Die Pollard-ρ-Methode und die Pollard-λ-Methode

Juli 2003

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Gegeben (immer noch):

Endliche abelsche Gruppe G

Ein Element $P \in G$

mit $\text{ord}(P) = n$

Ein Element $Q \in < P >$

Problem:

Bestimme $0 \leq k \leq n - 1$ so dass

$$\boxed{kP = Q}$$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Bisher vorhanden:

Enumeration

- Indiskutabel!

Babystep-Giantstep-Algorithmus

- Vorteil: *Garantierte* Laufzeit $O(\sqrt{n})$
- Nachteil: Ebenso hoher Platzbedarf

Pohlig-Hellman-Verfahren

- Arbeitet nicht selbst, sondern lässt arbeiten (reduziert aber wenigstens die Problemgröße)!

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Daher:

Suche also nach besseren Verfahren!

Leider:

Die schnelleren Algorithmen sind nicht länger deterministisch, sondern basieren auf dem *Geburtstagsparadoxon*

Geschätzte Laufzeit:	$O(\sqrt{n})$	
Garantierter Speicherbedarf:	$O(1)$	(!!!)

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Vereinbarung:

Betrachte das Problem in $(\mathbb{Z}/p\mathbb{Z})^*$
Dann ist $n = \phi(p) = p - 1$.

Dies ist keine Einschränkung!

Vorteile:

- Gewohntere (multiplikative) Schreibweise
- Existenz von Primitivwurzeln

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Gegeben (also nun):

$$
\begin{aligned}
\text{Gruppe} \quad & \mathbb{Z}_p^* \quad := \quad (\mathbb{Z}/p\mathbb{Z})^* \\
\text{Eine Primitivwurzel} \quad & g \\
\text{d.h.} \quad & < g >= \mathbb{Z}_p^* \\
\text{bzw.} \quad & n = \text{ord}(g) = \phi(p) = p-1 \\
\text{Ein Element} \quad & h \in \mathbb{Z}_p^*
\end{aligned}
$$

Problem:

Bestimme $0 \leq x \leq n-1$ so dass

$$\boxed{g^x = h}$$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Grundidee:

Stelle jedes Element y als

$$\boxed{y = g^r h^s}$$

dar.

Suche zwei Darstellungen eines Elements

$$y = g^r h^s = g^{r'} h^{s'}$$

und berechne daraus (wenn möglich) x

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Falls

$$g^r h^s \equiv g^{r'} h^{s'} \mod p \quad \Longleftrightarrow \quad h^{s-s'} \equiv g^{r'-r} \mod p$$

wäre wegen $h = g^x$

$$\begin{aligned} g^{x(s-s')} &\equiv g^{r'-r} \mod p \\ \Rightarrow x \cdot (s-s') &\equiv (r'-r) \mod n \end{aligned}$$

Diese Kongruenz ist zu lösen (Multipliziere mit $(s-s')^{-1}$)!

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Falls aber

$$d := \mathrm{ggT}(s - s', n) \neq 1$$

Teile Gleichung durch d:

$$\tilde{x} \cdot \tilde{s} \equiv \tilde{r} \mod \tilde{n}$$

$$\Rightarrow \tilde{x} \equiv \tilde{r} \cdot (\tilde{s})^{-1} \mod \tilde{n}$$

Achtung: Nicht eindeutig mod n!

Teste dann die Werte

$$x = \tilde{x} + k \cdot \tilde{n} \qquad \text{mit} \qquad 0 \leq k < d$$

Falls d dafür zu groß, starte neu!

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Wie findet man nun ein solches Element mit zwei Darstellungen?

Definiere eine Funktion $F : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$

Wähle einen Startpunkt y_0 (mit bekannten r, s)

Iteriere dann

$$y_{n+1} = F(y_n)$$

(wobei die neuen r, s auch bekannt sein müssen)

Es wird irgendwann ein Element auftauchen, dass schon einmal aufgetaucht ist!

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Wann wird dies denn aber passieren?

