Seminar Kryptographie und Datensicherheit

Public-key Kryptographie mit Diskreten Logarithmen

Peter Streng, Martin Schöne

Wintersemester 2006/2007

Gliederung

Problem des diskreten Logarithmus

- Sei $(G, \cdot)$ eine multiplikative Gruppe, $\alpha \in G$ ein Element der Ordnung n und $\beta \in \langle \alpha \rangle$.
- Fragestellung:
  Welche ganze Zahl $a, 0 \leq a \leq n-1$, erfüllt die Gleichung

$$\alpha^a = \beta$$

  a ergibt sich folglich aus der Berechnung von $\log_\alpha \beta$

ElGamal Public-key Kryptosystem

Sei p eine Primzahl, so dass das Lösen diskreter Logarithmen in $(\mathbb{Z}_p^*, \cdot)$ schwer ist, und sei $\alpha \in \mathbb{Z}_p^*$ ein primitives Element.
Definition:

$$\mathbf{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}$$

Die Werte p, α, β bilden den öffentlichen Schlüssel und a den privaten Schlüssel.
Für $K = (p, \alpha, a, \beta)$ und für eine (geheime) zufällige Zahl $k \in \mathbb{Z}_{p-1}$ und $y_1, y_2 \in \mathbb{Z}_p^*$ definiere:

- Verschlüsselung: $e_K(x, k) = (y_1, y_2)$
  $y_1 = \alpha^k \mod p$, $y_2 = x \cdot \beta^k \mod p$
- Entschlüsselung: $d_K(y_1, y_2) = y_2(y_1^a)^{-1} \mod p$

naiver Ansatz

- $a = \log_\alpha \beta$ lässt sich mit erschöpfender Suche finden:
- Berechne nacheinander $\alpha, \alpha^2, \alpha^3, ..., \alpha^i$, bis $\alpha^i = \beta$ und somit $a = i$ gefunden ist.

einfacher Algorithmus

$i \leftarrow 0$

while $\alpha^i \neq \beta$ **do**

$\alpha^i \leftarrow \alpha^{i-1} \cdot \alpha$

$i \leftarrow i + 1$

naiver Ansatz

- Beispiel:
  $\alpha = 3$, $\beta = 4$ und $p = 7$

  $\log_3 4 = a$

  $3^0 \equiv 1 \pmod 7$
  $3^1 \equiv 3 \pmod 7$
  $3^2 \equiv 2 \pmod 7$
  $3^3 \equiv 6 \pmod 7$
  $3^4 \equiv 4 \pmod 7 \Rightarrow a = 4$
  $3^5 \equiv 5 \pmod 7$

Shanks

Shanks(G, n, α, β)

- $m \leftarrow \lceil \sqrt{n} \rceil$
- **for** $j \leftarrow 0$ **to** $m - 1$
  do berechne α^{mj}
- Paare (j, α^{mj}) ordnen nach $\alpha^{mj} \rightarrow$ Liste L_1
- **for** $i \leftarrow 0$ **to** $m - 1$
  do berechne $\beta\alpha^{-i}$
- Paare $(i, \beta\alpha^{-i})$ sortieren nach $\beta\alpha^{-i} \rightarrow$ Liste L_2
- Finde ein Paar $(j, y) \in L_1$ und ein Paar $(i, y) \in L_2$
- $\log_\alpha \beta \leftarrow (mj + i) \mod n$

Beispiel

Gesucht: $\log_3 525$ in $(\mathbb{Z}_{809}{}^*, \cdot) \Rightarrow \alpha = 3,\ \beta = 525,\ n = 808,\ m = 29$

Beispiel

Gesucht: $\log_3 525$ in $(\mathbb{Z}_{809}{}^*, \cdot) \Rightarrow \alpha = 3,\ \beta = 525,\ n = 808,\ m = 29$

