weiterhin die Verwendung von relativen Koordinaten für die Adressierung eines Knotens. Erhält ein böswilliger Teilnehmer Routing-Informationen, die nicht für ihn bestimmt waren, so kann er keinerlei Rückschlüsse daraus ziehen, solange nicht klar ist, für welchen Vermittler diese Information bestimmt war. In [4] und [5] wird beschrieben, wie regelmäßige Gitterstrukturen aus lose gekoppelten Systemen, etwa Desktop-PCs, die über das Internet miteinander verbunden sind, als Overlay aufgebaut werden können. Sicherheit bzw. Anonymität werden in diesen Beiträgen nicht thematisiert. Die Beiträge in [9] und [10] beschrieben, wie zweidimensionale Gitterstrukturen für den Aufbau so genannter *Networked Virtual Marketplace Environments* verwendet werden können.

4 Anonyme Kommunikation auf dezentralen Gitterstrukturen

Voraussetzungen

Eine d-dimensionale Gitterstruktur ist definiert durch eine Menge $P \subset C$ von diskreten Koordinaten, $C := \{\vec{x} = (x_1, x_2, \ldots, x_d)^T \quad \forall x_1, \ldots, x_d \in \mathbb{Z}\}$. $N_{\vec{p}}$ bezeichnet für alle $\vec{p} \in P$ den Knoten an der Position $\vec{p}$. Jeder Knoten ist mit seinen direkten Nachbarn im Gitter verlinkt; zwei Knoten $N_{\vec{p}}$ und $N_{\vec{q}}$ sind also Nachbarn, wenn gilt: $|\vec{q} - \vec{p}| = 1$. Die Koordinaten in *P* bilden die logischen Adressen der Knoten im Grid. Die Weiterleitung von Nachrichten erfolgt nur zwischen Nachbarn - üblicherweise mit einem Greedy-Algorithmus. Das heißt, ein Knoten wählt zur Weiterleitung stets einen derjenigen Nachbarn, der online ist und die Distanz zum Empfänger der Nachricht reduziert. Weiter wird vorausgesetzt, dass die ungefähre Größe des regelmäßigen Gitters jedem Knoten bekannt ist. Die Beiträge beispielsweise in [2], [12] oder [13] zeigen unterschiedliche Wege, diese Information für dezentrale Netzwerke zu ermitteln. Jeder Knoten $N_{\vec{p}}$ verfügt über ein asymmetrisches Schlüsselpaar $(K_{\vec{p},pub}, K_{\vec{p},priv})$. Der öffentliche Schlüssel $K_{\vec{p},pub}$ ist zwar für jeden anderen Knoten zugänglich, kann jedoch niemals direkt von Knoten $N_{\vec{p}}$ selbst abgerufen werden, stattdessen existiert ein gesonderter Verteilungsmechanismus, der einen beliebigen öffentlichen Schlüssel $K_{\vec{p}}$ unter Angabe von $\vec{p}$ zur Verfügung stellt. Der private Schlüssel $K_{\vec{p},priv}$ ist nur dem Knoten $N_{\vec{p}}$ selbst bekannt.

