Universität Hamburg
Fachbereich Informatik

Das cMix-Verfahren

am Arbeitsbereich Sicherheit in Verteilten Systemen (SVS)

Maik Graaf, Merlin Koglin

14. Juni 2016

Proseminar

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit beherrschen Nachrichten, welche über das Internet und insbesondere auch über mobile Endgeräte versendet werden unsere Kommunikation. Zunehmend fragen sich die Benutzer, ob ihre Anonymität bei Benutzung von Mails und Messengern gewährleistet ist. Es gibt in der IT-Sicherheit Verfahren, die diese Anonymität des Nutzers sicherstellen können, unter anderem das cMix Verfahren, mit welchem wir uns in der vorliegenden Ausarbeitung auseinandersetzen. Dieses wurde von David Chaum konzipiert und ist eine Weiterentwicklung der ebenso von ihm entwickelten Chaumschen Mixe aus dem Jahr 1981.

Da die Chaumschen Mixe und bisherige Mixnetz Implementationen entweder nicht effizient genug für eine verzögerungsarme Kommunikation bzw. Datenübertragung sind oder diese nur mit eingeschränkter Sicherstellung der Anonymität anbieten, wurde das cMix Verfahren entwickelt, d.h. um einen Ansatz zu bieten, der effizient und sicher zugleich ist. Wir wollen uns in dieser Ausarbeitung näher damit beschäftigen, inwiefern genannte Probleme mit dem cMix-Verfahren angegangen werden und ob hierbei neue Probleme entstehen. Die Ausarbeitung orientiert sich dabei am Paper [CJK+16] von David Chaum et al.

2 Chaumsche Mixe

2.1 Verfahren

Die Chaumsche Mixe gewährleisten die Anonymität der Kommunikation, indem die zu verschickenden Nachrichten mehrere Stationen - sogenannte Mixe - durchlaufen, welche dafür sorgen, dass Nachrichten, Empfänger, sowie Sender nicht zueinander in Beziehung gesetzt werden können.

Zuerst werden mehrere Nachrichten gesammelt und diese dann zusammen an das Mixnetz weitergegeben. Dabei sind die Nachrichten für jeden Mix mit dessen zugehörigen öffentlichen Schlüssel verschlüsselt, es existiert also eine Art „Zwiebelschale“ von Verschlüsselungen. Nun kann jeder Mix nacheinander seine Verschüsselungsschicht entfernen, die Anordnung der Nachrichten vermischen und an den nächten Mix weiterreichen.

Durch diese Methode des Mischen gibt es zum Schluss keine Möglichkeit mehr, eine Beziehung zwischen Eingangs und Ausgangsnachrichten zu finden.

Mithilfe einer anonymen Rückadresse, welche als Teil einer Nachricht gesendet wird und keinen Rückschluss auf die tatsächliche Adresse des Senders zulässt, können sich zwei Nutzer nun gegenseitig Nachrichten senden und dabei anonym bleiben. [Cha81] [SP06]

2.2 Probleme

Die Chaumsche Mixe haben durchaus ihre Grenzen. Innerhalb eines Echtzeitsystems ist das Sammeln der Nachrichten sehr ineffizient, da man durchaus lange warten muss um eine bestimmte Anzahl an Nachrichten zu erreichen. Deshalb wird in solchen Echtzeitsystemen das Sammeln der Nachrichten weggelassen oder kurz gehalten. Daraus resultiert, dass die Sicherheit sinkt und das ganze Verfahren in Echtzeitsystemen somit angreifbarer wird.

Ein weiteres Problem ist, dass in Echzeit aufwändige Ver- und Entschlüsselungen berechnet werden. Je nach Schlüssellänge und Anzahl der Nachrichten entsteht hier ein großer Zeit- und Energieaufwand, sodass bisherige Verfahren für mobile Geräte ungeeignet sind. [Cha81] [SP06]

3 Das cMix Verfahren

3.1 Idee

Die grundsätzliche Idee des cMix Verfahrens ist es, Schlüsselberechnungen in Echtzeit zu vermeiden. Hierdurch entsteht auf der einen Seite eine Steigerung der Effizienz von Mix-Netzen, d.h. bessere Performance, also weniger verzögerte Kommunikation.
Auf der anderen Seite wird der Energiebedarf verringert, was z.B. zu längerer Akkulaufzeit eines Smartphones führen kann. Um dies zu erreichen werden vor der eigentlichen Kommunikation Schüssel berechechnet (Precomputation) und zwischen dem Sender und den Mixknoten ausgetauscht. Diese werden dann als Seed für einen Pseudozufallsgenerator verwendet, um weitere (gleiche) Schlüssel zu erzeugen.

