in: Patrick Horster (Hrsg.): Kommunikationssicherheit im Zeichen des Internet. DuD-Fachbeiträge, Vieweg, Wiesbaden, 2001, 235-249

# Praktischer Schutz vor »Flooding-Angriffen« bei Chaumschen Mixen

Oliver Berthold · Hannes Federrath · Stefan Köpsell

TU Dresden, Fakultät Informatik
01062 Dresden
{ob2, federrath, koepsell}@inf.tu-dresden.de

### Zusammenfassung

Dieses Papier beschreibt Verfahren, mit denen sich Angriffe der Klasse der Flooding- bzw. „*n*-1"-Angriffe auf Anonymisierungsdienste erkennen lassen und deren Erfolg verhindert werden kann.

## 1 Einführung

In der Literatur sind mehrere grundlegende Verfahren zur unbeobachtbaren Kommunikation in Rechnernetzen beschrieben. Eines der bedeutsamsten Verfahren ist das von David Chaum beschriebene Mix-Netz [Chau81], bei dem Nachrichten über anonymisierende Zwischenstationen (Mixe) geschickt werden. Ein Mix, der im Schub-Betrieb (engl. batch) arbeitet, *sammelt* mehrere Nachrichten, bevor er sie *umkodiert* und *umsortiert* wieder ausgibt. Ein Angreifer, der sämtliche Verbindungsleitungen überwacht, kann nicht beobachten, welche Ausgabe- zu welcher Eingabenachricht gehört. Alle Sender, die Nachrichten zu ein und demselben Schub beigesteuert haben, bilden eine sogenannte Anonymitätsgruppe (engl. anonymity set). Diese ist am größten, wenn alle Nachrichten von unterschiedlichen Sendern stammen.

## 2 Problemstellung

Befinden sich innerhalb der Anonymitätsgruppe eine bestimmte Anzahl von Angreifern, so verkleinert sich die tatsächliche Anonymitätsgruppe um eben diese Zahl. Im schlimmsten Fall besteht die Anonymitätsgruppe der Mächtigkeit *n* aus *n*-1 Angreifern und einem weiteren Nutzer. Dieser Nutzer ist dann nicht mehr in der Lage, unbeobachtbar Nachrichten zu verschicken. Der Begriff „*n*-1"-Angriff bezeichnet die bewußte Herbeiführung dieses Zustandes durch einen Angreifer. Einem Angreifer gelingt dies, indem er den Mix unbemerkt mit eigenen Nachrichten flutet und die zwischenzeitlich ankommenden Nachrichten anderer Sender blockiert, zwischenpuffert und im nächsten Schub selbst an den Mix sendet.

Um einen sicheren Anonymisierungsdienst realisieren zu können, ist es wichtig, sowohl das Fluten als auch das Blockieren von Nachrichten zu verhindern bzw. zu erkennen und die Nutzer entsprechend zu informieren.

# 3 Existierende Ansätze

## 3.1 Poolbetrieb

Neben dem Schub-Betrieb kann ein Mix auch im Pool-Betrieb arbeiten [Cott95]. Hier werden im Ausgabepuffer (Pool) *x* Nachrichten gespeichert. Sobald eine neue Nachricht den Pool erreicht, wird aus den *x* Nachrichten eine zufällig ausgewählt und ausgegeben. Um eine bestimmte Nachricht durch Fluten zu beobachten, muss der Angreifer den Pool zunächst mit seinen eigenen Nachrichten füllen, dann abwarten, bis die zu beobachtende Nachricht beim Mix eintrifft, um ihn erneut solange zu fluten, bis die eine ihm unbekannte Nachricht vom Mix ausgegeben wurde. Zwischenzeitlich eintreffende Nachrichten muß der Angreifer möglichst blockieren und darf sie erst nach Angriffsende an den Mix schicken.

Der Angriff wird aufwendiger, da im Mittel deutlich mehr Nachrichten an einen Mix gesendet werden müssen – er bleibt aber trotzdem erfolgreich.

## 3.2 Unabhängige Verzögerung

Eine weitere Möglichkeit ist, Nachrichten zeitgesteuert auszugeben. Der Ausgabezeitpunkt wird dabei entweder vom Mix (zufällig) bestimmt oder vom Sender festgelegt [KeBS99]. Da jetzt die Ausgabe einer Nachricht unabhängig von den anderen ist, muss der Angreifer den Mix nicht mehr fluten, sondern nur noch dafür sorgen, dass zwischenzeitlich keine weiteren Nachrichten den Mix erreichen.

Da ein Angreifer auch beim Fluten andere Nachrichten blockieren muss, ist diese Ausgabestrategie unsicherer als der Batch- und Poolbetrieb. Allerdings darf die Blockade anderer Nachrichten nicht zu lange dauern, wenn der Absender eine absolute Ausgabezeit in seine Nachricht hineinkodiert hat, da der Mix die Nachricht sonst wegwerfen wird, was zumindest ein Hinweis auf einen Angriff sein könnte.

## 3.3 Broadcast

Diese Methode ist eine Möglichkeit, um das Blockieren von Nachrichten zu erkennen. Dabei kontrolliert jeder Nutzer, ob seine Nachricht innerhalb des (erwarteten) Schubes vorhanden ist. Ein Mix sendet dazu alle Nachrichten eines Schubes an alle Teilnehmer. Stellt ein Nutzer fest, dass seine Nachricht enthalten ist, so teilt er dies dem Mix mit. Dieser bearbeitet den Schub nur, wenn genügend viele *unterschiedliche* Nutzer bestätigt haben, dass ihre Nachricht im Schub enthalten ist. Wesentlich bei diesem Verfahren ist, dass ein Angriff zunächst nur erkannt wird. In diesem Fall muss der Mix den kompletten Schub verwerfen, da nur so der Erfolg des Angriffs verhindert werden kann.

Um zu entscheiden, ob die Nachrichten von „genügend vielen unterschiedlichen Teilnehmern“ gesendet wurden, stehen folgende Möglichkeiten zur Verfügung:

- Melden sich die Teilnehmer vor Nutzung des Anonymitätsdienstes bei jedem Mix mit einem zertifizierten Pseudonym an (siehe Abschnitt 4.1), kann der Mix überprüfen, ob alle angemeldeten Teilnehmer eine Bestätigung gesendet haben.
- Melden sich die Teilnehmer nicht an, kann der Mix erwarten, dass jede Nachricht von einem anderen Teilnehmer gesendet wurde. Deshalb erwartet der Mix ebenso viele von ver-

schiedenen Teilnehmern signierte Bestätigungen wie Nachrichten im Schub enthalten sind.