Annahme: Die Folge $\{y_n\}$ verhält sich wie eine Zufallsfolge

Dann ist nach $\sqrt{\pi/2} \cdot \sqrt{n} \approx 1.25\sqrt{n}$ Folgengliedern ein doppeltes Vorkommen zu erwarten!

Begründung

$\rightarrow$ Geburtstagsparadoxon

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Exkurs: Geburtstagsparadoxon
Wie viele Personen müssen in einem Raum sein, damit zwei von ihnen mit mehr als 50% Wahrscheinlichkeit am gleichen Tag Geburtstag haben?

Seien k Personen im Raum. Dann ist die Wahrscheinlichkeit für das Gegenteil (alle Geburtstage verschieden):

$$\bar{P} = 1 \cdot \frac{364}{365} \cdot \frac{363}{365} \cdot \dots \cdot \frac{366-k}{365}$$

Dies ist bereits für $k = 23$ kleiner als 50%.

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Exkurs: Geburtstagsparadoxon (Fortsetzung)
Berechne die erwartete Anzahl von Folgengliedern, unter denen eine Wiederholung existiert:

Sei X die Zufallsvariable die angibt wie viele Ziehungen noch *ohne* Wiederholung gemacht werden konnten:

$$\begin{aligned}
P(X > k) &= 1 \cdot (1 - 1/n) \cdot (1 - 2/n) \cdot \cdots \cdot (1 - (k-1)/n) \\
&= \prod_{i=0}^{k-1} (1 - \frac{i}{n}) \\
& \qquad\qquad \text{und wg. } 1 + x \leq e^x \\
&\leq \prod_{i=0}^{k-1} e^{-i/n} \\
&= e^{-\sum_{i=0}^{k-1} i/n} = e^{-k(k-1)/(2n)} \approx e^{-k^2/(2n)}
\end{aligned}$$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Exkurs: Geburtstagsparadoxon (Fortsetzung)

Der Erwartungswert ist

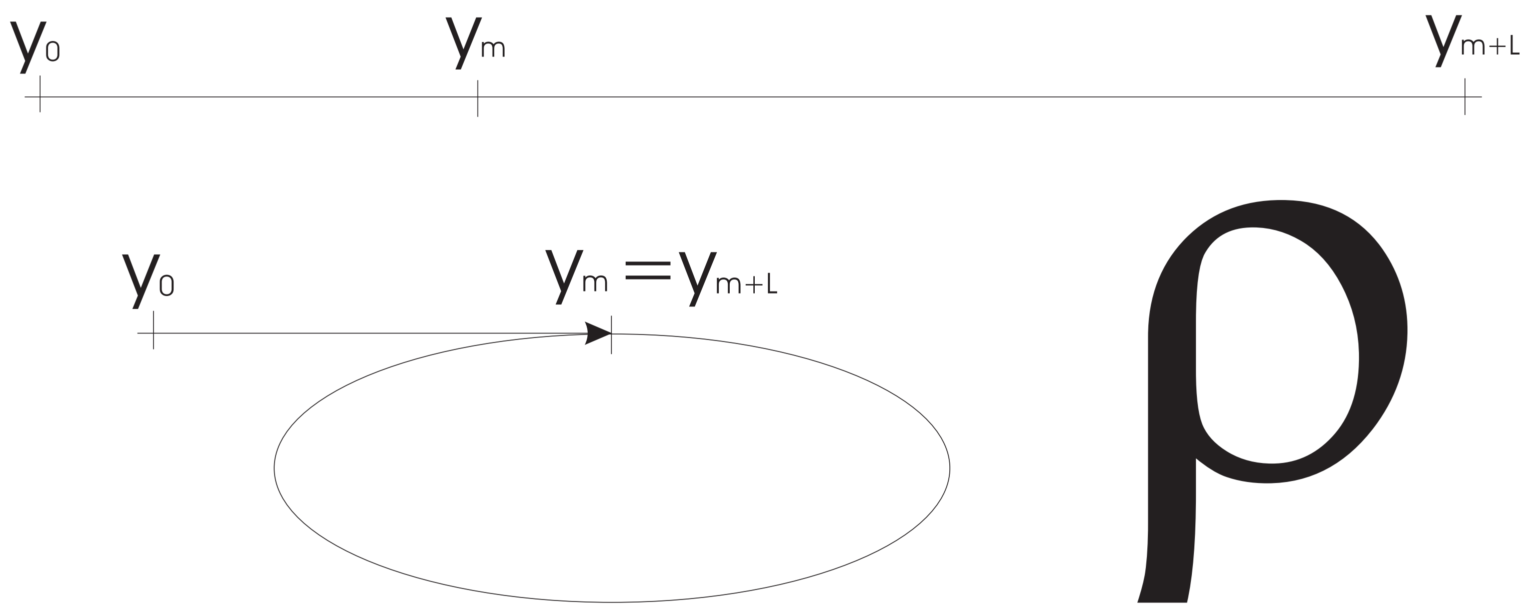
$$E(X) = \sum_{k=1}^{\infty} k \cdot P(X = k) \;=\; \sum_{k=1}^{\infty} k \cdot [P(X > k-1) - P(X > k)]$$
$$= \sum_{k=0}^{\infty} P(X > k)$$

