Performance Analyse des Babystep-Giantstep-Algorithmus

E-Voting Seminar

Juni 2011

Hochschule für Technik Rapperswil (HSR)

Marco Keller

ABSTRACT

Dieser Report untersucht die Anwendung des Babystep-Giantstep-Algorithmus im E-Voting. Der Algorithmus basiert auf der Idee, dass zuerst Zwischenergebnisse berechnet und in einer Tabelle gespeichert werden. In einem zweiten Schritt kann dann anhand dieser Tabelle der diskrete Logarithmus berechnet werden. Mit dieser Methode müssen viel weniger Gruppenoperationen gemacht werden als bei einer Enumeration nötig wären.

Eine Implementation des Babystep-Giantstep-Algorithmus wurde gemacht um sein Laufzeitverhalten zu messen. Aufbauend auf diesen Resultaten wurde ein Modell zur Vorhersage der Laufzeit entwickelt. Bei Wahlen mit grosser Teilnehmerzahl kann so die Laufzeit des diskreten Logarithmus abgeschätzt werden.

EINFÜHRUNG

Im E-Voting werden homomorphe Verfahren zum Auszählen von Stimmen verwendet. Das ElGamal [WEl11] oder das Paillier [WPa11] sind solche Verfahren, bei denen eine verschlüsselte Stimme mittels einer Gruppenoperation zu einem Wahlergebnis zusammengeführt werden können ohne dabei den Wert der verschlüsselten Stimmen zu verändern.

ElGamal ist ein kryptographisches Verfahren deren homomorphe Verschlüsselung eine multiplikative Gruppenfunktion ist. Um ElGamal beim E-Voting einzusetzen benötigt man aber additive Gruppenfunktionen, da das Ergebnis einer Wahl die Addition der Stimmen darstellt. Um dies mit ElGamal zu erreichen muss das Verfahren modifiziert werden. Dazu wird eine Stimme nicht normal verschlüsselt, sondern als Exponent einer gemeinsamen Zahl genommen und dann verschlüsselt. Ein Nachteil dieses modifizierten Verfahrens ist, dass es zur Entschlüsselung des Wahlergebnisses den diskreten Logarithmus benötigt. Dabei sind die Grösse der Gruppe und die Grösse des Suchraums entscheidende Faktoren für die Performance des diskreten Logarithmus. Der Suchraum ist dabei eine Teilmenge der Gruppe. Dessen Grösse kann auf die Anzahl der Stimmen beschränkt werden.

Bei anderen kryptographischen Verfahren, die auf der Schwierigkeit des diskreten Logarithmus beruhen, muss jeweils der ganze Gruppenraum durchsucht werden um einen Exponenten zu

finden. Beim E-Voting Verfahren mit Ja-Nein-Stimmen entspricht die maximale Grösse des Suchraums für den diskreten Logarithmus der Anzahl der Stimmen. Da die Grösse des Suchraums deutlich kleiner als die Gruppengrösse ist, kann der diskrete Logarithmus bei der Auswertung des Stimmergebnisses mit dem ElGamal Verfahren zur Anwendung kommen.

Dieser Report zeigt die Resultate einer Implementation des diskreten Logarithmus im Bezug auf verschiedene Gruppen- und Suchraumgrössen. Implementiert wurde dabei der Babystep-Giantstep-Algorithmus. Anhand der gemessenen Resultate wurde ein Modell entwickelt um Laufzeitvorhersagen machen zu können.

Es werden weitere Algorithmen zur Berechnung des diskreten Algorithmus betrachtet und deren Vor- und Nachteile im Bezug auf den Babystep-Giantstep-Algorithmus.

DISKRETER LOGARITHMUS

Der diskrete Logarithmus ist die Umkehrfunktion der diskreten Exponentiation. Die Sicherheit mancher kryptographischer Verfahren beruht darauf, dass der diskrete Logarithmus in bestimmten Gruppen schwierig zu lösen ist. Diese kryptographischen Verfahren sind z.B. ElGamal oder das Diffie-Hellman Schlüsselaustausch Verfahren. Der diskrete Logarithmus erlangt dadurch an Bedeutung für die Kryptographie.