Um Fehlertoleranz zu erreichen, kann auch ein bestimmter Prozentsatz der erwarteten Bestätigungen als ausreichend betrachtet werden. Der Angreifer kann in diesem Fall die Anonymitätsgruppe auf eben diesen Prozentsatz verkleinern.

Das Verfahren basiert auf gegenseitiger Kontrolle von Teilnehmer und Mix. Dies verhindert, dass ein Angreifer einfach ausreichend viele Bestätigungen generiert.

Methoden dieser Klasse verursachen zumindest in Vermittlungsnetzen wie dem Internet einen sehr hohen Übertragungsaufwand, da an alle Teilnehmer alle Nachrichten gesendet werden müssen. Da jeder Mix auf Bestätigungen von allen Teilnehmern warten muss, erhöht sich die Verzögerungszeit.

# 4 „*n*-1“-Angriffe auf Mix-Kaskaden

Im Falle einer Mix-Kaskade sind mehrere Mixe in einer beliebigen, aber festen Reihenfolge verbunden, wobei jeder Mix höchstens einen Vorgänger und höchstens einen Nachfolger besitzt. Betrachtet werden Mixe, die im Schub-Betrieb arbeiten.

Ein Angreifer könnte *n*-1 Nachrichten dieses Schubes selbst generieren und somit den oben beschriebenen Angriff durchführen. Die nachfolgend aufgeführten Lösungen beruhen darauf sicherzustellen, dass jeder Schub Nachrichten von genügend vielen unterschiedlichen Nutzern enthält. Dies erfordert, dass ein Mix die einzelnen Teilnehmer unterscheiden und überprüfen muss. Dies kann z. B. mit Hilfe von pseudonymen digitalen Zertifikaten realisiert werden. Ein Angreifer kann nicht mehr beliebig viele Nachrichten zu einem Schub beisteuern, es sei denn, er besitzt viele „digitale Identitäten“.

## 4.1 Pseudonyme digitale Zertifikate

Eine Möglichkeit zur sicheren Identifizierung von Teilnehmern in einem offenen Netz bieten digitale Zertifikate. Die Teilnehmer authentifizieren sich damit gegenüber allen verwendeten Mixen.

Die Authentifizierung dient lediglich dazu, die Anonymitätsgruppe für einen Schub genau bestimmen zu können, *ohne* jedoch offenzulegen, wer konkret welche Nachricht beigesteuert hat. Insofern können die Zertifikate auch auf *Pseudonyme* ausgestellt sein.

Da mit der Authentikation erreicht werden soll, dass kein Teilnehmer den Dienst unter verschiedenen Identitäten nutzen kann, müssen die Zertifikate genau einer der folgenden Bedingungen genügen.

- In den Zertifikaten ist eine eindeutige Zuordnung zu einer natürlichen Person definiert, so dass ein Mix erkennt, wenn sich eine Person mehrfach mit verschiedenen Zertifikaten anmeldet.
- Ein Mix akzeptiert nur (pseudonyme) Zertifikate einer einzigen Zertifizierungsstelle, die garantiert, dass sie einer natürlichen Person nur genau ein Zertifikat ausstellt.

Die zweite Möglichkeit entschärft den Gegensatz, dass zur Nutzung eines Anonymisierungsdienstes zuerst eine Identifikation der Teilnehmer erforderlich ist, da diese Zertifikate keinen direkten Personenbezug enthalten müssen.

Ein weiteres Problem ist die zentrale Position der Zertifizierungsstelle. Eine nicht korrekt arbeitende Zertifizierungsstelle kann dem Angreifer ausreichend viele Zertifikate ausstellen und somit einen „*n*-1“- Angriff ermöglichen.

Diesem Problem kann man dadurch begegnen, dass zur Nutzung eines Mixes mehrere Zertifikate verschiedener Zertifizierungsstellen notwendig sind, wodurch das nötige Vertrauen verteilt wird. Ein anderer Ansatz ist, dass der Mix für sich selbst als Zertifizierungsstelle in Aktion tritt, was für ihn einen hohen zusätzlichen Aufwand bedeutet.

## 4.2 Ticketmethode

In diesem Abschnitt wird ein Verfahren vorgestellt, welches jedem Mix ermöglicht, einen „*n*-1“-Angriff ohne zusätzliche Verzögerungszeit zu erkennen

Dabei werden sogenannte *anonyme Tickets*[1] verwendet, die es einem Mix ermöglichen, für jede Nachricht zu überprüfen, ob sie von einem berechtigten Teilnehmer gesendet wurde. Zudem wird sichergestellt, dass jeder Teilnehmer nur eine definierte Anzahl von Nachrichten zu einem Schub beitragen kann. Ein Mix bearbeitet einen Schub erst ab einer bestimmten Mindestanzahl von Nachrichten (bzw. Sendern). Daher kann der Angreifer einen „*n*-1“-Angriff nur durchführen, wenn er einen wesentlichen, durch die Wahl der Parameter festlegbaren, Anteil der zertifizierten Teilnehmer kontrolliert.

### 4.2.1 Verfahrensbeschreibung

Das Senden von Nachrichten geschieht in zwei Phasen:

**Phase 1:** Der Teilnehmer verbindet sich über eine verschlüsselte Verbindung mit dem Mix. Er authentifiziert sich gegenüber dem Mix mit Hilfe eines oder mehrerer digitaler Zertifikate (siehe Abschnitt 4.1). Ebenso authentifiziert sich der Mix gegenüber dem Teilnehmer. Nach erfolgreicher Authentikation fordert der Teilnehmer für jeden Schub, in dem er eine Nachricht durch diesen Mix senden möchte, ein Ticket an. Mit diesem Ticket wird in der später zu sendenden Nachricht nachgewiesen, dass sie von einem berechtigten Teilnehmer stammt.

An ein derartiges Ticket und das Ausgabeverfahren müssen folgende Anforderungen gestellt werden, um die erwünschte Sicherheit zu erzielen:

- Es darf nur dem Mix möglich sein, gültige Tickets zu generieren.
- Ein Ticket darf nur für einen festgelegten Schub gültig sein, da ansonsten ein Teilnehmer viele Tickets, die für verschiedene Schübe gedacht waren, im selben Schub verwenden kann.

[1] Die Idee der Tickets stammt von anonymen digitalen Zahlungssystemen: Nach dem gleichen Schema werden anonyme digitale Geldmünzen erzeugt.