Algorithmus

Ein Knoten $N_{\vec{s}}$, der eine Nachricht zum Knoten $N_{\vec{r}}$ senden will, wählt zunächst eine Anzahl von Vermittlern aus, Knoten $N_{\vec{v}_i}$, die die Nachricht weitervermitteln sollen. Dabei werden Weiterleitungsinformationen so verschlüsselt, dass ein Vermittler nur diejenigen Informationen lesen kann, die für ihn bestimmt sind. Der Rest der Nachricht bleibt verschlüsselt. Abbildung 2 zeigt die Schritte, die Sender und Vermittler bzw. Empfänger einer Nachricht abarbeiten.

```
function sendTo( r⃗ ∈ P )
    Wähle eine zufällige Zahl z ∈ ℕ, z > 1.
    Setze v⃗_0 := s⃗ und v⃗_{z+1} := r⃗.
    Wähle für alle 1 ≤ i ≤ z zufällige
    unterschiedliche Koordinaten v⃗_i ∈ P\{v⃗_0, v⃗_{z+1}}.
    Beziehe die öffentlichen Schlüssel K_{v⃗_i,pub} für die z
    Knoten N_{v⃗_i}.
    Bilde die Sprungvektoren Δ⃗_i := v⃗_{i+1} − v⃗_i   0 ≤
    i ≤ z.
    Verschlüssele für alle 1 ≤ i ≤ z die
    Sprungvektoren Δ⃗_i mit K_{v⃗_i,pub} und füge sie der zu
    sendenden Nachricht in zufälliger Reihenfolge
    hinzu.
    Verschlüssele den Payload mit K_{v⃗_{z+1}}.
    Sende die Nachricht an P_{v⃗_0+Δ⃗_0}.
end function
```

```
function recv/routemsg
    if (Payload von msg entschlüsselbar
                        mit K_{v⃗_j,priv}?) then
        N_{v⃗_j} ist Empfänger der Nachricht.
    elseif (kann einer der z Sprungvektoren
            Δ⃗_i mit K_{v⃗_j,priv} entschlüsselt werden?) then
        Es handelt sich bei diesem Vektor um Δ⃗_l und
        die Nachricht wird an N_{v⃗_l+Δ⃗_l} weitergeleitet.
    else
        Nachricht ist kompromittiert oder fehlgeleitet
        und wird verworfen.
    end if
end function
```

Abbildung 2: Algorithmen zum Senden und zur Weiterleitung bzw. zum Empfang einer Nachricht

Mit dem Parameter z wird die Anzahl der zu verwendenden Zwischenstationen festgelegt. Je größer z ist, desto sicherer wird der Kommunikationsweg, da umso mehr Knoten mit einbezogen werden. Allerdings wächst mit steigendem auch die Latenz. Es sei nochmal darauf hingewiesen, dass die Nachricht durch die letzte Anweisung nicht *direkt* an $N_{\vec{v}_0+\vec{\Delta}_0}$ gesendet wird, sondern über die vom Gitter-Overlay bereitgestellten Vermittlungstechniken von Nachbar zu Nachbar gereicht wird. Die Vermittler, die eine Nachricht erhalten, probieren nun, den Payload bzw. einen der Sprungvektoren, die das nächste Ziel angeben, mit ihren privaten Schlüsseln zu dekodieren. Man beachte, dass die Nachricht nur die relativen Sprungvektoren $\vec{\Delta}_i := \vec{v}_{i+1} - \vec{v}_i \quad 0 \le i \le z$ mit $\vec{\Delta} = \sum_{i=1}^{z} \vec{\Delta}_i = \vec{r} - \vec{s}$ enthält. Selbst, wenn die Nachricht an einen falschen Vermittler weitergeleitet wurde, und dieser in der Lage wäre, die Sprungvektoren zu entschlüsseln, könnte er weder auf die Quelle noch auf das Ziel noch auf den nächsten Vermittler für die Nachricht schließen.

Rückkanal

Mit dem bis hier beschriebenen Schema ist es einem Empfänger-Knoten nicht möglich, eine ggf. erforderliche Antwort auf eine Nachricht zurück zum Sender zu schicken, solange der Sender seine Absenderkoordinaten nicht im Payload der Nachricht hinterlegt. Dies würde jedoch bedeuten, dass der Empfänger Kenntnis vom Sender der Nachricht erhält. Um Antworten zuzulassen, ohne die Anonymität des Senders aufzudecken, fügt der Sender der Nachricht weitere Informationen hinzu, die dem Empfänger als „Rückumschlag“ für eine Antwort dienen. Dazu generiert der Sender für den Rückweg nach dem oben angegebenen Algorithmus eine weitere Liste von z' zufällig gewählten Vermittlern $N_{\vec{v}_i}{}'$, und mit $\vec{v}_0' := \vec{r}$, $\vec{v}_{z'+1}' := \vec{s}$ z'