Sei G eine zyklische Gruppe mit der Primzahl p und der Ordnung $n = p - 1$. Sei g ein Generator modulo p.
Für alle Gruppenelemente $a \in \{1, 2, \dots, n\}$ gibt es genau einen Exponenten $x \in \{0, 1, \dots, n - 1\}$, so dass

$$a \equiv g^x \bmod p \tag{1}$$

gilt.
Somit ist der Exponent x der diskrete Logarithmus von a zur Basis g.

$$x \equiv \log_g a \bmod p \tag{2}$$

ENUMERATION

Mit dem Enumerationsverfahren wird für alle Exponenten x = 0, 1, 2, ..., n-1 geprüft, ob sie (1) erfüllen. Ist dies der Fall hat man den diskreten Logarithmus gefunden. Diese vollständige Suche ist die einfachste Methode um den diskreten Logarithmus zu finden. Sie benötigt aber O(n) Gruppenoperationen und ist daher für grosse n in annehmbarer Zeit nicht durchführbar.

BABYSTEP-GIANTSTEP-ALGORITHMUS

Der Babystep-Giantstep-Algorithmus von Shanks [Sha71] ist ein Ansatz zur schnelleren Berechnung des diskreten Logarithmus. Der Algorithmus teilt die Suche in zwei Teile auf. Im ersten Teil wird eine Tabelle mit berechneten Werten erstellt. Im zweiten Teil wird ein Eintrag in der Tabelle gesucht der mit einer ergänzenden Berechnung übereinstimmt. Kann solch eine Übereinstimmung gefunden werden, ist der diskrete Logarithmus gefunden.

Zur Berechnung sind viel weniger Gruppenoperationen als bei der Enumeration nötig. Der Babystep-Giantstep-Algorithmus benötigt $O(\sqrt{n})$ Gruppenoperationen. Dafür ist Speicherplatz

für die Erstellung der Tabelle notwendig. Es müssen $O(\sqrt{n})$ Elemente in der Tabelle gespeichert werden.

BERECHNUNG
Zuerst wird $m = \lceil \sqrt{n} \rceil$ berechnet.

Der diskrete Logarithmus *x* aus (1) lässt sich zerlegen in

$$x = i * m + j, \qquad 0 \leq i < m \ und \ 0 \leq j < m \tag{3}$$

so dass *j* der Rest der Division von *i* durch *m* ist.

(1) lässt sich nun umformen in

$$a \equiv g^x \equiv g^{i*m+j} \ mod \ p. \tag{4}$$

Durch eine Umformung von (4) erhält man z.B.

$$(g^m)^i \equiv a g^{-j} \ mod \ p \tag{5}$$

Beide Seiten von (5) können nun für alle $i \in \{0..m\}$ und für alle $j \in \{0..m-1\}$ berechnet werden. Wenn eine Belegung für *i* und *j* gefunden wird, so dass (5) erfüllt ist, kann nach (4) der diskrete Logarithmus *x* berechnet werden mit

$$x = i * m + j. \tag{6}$$

Je eine Seite von (5) nennt man Giantstep bzw. Babystep.

Zuerst wird nun eine Tabelle für die Menge aller Babysteps angelegt. Dazu kann eine Hashtabelle verwendet werden. Die Menge der Babysteps ist wie folgt gegeben:

$$B = \{(a g^{-j} \ mod \ p, j) : 0 \leq j < m\} \tag{7}$$

Dabei entspricht $a g^{-j} mod \ p$ dem Key und *j* dem Value des Hashtabelleneintrages.

Wird bei der Erstellung der Babystep-Hashtabelle ein Paar der Form $(1, j)$ gefunden kann der diskrete Logarithmus direkt ausgelesen werden mit $i = 0$ und $x = j$.

