Public-Key-Verschlüsselung und Diskrete Logarithmen

Carsten Baum

Institut für Informatik
Universität Potsdam

10. Juni 2009

Inhaltsverzeichnis

Gruppen

Es sei $(G, \cdot)$ eine Gruppe.

Die Ordnung der Gruppe entspricht der Anzahl der Elemente in G.

Die Ordnung eines Elements $\alpha \in G$ ist die Zahl n, für die gilt: $\alpha^n = e$, wobei e das neutrale Element bzgl. $\cdot$ ist.

Korollar: Es sei $n \in \mathbb{N}$ und $\alpha^n = 1$ sowie $\alpha^{n/p} \neq 1$ für alle Primteiler p von n. Dann ist n die Ordnung von α.

Die Gruppe heißt zyklisch, wenn ein $\alpha \in G$ existiert, so dass $\forall a \in G \; \exists i \in \mathbb{N} : \alpha^i = \underbrace{\alpha \cdot \alpha \cdots \alpha}_{i} = a$.

α heißt Generator der zyklischen Gruppe.

Ordnung

Sei $(G, \cdot)$ eine Gruppe und $\alpha \in G$.

α habe eine endliche Ordnung. Man schreibt $\langle \alpha \rangle$ für die von α erzeugte Untergruppe.

Anmerkungen:

1. Ist G endlich, so teilt die Ordnung von $\langle \alpha \rangle$ stets die Ordnung von G (Satz von Lagrange).
2. Ist G zyklisch und α ein Generator, so ist die Ordnung von α gleich der Ordnung von G.
3. Die Gruppe $(\mathbb{Z}_n, \cdot)$, wobei n eine Primzahl ist, heißt **prime Restklassengruppe**. Die Gruppe ist zyklisch und hat die Ordnung n - 1.

Primitivwurzeln

Es sei $(G, \cdot)$ eine Gruppe, $\alpha \in G$ und $p = |G|$.

α heißt Primitivwurzel, wenn α die Ordnung p hat und $ggT(\alpha, p) = 1$.

Es sei $(\mathbb{Z}_p, \cdot)$ eine prime Restklassengruppe und $\alpha \in \mathbb{Z}_p$. Weiterhin seien $p_1 p_2 \cdots p_n$ die Primfaktoren von $\phi(p)$.

α ist eine Primitivwurzel, wenn $1 \leq r \leq n : \alpha^{p_r} \neq 1$.

Formulierung des DLP

Voraussetzungen:

Sei $(G, \cdot)$ eine multiplikative Gruppe, $\alpha \in G$ ein Element der Ordnung n und $\beta \in \langle \alpha \rangle$.

Problem:

Man berechne a $(0 \leq a \leq n-1)$, so dass
$\alpha^a = \beta$.

a nennt man den diskreten Logarithmus von β zur Basis α.

Anforderungen an ein Public-Key-Kryptosystem

Anforderungen an ein Public-Key-Kryptosystem:

1. $d(S, e(P, M)) = M$ für alle M und M ist beliebig.
2. Alle Schlüsselpaare (S, P) sind verschieden.
3. Die Herleitung von S aus P ist mindestens so aufwendig wie die Entschlüsselung von C ohne S.
4. Die Funktionen e und d sind einfach berechenbar.

Anforderungen an ein Public-Key-Kryptosystem

Notwendig sind Probleme, die NP-vollständig sind

1. Aufwand zur Lösung des Problems ohne Wissen über S steigt exponentiell mit dem Umfang des Problems.
2. Berechnung von d mit S ist in polynomialer Zeit möglich.

Beispiele für nutzbare Probleme:

- Faktorisierung großer Zahlen
- Berechnung des diskreten Logarithmus
- Knapsackproblem

Das Hellman-Merkle-Verfahren

Basiert auf dem 0-1-Knapsackproblem.

- 1978 von Merkle und Hellman vorgeschlagen.
- 1985 durch Brickell gebrochen.

Aber:
Angriff auf das Verfahren löst das zugrunde liegende Problem nicht.

Diffie-Hellman-Schlüsselaustausch

Alice und Bob wollen über eine unsichere Leitung kommunizieren.

Alice und Bob haben noch **keinen** gemeinsamen Schlüssel vereinbart.