- Der Mix darf die einzelnen für einen Teilnehmer erstellten Tickets nicht wiedererkennen können, d. h. er darf die später in der Nachricht mitgesendeten Tickets nur auf Gültigkeit für diesen Schub testen können, nicht jedoch (z.B. anhand gleichem Aussehen) eine Verkettung zu einem bestimmten Teilnehmer herstellen können.
- Der Mix muss speichern, an welche Teilnehmer(pseudonyme) er bereits welche Tickets ausgegeben hat, so dass kein Teilnehmer durch mehrfache (parallele) Abfragen mehr Tikkets erhalten kann als vorgesehen.
- Das jeweilige Ticket darf keinem Dritten bekannt werden.

Verwendet man für die Tickets blinde Signaturen [Chau83], so sind die obigen Forderungen erfüllbar. Dazu erstellt sich der Teilnehmer eine kurze Nachricht *m*, die er mit einer Zufallszahl *r* blendet. Für die Nachricht *m* ist es wichtig, dass sie einen zufälligen Anteil und eine durch den Mix eindeutig überprüfbare Redundanz enthält. Beispielsweise könnte sich die Nachricht aus einer Zufallszahl *z* und dem Wert einer öffentlichen Hashfunktion *h* über *z* zusammensetzen.

(1) $m = (z, h(z))$

Wird z.B. RSA als Signatursystem verwendet, so erfolgt das Blenden durch:

(2) $m' = m \cdot r^t \bmod(p \cdot q)$

*t* ... Testschlüssel für das digital signierte Ticket des betreffenden Schubes
*p,q* ... geheime Primzahlen des Mixes.

Der Teilnehmer sendet $m'$ an den Mix. Der Mix protokolliert die Erzeugung eines Tickets für diesen Schub und Teilnehmer in seiner Datenbank, um eine wiederholte Anforderung zu erkennen. Danach signiert er $m'$ und sendet das Ergebnis $sig(m')$ an den Teilnehmer zurück.

Der Teilnehmer kann nun aufgrund der multiplikativen Eigenschaften von RSA die Signatur entblenden, so dass sie für die entblendete Nachricht gültig ist.

(3) $sig(m) = sig(m') \cdot r^{-1} \bmod(p \cdot q)$

Der Mix kann $m$ nicht zu $m'$ verketten, da *r* in (2) zufällig gewählt ist, so dass *m* jeden möglichen Wert des Restklassenrings annehmen kann (unabhängig von $m'$). Der Teilnehmer kann keine (zusätzlichen) Tickets selbst erzeugen, da angenommen wird, dass er das Signatursystem nicht brechen kann. Die Entblendung ist nur mit der Kenntnis von *r* möglich.

Die Redundanz *h(z)* ist notwendig, da sonst gültige Tickets generiert werden können, indem man einen Wert für *sig(m)* wählt und die zugehörige Nachricht *m* berechnet.

(4) $m = \left(sig(m)\right)^t \bmod\left(p \cdot q\right)$

Da es bei diesem Signatursystem nicht möglich ist, Informationen zu signieren, die der Signierende vorher überprüfen kann (wie z.B. eine Schub-ID), muß der Mix für die Tickets jedes Schubs einen anderen Signierschlüssel wählen. Dabei genügt es, einen neuen Testschlüssel $t'$ zu wählen und den dazugehörigen Signierschlüssel $s'$ zu berechnen, ohne jedoch ein neues Geheimnis (*p, q*) zu verwenden. Diese Berechnung ist effizient möglich und nur einmal pro Schub notwendig.

**Phase 2:** Der Teilnehmer sendet seine Nachricht durch die Mix-Kaskade. Dabei wird in jeden Nachrichtenkopf[2] zusätzlich das erhaltene und entblendete Ticket eingefügt. Ein Mix prüft nun bei jeder erhaltenen Nachricht, ob sie ein gültiges Ticket enthält. Ist dies nicht der Fall, löscht der Mix die Nachricht. Ab einem festzulegenden Schwellwert von ungültigen Nachrichten löscht der Mix den gesamten Schub. Dieser Schwellwert ergibt sich aus einer Abwägung zwischen Fehlertoleranz und Robustheit gegen DoS-Angriffe und dem zu erreichenden Schutzniveau. Je mehr fehlerhafte Nachrichten vom Mix toleriert werden, desto stärker kann ein Angreifer die Anonymitätgruppe verkleinern. Bearbeitet ein Mix hingegen einen Schub nur dann, wenn die Anzahl ausgegebener Tickets mit der Anzahl gültiger Nachrichten übereinstimmt, besteht die Anonymitätsgruppe immer aus allen angemeldeten Teilnehmern.

### 4.2.2 Aufwandsbetrachtung

Für jeden Schub ist eine zusätzliche direkte Kommunikation mit jedem Mix der Kaskade erforderlich, um die Tickets anzufordern (Phase 1). Tickets müssen jedoch erst ab dem zweiten Mix ausgetauscht werden, da die Übermittlung der zu sendenden Nachrichten zwischen Teilnehmern und erstem Mix direkt und authentisiert erfolgen kann. Desweiteren kann man annehmen, dass der Aufwand für die gegenseitige Authentikation nur einmal pro Teilnehmer und Mix notwendig ist, da entweder die Tickets für alle gewünschten Schübe blockweise abgerufen werden oder die gesicherte Verbindung über längere Zeit bestehen bleibt. Der zusätzliche Kommunikationsaufwand für den Erhalt der Tickets beträgt somit pro Teilnehmer und Schub:

$$2 \cdot l \cdot (a-1)\,[Bit]$$

*a ... Anzahl Mixe der Kaskade*

*l ... Sicherheitsfaktor: Länge der Signaturen bzw. Signierschlüssel in Bit*

Der Aufwand setzt sich zusammen aus der Sendung der Nachricht $m'$ zum Mix und dem Empfang der Signatur $sig(m')$. Die für jeden Schub neu gewählten Testschlüssel *t* brauchen nicht übertragen zu werden, da diese Wahl auch anhand einer öffentlichen Funktion erfolgen kann. Hinzu kommt noch ein linearer zusätzlicher Kommunikationsaufwand für das Mitsenden der Tickets in den Mix-Nachrichten. Um wieviel Bit sich die Nachricht verlängert, ist abhängig davon, wie gut der Inhalt des Tickets zusätzlich für andere Zwecke verwendet werden kann. So ist es möglich, einen Teil der Zufallszahl *z* als symmetrischen Schlüssel für die hybride Verschlüsselung der Nachricht zu verwenden.