Ansonsten werden nun inkrementell solange Giantsteps berechnet bis eine Übereinstimmung mit einem Key in der Babystep Hashtabelle gefunden wird. Die Berechnung der Giantsteps erfolgt über

$$(g^m)^i \ mod \ p, \forall \ i \ \in \{1 \leq i \leq m\} \tag{8}$$

Wird ein Babystep Eintrag der Form $((g^m)^i \ mod \ p, j)$ gefunden kann nach (6) der diskrete Logarithmus berechnet werden.

BEISPIEL
Gegeben sei eine zyklische Gruppe mit dem primitiven Element g=2, der Primzahl p=211 und einem Element der Gruppe a=21.

Der diskrete Logarithmus von 21 zur Basis 2 wird berechnet.

Es wird die Gruppenordnung $n = \varphi(211) = 211 - 1 = 210$ und $m = \lceil\sqrt{210}\rceil = 15$ berechnet. Die Menge der Babysteps wird berechnet mit $B = \{(21 * 2^{-j} \; mod \; 211, j) : 0 \leq j < 15\}$ und ergibt folgende Tabelle:

B = {(21, 0), (116, 1), (58, 2), (29, 3), (120, 4), (60, 5), (30, 6), (15, 7), (113, 8), (162, 9), (81, 10), (146, 11), (73 ,12), (142, 13), (71, 14)}

Für die Berechnung der Giantsteps wird nun $(2^{15})^i \; mod \; 211, \forall \; i \in \{1 \leq i \leq 15\}$ berechnet. Dies ergibt folgende Giantsteps:

63, 171, 12, 123, 153, 144, 210, 148, 40, 199, 88, 58.

Beim Giantstep mit i=12 findet man in der Babystep-Tabelle den entsprechenden Eintrag (58, 2). Somit gilt nach (5)

$$(2^{15})^{12} \equiv 21 * 2^{-2} \; mod \; 211$$

Nach (4) gilt

$$21 \equiv 2^{12*15+2} \; mod \; 211$$

Damit ist der diskrete Logarithmus nach (6)

$$x = 12 * 15 + 2 = 182.$$

IMPLEMENTATION

LAUFZEITABSCHÄTZUNG

Die Platzbedarf- und Laufzeitabschätzung für die Berechnung des diskreten Logarithmus bei einer Gruppengrösse mit Ordnung $n = 2^{80}$ ergibt folgendes.

Ein Eintrag in der Hashtabelle benötigt ein Key-Value-Paar. Der Key kann einen Wert im Range $[0 \ldots 2^{80}]$ annehmen und benötigt 80Bit. Der Value kann einen Wert im Range $[0 \ldots 2^{40}]$ annehmen und benötigt 40Bit. Ein Eintrag benötigt somit 120Bit. Vernachlässigt man den Platzbedarf für die Buckets der Hashtabelle und die Verknüpfungen der Einträge, ergibt sich bei 2^{40} Einträgen ein Platzbedarf von 15TB.

Für die Berechnung der Giantsteps, welche dann in der Tabelle gesucht werden müssen wird folgende Abschätzung gemacht. Wenn man annimmt, ein Prozessor kann pro Sekunde 10^9 Gruppenoperationen berechnen, werden für 2^{40} Gruppenoperationen etwa 18 Minuten benötigt.

Die Berechnung der Babystep-Werte, ebenfalls über 2^{40} Gruppenoperationen, wird etwa gleich lange dauern. Jedoch kommt dann noch das Einfügen des Eintrags in die Hashtabelle hinzu. Dies kann je nach Implementation oder auftretenden Kollisionen unterschiedlich lange dauern.

