Universität Hamburg
Fachbereich Informatik

Das cMix-Verfahren

am Arbeitsbereich Sicherheit in Verteilten Systemen (SVS)

todo

12. Juni 2016

Inhaltsverzeichnis

Abstract / Einleitung

Das ist die Zusammenfassung

In der heutigen Zeit sind Nachrichten, die über das Internet versendet werden, nicht mehr weg zu denken. Zunehmend fragen sich die Benutzer ob ihre Anonymität bei Benutzung von Mails und Messengern gewährleistet wird. Deshalb gibt es in der IT- Sicherheit Verfahren, die Mithilfe von Kryptographischen-Techniken versendete Nachrichten verschlüsseln. Das C-Mix Verfahren, mit welchem wir uns in der vorliegenden Ausarbeitung auseinandersetzen, ist ein solches Verfahren. Es wurde von David Chaum, konzipiert und ist eine Weiterentwicklung der ebenso von ihm entwickelten Chaumschen Mixe aus dem Jahr 1981. Da die Chaumsche Mixe bei heutiger Technik nicht mehr so effizient sind, wurde das C-Mix Verfahren entwickelt. Wir wollen uns in dieser Ausarbeitung näher damit beschäftigen, ob das Problem mit dem C-Mix verfahren gelöst wurden ist und ob das Verfahren neue Probleme bereit hält. Die Ausarbeitung orientiert sich am Paper [CJK+16] von David Chaum et al.

1 Übersicht Chaumsche Mixe

1.1 Funktionsprinzip

Die Chaumsche Mixe gewährleisten die Anonymität der Kommunikation, in dem sie Sender und Empfänger voreinander anonym halten. Dies gelingt, indem die zu verschickenden Nachrichten mehrere Stationen - sogenannte Mixe durchlaufen. Diese sorgen dafür, dass die Nachrichten sowohl Empfänger als auch Sender nicht zueinander in Beziehung gesetzt werden können. Damit dies gelingt haben die Mixe verschiedene Aufgaben. Zum einen muss ein Mix die Nachrichten mithilfe eines Verschlüsselungssystems verschlüsseln und ein anderer später wieder entschlüsseln. Durch diese Methode des Umkodieren ist es nicht mehr möglich eine Beziehung zwischen Eingangs und Ausgangsnachrichten zu finden. Ein anderer Mix muss die Nachrichten sammeln und ein weiterer umsortieren, damit man nicht ausgehend von der Reihenfolge des Eintreffens und Weiterleitens der Nachrichten am Mix eine Beziehung zwischen Sender und Empfänger vorfinden kann. Mithilfe einer Rückadresse, die als Teil einer Nachricht gesendet wird und einem Mix, der diese Rückadresse zwischen speichert und umkodiert können sich zwei Nutzer nun gegenseitig Nachrichten senden und dabei anonym bleiben.[Cha81] [SP06]

1.2 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 mehrere Nachrichten zusammen zu bekommen. Deshalb wird in solchen Echtzeitsystemen das Sammeln der Nachrichten weggelassen oder kurz gehalten. Das Umsortieren der Nachrichten kann deshalb nicht oder nur mit wenig Nachrichten erfolgen. Daraus resultiert, dass die Sicherheit sinkt und das ganze Verfahren in Echtzeitsystemen somit angreifbarer wird.[Cha81] [SP06]

1.3 vorherige Ansätze

2 Das cMix Verfahren

2.1 Idee

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

2.2 Funktionsprinzip

2.2.1 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 weiterer Kommunikation dient.

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

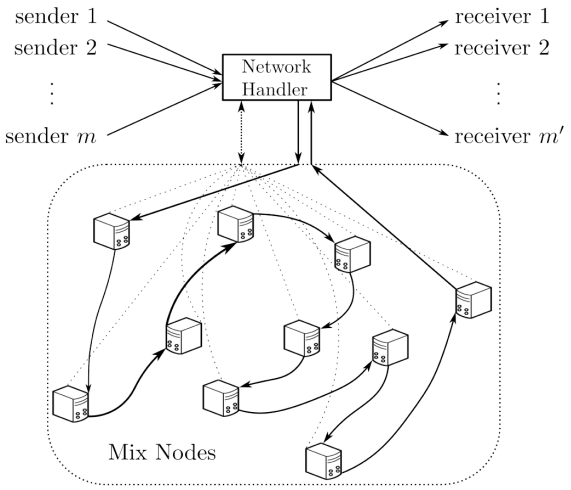

Abbildung 2.1: Communication Overview

Nun gibt es eine Precomputation Phase, bei der die Knoten eine Permutation von β Nachrichten festlegen, diese Permutation wird dann in der Echzeit-Phase zum Mixen verwendet. Die