Also ist

$$E(X) = \sum_{k=0}^{\infty} P(X > k) \approx \sum_{k=0}^{\infty} e^{-(k^2)/(2n)} \approx \int_0^{\infty} e^{-(k^2)/(2n)}\, dk = \sqrt{\frac{\pi n}{2}}$$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Nach ca. $\sqrt{\pi/2} \cdot \sqrt{n}$ Elementen wird also die Folge zyklisch mit Periodenlänge L.



Die Pollard-ρ-Methode und die Pollard-λ-Methode

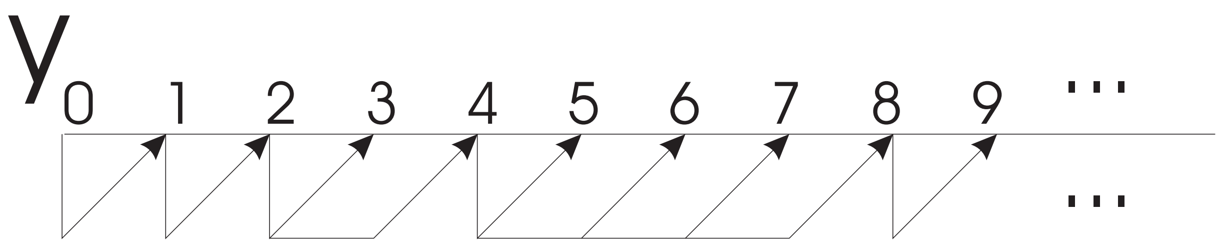
Kann man vermeiden alle Elemente speichern zu müssen?

Antwort: Ja!

$\rightarrow$ Algorithmus von Brent:

Vergleiche

- y_0 mit y_1
- y_1 mit y_2
- y_2 mit y_3, y_4
- y_4 mit y_5, y_6, y_7, y_8
- y_8 mit ...
- ...



Die Pollard-ρ-Methode und die Pollard-λ-Methode

Das heißt:

Abgespeichert wird: y_i
Verglichen wird mit y_k wobei $k = (i+1), \ldots, 2i$
Dann wird abgespeichert: y_{2i}

Übereinstimmung wird gefunden wenn

$$2i - i \geq L$$

Klar ist:

Brent's Algorithmus findet fast immer *nicht* die erste Übereinstimmung.

Aber:

Die Übereinstimmung wird nach ca. $\boxed{1.97\sqrt{n}} = O(\sqrt{n})$ Iterationen gefunden.

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Letzte konkrete Frage:

Wie sieht die Abbildung F aus?