Der Kommunikationsaufwand der Ticketmethode ist folglich $O(a)$, wie auch das Senden einer Nachricht über *a* Mixe. Für die auf Broadcast basierenden Verfahren (Abschnitt 3.3) ist hingegen der Aufwand quadratisch: $O(a \cdot n)$; *n* ... Anzahl Nachrichten pro Schub.

Verwendet man für das Senden der Nachrichten in Phase 2 die im folgenden beschriebene Methode, so reduziert sich der Berechnungsaufwand für die Überprüfung der Tickets auf das Testen der Redundanz.

---

[2] Eine Mix-Nachricht enthält für jeden Mix einen separaten asymmetrisch mit $c_{Mix}$ verschlüsselten Nachrichtenkopf und einen symmetrisch mit $k_{Mix}$ verschlüsselten Nachrichtenteil, der ggf. Nachrichtenköpfe für weitere Mixe enthält. Ein Nachrichtenkopf enthält u.a. den zur symmetrischen Entschlüsselung notwendigen Schlüssel $k_{Mix}$.

$c_{Mix}(k_{Mix}, ...)$, $k_{Mix}(...)$

Bei Anwendung der Ticketmethode muss im Nachrichtenkopf zusätzlich das jeweilige Ticket enthalten sein. Verwendet man für die Verschlüsselung und die Ticketerzeugung RSA, kann man (ohne zusätzliches Risiko) für beide Operationen das gleiche Geheimnis wählen.

Dies bedeutet, dass nur genau ein Paar (*p, q*) vom Mix genutzt wird, wobei der Mix einen öffentlichen Exponenten *c* für die Verschlüsselung und eine Bildungsvorschrift für die Testschlüssel *t* der Tickets veröffentlicht.

Der Nachrichtenkopf sieht nun folgendermaßen aus:

$c(sig(m))$

Der symmetrische Schlüssel *k* ist wie oben beschrieben Teil des Tickets:

$m=(k, z, h(k, z))$

Der Mix muß diesem Nachrichtenkopf zuerst mit seinem privaten Schlüssel *d* entschlüsseln (potenzieren) und danach die Signatur des Tickets mit dem öffentlichen Testschlüssel *t* des aktuellen Schubes testen, um *m* zu erhalten und zu prüfen.

$$m = \left[ \left( c(sig(m)) \right)^{d} \right]^{t} \bmod \left( p \cdot q \right) = \left( c(sig(m)) \right)^{d \cdot t} \bmod \left( p \cdot q \right) \tag{6}$$

Das Produkt $d \cdot t$ kann vorausberechnet werden. Für den Test des Tickets ist keine zusätzliche asymmetrische Operation erforderlich. Es ist lediglich noch notwendig, die Redundanz zu prüfen.

Da Nachrichten eines Teilnehmers nur akzeptiert werden, wenn sie ein gültiges Ticket enthalten, kann die Abrechnung von Mixdienstleistungen zwischen Teilnehmer und dem einzelnen Mix mit der Ausgabe der Tickets verbunden werden. Da sich die Teilnehmer dabei gegenüber dem Mix authentifizieren, können herkömmliche Zahlungsverfahren eingesetzt werden. Dies wiederum vereinfacht die Abrechnung von Anonymitätsdienstleistungen gegenüber den in [BaNe99, FrJe98] beschriebenen Ansätzen in der Praxis erheblich.

## 4.3 Hashwertmethode

Im Vergleich zu der im letzten Abschnitt beschriebenen Methode zur Verhinderung von „*n*-1“- Angriffen wird bei der Hashwertmethode ein anderer Ansatz verfolgt.

Ziel ist es nicht mehr, den Angreifer am Erzeugen und Senden von Nachrichten zu hindern, sondern es soll sichergestellt werden, dass alle von angemeldeten Teilnehmern gesendeten Nachrichten korrekt durch alle Mixe der gegebenen Kaskade geleitet werden. Trotz Einsatz der Hashwertmethode ist es einem Angreifer leicht möglich, zusätzliche Nachrichten von einem Mix verarbeiten zu lassen. Die Sicherheit gegen „*n*-1“-Angriffe basiert darauf, dass der Angreifer keine Nachrichten anderer Teilnehmer unbemerkt löschen oder verändern kann.

Grundidee der Hashwertmethode ist die Überprüfung des Eingabeschubes durch den Mix, wobei für den Schub *insgesamt* überprüft wird, ob die enthaltenen Nachrichten gültig sind. Zur Erinnerung: Bei der Ticketmethode wurde die Überprüfung der Gültigkeit *jeder einzelnen* Nachricht durch anonyme Tickets realisiert.

Ein Mix vergleicht die bitweise Überlagerung (XOR-Verknüpfung) aller Hashwerte[3] der (entschlüsselten) Nachrichten seines Eingabeschubes mit einem Referenzwert. Durch dieses Vorgehen wird nun nur noch erkannt, ob der Schub insgesamt fehlerfrei ist oder nicht. Wesentlich für die Sicherheit der Hashwertmethode ist, dass eine Trennung zwischen der einzelnen Nachricht und dem zur Überprüfung verwendeten Referenzwert erreicht wird.

Zur Verdeutlichung: In dem Referenzwert müssen die einzelnen Hashwerte der Nachrichten bitweise überlagert enthalten sein, damit der Vergleich positiv ausfallen kann. Diese Hashwerte müssen von den Teilnehmern mitgesendet werden, da außer dem Mix nur der jeweilige Sender die entschlüsselte Nachricht kennt. Einem Angreifer dürfen die einzelnen Hashwerte nicht bekannt werden, da er ansonsten eine Zuordnung zwischen Teilnehmer und Nachricht vornehmen kann.

Die Hashwertmethode erreicht dieses durch eine geschickte Verschlüsselung der vom Teilnehmer auszugebenden Hashwerte. Dabei muß diese Verschlüsselung folgende Bedingungen erfüllen:

- Eine Entschlüsselung darf außer dem Teilnehmer nur noch dem entsprechenden Mix möglich sein, damit nicht beispielsweise einer zentralen Station, welche die Überlagerung der Hashwerte aller Teilnehmer vornimmt, vertraut werden muss.
- Die Entschlüsselung muss auch noch möglich sein, nachdem verschiedene Hashwerte überlagert wurden.