Sprungvektoren $\vec{\Delta}_i' = \vec{v}_{i+1}' - \vec{v}_i'$, sodass $\vec{\Delta}' = \sum_{i=1}^{z'} \vec{\Delta}_i' = \vec{s} - \vec{r} = -\vec{\Delta}$. Dies sind also die Vermittlungsinformationen für den Rückweg von $N_{\vec{r}}$ zu $N_{\vec{s}}$. Wie die Sprungvektoren für den Hinweg werden auch die $\vec{\Delta}_i'$ mit den entsprechenden Schlüsseln $K_{\vec{v}_i', pub}$ für alle $0 \leq i \leq z'$ verschlüsselt. Zusätzlich generiert der Sender einen Einmal-Schlüssel $K_{\vec{s}}^*$, den der Empfänger $N_{\vec{r}}$ nutzen kann, um die Antwort für den Sender zu verschlüsseln. $K_{\vec{s}}^*$ wird ebenfalls mit $K_{\vec{r}, pub}$ verschlüsselt. Die verschlüsselten $\vec{\Delta}_i'$ und $K_{\vec{r}}^*$ werden der Nachricht als *Rückumschlag* zugefügt. Will der Empfänger eine Antwort auf eine Nachricht senden, bildet er den Payload, verschlüsselt ihn mit $K_{\vec{r}}^*$, fügt der Nachricht die (nach wie vor verschlüsselten) Rücksendeinformationen $\vec{\Delta}_i'$ für $i \geq 1$ hinzu, entschlüsselt $\vec{\Delta}_0'$ und schickt die Antwort an den ersten für den Rückweg vorgesehenen Vermittler $N_{\vec{r}+\vec{\Delta}_0'}$. Abbildung 2 zeigt eine Nachricht mit Rückumschlag und die darin enthaltenen verschlüsselten Informationen. Abbildung 3 veranschaulicht den Weg den die Nachricht und ihre Antwort durch das Netzwerk anhand der in der Nachricht enthaltenen Informationen nehmen, und welche Vermittler dabei welche Informationen entschlüsseln und verarbeiten, um die Nachricht weiterzuleiten.

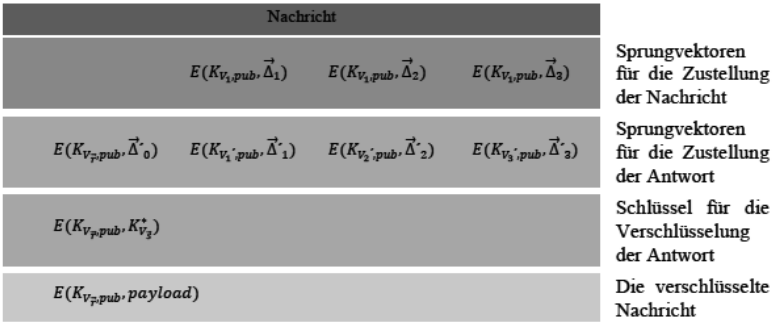

Abbildung 3: Eine Nachricht zum Versand über drei Zwischenstationen. Ihr liegt ein „Rückumschlag" bei, mit dem eine Antwort zurück an schicken kann.

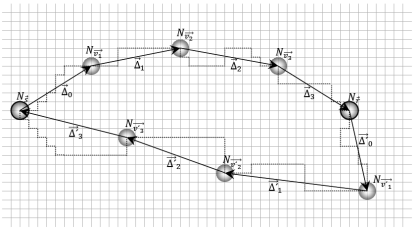

Abbildung 4: Der Weg, den die in Abb. 3 angegebene Nachricht und die Rückantwort durch das Gitter nehmen.

Dadurch, dass die verschlüsselten Sprungvektoren in zufälliger Reihenfolge abgelegt werden, kann kein Vermittler darauf schließen, an welcher Stelle im Pfad er sich befindet. Darüber hinaus ist einem Vermittler nicht einmal der vorangegangene Vermittler bekannt, weil dessen Koordinaten nicht in der Nachricht enthalten sind und er die Nachricht nur durch einen seiner direkten Nachbarn im Gitter erhält. Ein Vermittler kennt zwar die Koordinaten seines Nachfolgers, er kann jedoch nicht feststellen, ob jener Empfänger der Nachricht oder ein weiterer Vermittler ist. Der Empfänger kann entsprechend auch nicht feststellen, wer der Sender ist, sofern diese Information nicht im Payload enthalten ist.