Pollard schlug vor:

$$F(y) = \begin{cases} yg & \text{falls } \quad 0 < y \leq 1/3p \\ y^2 & \text{falls } 1/3p < y \leq 2/3p \\ yh & \text{falls } 2/3p < y \leq p \end{cases}$$

Erinnerung:

Darstellung $y = g^r h^s$
Identifiziere dies mit $\{y, r, s\}$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Auf $y = g^r h^s \simeq \{y, r, s\}$ arbeitet F konkret also so:

$$F(\{y,r,s\}) = \begin{cases} \{yg, & r+1, & s\} & \text{falls} \quad 0 < y \leq 1/3p \\ \{y^2, & 2r, & 2s\} & \text{falls } 1/3p < y \leq 2/3p \\ \{yh, & r, & s+1\} & \text{falls } 2/3p < y \leq p \end{cases}$$

Damit kann die Methode implementiert werden!

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Was ist wenn nicht alle Lösungen

$$0 \leq x \leq n-1$$

in Betracht kommen, sondern nur

$$a \leq x \leq b$$

Fakt:

Die Pollard-Rho-Methode kann diese Zusatzinformation nicht ausnutzen!

Suche nach besserer Methode:

$\longrightarrow$ Pollard-Lambda-Methode

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Die Pollard-Lambda-Methode heißt auch

„Pollard-Kangaroo-Algorithm“
oder
„Catching Kangaroos“

Man hat

- Ein zahmes Känguru (= `TameKangaroo`)
- Ein wildes Känguru (= `WildKangaroo`)
- Eine Menge von Sprüngen $J = \{j_0, \ldots, j_r\}$ mit

$$j_i = g^{(\delta_i)}$$

- Eine Hash-Abbildung $f : \mathbb{Z}_p^* \to \{0, \ldots, r\}$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Die Kängurus springen nun deterministisch in $\mathbb{Z}_p^*$ herum, d.h. eine Folge wird rekursiv so definiert:

$$y_{k+1} = y_k \cdot j_{f(y_k)}$$

Grundidee:

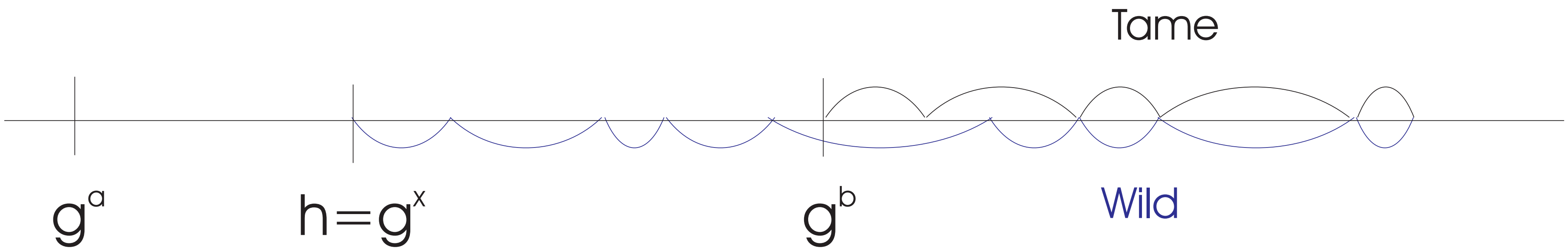
Lasse das zahme Känguru eine gewisse Anzahl Sprünge machen, und lege dort eine Falle aus.

Lasse das wilde Känguru springen. Vielleicht tappt es in die Falle, und ermöglicht die Berechnung von x.

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Das zahme Känguru startet bei g^b
und legt die Exponentendistanz δ_t zurück

Das wilde Känguru startet bei $h = g^x$
und legt die Exponentendistanz δ_w zurück