Charlie will die Kommunikation von Alice und Bob belauschen.

Problem:

Wie können sich Alice und Bob über eine unsichere Leitung auf auf einen geheimen Schlüssel einigen, ohne das Charlie ihn erfährt?

Diffie-Hellman-Schlüsselaustausch

Lösung:

Schlüsselaustausch nach Diffie und Hellman

1. Alice und Bob einigen sich auf eine Primzahl p und eine Primitivwurzel g mod p.
   Es gilt $2 \leq g \leq p-2$.
2. Alice wählt ein $a \in \{0, 1, \cdots, p-2\}$ und berechnet $A = g^a \bmod p$

   Bob wählt ein $b \in \{0, 1, \cdots, p-2\}$ und berechnet $B = g^b \bmod p$
3. Alice schickt A an Bob, Bob schickt A an Alice.
4. Alice berechnet $K_1 = B^a \bmod p$

   Bob berechnet $K_2 = A^b \bmod p$

Sicherheit von Diffie-Hellman

$K = K_1 = K_2$, da $B^a \bmod p = g^{ab} \bmod p = A^b \bmod p$

Charlie hört über die Leitung g, p, A und B mit.

Um den gemeinsamen Schlüssel von Alice und Bob zu berechnen, muss Charlie $log_g A$ oder $log_g B$ in $\mathbb{Z}_p$ lösen.

Könnte Charlie auch aus g, p, A und B den Wert g^{ab} errechnen? Dies nennt man das **Diffie-Hellman-Problem**.

Die Frage, ob für bekannte g^a, g^b und g^c gilt $g^{ab} \equiv g^c$ nennt man das **Diffie-Hellman-Decision-Problem**.

Es ist nicht bekannt, ob eine effiziente Lösung des Diffie-Hellman-Problems oder des Diffie-Hellman-Decision-Problems das DLP löst.

Schlüsselerzeugung

ElGamal-Verschlüsselung

Alice erzeugt eine Primzahl p und eine Primitivwurzel g mod p.
Es gilt $2 \leq g \leq p-2$.

Weiterhin erzeugt Alice zufällig und gleichverteilt ein $a \in \{0, 1, \cdots, p-2\}$

Alice berechnet $A = g^a \bmod p$.

(p, g, A) ist der öffentliche Schlüssel, a bleibt geheim.

Verschlüsselung

Bob hat die Nachricht m mit $m \in \{0, 1, \cdots, p-1\}$.

Er wählt zufällig ein $b \in \{1, 2, \cdots, p-2\}$ und berechnet $B = g^b \bmod p$.

Weiterhin berechnet Bob
$c = A^b m \bmod p$
und verschickt den Schlüsseltext (B, c) an Alice.

Entschlüsselung

Alice berechnet den Exponenten $x = p - 1 - a$.

Mit x berechnet Alice $B^x c$ mod p. Dies entspricht dem Klartext.

Beweis:

$$B^x c \equiv \underbrace{g^{b(p-1-a)}}_{B^x} \underbrace{A^b m}_{c}$$

$$\equiv (\underbrace{g^{p-1}}_{=1})^b (\underbrace{g^a}_{=A})^{-b} A^b m$$

$$\equiv A^{-b} A^b m \equiv m \text{ mod } p$$

Anmerkungen

- ElGamal benötigt 2 Exponentiationen pro Verschlüsselung, nämlich
  A^b mod p und $B = g^b$ mod p.
- Bob muss zusätzlich zur verschlüsselten Nachricht auch noch B versenden(Nachrichtenexpansion).
- Wird der gleiche Klartext m mit verschiedenen b verschlüsselt, so ergeben sich verschiedene c.
  Wird b zufällig und gleichverteilt gewählt, so sind die Schlüsseltexte ebenfalls zufällig und gleichverteilt(randomisiert).

weitere Anmerkungen

- Die Sicherheit des ElGamal-Verfahrens basiert darauf, dass das Berechnen des diskreten Logarithmus im zugrunde liegenden Körper schwierig ist.
- Existiert ein effizienter Algorithmus um den geheimen Exponenten a zu berechnen, so ist m berechenbar.
- Ob eine Attacke auf ElGamal Rückschlüsse auf den diskreten Logarithmus zulässt, ist nicht bekannt.