$L1$

(0, 1)	(1, 99)	(2, 93)	(3, 308)
(4, 559)	(5, 329)	(6, 211)	(7, 664)
(8, 207)	(9, 268)	(10, 644)	(11, 654)
(12, 26)	(13, 147)	(14, 800)	(15, 727)
(16, 781)	(17, 464)	(18, 632)	(19, 275)
(20, 528)	(21, 496)	(22, 564)	(23, 15)
(24, 676)	(25, 586)	(26, 575)	(27, 295)
(28, 81)			

Beispiel

Gesucht: $\log_3 525$ in $(\mathbb{Z}_{809}{}^*, \cdot) \Rightarrow \alpha = 3,\ \beta = 525,\ n = 808,\ m = 29$

$L1$

(0, 1)	(1, 99)	(2, 93)	(3, 308)
(4, 559)	(5, 329)	(6, 211)	(7, 664)
(8, 207)	(9, 268)	(10, 644)	(11, 654)
(12, 26)	(13, 147)	(14, 800)	(15, 727)
(16, 781)	(17, 464)	(18, 632)	(19, 275)
(20, 528)	(21, 496)	(22, 564)	(23, 15)
(24, 676)	(25, 586)	(26, 575)	(27, 295)
(28, 81)			

$\Rightarrow$

$L1$ *sortiert*

(0, 1)	(23, 15)	(12, 26)	(28, 81)
(2, 93)	(1, 99)	(13, 147)	(8, 207)
(6, 211)	(9, 268)	(19, 275)	(27, 295)
(3, 308)	(5, 329)	(17, 464)	(21, 496)
(20, 528)	(4, 559)	(22, 564)	(26, 575)
(25, 586)	(18, 632)	(10, 644)	(11, 654)
(7, 664)	(24, 676)	(15, 727)	(16, 781)
(14, 800)			

Beispiel

Gesucht: $\log_3 525$ in $(\mathbb{Z}_{809}{}^*, \cdot) \Rightarrow \alpha = 3,\ \beta = 525,\ \mathrm{n} = 808,\ \mathrm{m} = 29$

L1

(0, 1)	(1, 99)	(2, 93)	(3, 308)
(4, 559)	(5, 329)	(6, 211)	(7, 664)
(8, 207)	(9, 268)	(10, 644)	(11, 654)
(12, 26)	(13, 147)	(14, 800)	(15, 727)
(16, 781)	(17, 464)	(18, 632)	(19, 275)
(20, 528)	(21, 496)	(22, 564)	(23, 15)
(24, 676)	(25, 586)	(26, 575)	(27, 295)
(28, 81)			

$\Rightarrow$

L1 sortiert

(0, 1)	(23, 15)	(12, 26)	(28, 81)
(2, 93)	(1, 99)	(13, 147)	(8, 207)
(6, 211)	(9, 268)	(19, 275)	(27, 295)
(3, 308)	(5, 329)	(17, 464)	(21, 496)
(20, 528)	(4, 559)	(22, 564)	(26, 575)
(25, 586)	(18, 632)	(10, 644)	(11, 654)
(7, 664)	(24, 676)	(15, 727)	(16, 781)
(14, 800)			

L2

(0, 525)	(1, 175)	(2, 328)	(3, 379)
(4, 396)	(5, 132)	(6, 44)	(7, 554)
(8, 724)	(9, 511)	(10, 440)	(11, 686)
(12, 768)	(13, 256)	(14, 355)	(15, 388)
(16, 399)	(17, 133)	(18, 314)	(19, 644)
(20, 754)	(21, 521)	(22, 713)	(23, 777)
(24, 259)	(25, 356)	(26, 658)	(27, 489)
(28, 163)			

Beispiel

Gesucht: $\log_3 525$ in $(\mathbb{Z}_{809}{}^*, \cdot) \Rightarrow \alpha = 3,\ \beta = 525,\ \mathrm{n} = 808,\ \mathrm{m} = 29$

L1

(0, 1)	(1, 99)	(2, 93)	(3, 308)
(4, 559)	(5, 329)	(6, 211)	(7, 664)
(8, 207)	(9, 268)	(10, 644)	(11, 654)
(12, 26)	(13, 147)	(14, 800)	(15, 727)
(16, 781)	(17, 464)	(18, 632)	(19, 275)
(20, 528)	(21, 496)	(22, 564)	(23, 15)
(24, 676)	(25, 586)	(26, 575)	(27, 295)
(28, 81)			