Diese Anforderungen erfüllt das Pseudo-One-Time-Pad. Dabei wird eine Nachricht dadurch verschlüsselt, dass eine gleichlange Zufallszahl (Schlüssel) mit der Nachricht bitweise überlagert (XOR-verknüpft) wird. Der Schlüsselaustausch wird realisiert, indem Sender und Empfänger die Initalisierungsparameter eines Pseudozufallszahlengenerators (PZG) austauschen, welcher die eigentlichen Schlüssel erzeugt.

Die bitweise Überlagerung ist eine kommutative und assoziative Operation. Deshalb kann die Entschlüsselung auch nach erfolgter Überlagerung der Hashwerte mehrerer Teilnehmer durchgeführt werden.

Ganz grob beschrieben arbeitet das Verfahren folgendermaßen:

0. Die Teilnehmer melden sich bei jedem Mix an und tauschen die notwendigen Parameter für das Pseudo-One-Time-Pad aus.
1. Die Teilnehmer erzeugen die Nachrichten, welche sie durch die Mix-Kaskade senden wollen. Dabei verschlüsselt jeder Teilnehmer seine Nachricht sequentiell für alle Mixe *und* berechnet zusätzlich vor jeder Verschlüsselungsstufe den entsprechenden Hashwert.
2. Der Teilnehmer verschlüsselt diese Hashwerte mit dem Pseudo-One-Time-Pad.
3. Der Teilnehmer sendet die Nachricht sowie die verschlüsselten Hashwerte an eine Station, die die Überlagerung aller Teilnehmer-Hashwerte für diesen Schub vornimmt. Diese Aufgabe kann auch der erste Mix übernehmen.
4. Die Mixe entschlüsseln die Referenzwerte und überprüfen, ob die Überlagerung der Hashwerte der erhaltenen Nachrichten mit dem für sie bestimmten Referenzwert überein-

[3] Für die Berechnung von Hashwerten werden kryptographisch starke Hashfunktionen eingesetzt.

stimmt. Nur in diesem Fall bearbeiten sie den Schub weiter. Andernfalls wird der gesamte Schub gelöscht, da ein „*n*-1"-Angriff stattgefunden haben könnte.

Bei Schritt 2 ist zu beachten, dass es nicht genügt, wenn die Teilnehmer den Hashwert nur für genau den Mix $M_i$ verschlüsseln, für den er bestimmt ist. Sollte dieser Mix mit dem Angreifer zusammenarbeiten, könnte er die Hashwerte der Nachrichten seines Schubes mit den in Schritt 3 gesendeten (da der Angreifer alle Netzverbindungen überwacht) vergleichen und so die Nachrichten den Teilnehmern zuordnen.

Um dies zu verhindern, muß sichergestellt werden, dass jeder Mix nur die Überlagerungssumme aller Hashwerte erfährt. Dies wird erreicht, indem jeder Teilnehmer den für Mix $M_i$ bestimmten Hashwert sequentiell für alle Mixe $M_1$, ..., $M_i$ verschlüsselt. Ist einer der Mixe $M_1$, ..., $M_{i-1}$ vertrauenswürdig, fehlt dem Angreifer die Zufallszahl dieses Mixes, um den Hashwert des Teilnehmers entschlüsseln zu können. Ist keiner der Vorgänger-Mixe vertrauenswürdig, kennt der Angreifer die Zuordnung sowieso.

Die Integrität wird mit diesem Verfahren wie gewünscht geschützt: Möchte der Angreifer eine Nachricht aus dem Schub von Mix $M_i$ entfernen, um einen Angriff durchzuführen, muss er den Hashwert dieser Nachricht aus dem Referenzwert entfernen, um eine Angriffswarnung zu vermeiden. Da der Angreifer den Hashwert nicht kennt, wird ein Versuch mit der Wahrscheinlichkeit

$$p = 1 - \frac{1}{2^l} \text{ ; } l \text{ ... Länge des Hashwertes}$$

zu einer Angriffswarnung führen.

Im folgenden wird die vollständige Hashwertmethode beschrieben und deren Sicherheit bewiesen.

### 4.3.1 Verfahrensbeschreibung

Wie bereits beschrieben, muß sich jeder Teilnehmer $T_k$ vor Nutzung des Dienstes bei jedem Mix $M_i$ der $m$ Mixe der Kaskade anmelden. Dies geschieht über eine einmalige Anmeldeprozedur, bei der sich Teilnehmer und Mix gegenseitig mit Hilfe digitaler Zertifikate authentifizieren. Unter der Annahme, dass es einem Angreifer nicht möglich ist, unbemerkt Anmeldungen zu verhindern, braucht nicht überprüft zu werden, ob sich ein Teilnehmer (Angreifer) mehrfach anmeldet.

Sind die Partner authentisiert, werden die Parameter für das Pseudo-One-Time-Pad vertraulich ausgetauscht. Ein Aufruf des Pseudozufallszahlengenerators wird im folgenden mit $PZG(T_k)$ bzw. $PZG(M_i)$ symbolisiert. Dabei gibt $M_i$ bzw. $T_k$ an, mit welchem anderen Teilnehmer resp. Mix die Startwerte ausgetauscht wurden. Mit der Anmeldung vereinbaren Teilnehmer und Mix, dass der Teilnehmer ab einem bestimmten Schubindex $s'$ in jedem Schub eine Nachricht sendet.

Im folgenden wird die obige Schrittfolge detailliert beschrieben:

1. In jedem Schub $s \geq s'$ generiert jeder Teilnehmer $T_k$ eine Nachricht $N_g$ und berechnet die $m$ Hashwerte $h_{s,M_i,N_g}$. Bei den folgenden Erläuterungen wird auf den Schubindex $s$ verzichtet, da nur jeweils ein einzelner Schub betrachtet wird.

2. Für jeden Hashwert führt der Teilnehmer $T_k$ folgende Verschlüsselung mit den Pseudo-One-Time-Pads aus

$$\underset{i \le m}{\forall} u_{M_i,T_k} = h_{M_i,T_k} \oplus \left( \underset{j \le i}{\oplus} PZG(M_j) \right) \tag{7}$$

und sendet die Nachricht *N* zusammen mit den *m* berechneten Werten *u* an den ersten Mix der Kaskade:

$$N_g, u_{M_1,T_k}, ..., u_{M_m,T_k}$$

3. Der erste Mix führt eine bitweise Überlagerung der von allen Teilnehmern erhaltenen Werte $u_{M_i,T_k}$, die für denselben Mix $M_i$ bestimmt sind, durch:

$$\underset{i \le m}{\forall} v_{M_i,0} = \underset{T_k \in T}{\oplus} u_{M_i,T_k}$$

Die Ergebnisse verwendet der erste Mix im Schritt 4 weiter. Jeder folgende Mix $M_i$ erhält vom Mix $M_{i-1}$ die Überlagerungssummen $v_{M_j,M_{i-1}}$ für alle $j \ge i$. Der zweite Parameter gibt dabei an, bis zu welchem Mix die Überlagerungssumme $v_{M_j,0}$ bereits entschlüsselt wurde.