Äquivalenz der Schwierigkeit von Diffie-Hellman und ElGamal

Angenommen, Charlie könnte das Diffie-Hellman-Problem lösen.

Charlie besitzt (p, g, A) und (B, c). Er kann $K = g^{ab}$ mod p berechnen. Damit erhält er $m = K^{-1}c$ mod p.

Angenommen, Charlie könnte ElGamal brechen, also aus p,g,A,B,c das zugehörige m berechnen.

Dann setzt Charlie $c = 1$ und erhält m.
Es gilt $1 = g^{ab}m$ mod p und damit
$m^{-1} = g^{ab}$ mod p.

Andere Gruppen

Die vorgestellten Algorithmen operieren in primen Restklassengruppen.
Angenommen, man könnte diskrete Logarithmen in primen Restklassengruppen effizient berechnen.
Können die Verfahren weiter verwendet werden?

Die Verfahren können in anderen zyklischen Gruppen eingesetzt werden, in denen das Problem des diskreten Logarithmus ebenfalls schwer berechenbar ist.

Beispiele:

1. Z_{p^k}, p ist prim.
2. Punktgruppe einer elliptischen Kurve über einem endlichen Körper K.

Man-in-the-middle-Attacke

Voraussetzung:
Charlie kann Kommunikation zwischen Alice und Bob unterbrechen.

1. Charlie handelt Schlüssel K_1 mit Alice und Schlüssel K_2 mit Bob aus.
2. Nachrichten von Alice an Bob muss Charlie mit K_1 entschlüsseln und mit K_2 verschlüsseln.

   Nachrichten von Bob an Alice muss Charlie umgekehrt mit K_2 entschlüsseln und mit K_1 verschlüsseln.
3. Charlie kann Nachrichten mitlesen und beliebig manipulieren.

Klartextmanipulation

Eine Nachricht besteht bei ElGamal aus (B, c), wobei c der verschlüsselte Text von m ist.

Dann ist (B, dc) der verschlüsselte Text von dm.

$$B^x dc \equiv d \underbrace{g^{b(p-1-a)}}_{B^x} \underbrace{A^b m}_{c}$$

$$\equiv d(\underbrace{g^{p-1}}_{=1})^b(\underbrace{g^a}_{=A})^{-b} A^b m$$

$$\equiv dA^{-b}A^b m \equiv dm \bmod p$$

Absicherung von ElGamal gegen diese Adaptive-Chosen-Ciphertext-Attacke ist möglich, macht das Verfahren aber ineffizient.

Angriff bei bekanntem Klartext

Es seien $c = A^b m \mod p$
und $c' = A^b m' \mod p$.

Es seien weiterhin c und m bekannt.

Da wir in $\mathbb{Z}_p$ rechnen, können wir das multiplikative Inverse von m mittels des erweiterten euklidischen Algorithmus berechnen.

$c = A^b m \mod p \Rightarrow cm^{-1} = A^b \mod p$

Man berechnet das multiplikative Inverse zu A^b und erhält so m' durch
$c' = A^b m' \mod p \Rightarrow c'(A^b)^{-1} = m'$

Zur Erinnerung - Das DLP

Voraussetzungen:

Sei $(G, \cdot)$ eine multiplikative Gruppe, $\alpha \in G$ ein Element der Ordnung n und $\beta \in \langle \alpha \rangle$.

Problem:

Man berechne a $(0 \leq a \leq n-1)$, so dass
$\alpha_a = \beta$.

a nennt man den diskreten Logarithmus von β zur Basis α.

Brute-Force-Attacken

2 einfache Angriffe:

Gegeben sei $\alpha^a = \beta \bmod n$.

1. Iteriere r von 0 bis n-1, berechne α^r und prüfe, ob $\alpha^r = \alpha^a$.

   Laufzeit: $O(n)$.
   Speicherplatz: $O(1)$.

2. Berechne eine Hashtabelle nach Verfahren 1 und speichere r an der Adresse α^r.

   Initialisierung: $O(n)$
   Suche: $O(1)$
   Speicherplatz: $O(n)$

Shanks Algorithmus

Auch Babystep-Giantstep-Algorithmus genannt.