Folgendes Funktionprinzip und die Analyse stammt aus [CJK+16].

3.2 Funktionsprinzip

3.2.1 Übersicht

Seien m die Anzahl der Nutzer des cMix-Systems mit n Mixknoten $N_1, N_2, ..., N_n$.

Sei T ein Netzwerkknoten, welcher eingehende Nachrichten in Bündeln sortiert und zur weiteren Kommunikation dient.

β sei die Anzahl der Nachrichten, die ein Knoten gleichzeitig verarbeiten kann, wobei $\beta \geq m$ gilt.

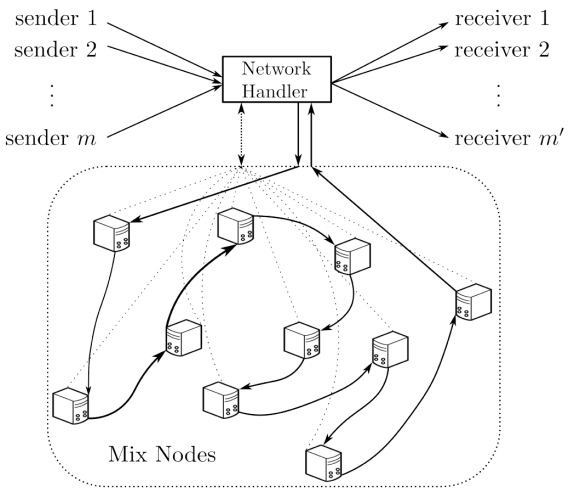

Abbildung 3.1: Übersicht der Kommunikation

3.2.2 Pre-Communication

Vor der Benutzung des System muss jeder Nachrichtensender jeweils einen symmetrischen Schlüssel mit jedem der Mixknoten austauschen. Für jeden Knoten N_i und jeden Nutzer U_j sei dieser Schlüssel $MK_{i,j}$.
Dieser Schlüssel wird z.B. mit Hilfe des Diffie-Hellman Verfahrens ausgetauscht.

Bei Kommunikation mit dem Mixnetz ver- oder entschlüsselt jeder Teilnehmer jede seiner Nachrichten mit Schlüsseln, die aus den gemeinsamen Schlüsseln $MK_{i,j}$ erzeugt werden. Genauer gesagt werden diese Schlüssel als Ausgabe eines Pseudozufallszahlengeneratoren berechnet, welche bei gleichem Startwert die gleiche Zahlenfolge generieren.
Als Startwert werden hier die Schüsselpaare $MK_{i,j}$ verwendet, sodass Nutzer und Mixknoten jeweils die gleichen Keys erzeugen. Die pseudozufallsgenerierten Schlüssel sind als $ka_{i,j}$ bezeichnet. Zum Verschlüsseln einer Nachricht berechnet der Nutzer aus den Schlüsseln $MK_{i,j}$ einen zusammengesetzten Schüssel: $Ka_j = \prod_{i=1}^{n} ka_{i,j}$, dann kann die Nachricht M_j durch $M_j \times Ka_j^{-1}$ verschlüsselt werden.

Außerdem legt jeder Mixknoten eine zufällige Permutation P_i fest, die später zum Mixen verwendet wird.

cMix verabeitet jedes Bündel von Nachrichten in zwei Phasen:

- Vorberechungsphase (precomputation)
- Echzeitphase (real-time)

Für den Hinweg von Nachrichten gibt es drei Phasen, für den Rückweg - also eine Antwort - zwei Phasen.

3.2.3 Precomputation Phase

In der Vorberechnungsphase werden die Werte und Schlüssel berechnet, die später für die Echtzeitphase benötigt werden.

Vorwärts

1. *Preprocessing*
   Für die Precomputation wird als erster Schritt von jedem Mixknoten N_i eine zufälliger Wert $r_{i,j}$ für jede spätere Nachricht M_j generiert. Jeder Mixknoten verschlüsselt mittels Elgamal-Verschlüsselungsverfahren das Inverse des jeweiligen Wertes, also $r_{i,j}^{-1}$ und sendet das Resultat $\mathscr{E}(r_i^{-1})$ an den Netzwerknoten T.
   Der Netzwerknoten T berechnet das komponentenweise direkte Produkt $\mathscr{E}(R_n^{-1}) = \prod_{i=1}^{n} \mathscr{E}(r_i^{-1})$ der verschlüsselten Vektoren $\mathscr{E}(r_i^{-1})$ und sendet das Resultat an den ersten Mixknoten.
2. *Mixing*
   Im zweiten Schritt der Precomputation permutiert jeder Mixknoten N_i nacheinander das direkte Produkt mit der anfangs festgelegten Permutation P_i, es wird ein weiterer Vektor S_i^{-1} mit zufälligen Werten hinzumultipliziert und alles an den nächsten Mixknoten gesendet.