Echtzeit-Phase wird wiederum in Runden aufgeteilt, wobei jede Runde eine Permutation auf die Nachrichten angewendet wird. [RA12]

2.2.2 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 kann z.B. mit Hilfe des Diffie-Hellman Verfahrens ausgetauscht werden.

Bei Kommunikation mit dem Mixnetz, ver- oder entschüsselt 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, so dass Sender und Node jeweils die gleichen Keys erzeugen. 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_1 durch $M_1 \times Ka_j^{-1}$ verschlüsselt werden.

cMix verabeitet jedes Bündel von Nachrichten in zwei Phasen: precomputation and real-time Während jeder dieser Phasen führt cMix einen bestimmten Hin- und Rückweg an Berechnungschritten aus. Dafür werden die Nachrichten von jedem Knoten in einem Puffer gespeichert (auch map gennant). Bei jeden Schritt werden die Berechnungen in diesem Puffer ausgeführt. Jeder Knoten verbindet jeden gemeinsamen Schlüssel $MK_{i,j}$ mit einem bestimmten Abschnitt im Puffer. Während des Hinwegs der Echzeit-Phase wird jeder gemeinsamer Key für jeden Abschnitt mit den generierten Schlüssen aus der Precomputation Phase ausgetauscht. Auf dem Rückweg werden die gemeinsamen Schlüssel wieder hinein multipliziert. So werden in den Echzeit-Phase rechenintesive public-key Operationen vermieden

Festlegung der Permutation?

2.2.3 cMix Protokoll

Precomputation Phase

1. *Preprocessing*
   Für die Precomputation wird als erster Schritt von jedem Knoten N_i eine zufälliger Wert $r_{i,j}$ für jeden Bereich j im Nachrichtenpuffer generiert. Jeder Mixknoten verschlüsselt mittels Elgamal-Verschlüsselungsverfahren das Inverse des jeweiligen Wertes $r_{i,j}^{-1}$ und sendet das Resultat $\varepsilon(r_i^{-1})$ an den Verteilerknoten. Der Verteilerknoten berechnet das komponentenweise direkte Produkt $\varepsilon R_n - 1$ der verschlüsselten Vektoren $\varepsilon(r_i^{-1})$ und sendet das Resultat an den ersten Mixknoten.
2. *Mixing*
   Im zweiten Schritt der Precomputation permutiert jeder Mixknoten N_i nacheinander den Nachrichten Puffer mit der Anfangs festgelegten Permutation p_i. Außerdem wird ein weiterer Vektor mit zufälligen Werten s_i^{-1} hinzumultipliziert. Der letzte Mixknoten hat also den Output $\varepsilon((P(R) \times S)^{-1})$, wobei P aus den Kompisitionen aller p_i's besteht und S

das direkte Produkt der s_i's darstellt. Dieser Output wird wiederum an den Netzwerkknoten und den ersten Mixknoten gesendet.

3. *Postprocessing*
   Dann wird im dritten Schritt der Precomputation von jedem Mixknoten N_i der Entschüsselungsteil $D(i,r)$ aus $\varepsilon((P(R) \times S)^{-1})$ berechnet. Um die Entschlüsselung durchzuführen, müsste ein Knoten alle $D(i,r)$ kennen.

Echzeit Phase

1. *Preprocessing*
   Als erster Schritt sendet jeder Mixknoten N_i das Produkt aus den gemeinsam genutzen 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 $M \times R$, wobei nur Multiplikation verwendet hier.

2. *Mixing*
   Im zweiten Schritt der Echzeit-Phase ordnet jeder nacheinander Mixknoten N_i seinen Nachrichtenpuffer mit der festgelegten Permutation p_i und multipliziert den in Precomputation Phase, Schritt 2 generierten Vektor von zufälligen Werten s_i hinzu. Das Resultat des letzten Mixknotens ist $P(M \times R) \times S$. Dieses wird an den Netzwerkknoten gesendet.