4. Jeder Mix führt nun folgende Schritte aus, um den Schub zu bearbeiten:

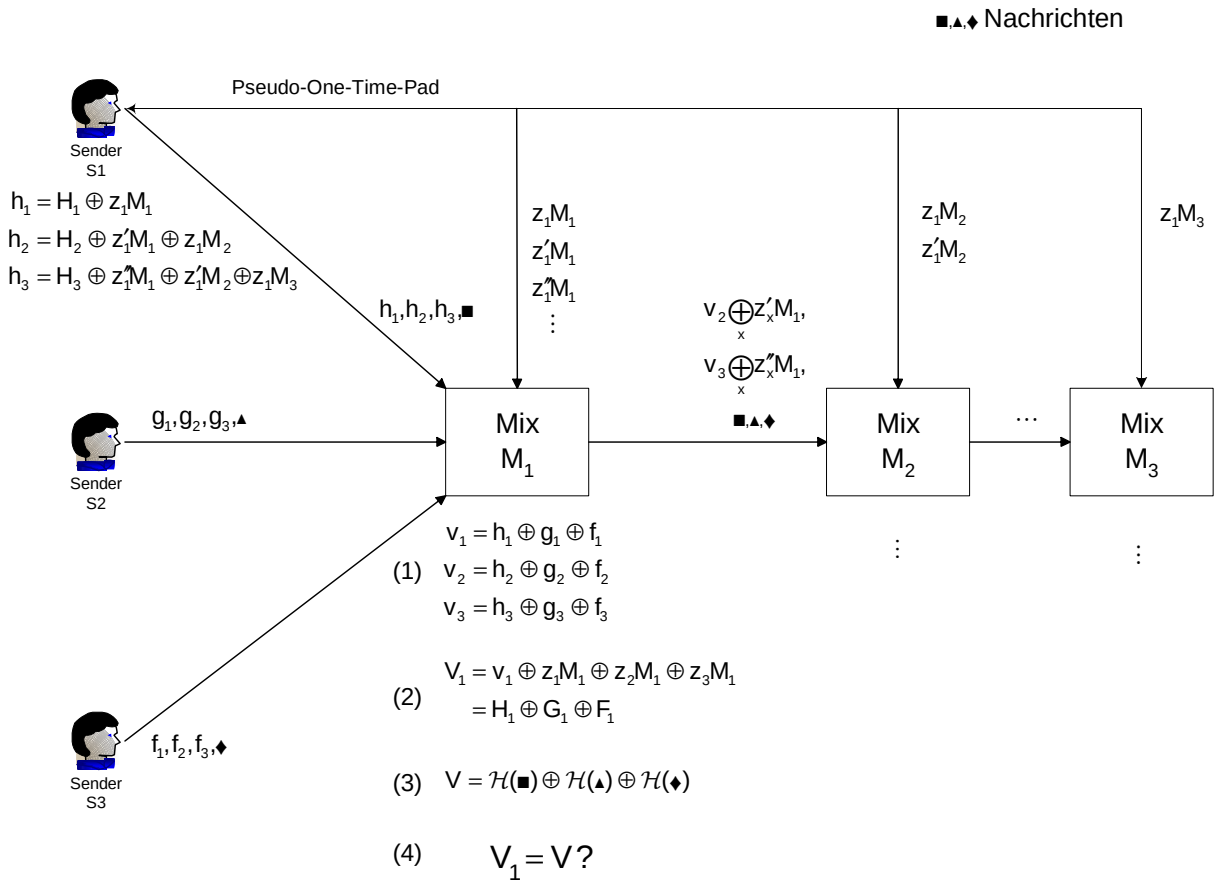
- Überlagerung der erhaltenen Überlagerungssummen $v_{M_j,M_{i-1}}$ mit den Zufallszahlen aller Teilnehmer *T* für jeweils den gleichen (Mix-)Index *j*. Im Ergebnis entstehen die Überlagerungssummen $v_{M_j,M_i}$ für alle Mixe $j \ge i$, die bis Mix $M_i$ entschlüsselt worden sind.

$$\underset{j \ge i}{\forall} v_{M_j,M_i} = \left( \underset{k \in T}{\oplus} PZG(T_k) \right) \oplus v_{M_j,M_{i-1}}$$

- Der Mix entschlüsselt die im Schub *s* enthaltenen Nachrichten und berechnet die Überlagerung der Hashwerte der Nachrichten. Diese Überlagerung vergleicht der Mix mit $v_{M_i,M_i}$. Ist der Vergleich negativ, löscht er alle Nachrichten und alle Überlagerungssummen *v*.

$$v_{M_i,M_i} = \underset{N_g \in N}{\oplus} h_{M_i,N_g} \; ?$$

- Fällt der Vergleich positiv aus, sendet der Mix *i* den Schub entsprechend der normalen Mixfunktionalität zum Mix $M_{i+1}$. Zusätzlich sendet der Mix $M_i$ mit dem Schub alle $v_{M_j,M_i}$ mit $j > i$ zum Mix $M_{i+1}$. Ist $i = m$, werden die bearbeiteten Nachrichten zu den Empfängern gesendet.




**Schematische Darstellung der Hashwertmethode.** Die Zufallszahl $z_xM_y$ wird für die Pseudo-One-Time-Pad Verschlüsselung zwischen Sender $S_x$ und Mix $M_y$ verwendet. $H_y$ ist der von Sender $S_1$ für Mix $M_y$ bestimmte Hashwert ($G_y$ und $F_y$ entsprechend für Sender $S_2$ bzw. $S_3$). $\mathcal{H}$ bezeichnet die Hashfunktion.

Hat jeder Mix der Kaskade alle Bearbeitungsschritte ausgeführt, wurden alle Nachrichten des Schubes anonym an die adressierten Empfänger gesendet. Im folgenden Abschnitt wird bewiesen, daß den vertrauenswürdigen Mix entweder alle Nachrichten und Überlagerungssummen *v* unmanipuliert (abgesehen von protokollgemäßen Entschlüsselungen) verlassen oder der Schub überhaupt nicht ausgegeben wird. Zusätzlich wird bewiesen, dass durch die Hashwertmethode keine zusätzlichen Möglichkeiten zur Deanonymisierung entstehen.