Der letzte Mixknoten hat also den Output $\mathscr{E}((P_n(R_n) \times S_n)^{-1})$, wobei P_n aus den Kompisitionen aller P_i's besteht und S_n das direkte Produkt der S_i's darstellt. Dieser Output wird wiederum an den Netzwerkknoten gesendet.

3. *Postprocessing*
   Dann wird im dritten Schritt der Precomputation von jedem Mixknoten N_i der Entschüsselungsteil $D(i,x)$ aus $\mathscr{E}((P_n(R_n) \times S_n)^{-1})$ berechnet.

   Hierfür wird eine Methode verwendet, die auf der ElGamal Verschlüsselung basiert, Eigenschaften von Gruppenhomomorphismen verwendet und hier nicht weiter behandelt wird. Der interessierte Leser kann in [Ben06] mehr dazu erfahren.

   Um die Entschlüsselung durchzuführen, müsste ein Knoten alle $D(i,x)$ kennen.

Rückwärts

1. *Mixing*
   Der Rückweg funktioniert ähnlich wie der Hinweg, allerdings beginnt hier der letzte Mixknoten mit dem Mixing und die Permutationen sind alle umgekehrt. Außerdem wird hier kein zufälliger Wert $r_{i,j}$ mit einbezogen.

2. *Postprocessing*
   Auch für den Rückweg wird wieder ein Entschlüsselungsteil berechnet.

3.2.4 Echtzeit Phase

In der Echtzeitphase generiert nun jeder Nutzer U_j aus dem Schlüssel $MK_{i,j}$ für jeden Mixknoten M_i einen Schlüssel $ka_{i,j}$ und berechnet dadurch das Produkt: $Ka_j = \prod_{i=1}^{n} ka_{i,j}$.
Dann sendet jeder Nutzer $M_j \times Ka_j^{-1}$ an den Netzwerkknoten, welcher diese zu $M_j \times Ka^{-1}$ kombiniert.

Vorwärts

1. *Preprocessing*
   Als erster Schritt sendet jeder Mixknoten N_i das Produkt aus den gemeinsam genutzten Schlüsseln ka_i und seinen zufällig generierten Werten als Vektor r_i an den Netzwerkknoten. Dieser multipliziert diese Werte dann mit den Nachrichten von den Sendern. Damit wird aus $M \times Ka^{-1}$ das Produkt $M \times R = M_j \times Ka^{-1} \times \prod_{i=1}^{n} ka_i \times r_i$.

2. *Mixing*
   Im zweiten Schritt der Echzeit-Phase permutiert jeder Mixknoten N_i nacheinander die Nachrichten mit der festgelegten Permutation p_i und multipliziert den in der Vorberechnungsphase - Schritt 2 generierten Vektor S_i von zufälligen Werten s_i hinzu, also entsteht $P_i(M \times R_n) \times S_i$.
   Das Resultat des letzten Mixknotens ist $P_n(M \times R_n) \times S_n$. Dieses wird an den Netzwerkknoten gesendet.

