



# Das c-mix Verfahren

Merlin Koglin, Maik Graaf



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

# Agenda

---

1. Motivation
2. Elgamal-Verschlüsselungsverfahren
3. c-mix Verfahren
4. Sicherheit
5. Performance
6. PrivaTegrity

## Ansatz der Chaumische Mixe

---

- Gewährleistung der Anonymität
  - Mixe werden durchlaufen um die Beziehung zwischen Sender und Empfänger zu verschleiern
  - Im Mixnetz findet eine „Zwiebelschalen“ artige Entschlüsselung statt
  - Mithilfe der Rückadresse wird der Empfänger einer Nachricht bestimmt

## Probleme bisheriger Mix Verfahren

---

- In Echtzeitsystemen
  - Lange warte Zeiten beim Sammeln der Nachrichten
  - Der Sammelschritt wird deshalb kurz gehalten oder sogar weggelassen
  - Das Verfahren wird somit angreifbarer
  - Für mobile Geräte ungeeignet aufgrund des großen Zeit und Energie Aufwands

## Idee von David Chaum

---

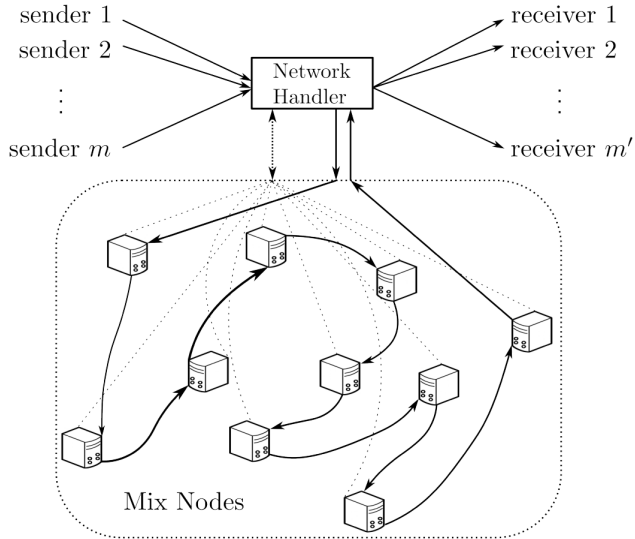
- Vermeidung von Schlüsselberechnungen in Echtzeit
  - Steigerung der Effizienz von Mix-Netzen
  - Energieaufwand verringern
  - Schlüsselberechnung vor der Kommunikation
  - Schlüssel Austausch zwischen Sender und Mixknoten

# Elgamal

---

- Elgamal-Verschlüsselungsverfahren
  - asymmetrisch
  - basiert auf Diffie-Hellman-Schlüsselaustausch
  - diskreter Logarithmus als Einweg-Funktion
  - diskrete Exponentialfunktion  $b^x \bmod m$  -> einfach zu berechnen
  - bisher keine effiziente Berechnung der Umkehrfunktion bekannt

# Kommunikationsübersicht



## Übersicht der zwei Phasen

---

- Precomputation Phase
  - 1
- Realtime Phase
  - 2



## Vorbereitung

---

- User
  - Symmetrischer Schlüssel  $K_{i,j}$  zwischen jedem User  $U_j$  user jedem Node  $N_i$
  - Austausch des Schlüssels z.B. mit Diffie-Hellman
  - $K_{i,j}$  wird später als Eingabe für Pseudozufallszahlengenerators verwendet
  - User und Node könne für jede Runde den gleichen Schlüssel generieren.

## Precomputation - Step 1

---

- Pre Processing
- Knoten  $N_1, \dots, N_n$  erzeugt einen Vektor  $r_i$  aus zufälligen Werten für jede Nachricht
- Verschlüsselung mittels ElGamal  $\rightarrow E(r_i^{-1})$ , Resultat wird an den Network Handler gesendet
  - Diese Verschlüsselung muss dann in der Echtzeitphase nicht mehr durchgeführt werden
- NH berechnet Produkt aus allen  $E(r_n) \rightarrow E(R_n^{-1})$

## Precomputation - Step 2

- Mixing
  1. Jeder Knoten legt Permutation  $P(X)$  fest.
  2. Jeder Knoten erzeugt einen weiteren zufälligen Vektor  $s_i$
  3.  $E(R_n^{-1})$  wird von jedem Knoten nacheinander mit der jeweils festgelegten Permutation permutiert (Mixing) und gleichzeitig der erzeugte  $s_i^{-1}$  hineinmultipliziert
  4.  $s_i$  wird später für eine Nachrichtenantwort verwendet
  5. Der letzte Knoten erzeugt damit  $E(P_n(R_n^{-1}) \times S_n^{-1})$

## Precomputation - Step 3

- Post Processing
  1. Jeder Knoten berechnet nun aus  $E(P_n(R_n^{-1}) \times S_n^{-1})$  seinen Entschlüsselungsanteil  $D(i, r)$  für den zufälligen Vektor  $r_i$  aus Schritt 1.
  2. Das jeder Knoten einen eigenen Entschlüsselungsanteil berechnen kann, liegt an der ElGamal Verschlüsselung, die diese Möglichkeit bietet.
  3.  $E(P_n(R_n^{-1}) \times S_n^{-1})$  kann nur mit allen Anteilen entschlüsselt werden

