Das c-mix Verfahren

Merlin Koglin, Maik Graaf

Agenda

1. Motivation
2. c-mix Verfahren
3. Sicherheit
4. Performance
5. 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

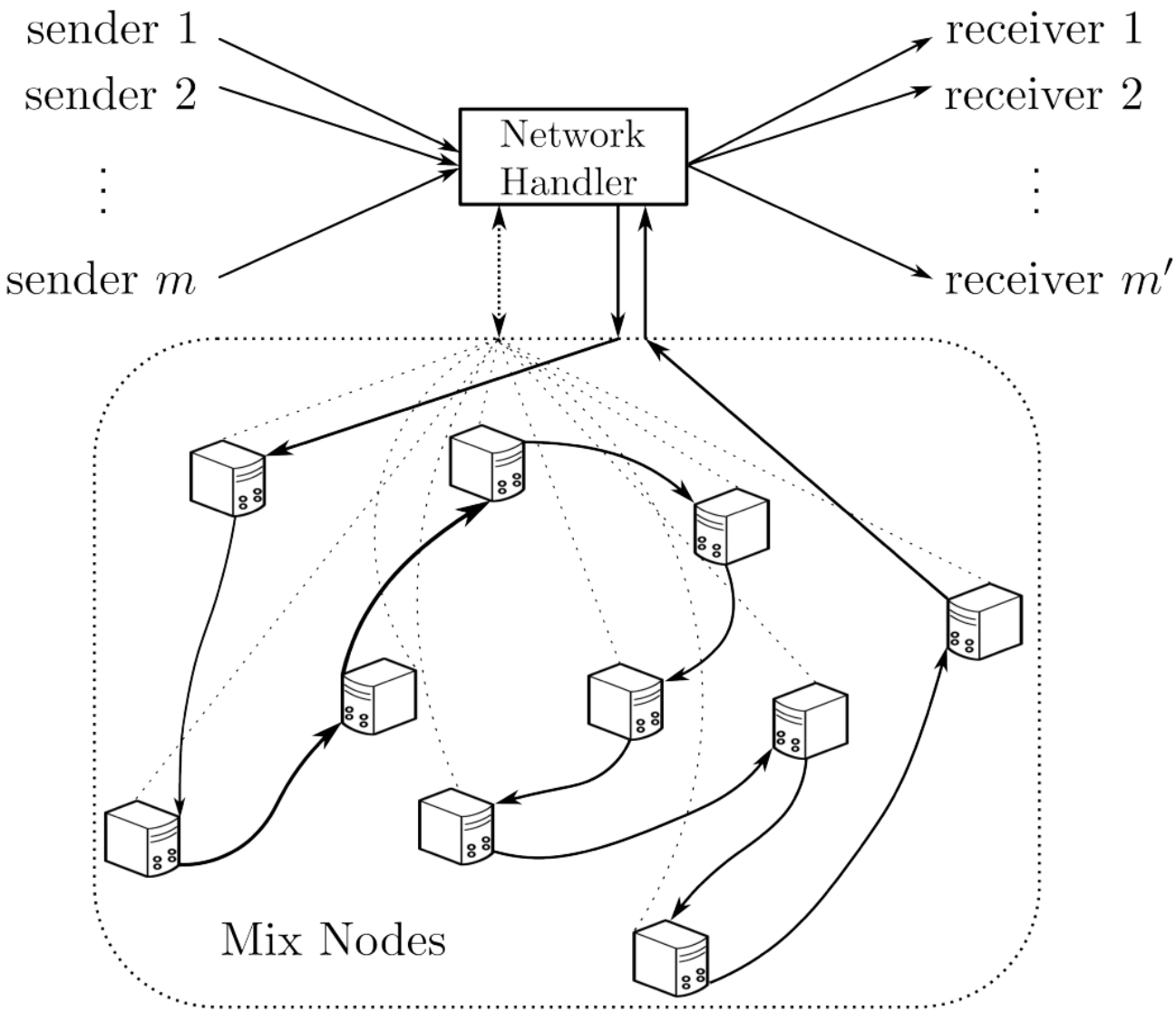
Probleme bisheriger Mix Verfahren

- In Echtzeitsystemen
  - Lange Wartezeiten 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 Energieaufwands

Idee von David Chaum

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

Kommunkationsübersicht



Übersicht der zwei Phasen

- Vorbereitung
- Precomputation Phase
  - Hier werden aufwändige Public-Key Verschlüsselungen vorberechnet
  - 3 Schritte, Preprocessing, Mixing, Postprocessing
- Realtime Phase
  - In dieser Phase findet die Kommunikation zwischen Sender und Empfänger statt
  - 3 Schritte, Preprocessing, Mixing, Postprocessing