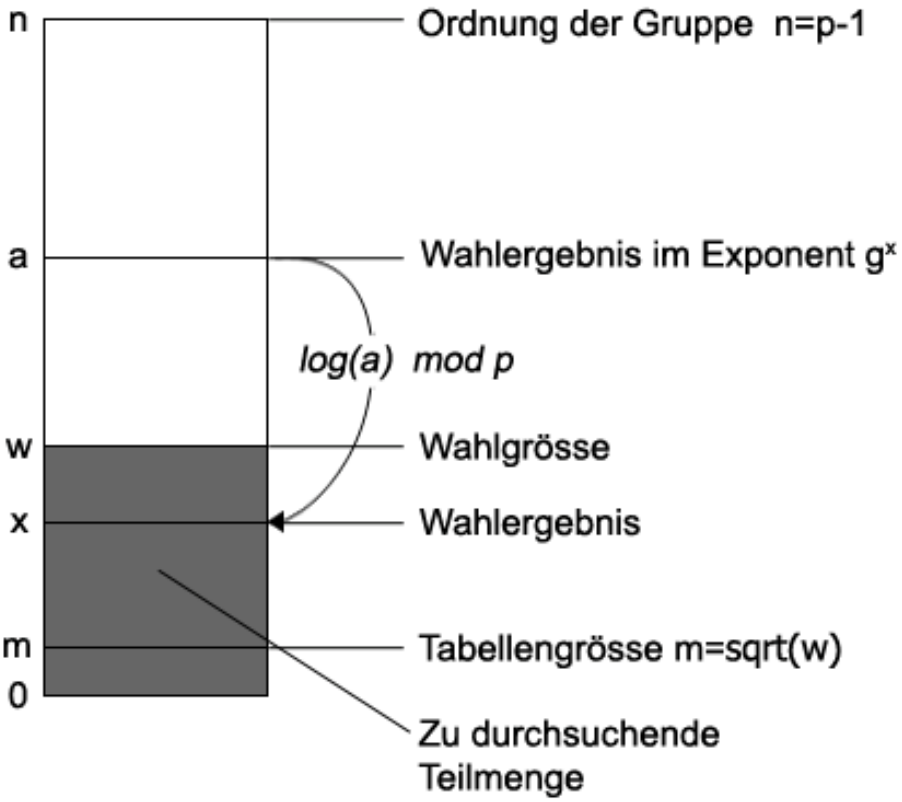
Eine obere Grenze für die Berechnung des diskreten Logarithmus liegt nach Buchmann [Buc03] bei einer Gruppengrösse mit der Ordnung $n = 2^{160}$. Für grössere n ist der Babystep-Giantstep-Algorithmus in der Praxis nicht mehr machbar.

E-VOTING ANFORDERUNGEN

Bei einer Wahl mit dem ElGamal Verfahren werden Gruppengrössen von $n = 2^{1024}$ bis $n = 2^{2048}$ verwendet.

Da der Babystep-Giantstep-Algorithmus mit einer vollständigen Durchsuchung der Gruppe, ein Limit bei $n = 2^{160}$ hat, kann für die Bestimmung des diskreten Logarithmus mit Werten von $n = 2^{1024}$ die Gruppe nicht mehr vollständig durchsucht werden. Beim E-Voting genügt aber die Durchsuchung einer Teilmenge der Gruppe.

Figur 1 zeigt die Teilmenge der Gruppe, die durchsucht werden muss. Nach der Wahl erhält man ein Wahlergebnis *a*. Aus diesem Wahlergebnis wird der diskrete Logarithmus *x* gesucht. Dieses *x* kann sich nur in der grauen Teilmenge befinden, da alle möglichen Wahlergebnisse in dieser Teilmenge enthalten sind. Die Teilmenge enthält alle Werte von 0 bis *w*. Die Wahlgrösse *w* bezeichnet den grössten Wert, der als Wahlergebnis auftreten kann. Die Tabellengrösse *m*, welche beim Babystep-Giantstep-Algorithmus die Grösse der Hashtabelle und die Anzahl der Suchiterationen ausmacht, wird mit $m = \lceil \sqrt{w} \rceil$ berechnet.