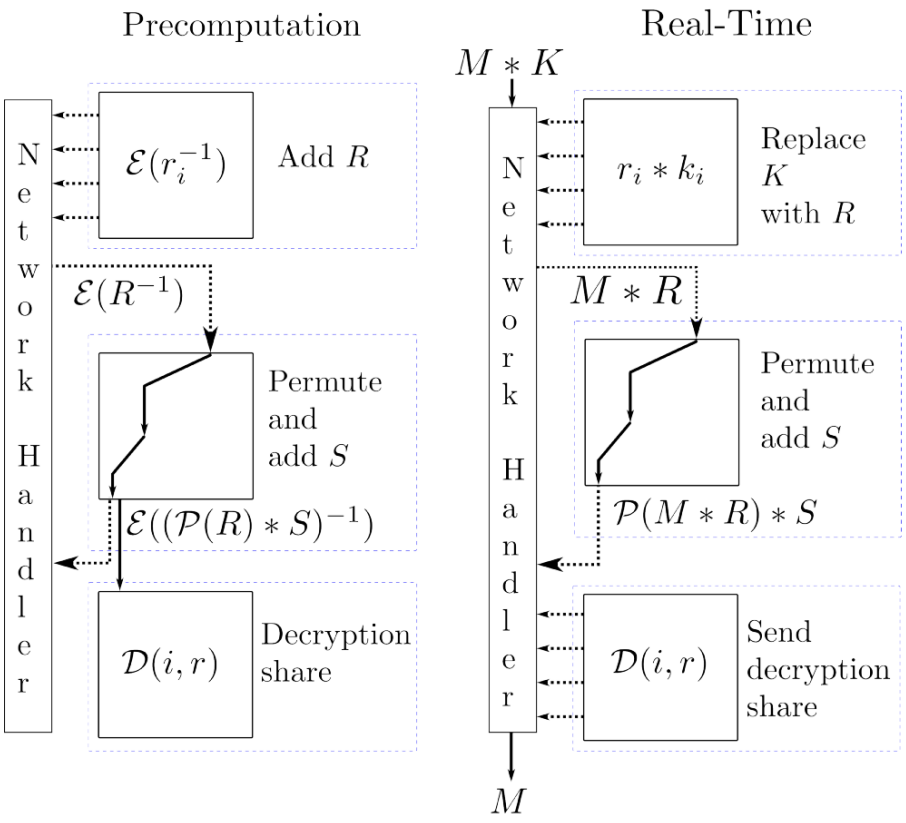

Abbildung 3.2: cMix Overview - Hinweg

3. *Postprocessing*
   Im letzten Schritt des Hinwegs, gibt jeder Mixknoten seinen Entschüsselungsanteil $D(i,x)$ an den Netzwerkknoten, welcher dadurch die Nachrichten entschlüsseln, aber nicht mehr mit einem Sender verknüpfen kann.

$$\begin{aligned} & P_n(M \times R_n) \times S_n \times \prod_{i=1}^{n} D(i,x) \\ = \; & P_n(M \times R_n) \times S_n \times (P_n(R_n) \times S_n)^{-1} \\ = \; & P_n(M) \end{aligned}$$

Rückwärts

1. *Mixing*
   Der Rückweg funktioniert ähnlich wie der Hinweg, nur dass der letzte Mixknoten beginnt, die Permutationen alle umgekehrt sind und hier kein zufälliger Wert $r_{i,j}$ mit einfließt.

2. *Postprocessing*
   Für den Rückweg wird der Entschlüsselungsanteil $D(i,x)$ in die Schlüssel ka'_i mit einberechnet, sodass jeder User U_J aus dem Resultat $P'_n(M') \times Ka'$ mit seinem Schlüssel Ka'^{-1}_j seinen Anteil der Nachricht entschlüsseln kann.

3.3 Analyse

3.3.1 Sicherheit

Chaum et al. untersuchen in ihrer Arbeit über cMix auch die Sicherheit des entwickelten Verfahrens.

Anonymität

Hierfür wird zuerst eine „ideale Welt“ betrachtet, ein Modell, bei der jeder Mixknoten jeweils privat mit den anderen Mixknoten und zusätzlich mit einem vertraulichen dritten Punkt kommunizieren kann.
In diesem Modell gibt es keine kryptografischen Operationen wie Ver- und Entschlüsselung, diese werden hier durch den vertraulichen dritten Punkt sichergestellt.
Dann wird gezeigt, dass dieses Modell die Anonymität des Senders gewährleistet.

Von diesem Modell wird eine „reale Simulation“ abstrahiert, wobei Eigenschaften des Modells durch Eigenschaften des cMix Protokolls ersetzt werden. Damit wird gezeigt, dass cMix das Modell erfüllt und somit Anonymität sicherstellt.

Integrität

Die Integrität ist nach Chaum nur gegeben, wenn von den folgenden Bedingungen eine zutrifft.

- Die Nachricht M wird unmodifiziert an den Empfänger weitergeleitet.
- Alle Mixknoten wissen, dass das cMix Protokoll nicht richtig durchgeführt wurde.

Auf den ersten Punkt wird in diesem Paper nicht weiter eingegangen. Für den zweiten Punkt wird ein Mechanismus mit dem Namen „Randomized Partial Checking“ (vergleiche [JJR]) verwendet, welches eine deterministische Verifikation aller ausgehenden Nachrichten im Vergleich zu deren permutierten Eingangsnachrichten durchführt. Hierbei wird neben der Integrität auch sichergestellt, dass die Permutationen korrekt ausgeführt werden.