3. *Postprocessing*
   Im letzten Schritt, gibt jeder Mixknoten seinen Entschüsselungsanteil an den Netzwerkknoten, welcher dadurch die Nachrichten entschlüsseln, aber nicht mehr mit einem Sender verknüpfen kann.

Ergänzungen

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. duch einen öffentlichen Schlüssel einer assymetrischen Verschlüsselung). Es ist aber auch möglich, diese Verschüsselung ohne großen Rechenaufwand in das cMix Verfahren einzubauen, wodurch wiederum aufwändige Publickey-Operationen vermieden werden.

2.3 Analyse

2.3.1 Sicherheit

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

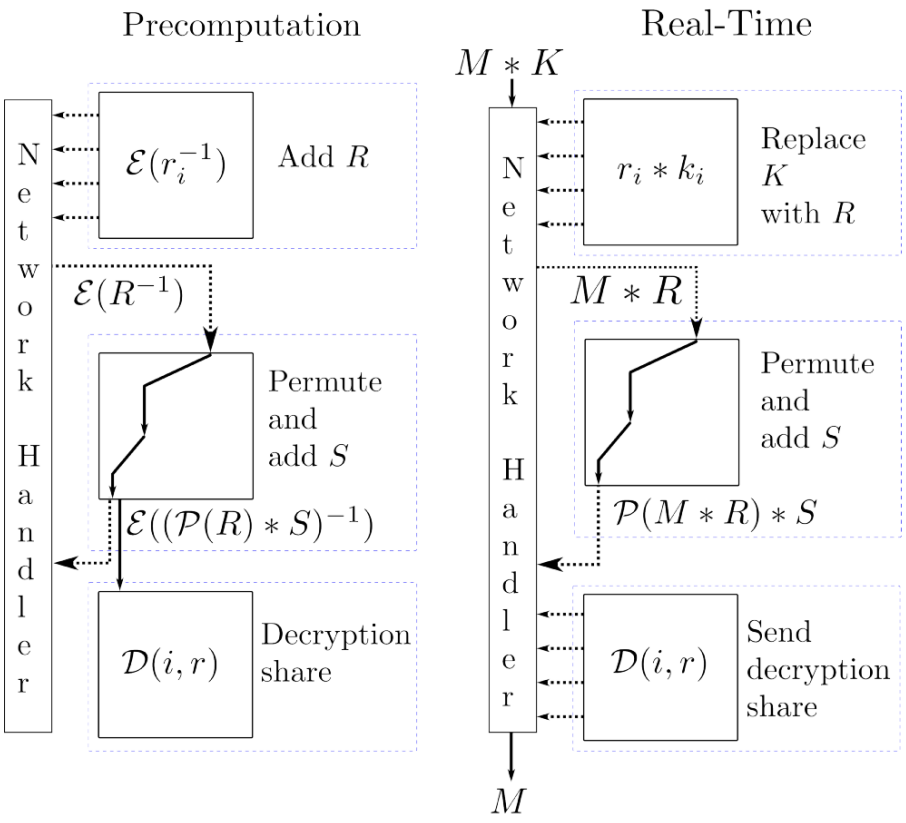

Abbildung 2.2: cMix Overview

Anonymität

Hierfür wird zuerst eine ïdeale 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 Verifkation aller ausgehenden Nachrichten im Vergleich zu deren permutierten Eingangsnachrichten durchführt. Hierbei wird neben der Integrität auch sichergestellt, dass die Permutationen korrekt ausgeführt werden.

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 2.3: Messungen anhand eines Prototypens

2.3.2 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 Echzeit-Phasen mit bis zu 1000 Nachrichten getestet, wobei eine 1024-bit ElGamal-Verschlüsselung verwendet wurde.

Die Tabelle 2.3 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 Schlüsselänge von 512-bit bis zu 40 Sekunden, bei einer Schlüssellänge von 1024-bit sogar bis zu 250 Sekunden.[RA12]

3 Schlussbemerkungen / Fazit

Literaturverzeichnis

[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.

[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.