Vorbereitung

Vorbedingungen an die User und Nodes

- 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

Precomputation - Step 1

- Knoten N_1, ... N_n erzeugt einen Vektor r_i aus zufälligen Werten für jede Nachricht
- Verschlüsselung mittes ElGamal $\rightarrow E(r_i^{-1})$
  - Diese Verschlüsselung muss dann in der Echzeitphase nicht mehr durchgeführt werden
- ElGamal
  - asymmetrisches Verschlüsselungsverfahren
  - jeder Knoten hält einen Teil des privaten Schlüssels
- NH fasst diese zusammen $E(r_i) \rightarrow E(R_n^{-1})$

Precomputation - Step 2

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

Precomputation - Step 3

- Entschüsselungsanteil
  1. Jeder Knoten berechnet nun aus $E(P_n(R_n^{-1}) \times S_n^{-1})$ seinen Entschlüsselungsanteil.
  2. Das jeder Knoten einen eigenen Entschlüsselungsanteil berechnen kann, liegt an der ElGamal Verschlüsselung, die diese Möglichkeit bietet.
- $E(P_n(R_n^{-1}) \times S_n^{-1})$ kann nur mit allen Anteilen entschlüsselt werden

Precomputation - Return Path

- Step 1
  1. Nodes erzeugen zufällige Vektoren $E(s'^{-1}_i)$ (ElGamal verschlüsselt).
  2. Permutation rückwärts, der letzte Knoten beginnt, gleichzeitig werden s'^{-1} hinzugefügt
  3. Der erste Knoten erhält $E(S'^{-1}_1)$
- Step 2
  1. Wie vorher werden wieder Entschüsselungsanteile D'_i für $E(S'^{-1}_1)$ von allen Knoten berechnet

Precomputation - Resultat

- Hinweg
  1. $E(P_n(R_n^{-1}) \times S_n^{-1})$, Entschlüsselungsanteil D_i
- Rückweg
  1. $E(S'^{-1}_1)$, Entschlüsselungsanteil D'_i

Echzeit Phase - Step 1

- Generierung mit gleichem Seed
  1. User U_j generiert für jeden Knoten mittels Pseudozufallszahlengenerator einen neuen Schlüssel und fasst diese zusammen Ka_j^{-1}
  2. Startwert: der anfangs ausgetauschte symmetrische Schlüssel
  3. Verschlüsselung einer Nachricht mit $M_j \times Ka_j^{-1}$
  4. Network Handler führt alle zusammen $M \times Ka^{-1}$
  5. Knoten N_i generiert für jeden User wie oben Schlüssel und sendet $ka_i \times r_i$ an den NH.

Echzeit Phase - Step 1

- 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 fügt Vektor S_i aus Precomputation Phase ein
  - Der letzte Knoten erhält $P_n(M \times R_n) \times S_n)$

Echzeit Phase - Step 3

- Sammeln der Entschlüsselungsanteile
  - Die Knoten senden Entschlüsselungsanteile an den NH $\rightarrow D(n, x)$
- 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

Schritt 1

- Antworten werden gesammelt (M')
- Die Mixknoten permutieren diese nun rückwärts und fügen $E(s')$ hinzu
- Erste Knoten erzeugt dann $P'(M') \times E(S')$
- Alle Knoten erzeugen neuen Key mittels PZG Ka'_j und verknüpfen diesen mit Entschüsselungsanteil
- $E(S'^{-1})$ kann entschüsselt werden
- Resultat
  - $P'(M') \times S' \times E(S'^{-1}) \times D' \times Ka'$
    $= P'(M') \times S' \times S'^{-1} \times Ka'$

Echzeit Phase - Antwort

Schritt 2

- $P'(M') \times S' \times S'^{-1} \times Ka' = P'(M') \times Ka'$
- User U_j kann Ka'_j für seinen Nachrichtenslot erzeugen und Nachricht entschüsseln.

Anonymität

- Anhand eines Modells
  - Private Kommunikation der Mixknoten untereinander und eines vertraulichen dritten Punktes
  - Keine Kryptographischen Operationen, 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 Public-Key-Operationen

Prototyp

- 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	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

Einbettung in PrivaTegrity

PrivaTegrity

- Sicherheitssystem basierend auf c-mix
- Nachrichten werden in 1-Sekunden Intervallen gesammelt und gesendet
- 10 Mixknoten in 10 verschiedenen Ländern
- Wenn alle Mixknoten Betreiber kooperieren, kann Anonymität aufgedeckt werden

Fazit

- Prototyp ist effizienter und schneller als bisherige Verfahren
- Anpassung für heutige Kommunikationsgeräte (Smartphone, Laptop)
- Anonymes Chatten, Fotosharing, Bezahlsysteme, Suche
- Möglichkeit zur gezielten Aufdeckung wird teilweise kritisch betrachtet