Verteilung und Bezug der öffentlichen Schlüssel

Für die Verschlüsselung der Sprungvektoren in der Nachricht benötigt der Sender die öffentlichen Schlüssel der gewählten Vermittler. Sie sollten nicht direkt beim Vermittler bezogen werden; dies würde es einem Angreifer erleichtern, herauszufinden, welche Vermittler der Sender für die Weiterleitungen ausgewählt hat. Die Bezugsquelle für einen öffentlichen Schlüssel sollte also keinerlei Rückschlüsse auf den Besitzer dieses Schlüssels zulassen. Ein Server, der über eine gesicherte Verbindung sämtliche öffentliche Schlüssel bereitstellt, ist laut Voraussetzung ausgeschlossen; die Beschaffung der Schlüssel muss auf Basis eines völlig dezentral organisierten Overlays möglich sein. Für die Bereitstellung der öffentlichen Schlüssel kann ein völlig eigenes dezentral organisiertes Netzwerk verwendet werden. Alternativ können die Schlüssel auch auf die Knoten des Gitters selbst verteilt werden. Benötigt ein Knoten $N_{\vec{p}}$ den öffentlichen Schlüssel des Knotens $N_{\vec{q}}$ ermittelt er mögliche Koordinaten für Knoten, die diesen Schlüssel besitzen, wählt zufällig eine unter ihnen aus und bezieht über eine gesicherte Verbindung den gewünschten Schlüssel vom entsprechenden Knoten. Da durch die redundante Verteilung der öffentlichen Schlüssel über mehrere Knoten jeder Knoten über viele Schlüssel verfügt, kann ein Beobachter aufgrund der bloßen Observation der Anforderung eines öffentlichen Schlüssels keinen eindeutigen Rückschluss auf einen konkreten Vermittler ziehen. Untersuchungen darüber, wie eine Verteilung öffentlicher Schlüssel über die Knoten eines Gitters konkret aussehen kann, werden Bestandteil weiterer Arbeiten sein.

5 Vergleich und Abgrenzung zu existierenden Arbeiten

Das vorgestellte Schema hat gewisse Gemeinsamkeiten mit den in Kapitel 2 vorgestellten Arbeiten. Allerdings gibt es wesentliche Unterschiede, die an dieser Stelle zur Abgrenzung angesprochen werden sollen:

- Die Mix Kaskaden beim Onion Routing bestehen aus einigen wenigen dedizierten Servern, während in diesem Ansatz prinzipiell jeder Knoten Teil einer Mix-Kaskade sein kann. Dies sorgt für eine höhere Sicherheit, da nicht vorhersagbar ist, welche Knoten als Vermittler verwendet werden.

- Die Nachrichten enthalten keinerlei absolute und damit identifizierende Adressen. Die (logischen) Adressen der Vermittler sind als relative Koordinaten abgelegt, die nur von den Vermittlern richtig interpretiert werden können, für die sie bestimmt sind.
- Beim Onion Routing wird eine Nachricht mehrfach verschlüsselt, wobei bei jedem Verschlüsselungsvorgang die Adresse des nächsten Mixes hinterlegt wird. Im vorgestellten Ansatz wird dagegen die Adresse jedes Vermittlers nur einmal verschlüsselt. Dies und die Tatsache, dass die Vermittleradressen relativ gespeichert werden führt dazu, dass ein Angreifer selbst bei Kenntnis der privaten Schlüssel keine Aussagen darüber treffen kann, welche Adressen bereits zur Vermittlung herangezogen wurden – entsprechend kann aus diesen Informationen nicht auf das Ziel einer Nachricht geschlossen werden.
- Auch Freenet bietet anonyme Kommunikation zwischen Teilnehmern an. Allerdings handelt es sich bei Freenet um ein unstrukturiertes Netzwerk, was zur Folge hat, dass Informationen ‚blind' gesucht werden müssen, und es keine Garantie gibt, eine Information – selbst wenn sie im Netzwerk vorhanden ist – aufzufinden. Das hier zugrundegelegten Gitter bieten hingegen eine durch Gitter-Koordinaten beschriebene Struktur. Fasst man die Gitter-Koordinaten etwa als Schlüsselraum für eine verteilte Hashtabelle auf, so ist das deterministische Ablegen und Auffinden von Informationen leicht möglich.