$\Rightarrow$

L1 sortiert

(0, 1)	(23, 15)	(12, 26)	(28, 81)
(2, 93)	(1, 99)	(13, 147)	(8, 207)
(6, 211)	(9, 268)	(19, 275)	(27, 295)
(3, 308)	(5, 329)	(17, 464)	(21, 496)
(20, 528)	(4, 559)	(22, 564)	(26, 575)
(25, 586)	(18, 632)	(10, 644)	(11, 654)
(7, 664)	(24, 676)	(15, 727)	(16, 781)
(14, 800)			

L2

(0, 525)	(1, 175)	(2, 328)	(3, 379)
(4, 396)	(5, 132)	(6, 44)	(7, 554)
(8, 724)	(9, 511)	(10, 440)	(11, 686)
(12, 768)	(13, 256)	(14, 355)	(15, 388)
(16, 399)	(17, 133)	(18, 314)	(19, 644)
(20, 754)	(21, 521)	(22, 713)	(23, 777)
(24, 259)	(25, 356)	(26, 658)	(27, 489)
(28, 163)			

$\Rightarrow$

L2 sortiert

(6, 44)	(5, 132)	(17, 133)	(28, 163)
(1, 175)	(13, 256)	(24, 259)	(18, 314)
(2, 328)	(14, 355)	(25, 356)	(3, 379)
(15, 388)	(4, 396)	(16, 399)	(10, 440)
(27, 489)	(9, 511)	(21, 521)	(0, 525)
(7, 554)	(26, 658)	(19, 644)	(11, 686)
(22, 713)	(8, 724)	(20, 754)	(12, 768)
(23, 777)			

Beispiel

Gesucht: $\log_3 525$ in $(\mathbb{Z}_{809}{}^*, \cdot) \Rightarrow \alpha = 3,\ \beta = 525,\ n = 808,\ m = 29$

L1

(0, 1)	(1, 99)	(2, 93)	(3, 308)
(4, 559)	(5, 329)	(6, 211)	(7, 664)
(8, 207)	(9, 268)	(10, 644)	(11, 654)
(12, 26)	(13, 147)	(14, 800)	(15, 727)
(16, 781)	(17, 464)	(18, 632)	(19, 275)
(20, 528)	(21, 496)	(22, 564)	(23, 15)
(24, 676)	(25, 586)	(26, 575)	(27, 295)
(28, 81)			

$\Rightarrow$

L1 sortiert

(0, 1)	(23, 15)	(12, 26)	(28, 81)
(2, 93)	(1, 99)	(13, 147)	(8, 207)
(6, 211)	(9, 268)	(19, 275)	(27, 295)
(3, 308)	(5, 329)	(17, 464)	(21, 496)
(20, 528)	(4, 559)	(22, 564)	(26, 575)
(25, 586)	(18, 632)	**(10,644)**	(11, 654)
(7, 664)	(24, 676)	(15, 727)	(16, 781)
(14, 800)			

L2

(0, 525)	(1, 175)	(2, 328)	(3, 379)
(4, 396)	(5, 132)	(6, 44)	(7, 554)
(8, 724)	(9, 511)	(10, 440)	(11, 686)
(12, 768)	(13, 256)	(14, 355)	(15, 388)
(16, 399)	(17, 133)	(18, 314)	(19, 644)
(20, 754)	(21, 521)	(22, 713)	(23, 777)
(24, 259)	(25, 356)	(26, 658)	(27, 489)
(28, 163)			

$\Rightarrow$

L2 sortiert

(6, 44)	(5, 132)	(17, 133)	(28, 163)
(1, 175)	(13, 256)	(24, 259)	(18, 314)
(2, 328)	(14, 355)	(25, 356)	(3, 379)
(15, 388)	(4, 396)	(16, 399)	(10, 440)
(27, 489)	(9, 511)	(21, 521)	(0, 525)
(7, 554)	(26, 658)	**(19,644)**	(11, 686)
(22, 713)	(8, 724)	(20, 754)	(12, 768)
(23, 777)			