## 4.3.2 Sicherheitsbeweis

Zu Beginn des Abschnittes 4.3 wurde erwähnt, dass der Erfolg eines „*n*-1“-Angriffs auf eine Mixkaskade verhindert werden soll, indem jeder Mix erkennen kann, wenn eine Nachricht des Schubs verändert oder entfernt wurde.

### 4.3.2.1 Entfernen/Manipulieren einer Nachricht

Da jeder Mix im Schritt 3 (Abschnitt 4.3.1) die Überlagerung der Hashwerte der Schub-Nachrichten mit der Überlagerungssumme $v_{M_i,M_i}$ vergleicht, kann der Angreifer eine Nachricht nur dann unerkannt aus dem Schub *entfernen*, wenn er aus $v_{M_i,M_i}$ bzw. $v_{M_i,M_{i-1}}$ den Hashwert der Nachricht entfernt oder eine andere Nachricht mit gleichem Hashwert einspielt.

Da kryptographisch starke Hashfunktionen verwendet werden, ist es nicht effizient möglich, zwei unterschiedliche Nachrichten mit dem selben Hashwert zu finden.

Aus dem gleichen Grund ist es nicht möglich, eine Nachricht zu *manipulieren*, ohne dass sich deren Hashwert ändert. Will der Angreifer eine Nachricht unbemerkt manipulieren, muss er den Hashwert der alten Nachricht aus der Überlagerungssumme entfernen und den der geänderten Nachricht einfügen. Dies zu erreichen ist mindestens genauso schwer wie das Entfernen einer Nachricht, da der Hashwert der alten Nachricht in jedem Fall ermittelt werden muss. Es genügt deshalb zu beweisen, dass das Entfernen einer Nachricht dem Angreifer nicht möglich ist.

Um den Hashwert aus der Überlagerungssumme zu entfernen, muß der Angreifer diesen kennen. Im folgenden wird bewiesen, dass der Angreifer den Hashwert nicht kennen kann. Für den Beweis wird ein stärkerer Angreifer vorausgesetzt als in anderen Abschnitten des Papiers.

**Voraussetzung:**

1. Der Angreifer sendet *n*-1 Nachrichten selbst; ihm ist nur die zu überwachende Nachricht *x* nicht bekannt.
2. Der Angreifer kontrolliert alle Datenleitungen.
3. Nur der Mix $M_i$ ist vertrauenswürdig.
4. Die kryptographischen Funktionen können nicht gebrochen werden.

**Behauptung:** Der Angreifer kann den Hashwert $h_{M_i,N_g}$ nicht ermitteln und deshalb nicht aus der Überlagerungsumme $v_{M_i,M_i}$ entfernen.

**Beweis:**

Es gibt nur zwei Möglichkeiten, den Hashwert $h_{M_i,N_g}$ einer dem Angreifer unbekannten Nachricht $N_g$ zu ermitteln:

1. *Der Angreifer könnte versuchen den Hashwert aus der Nachricht zu ermitteln:* Da der Hashwert jedoch erst über die vom Mix $M_i$ **entschlüsselte** Nachricht gebildet wird, müsste der Angreifer zuerst diese Entschlüsselung der Nachricht durchführen und dann den Hashwert bilden. Die Entschlüsselung der für Mix $M_i$ bestimmten Nachricht $N_g$ ist jedoch nach Voraussetzung (Punkt 4) dem Angreifer nicht möglich.

2. *Der Angreifer könnte versuchen, den Hashwert $h_{M_i,T_k}$ aus den bekannten, vom Teilnehmer ausgegebenen Daten zu berechnen:*

   Da der Angreifer den Teilnehmer $T_k$ nicht kontrolliert, kennt er nur dessen Ausgabewert $u_{M_i,T_k}$, in welchem $h_{M_i,T_k}$ überlagert enthalten ist:

   $$u_{M_i,T_k} = h_{M_i,T_k} \oplus PZG(M_i) \oplus \left( \bigoplus_{j<i} PZG(M_j) \right)$$

   Von diesem Wert kennt der Angreifer nur die mit den (Angreifer-)Mixen $M_j$ mit $j \neq i$

ausgetauschten Zufallszahlen (den letzten Term der Formel). Da in die Berechnung von $u_{M_i,T_k}$ ein dem Angreifer unbekanntes Element $PZG(M_i)$ einfließt, kann $u_{M_i,T_k}$ jede beliebige Bitfolge ergeben, unabhängig vom Wert von $h_{M_i,T_k}$. Der Angreifer kann durch die Auswertung von $u_{M_i,T_k}$ somit keine Informationen über $h_{M_i,T_k}$ erhalten.

Somit folgt, dass der Angreifer weder den Hashwert $h_{M_i,N_g}$ anhand der Nachricht $N_g$ noch $h_{M_i,T_k}$ aus den zusätzlichen Ausgaben des Teilnehmers $T_k$ berechnen kann. Damit ist es dem Angreifer unmöglich, die Nachricht $N_g$ unbemerkt aus dem Schub zu entfernen.

#### 4.3.2.2 Zuordnung Nachricht – Teilnehmer nach dem Mix $M_i$

Im letzten Abschnitt wurde gezeigt, dass die Hashwertmethode aktive Angriffe verhindert. In diesem Abschnitt soll nun bewiesen werden, dass durch die Hashwertmethode keine neuen passiven Angriffe durch die Auswertung der mitgesendeten Daten möglich sind.

Es soll weiterhin von der Annahme ausgegangen werden, dass Mix $M_i$ der einzige vertrauenswürdige Mix der Kaskade ist. Bewiesen wird, dass die Zuordnung zwischen Nachricht und Teilnehmer für die Mixe $M_j$ mit $j > i$ und auf allen Datenleitungen *nach* Mix $M_i$ verborgen ist.

Jeder Mix $M_j$ kennt $h_{M_j,N_g}$ und $u_{M_j,T_k}$ und alle Elemente von $PZG(M_b)$ (bezüglich aller Mixe $M_b \neq M_i$) und aller Teilnehmer $T_k$. Dem Angreifer sind zusätzlich die Elemente von $PZG(M_i)$ bei von ihm kontrollierten Teilnehmern bekannt.