6 Zusammenfassung

Das hier vorgestellte Kommunikationsschema ermöglicht anonyme Kommunikation auf Basis von dezentral organisierten regelmäßigen Gitterstrukturen mit folgenden Eigenschaften:

- Der Sender einer Nachricht ist für den Empfänger und die Vermittler unbekannt. Der Empfänger einer Nachrcht ist nur dem Sender bekannt.
- Ein Vermittler kennt nur einen Teil des Weges einer Nachricht von ihm zum nächsten Vermittler (Secret Sharing)
- Die in einer Nachricht enthaltenen Vermittlungsinformationen gelten nur relativ zu den Vermittlern für die sie bestimmt sind. Erhält ein anderer Knoten diese Information, kann er keine Rückschlüsse auf Sender, Empfänger oder weitere Vermittler ziehen, solange er nicht weiß für welchen Vermittler diese Information bestimmt war.
- Nachrichten können mit einem Rückumschlag versehen werden, der es einem Empfänger ermöglicht, Antwortnachrichten an den Sender abzusetzen ohne seine Identität aufzudecken.
- Die Weiterleitungsinformationen in einer Nachricht sind nicht (wie beim Onion Routing) mehrfach verschlüsselt, wodurch die in Abschnitt 7 angedeutete Pfadverschränkung zur Erhöhung der Zuverlässigkeit möglich wird.

7 Zukünftige Arbeiten

Performance vs. Sicherheit

Die Kommunikation nach dem hier vorgestellten Schema verläuft in der Regel über eine Vielzahl von Knoten und ist mit entsprechend hohen Latenzzeiten behaftet. Es muss genauer untersucht werden, welchen Einfluß etwa die Abstände der Vermittler oder ihre Anzahl auf die Sicherheit des Systems haben.

Zuverlässigkeit, Ausfallsicherheit

Weiterhin wird untersucht werden, welche Maßnahmen zur Erhöhung der Zuverlässigkeit ergriffen werden können. So wäre es etwa möglich eine Nachricht auf mehreren, disjunkten Pfaden zu versenden, um trotz eines Ausfalls auf einem Pfad die Ankunft einer Nachricht sicherzustellen. Alternativ könnten Sprungvektoren mehrfach aber mit verschiedenen Schlüsseln kodiert vorhanden sein, so dass ein Vermittler nicht nur den Sprungvektor für einen einzigen Nachfolger, sondern für f Nachfolger entschlüsseln kann. Sollte ein Vermittler dann feststellen, dass das Weiterleiten einer Nachricht fehlgeschlagen ist, wählt er einfach einen anderen der $\mathbf{f-1}$ anderen Vermittler. Die Pfadlänge wird so zwar effektiv verkürzt, denn im Extremfall werden vom jeden Vermittler $\mathbf{f-1}$ folgende Vermittler übersprungen. Allerdings existiert „im Schlepptau“ und verschränkt ein f-redundanter Pfad.

Schlüsselverteilung

Auch die Verteilung der öffentlichen Schlüssel verdient ein besonderes Augenmerk. Neben der Möglichkeit, eine völlig eigene Infrastruktur für die Bereitstellung der öffentlichen Schlüssel aller Teilnehmer zu schaffen, könnten die Schlüssel auch im Gitter selbst abgelegt sein. Um die Verfügbarkeit aller Schlüssel zu gewährleisten würde man sie mehrfach und gleichmäßig über die Knoten verteilt ablegen, etwa mit Hilfe einer indizierten Hashfunktion, oder entlang einer Space-Filling-Curve, die auf dem Gitter definiert wird.

Anbieteranonymität

Schließlich sind Erweiterungen in Arbeit, die eine totale Anonymisierung erlauben, d.h. ein Angreifer wird nicht einmal mehr feststellen können, wer der Urherber einer abgerufenen Information ist.