Beispiel

Gesucht: $\log_3 525$ in $(\mathbb{Z}_{809}{}^*, \cdot) \Rightarrow \alpha = 3,\ \beta = 525,\ n = 808,\ m = 29$

L1

(0, 1)	(1, 99)	(2, 93)	(3, 308)
(4, 559)	(5, 329)	(6, 211)	(7, 664)
(8, 207)	(9, 268)	(10, 644)	(11, 654)
(12, 26)	(13, 147)	(14, 800)	(15, 727)
(16, 781)	(17, 464)	(18, 632)	(19, 275)
(20, 528)	(21, 496)	(22, 564)	(23, 15)
(24, 676)	(25, 586)	(26, 575)	(27, 295)
(28, 81)			

$\Rightarrow$

L1 sortiert

(0, 1)	(23, 15)	(12, 26)	(28, 81)
(2, 93)	(1, 99)	(13, 147)	(8, 207)
(6, 211)	(9, 268)	(19, 275)	(27, 295)
(3, 308)	(5, 329)	(17, 464)	(21, 496)
(20, 528)	(4, 559)	(22, 564)	(26, 575)
(25, 586)	(18, 632)	**(10,644)**	(11, 654)
(7, 664)	(24, 676)	(15, 727)	(16, 781)
(14, 800)			

L2

(0, 525)	(1, 175)	(2, 328)	(3, 379)
(4, 396)	(5, 132)	(6, 44)	(7, 554)
(8, 724)	(9, 511)	(10, 440)	(11, 686)
(12, 768)	(13, 256)	(14, 355)	(15, 388)
(16, 399)	(17, 133)	(18, 314)	(19, 644)
(20, 754)	(21, 521)	(22, 713)	(23, 777)
(24, 259)	(25, 356)	(26, 658)	(27, 489)
(28, 163)			

$\Rightarrow$

L2 sortiert

(6, 44)	(5, 132)	(17, 133)	(28, 163)
(1, 175)	(13, 256)	(24, 259)	(18, 314)
(2, 328)	(14, 355)	(25, 356)	(3, 379)
(15, 388)	(4, 396)	(16, 399)	(10, 440)
(27, 489)	(9, 511)	(21, 521)	(0, 525)
(7, 554)	(26, 658)	**(19,644)**	(11, 686)
(22, 713)	(8, 724)	(20, 754)	(12, 768)
(23, 777)			

$\Rightarrow \log_3 525 = ((29 * 10) + 19)\ mod\ 808 = 309$

Pollard Rho

Idee

Der Algorithmus baut eine Folge von Gruppenelementen mit einer "zufälligen" Funktion. Gesucht werden zwei Elemente der Folge, die gleich sind - aber einen anderen Index haben. Ist die Kollision gefunden, kann man (wahrscheinlich) den Logarithmus berechnen.

- Problem: $\beta = \alpha^a$
  Die Gruppe G wird in 3 paarweise disjunkte Teilmengen gegliedert G_1, G_2 und G_3.
  $f : G \to G$ definiert durch

$$f(\gamma) = \begin{cases} \alpha\gamma & \text{falls } \gamma \in G_1, \\ \gamma^2 & \text{falls } \gamma \in G_2, \\ \beta\gamma & \text{falls } \gamma \in G_3. \end{cases}$$

Pollard Rho

- Nehme Zufallszahl x_0 aus $\{1, ..., n\}$ und setze $\gamma_0 = \alpha^{x_0}$
  Berechne Folge (γ_i) rekursiv:

$$\gamma_{i+1} = f(\gamma_i)$$

- Glieder dieser Folge sind in der Form

$$\gamma_i = \alpha^{x_i} \beta^{y_i},\ i \geq 0$$

- x_0 ist der zufällig gewählte Startwert, $y_0 = 0$. Es gilt:

$$x_i + 1 = \begin{cases} x_i + 1 \mod n & \text{falls } \gamma_i \in G_1, \\ 2x_i + 1 \mod n & \text{falls } \gamma_i \in G_2, \\ x_i & \text{falls } \gamma_i \in G_3 \end{cases}$$

$$y_i + 1 = \begin{cases} y_1 & \text{falls } \gamma_i \in G_1, \\ 2y_i + 1 \mod n & \text{falls } \gamma_i \in G_2, \\ y_i + 1 \mod n & \text{falls } \gamma_i \in G_3 \end{cases}$$