Gegeben sei ein $m = \lceil \sqrt{n} \rceil$.

a lässt sich darstellen als $a = mj + i$, wobei $0 \leq i \leq m - 1$ und $0 \leq j \leq m - 1$.

Es ist $\alpha^{mj+i} = \beta \Rightarrow \alpha^{mj} = y = \beta\alpha^{-i}$.

Wir berechnen nun die Mengen
$L_1 = \{(j, y) | \alpha^{mj} = y\}$ und
$L_2 = \{(i, y) | \beta\alpha^{-i} = y\}$.

Man finde $(j, y) \in L_1$ und $(i, y) \in L_2$ und erhält $a = mj + i$.

Shanks Algorithmus

Shanks Algorithmus - Pseudocode

1. $m \leftarrow \lceil \sqrt{n} \rceil$
2. **for** $j \leftarrow 0$ **to** $m - 1$ do
   compute $y = \alpha^{mj}$
   put (j, y) into hashtable L_1 indexed by y
3. **for** $i \leftarrow 0$ **to** $m - 1$ do
   compute $y = \beta\alpha^{-i}$
   put (i, y) into hashtable L_2 indexed by y
4. find y such that $(j, y) \in L_1$ and $(i, y) \in L_2$ exists
5. print $a \leftarrow mj + i$

Index-Calculus-Methode

Ist schneller als die bisher vorgestellten Methoden.

Funktioniert nur in Gruppen der Form $(\mathbb{Z}_p, \cdot)$.

Idee:
Wir berechnen den 'großen' Logarithmus, indem wir Logarithmen für 'kleine' Elemente aus $\mathbb{Z}_p$ berechnen und damit Rückschlüsse auf den 'großen' Logarithmus ziehen.

Index-Calculus-Methode - Vorberechnung

1. Man bestimmt eine Zahl $B < n$ und die Menge $F(B) = \{p_1, p_2, \cdots, p_B\}$ von Primzahlen, die sogenannte Faktorbasis.
2. Man wähle ein C größer als B, z.B. $B + 10$.
3. Man erhält nun C Kongruenzen der Form $\alpha^{x_j} \equiv p_1^{a_{1j}} p_2^{a_{2j}} \cdots p_B^{a_{Bj}} \ mod\ p$ für $1 \leq j \leq C$.
4. Dies kann man in ein lineares Gleichungssystem mit C Gleichungen umwandeln mit Gleichungen der Form $x_j \equiv a_{1j} log_\alpha p_1 + \cdots + a_{Bj} log_\alpha p_B \ mod\ p - 1$ für $1 \leq j \leq C$.
5. Man bestimmt x-Werte, so dass α^x nur Primfaktoren in $F(B)$ hat, und berechnet die Exponenten der Primfaktoren durch Division.
6. Anschließend löst man das lineare Gleichungssystem mit dem Gauss'schen Algorithmus.

- Die Vorberechnung terminiert in $O(e^{(1+o(1))\sqrt{ln(p)ln(ln(p))}})$.

Index-Calculus-Methode - Bestimmen des diskreten Logarithmus

1. Man wählt zufällig ein s$(1 \leq s \leq p-2)$ und berechnet $\gamma = \beta\alpha^s \bmod p$.
2. Wenn s nur Primfaktoren in $F(B)$ hat, so erhält man $\beta\alpha^s \equiv \underbrace{p_1^{c_1} p_2^{c_2} \cdots p_B^{c_B}}_{\gamma} \bmod p$,
   ansonsten muss man ein neues s wählen.
3. Dies kann man umformen nach $log_\alpha\beta + s \equiv c_1 log_\alpha p_1 c_2 log_\alpha p_2 \cdots c_B log_\alpha p_B \bmod p-1$.

- Dieser Algorithmus terminiert in $O(e^{(1/2+o(1))\sqrt{ln(p)ln(ln(p))}})$.

Literatur

Cryptography - Theory and Practice, Stinson, 3. Auflage, Chapman & Hall

Einführung in die Kryptographie, Buchmann, 3. Auflage, Springer Science

CrypTool-Skript, Esslinger et al., 9. Auflage, www.cryptool.org