Literaturverweise

[1] Androutsellis-Theotokis, Spinellis. A survey of peer-to-peer content distribution technologies. ACM Computing Surveys, 36:335-371, 2004.

[2] Bawa, Garcia-Molina, Gionis, Motwani. Estimating Aggregates on a Peer-to-Peer Network. Technical Report, 2003-24, Stanford InfoLab, 2003.

[3] Berg, Sukjit, Unger, Nicolaysen, Jens G. M. ICE - a self-organizing infrastructure for a flexible service management within supply chains. Proceedings of the IASTED International Conference on Communication Systems, Networks, and Applications in CSNA ,07, pages 242-247, Anaheim, CA, USA, 2007. ACTA Press.

[4] Berg, Unger. n-Dimensional Border Growth. In Eichler, Gerald and Kropf, Peter G. and Lechner, Ulrike and Meesad, Phayung and Unger, Herwig, editors, IICS in LNI, pages 296-305, 2010. GI.

[5] Berg, Unger, Sukjit. Borderline-growth: a new method to build complete, dense grids with local algorithms. 2nd International Workshop on Nonlinear Dynamics and Synchronization in INDS, pages 95-99, 2009. Shaker.

[6] Buyya, Stockinger, Giddy, Abramson. Economic Models for Management of Resources in Peer-to-Peer and Grid Computing. 2001.

[7] Chothia, Chatzikokolakis. A Survey of Anonymous Peer-to-Peer. in Proceedings of the IFIP International Symposium on Network-Centric Ubiquitous Systems (NCUS 2005), Lecture Notes in Computer Science, pages 744-755. Springer.

[8] Clarke, Sandberg, Wiley, Brandon, Hong, Theodore. Freenet: A Distributed Anonymous Information Storage and Retrieval System. Internaltional Workshop o Designing Privacy enhancing technologies: Design Issues in Anonymity and Unobservability , pages 46-66, 2001. Springer-Verlag New York, Inc.

[9] Coltzau. P2Life: An Infrastructure for Networked Virtual Marketplace Environments. IJIIP, 1(2):1-13, 2010.

[10] Coltzau, Ulke. Navigation in the P2Life Networked Virtual Marketplace Environment. In Unger, Herwig and Kyamaky, Kyandoghere and Kacprzyk, Janusz, editors, Autonomous Systems: Developments and Trends in Studies in Computational Intelligence, pages 213-227. Springer Berlin / Heidelberg, 2012.

[11] Gara, Blumrich, Chen, Chiu, Coteus, Giampapa, Haring, Heidelberger, Hoenicke, Kopcsay, Liebsch, Ohmacht, Steinmacher-Burow, Takken, Vranas. Overview of the Blue Gene/L system architecture. IBM J. Res. Dev., 49(2):195-212, 2005.

[12] Horowitz, Malkhi. Estimating Network Size from Local Information. Information Processing Letters, 88:237-243, 2003.

[13] Jelasity, Montresor. Epidemic-Style Proactive Aggregation in Large Overlay Networks. Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS‘04) in ICDCS ,04, pages 102-109, Washington, DC, USA, 2004. IEEE Computer Society.

[14] Lerstuwanakul. Multiple Criteria Routing Algorithms in Mesh Overlay Networks. PhD thesis, Fernuniversität Hagen, 2012.
[15] Milojicic, Kalogeraki, Lukose, Nagaraja, Pruyne, Richard, Rollins, Xu. Peer-to-Peer Computing. Technical report, 2002.
[16] Pourebrahimi, Bertels, Vassiliadis. A survey of peer-to-peer networks. Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Proessing, 2005.
[17] Ratnasamy, Francis, Shenker, Karp, Handley. A Scalable Content-Addressable Network. In Proceedings of ACM SIGCOMM, pages 161-172, 2001.
[18] IBM journal of Research and Development staff. Overview of the IBM Blue Gene/P project. IBM J. Res. Dev., 52(1/2):199-220, 2008.
[19] Sarmady. A Survey on Peer-to-Peer and DHT. 2007.
[20] Shamir. How to share a secret. Commun. ACM, 22(11):612-613, 1979.
[21] Stoica, Morris, Karger, Kaashoek, Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. pages 149-160, 2001.