Pollard Rho

- Zyklische Gruppe G hat nur endlich viele verschiedene Elemente $\Rightarrow$ in der Folge müssen 2 gleiche Gruppenelemente vorkommen. Es gibt also $i \geq 0$ und $k \geq 1$ mit $\gamma_{i+k} = \gamma_i$.

$$\alpha^{x_i}\beta^{y_i} = \alpha^{x_{i+k}}\beta^{y_{i+k}}$$

daraus folgt

$$\alpha^{x_i - x_{i+k}} = \beta^{y_{i+k} - y_i}$$

- Für den diskreten Logarithmus a von β zur Basis α gilt also

$$(x_i - x_{i+k}) \equiv a(y_{i+k} - y_i) \mod n$$

- Falls die Lösung mod n nicht eindeutig ist, muss probiert werden. Wenn zu viele Möglichkeiten bestehen, wiederholt man die gesamte Berechnung mit einem neuen Startwert x_0

Pollard Rho

Pollard Rho(G, n, α, β)

- **procedure** $f(x, a, b)$
  if $x \in G_1$
  then $f \leftarrow (\beta \cdot x, a, (b+1) \mod n)$
  else if $x \in G_2$
  then $f \leftarrow (x^2, 2a \mod n, 2b \mod n)$
  else $f \leftarrow (\alpha \cdot x, (a+1) \mod n, b)$
  return(f)
- **main**
  definiere $G = G_1 \cup G_2 \cup G_3$
  $(x, a, b) \leftarrow f(1, 0, 0)$
  $(x', a', b') \leftarrow f(x, a, b)$
  while $x \neq x'$
  do
  $$\begin{cases} (x, a, b) & \leftarrow f(x, a, b) \\ (x', a', b') & \leftarrow f(x', a', b') \\ (x', a', b') & \leftarrow f(x', a', b') \end{cases}$$
  if $gcd(b' - b, n) \neq 1$
  then return (''failure'')
  else return $((a - a')(b - b')^{(-1)} \mod n)$

Pohlig-Hellman-Algorithmus

Idee

Anstatt den diskreten Logarithmus modulo einer sehr großen Primzahl p zu berechnen, werden die diskreten Logarithmen der Primfaktoren von $p - 1 = n$ ermittelt

- ist die Primfaktorzerlegung von $|G|$ bekannt, lässt sich mit dem Pohlig-Hellman-Algorithmus das Problem der Berechnung des diskreten Logarithmus in G auf dasselbe Problem in Gruppen von Primzahlordnungen reduzieren
- $n = |G| = \prod_{i=1}^{k} p_i^{c_i}$ ist die Primfaktorzerlegung von n

Pohlig-Hellman-Algorithmus

- der Algorithmus berechnet den Wert von $x = a \mod q^c$ für jeden Primfaktor q, wobei x auch als $x = a_0 + a_1 \cdot q^1 + a_2 \cdot q^2 + ... + a_{c-1} \cdot q^{c-1}$ geschrieben werden kann
- wichtige Beobachtung für den Algorithmus: $\beta^{n/q} = \alpha^{a_0 \cdot n/q}$ und allgemein: $\beta_j^{n/q^{j+1}} = \alpha^{a_j \cdot n/q}$

Pohlig-Hellman-Algorithmus

- Vorarbeit: Prim-Faktorisieren der Gruppenordnung $|G|$
- Berechnung von i erfolgt mittels *SHANKS* oder *POLLARD* ρ