Das Ziel ist es, herauszufinden, in welchem $u_{M_j,T_k} = h_{M_j,N_g} \oplus \left( \bigoplus_{b<j} PZG(M_b) \right)$ das gesuchte $h_{M_j,N_g}$ enthalten ist.

In jedem $u_{M_j,T_k}$ eines vertrauenswürdigen Teilnehmers wird aber ein $PZG(M_i)$ überlagert, welches der Angreifer nicht kennt. Man kann nun für *jeden* vertrauenswürdigen Teilnehmer $T_k$ ein $PZG(M_i)$ berechnen, mit welchem der gesuchte Hashwert $h_{M_j,N_g}$ in die obige Formel zur Berechnung des bekannten $u_{M_j,T_k}$ eingehen würde:

$$(*) \qquad PZG(M_i) = u_{M_j,T_k} \oplus h_{M_j,N_g} \oplus \left( \bigoplus_{b \leq j \wedge b \neq i} PZG(M_b) \right)$$

Es ist folglich nicht möglich, allein mit den oben genannten Informationen die Zuordnung zu ermitteln.

Die unbekannten $PZG(M_i)$ der vertrauenswürdigen Teilnehmer werden aber vom Mix $M_i$ aus der Überlagerungssumme $v_{M_j,M_{i-1}}$ entfernt, so dass im Ausgabewert $v_{M_j,M_i}$ des Mixes $M_i$

keine unbekannten Zufallszahlen mehr enthalten sind. Da jedoch die Hashwerte aller Nachrichten überlagert ausgegeben werden, ist die Zuordnung zwischen $h_{M_j,N_g}$ und Teilnehmer $T_k$ nicht mehr möglich.

Dabei ist es unerheblich, ob dem Mix $M_i$ die korrekte Überlagerungssumme $v_{M_j,M_{i-1}}$ vorgelegt wird oder nicht. Der vertrauenswürdige Mix $M_i$ wird immer nur die Überlagerung der $PZG(M_i)$ aller Teilnehmer *T* mit dem erhaltenen Wert überlagern, weshalb für einen Angreifer die Zufallszahlen $PZG(M_i)$ nicht einzeln ermittelbar sind. Anders ausgedrückt ist es für jede Kombination zwischen Nachrichten und Teilnehmern möglich, entsprechende Zufallszahlen für die Teilnehmer zu ermitteln (nach ähnlicher Formel wie (*)), deren Überlagerung den gleichen Wert ergibt, den auch der Mix auf $v_{M_j,M_{i-1}}$ überlagert hat.

Damit folgt, daß die zusätzlich von den Teilnehmern gesendeten Daten für die Ermittlung der Zuordnung zwischen Nachricht und Teilnehmer durch einen Angreifer wertlos sind. Vor dem vertrauenswürdigen Mix $M_i$ werden die Daten durch bitweise Überlagerung mit Zufallszahlen verschlüsselt und nach diesem Mix durch die Tatsache, dass die Entschlüsselung nur für den Schub insgesamt durchgeführt wird.

### 4.3.3 Aufwandsbetrachtung

Sowohl der Kommunikations- als auch der Berechnungsaufwand für einen Teilnehmer steigt linear mit Anzahl der Mixe. Allerdings muß festgestellt werden, daß dieser Aufwand sehr gering ist, wenn ein effizienter Zufallszahlengenerator und ein effizientes Hashverfahren verwendet wird. Da für die Hashwerte Größen von $50\,\text{Bit}$ durchaus genügen, ist der Kommunikationsaufwand mit $m \cdot 50\,\text{Bit}$ sehr gering.

Für die Mixe ist der Berechnungsaufwand linear abhängig von der Anzahl der Teilnehmer (und von der Anzahl der Mixe). Der Übertragungsaufwand ist dagegen unabhängig von der Anzahl der Teilnehmer.

# 5 Zusammenfassung und Vergleich

Die bekannten auf Broadcast basierenden Verfahren zur Verhinderung von „*n*-1“-Angriffen sind sehr aufwendig und verursachen wesentliche zusätzliche Verzögerungszeiten. Aus diesem Grund wurden in diesem Paper weitere Verfahren vorgestellt und bewertet, die weniger Nachteile haben und als Nebeneffekt eine Abrechnung der Mixdienstleistung ermöglichen.

Die Ticketmethode ist zwar noch sehr aufwendig, benötigt jedoch deutlich weniger Ressourcen als die auf Broadcast basierenden Systeme. Die Verarbeitung der Nachrichten wird praktisch nicht zusätzlich verzögert.

Gegen „*n*-1“-Angriffe bietet die Hashwertmethode u.U. einen höheren Schutz als die Ticket-Methode, da der Angreifer keine einzige Nachricht blockieren kann und so selbst sog. Schnittmengenangriffe [Bert99], die schon beim Blockieren einiger Nachrichten erfolgreich sind, verhindert werden. Bei der Ticketmethode ist dies im Prinzip auch erreichbar, wenn je-

der Mix einen Schub nur unter der Voraussetzung akzeptiert, dass er genau so viele Nachrichten enthält, wie er vorher Tickets ausgegeben hat.

## Literatur


BaNe99 Matthias Baumgart, Heike Neumann: Bezahlen von Mix-Netz-Diensten. Verläßliche IT-Systeme, GI-Fachtagung VIS '99, DuD Fachbeiträge, Vieweg, Braunschweig 1999, 19-33.

Bert99 Oliver Berthold: Effiziente Realisierung von Dummy Traffic zur Gewährleistung von Unbeobachtbarkeit im Internet. Diplomarbeit, TU Dresden, Institut für Theoretische Informatik, Dezember 1999.

Chau81 David Chaum: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communication of the ACM 24/2 (1981) 84-88.

Chau83 David Chaum: Blind Signature System. Crypto ’83, Plenum Press, New York 1984, 153.

Cott95 Lance Cottrel: Mixmaster & Remailer Attacks. http://www.obscura.com/~loki/remailer-essay.html.

FrJe98 Elke Franz, Anja Jerichow: A Mix-Mediated Anonymity Service and Its Payment. ESORICS '98 (5th European Symposium on Research in Computer Security), LNCS 1485, Springer-Verlag, Berlin 1998, 313-327.

KeBS99 Dogan Kesdogan, Roland Büschkes, Otto Spaniol: Stop-And-Go-MIXes Providing Probabilistic Anonymity in an Open System. in: Günter Müller, Kai Rannenberg (ed.): Multilateral Security in Communications, Vol. 3: Technology, Infrastructure, Economy; Addison-Wesley, München 1999, 365-380.