Vertraulichkeit

Das cMix Protokoll dient in erster Linie dem Schutzziel *Anonymität*. Damit für die zu übermitteldenen Nachrichten auch *Vertraulichkeit* sichergestellt werden soll, müssen diese vorher vom Sender verschlüsselt werden (z.B. durch einen öffentlichen Schlüssel einer asymmetrischen Verschlüsselung).
Es ist aber auch möglich, diese Verschüsselung ohne großen Rechenaufwand in das cMix Verfahren einzubauen, wodurch wiederum aufwändige Publickey-Operationen vermieden werden.

3.3.2 Performance

Die Performance wird durch Chaum et al. auf der einen Seite analysiert, aber auch mit Hilfe eines entwickelten Protoypens gemessen.

Der Prototyp wurde in Python implementiert und lief während der Tests auf Instanzen des *Amazon Web Service EC2*, wobei für jeden Mixknoten zwei Intel Xeon E5-2680 und 3,75 GB Arbeitsspeicher zur Verfügung standen.
Es wurden 100 Vorberechnungen und verschiedene Echtzeit-Phasen mit bis zu 1000 Nachrichten getestet, wobei eine 1024-bit ElGamal-Verschlüsselung verwendet wurde. Die Tabelle 3.3 zeigt einige gemessene Werte des Tests.

Zu bestehenden Mixnetzen ist dies eine eindeutige Verbesserung, z.B. braucht das re-encryption Mixnet für die Verarbeitung von 1000 Nachrichten bei einer Schlüssellänge von 512-bit bis zu 40 Sekunden, bei einer Schlüssellänge von 1024-bit sogar bis zu 250 Sekunden. [RA12]

Dies ist dadurch zu erklären, dass die aufwändigen Schlüsseloperationen in die Vorausberechnungsphase gelegt werden. In der Echtzeitphase werden nur noch simple Multiplikationen durchgeführt, die für heutige Prozessoren kein großer Aufwand sind.

Anzahl Nachrichten	Vorberechnung (Durchschnitt in Sekunden)	Echtzeit (Durchschnitt in Sekunden)
50	1.56	0.20
100	3.02	0.33
500	14.59	1.51
1000	28.87	3.09

Abbildung 3.3: Messungen anhand eines Prototypens

4 Fazit

Chaum et al. haben ein Mix-Verfahren entwickelt, welches effizient ist und sich damit wahrscheinlich auch auf mobilen Endgeräten, sowie für zeitkritische Anliegen mit hohem Datenfluss, wie Suchen oder Online-Shopping einsetzen lässt.

Für eine Implementierung, die Anonymität zusichert, dürfen die Betreiber der Mixknoten sich untereinander nicht austauschen, da sonst mit allen Schlüsselanteilen die Kommunikation aufgedeckt werden kann. Das Verfahren bietet also Anonymität mit „Hintertür“.
David Chaum sieht den Betrieb der Mixknoten in verschiedenen westlichen Ländern vor, welche in besonderen Fällen kooperieren, die Schlüsselanteile austauschen und damit bestimmte Kommunikationspartner identifizieren können. Auch wenn dies eine Ausnahme sein soll, wird diese Möglichkeit zum Teil sehr kritisch betrachtet. [Gre16]

Abgesehen davon hat Chaums Protoyp im Vergleich zu anderen Mix-Verfahren gute Leistungen gezeigt, durch cMix werden Mixnetze wohl auch zur Kommunikation mit einer sehr geringen Verzögerung eingesetzt werden können.

Literaturverzeichnis

[Ben06] Josh Benaloh. Simple verifiable elections. 2006.

[Cha81] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, February 1981.

[CJK+16] David Chaum, Farid Javani, Aniket Kate, Anna Krasnova, Joeri de Ruiter, and Alan T. Sherman. cmix: Anonymization byhigh-performance scalable mixing. *IACR Cryptology ePrint Archive*, 2016:8, 2016.

[Gre16] Andy Greenberg. The father of online anonymity has a plan to end the crypto war. http://web.archive.org/web/20160402045225/http://www.wired.com/2016/01/david-chaum-father-of-online-anonymity-plan-to-end-the-crypto-wars/, 2016.

[JJR] Markus Jakobsson, Ari Juels, and Ronald L Rivest. Making mix nets robust for electronic voting by randomized partial checking.

[RA12] Pance Ribarski and Ljupcho Antovski. Mixnets: Implementation and performance evaluation of decryption and re-encryption types. *CIT. Journal of Computing and Information Technology*, 20(3):225–231, 2012.

[SP06] Krishna Sampigethaya and Radha Poovendran. A survey on mix networks and their secure applications. *Proceedings of the IEEE*, 94(12):2142–2181, 2006.