Pohlig-Hellman$(G, n, \alpha, \beta, q, c)$

$j \leftarrow 0$

$\beta_j \leftarrow \beta$

solange $j \leq c - 1$

tue
$$\begin{cases} \delta \leftarrow \beta_j^{n/q^{j+1}} \\ \text{suche } i, \text{ sodass } \delta = \alpha^{in/q} \\ a_j \leftarrow i \\ \beta_{j+1} \leftarrow \beta_j \alpha^{-a_j q^j} \\ j \leftarrow j + 1 \end{cases}$$

Rückgabe $(a_0, ..., a_{c-1})$

Pohlig-Hellman-Algorithmus

- Beispiel: $p = 29,\ n = 28 = 2^2 \cdot 7^1,\ \alpha = 2,\ \beta = 18$

Pohlig-Hellman($\mathbb{Z}_{29}*, 28, 2, 18, 2, 2$)

$j \leftarrow 0$

$\beta_0 \leftarrow 18$

solange $j \leq 1$

tue
- $\delta \leftarrow 18^{28/2^{0+1}}\ (\delta = 28)\ 9^{28/2^{1+1}}\ (\delta = 28)$
- suche i, sodass $28\ 28 = 2^{i \cdot 28/2}$
- $a_j \leftarrow 1\ 1$
- $\beta_{j+1} \leftarrow 18 \cdot 2^{-1 \cdot 2^0}\ (\beta_1 = 9)\ 9 \cdot 2^{-1 \cdot 2^1}\ (\beta_2 = 24)$
- $j \leftarrow 1\ 2$

Rückgabe $(1\ 1)$

- $a(2) = 1 + 1 \cdot 2^1 = 3 \rightarrow a \equiv 3 \pmod 4$

Pohlig-Hellman-Algorithmus

Pohlig-Hellman($\mathbb{Z}_{29}*, 28, 2, 18, 7, 1$)

$j \leftarrow 0$

$\beta_0 \leftarrow 18$

solange $j \leq 0$

tue
- $\delta \leftarrow 18^{28/7^{0+1}}$ $(\delta = 25)$
- suche i, sodass $25 = 2^{i \cdot 28/7}$
- $a_j \leftarrow 4$
- $\beta_{j+1} \leftarrow 18 \cdot 2^{-4 \cdot 7^0}$ $(\beta_1 = 12)$
- $j \leftarrow 1$

Rückgabe (4)

- $a(7) = 4 \rightarrow a \equiv 4 \pmod 7$

Pohlig-Hellman-Algorithmus

Lösung zusammenfügen

man erhält folgendes Gleichungssystem:

$$a \equiv 3 \pmod 4$$

$$a \equiv 4 \pmod 7$$

Anwendung des Chinesischen Restsatz ergibt:

$$a \equiv 11 \pmod{28}$$

$\rightarrow \log_2 18 = 11$ in $\mathbb{Z}_{29}$

Pohlig-Hellman-Algorithmus

- die Laufzeit des Pohlig-Hellman-Algorithmus zur Berechnung diskreter Logarithmen ist stark abhängig von der Quadratwurzel des größten Primteilers der Gruppenordnung
- wenn der größte Primteiler zu klein ist, kann der diskrete Logarithmus "leicht" berechnet werden

Beispiel

Primzahl $p = 2 \cdot 3 \cdot 5^{278} + 1$ (binär 649-Bit), die Ordnung der primen Restklassengruppe mod p hat als größten Primteiler die 5
$\longrightarrow$ p ist nicht für das ElGamal-Verfahren geeignet

Index Calculus Method

- sehr effiziente Methode, wenn diskrete Logarithmen in primen Restklassengruppen bestimmt werden sollen ($\mathbb{Z}_p*$)
- ist eng verwandt mit Faktorisierungsverfahren wie dem Quadratischen Sieb u. dem Zahlkörpersieb

Idee

- Bestimmen einer Menge $B = \{p_1, p_2, ..., p_B\}$ von "kleinen" Primzahlen, der Faktorbasis
- im Vorberechnungsschritt werden die Logarithmen aller Elemente der Faktorbasis berechnet
- im 2. Schritt werden die Ergebnisse aus dem 1. Schritt verwendet, um den diskreten Logarithmus von β zu bestimmen