## Precomputation - Return Path

- Step 1
  1. Nodes erzeugen zufällige Vektoren  $E(s_i'^{-1})$  (ElGamal verschlüsselt).
  2. Permutation rückwärts, der letzte Knoten beginnt, gleichzeitig werden  $s'^{-1}$  dazumultipliziert
  3. Der erste Knoten erhält  $E(S_1'^{-1})$
- Step 2
  1. Wie vorher werden wieder Entschlüsselungsanteile für  $E(S_1'^{-1})$  von allen Knoten berechnet

## Precomputation - Resultat

---

- Hinweg
  1.  $E(P_n(R_n^{-1}) \times S_n^{-1})$
- Rückweg
  1.  $E(S_1'^{-1})$

## Echzeit Phase - Step 1

- Generierung mit gleichem Seed
  1. User  $U_j$  generiert aus  $K_{i,j}$  Zufallszahl  $ka_{i,j}$  für jeden Knoten,  
 $Ka_j = \sum_{i=1}^n ka_{i,j}$
  2. Verschlüsselung einer Nachricht mit  $M_j \times Ka_j^{-1}$
  3. Network Handler  $Ka_j^{-1} \rightarrow Ka^{-1} \rightarrow M \times Ka^{-1}$
  4. Knoten  $N_i$  generiert  $ka_i = \sum_{j=1}^m ka_{i,j}$  und sendet  $ka_i \times r_i$  an den NH.
- Austausch der Verschlüsselung
  1. Der NH kann damit die  $Ka^{-1}$  mit den zufälligen Vektoren  $r_i$  der Knoten austauschen
  2.  $M \times Ka^{-1} \times \sum_{i=1}^n ka_i \times r_i = M \times R_n$

## Echzeit Phase - Step 2

---

- Mixing
  - Jeder Knoten permutiert (Nachrichten werden getauscht) nacheinander  $M \times R_n$  und multipliziert den zufälligen Vektor  $S_i$  mit ein
  - Der letzte Knoten erhält damit  $P_n(M \times R_n) \times S_n$



## Echzeit Phase - Step 3

- Sammeln der Entschlüsselungsanteile
  - Die Knoten  $N_1$  bis  $N_i$  senden ihren Entschlüsselungsanteil  $D(i, x)$  an den NH
- Entschlüsselung
  - Der NH Entschlüsselt  $E(P_n(R_n^{-1}) \times S_n^{-1})$  mittels  $D(n, x)$
  - $P_n(M \times R_n) \times S_n \times P_n(R_n^{-1}) \times S_n^{-1} = P_n(M)$
- Resultat
  - $P_n(M)$  Ursprüngliche Nachrichten in vertauschter Reihenfolge
  - Keine direkte Verbindung zum Sender möglich

## Echzeit Phase - Antwort

---

- Sammeln der Entschlüsselungsanteile
  - Die Knoten  $N_1$  bis  $N_i$  senden ihren Entschlüsselungsanteil  $D(i, x)$  an den NH

# Anonymität

---

- Anhand eines Modells
  - Private Kommunikation der Mixknoten untereinander und eines vertraulichen dritten Punktes
  - Keine Kryptographischen Operation, Sicherstellung durch den vertraulichen dritten Punkt
  - „reale Simulation“ des Modells mit Eigenschaften des cMix Protokolls zeigt Anonymität

# Integrität

---

- Die Integrität ist gegeben wenn
  - Die Nachricht M unmodifiziert und an den Empfänger weitergeleitet wird oder...
  - Alle Mixknoten wissen, dass das cMix Protokoll nicht richtig durchgeführt wurde
    - Sicherstellung durch den Mechanismus „Randomized Partial Checking“

# Vertraulichkeit

---

- **Schutzziel Anonymität**
  - Wird sichergestellt indem Nachrichten vom Sender verschlüsselt werden
    - z.B durch einen öffentlichen Schlüssel einer asymmetrischen Verschlüsselung
  - Diese Verschlüsselung vermeidet aufwändige Publickey-Operationen

# Protoyp

- Performance Messung
  - In Python implementiert
  - Auf Instanzen des Amazon Web Service EC2 getestet
  - Jeder Mixknoten hatte zwei Intel Xeon E5-2680 und 3,75 GB Arbeitsspeicher zur Verfügung
  - Bei einer 1024-bit ElGamal-Verschlüsselung
  - Starke Verbesserung das re-encryption Mixnet ist bis zu 8 mal langsamer

Anzahl Nachrichten	Vorbereitung (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

## Einbettung in PrivataTegrity

---

- Zweck
  - Nur mit berechtigten Partnern weiter kommunizieren
  - Verhindert unbefugte Inanspruchnahme von Betriebsmitteln

# Backdoor

---

- Zweck
  - Nur mit berechtigten Partnern weiter kommunizieren
  - Verhindert unbefugte Inanspruchnahme von Betriebsmitteln