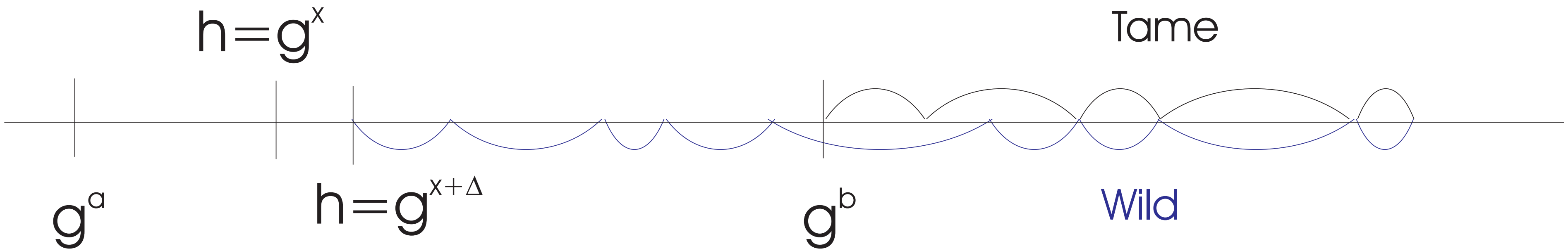
Man kann dann x direkt (!) berechnen:

$$x = b + \delta_t - \delta_w$$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Tappt das wilde Känguru nicht in die Falle lässt man es von einer anderen (nahen) Position

$$h \cdot g^{\Delta} = g^{x+\Delta}$$



erneut starten, und berechnet

$$x = b + \delta_t - \delta_w - \Delta$$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Frage:

Wie sehen nun konkrete Parameter aus?

Gute Werte in der Praxis sind folgende:

- Wähle r so dass der Mittelwert von $\{2^0, 2^1, \ldots, 2^r\} \approx \frac{\sqrt{b-a}}{2}$
- $$J = \{g^{(2^0)}, g^{(2^1)}, \ldots, g^{(2^r)}\}$$
- Das zahme Känguru macht $T = 0.7\sqrt{b-a}$ Sprünge
- Das wilde Känguru macht $W = 2.7\sqrt{b-a}$ Sprünge
- Hash-Funktion

$$\begin{aligned} f : \mathbb{Z}_p^* &\rightarrow \{0, \ldots, r\} \\ x &\mapsto x \mod (r+1) \end{aligned}$$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Wie lange braucht nun diese Methode?

Empirische Antwort:

- Immer:

  $0.7\sqrt{b-a}$ Operationen für das zahme Känguru

- Mit $p = \frac{3}{4}$:

  Das wilde Känguru <u>fällt</u> in die Falle.

  Erwartete Anzahl von Sprüngen dann: $1.7\sqrt{b-a}$

- Mit $p = \frac{1}{4}$:

  Das wilde Känguru <u>fällt nicht</u> in die Falle.

  Erwartete Anzahl von Sprüngen dann:

$$\left(\left(\frac{1}{4}\right) + \left(\frac{1}{4}\right)^2 + \ldots\right) 2.7\sqrt{b-a} = \frac{1}{3} \cdot 2.7 \cdot \sqrt{b-a}$$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Die Pollard-Lambda-Methode braucht also ca.

$$(0.7 + 0.33 \cdot 2.7 + 1.7)\sqrt{b-a}$$

$$\boxed{\approx 3.3\sqrt{b-a}}$$

Gruppenoperationen.

Erinnerung:

Pollard-Rho:

$$\boxed{\approx 1.97\sqrt{n}}$$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Vergleich:

$\Rightarrow$ Die Pollard-Lambda-Methode lohnt erst, wenn

$$\sqrt{b-a} < 0.6\sqrt{n}$$

bzw.

$$b-a < 0.4 \cdot n$$

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Vorteile beider Methoden:

Beide lassen sich sehr gut parallelisieren!

Bei m Prozessoren lässt sich ein
Speed-Up-Faktor von m erreichen.

(Dies ist *nicht* sofort klar, denn der intuitive Ansatz bringt nur einen Faktor von $\sqrt{m}$)

Die Pollard-ρ-Methode und die Pollard-λ-Methode

Weiterführende Literatur:

- *Monte Carlo methods for index computation* (*mod* p)
  J. M. Pollard

- Diverse Texte zu Diskreten Logarithmen
  Edlyn Teske

- *Parallel Collision Search with Cryptanalytic Applications*
  Paul C. van Oorschot and Michael J. Wiener