Index Calculus Methode

- in der Vorberechnung erzeugt man sich $|B|$ Kongruenzen der Form:

$$\alpha^{x_j} \equiv p_1^{a_{1_j}} \cdot p_2^{a_{2_j}} \cdot \ldots \cdot p_B^{a_{B_j}} \pmod{p}$$

  umgeformt:

$$x_j \equiv a_{1_j} \log_\alpha p_1 + a_{2_j} \log_\alpha p_2 + \ldots + a_{B_j} log_\alpha p_B \pmod{p-1},$$

  für die es modulo $p-1$ eine eindeutige Lösung gibt (x_j wird dabei zufällig gewählt)
- die diskreten Logarithmen der Elemente der Faktorbasis ergeben sich, wenn man das System aus Kongruenzen modulo jedes Primteilers löst (z.B mit Gaußalgorithmus)

Index Calculus Methode

- im 2. Schritt wählt man sich ein zufälliges s $(1 \leq s \leq p-2)$ und berechnet:

$$\gamma \equiv \beta\alpha^s \mod p$$

- wenn man nun γ über der Faktorbasis faktorisieren kann, erhält man folgende Kongruenz:

$$\beta\alpha^s \equiv p_1^{c_1} \cdot p_2^{c_2} \cdot \ldots \cdot p_B^{c_B} \pmod{p},$$

umgeformt:

$$\log_\alpha \beta + s \equiv c_1 \log_\alpha p_1 + c_2 \log_\alpha p_2 + \ldots + c_B \log_\alpha p_B \pmod{p-1}$$

- da alle Terme außer $\log_\alpha \beta$ in der Kongruenz bekannt sind, kann man die Lösung einfach berechnen

Index Calculus Methode

- $p = 10007$, $\alpha = 5$, Faktorbasis $B = \{2, 3, 5, 7\}$, zufälliges $x_1 = 4063$, $x_2 = 5136$, $x_3 = 9865$
- es werden nur 3 Kongruenzen benötigt, da $\log_5 5 = 1$

Beispiel - Schritt 1

- $5^{4063} \mod 10007 = 42 = 2 \cdot 3 \cdot 7$
  $5^{5136} \mod 10007 = 54 = 2 \cdot 3^3$
  $5^{9865} \mod 10007 = 189 = 3^3 \cdot 7$
- $\log_5 2 + \log_5 3 + \log_5 7 \equiv 4063 \pmod{10006}$
  $\log_5 2 + 3\log_5 3 \equiv 5136 \pmod{10006}$
  $3\log_5 3 + \log_5 7 \equiv 9865 \pmod{10006}$
- Lösen des Kongruenzensystems ergibt: $\log_5 2 = 6578$, $\log_5 3 = 6190$, $\log_5 7 = 1301 \pmod{10006}$

Index Calculus Methode

- $\beta = 9451$, zufälliger Wert $s = 7736$
- es soll also $\log_5 9451$ gelöst werden

Beispiel - Schritt 2

- $9451 \cdot 5^{7736} \mod 10007 = 8400 = 2^4 \cdot 3 \cdot 5^2 \cdot 7$
- $\log_5 9451 = (4 \log_5 2 + \log_5 3 + 2 \log_5 5 + \log_5 7 - s) \mod 10006$
  $\log_5 9451 = (4 \cdot 6578 + 6190 + 2 \cdot 1 + 1301 - 7736) \mod 10006$
  $\log_5 9451 = 6057$
- Probe: $5^{6057} \equiv 9451 \pmod{10007}$

Komplexitätsbetrachtung

- Problem lässt sich vereinfachen, wenn wir $(\mathbb{Z}_p, \cdot)$ verlassen
- Gesucht wird ein Isomorphismus $\phi : (\mathbb{Z}_n, \cdot) \rightarrow (\mathbb{Z}_n, +)$
- Das Problem diskreter Logarithmus in der additiven Gruppe:

$$\alpha a \equiv \beta \pmod{n}$$
$$\text{also } \log_\alpha \beta = \beta\alpha^{-1} \mod n$$

- Ein effizienter Algorithmus zur Bestimmng dieses Isomorphismus ϕ ist allerdings nicht bekannt
- ϕ zwischen 2 willkürlich gewählten Gruppen zu finden, ist ebenso schwer wie das DL Problem

Komplexitätsbetrachtung

ein generischer Algorithmus

- Kodierung von $(\mathbb{Z}_n, +)$ ist injektive Abbildung:
$$\sigma : \mathbb{Z}_n \to G$$
- Eingaben für den Algorithmus: $\sigma_1 = \sigma(1)$, $\sigma_2 = \sigma(a)$
- Orakel liefert die Kodierungen für $\sigma(i)$ und $\sigma(j)$, so dass:
$$\sigma((ci \pm dj) \mod n), \text{ wobei } c, d \in \mathbb{Z}_n$$
- Erzeuge m Paare $(c_i, d_i) \in \mathbb{Z}_n \times \mathbb{Z}_n, 1 \leq m$
- Da σ injektiv, gilt $c_i + d_i a \equiv c_j + d_j a \pmod n \Leftrightarrow \sigma_i = \sigma_j$
- Wenn jetzt $\sigma_i = \sigma_j$, $(i \neq j)$, dann gilt:
$$a = (c_i - c_j)(d_j - d_i)^{-1} \mod n$$

Komplexitätsbetrachtung

- Wahrscheinlichkeit, dass der Algorithmus den richtigen Wert zurückgibt

$$\leq \frac{\binom{m}{2}+1}{n}$$

- Wenn der generische Algorithmus garantiert den DL berechnet, ist die Erfolgswahrscheinlichkeit 1. Also:

$$\frac{\binom{m}{2}+1}{n} \geq 1, \text{ oder } m^2 + m \geq 2n$$

- $m = \Omega\sqrt{n}$ - also unsere untere Grenze für das diskrete Logarithmenproblem von jeglichen generischen Algorithmen in einer Gruppe der Ordnung n.

Zusammenfassung

Laufzeiten

einfacher Algorithmus	$O(n)$
Shanks	$O(\sqrt{n})$
Pollard Rho	$O(\sqrt{n})$
Pohlig-Hellman	$O(c\sqrt{q})$ ($\cdot$ Anzahl der Primfaktoren von n)
Index Calculus Methode	$O(e^{(1+o(1))\sqrt{\ln p \ln\ln p}})$ $O(e^{(1/2+o(1))\sqrt{\ln p \ln\ln p}})$

- *Pohlig-Hellman* nur schneller, weil Vorberechnung angenommen wird - Primfaktorisierung
- *Index Calculus* ist nur auf prime Restklassengruppen anwendbar $\rightarrow$ kein generischer Algorithmus

Dec 22, 2006

$p = \lfloor 2^{446}\pi \rfloor + 63384 = 57085\ldots$ (135 Dezimalstellen, 448 Bit)
$g^x = 11$ $g = 7$
$x = 263809415442532684357793832777626704483700110050961631240336610545143645723034872275030016383962573841181649388892154031068496007427 12$
$\approx$ 44623 MIPS Jahre
In einem Zwischenschritt waren es dann 2.541.033 lineare Gleichungen mit 2.739.345 Unbekannten

Andrey Dorofeev, Denis Dygin, Dmitry Matyukhin[1]
$\approx 42,4$ Jahre auf einem 2 GHz Rechner.

[1] http://www.nabble.com/Discrete-logarithm-in-GF%28p%29—135-digits-tf2870677.html#a8023511

Quellen

- Johannes Buchmann: Einführung in die Kryptographie. Springer Verlag 2003
- Douglas R. Stinson: Cryptography Theory and Practice Second Edition. Chapman & Hall/CRC 2002
- http://de.wikipedia.org
- Vortrag "Public-Key Kryptography mit Diskreten Logarithmen" Jan Schwarz, Kristine Jetzke - 11.01.2005[2]

[2] Dank für die Latexquellen zum Shanksbeispiel!