Figur 1: Die Teilmenge der Gruppe, die für die Bestimmung des diskreten Logarithmus im E-Voting durchsucht werden muss

Zur Erläuterung wird folgendes Szenario angenommen:

An einer Wahl nehmen 10^6 Wähler teil. Es stehen 2 Kandidaten zur Wahl. Für den Kandidaten A wird eine 0 zum Wahlergebnis addiert, für Kandidat B wird eine 1 zum Wahlergebnis addiert. Das Wahlergebnis muss somit einen Wert im Range $[0 \ldots 10^6]$ annehmen. Dieser Range

entspricht der Teilemenge die mit dem Babystep-Giantstep-Algorithmus durchsucht werden muss.

Die Wahlgrösse *w* entspricht somit dem maximalen Wahlergebnis bei Einstimmigkeit von $w = 10^6$. Daraus ergibt sich eine Tabellengrösse von $m = 10^3$.

Mit einer Hashtabellen-Eintragsgrösse von 8 Byte für ein Key-Value-Paar benötigt man nun 8KB Speicherplatz für *m* Tabelleneinträge. Die Zeit für die Berechnung aller Gruppenoperationen in der Teilmenge beträgt $\frac{10^9/s}{10^3} = 1\mu s$.

VARIANTEN

Je nach Umformung der Gleichung (4) können mehrere Varianten des Babystep-Giantstep-Algorithmus implementiert werden. Da die Berechnung der Babystep-Tabelle aber am meisten Zeit in Anspruch nimmt, ist eine Variante des Algorithmus gefragt, die es erlaubt die Tabelle vor der Wahl zu berechnen. Bei der Wahl selbst muss dann die vorberechnete Tabelle nur noch durchsucht werden.

Gegeben sei das Wahlset S:

*S = {p:*Primzahl*,*
*g:*Generator*,*
*m:*Tabellengrösse*,*
*a:*Wahlergebnis*}*

Der einzige Parameter aus S der von der Wahl abhängt und somit bei der Vorausberechnung nicht zur Verfügung steht, ist das Wahlergebnis *a*.

Betrachtet man die Gleichung $(g^m)^i \equiv ag^{-j} \; mod \; p$ (5), so befindet sich a im Babystep-Teil der Gleichung. Wird nun zur Erstellung der Babystep-Tabelle aber der Teil der Gleichung ohne den Parameter *a* verwendet, kann die Babystep-Tabelle bereits im Voraus berechnet werden.

ALGORITHMUS

Die Implementation basiert auf eine Variante des Babystep-Giantstep-Algorithmus, bei der eine Vorausberechnung der Hashtabelle möglich ist. Aus Gleichung (4) wird folgende Variante gewählt:

$$g^j \equiv a(g^{-m})^i \; mod \; p \qquad (9)$$

Die Babystep-Tabelle wird mit $g^j mod \; p$ erstellt, für alle j von 0 bis *m-1*.

Die Giantstep-Suche wird mit $a((g^{-m})^i \; mod \; p$, für all i von 0 bis *m* gemacht.

Für eine schnellere Berechnung in der Giantstep-Suchschleife wir der Zwischenergebniswert $\alpha \equiv g^{-m} \; mod \; p$ ebenfalls im Voraus berechnet.

LISTING 1
Eingabe: Ein Wahlset S

Ausgabe: Ein Wert *x*, der die Bedingung $a \equiv g^x \ mod\ p$ erfüllt.

```
1. For all j from 0 to (m-1):
2.   δ = g^j mod p
3.   Erstelle einen Eintrag in der Hashtabelle mit dem Paar (δ, j).
4. α = g^(-m) mod p
5. y = a
6. For all i from 0 to m:
7.   Suche Key y in der Hashtabelle
8.   Wenn ein Eintrag gefunden wird:
9.   return x = i * m + j
10.  sonst:
11.     y = y * α mod p
```

BIBLIOTHEK FÜR GROSSE ZAHLEN
Da der Algorithmus mit grossen Zahlen umgehen können muss, wurde die GNU Multiple Precision Arithmetic Library (GMP) [GMP91] verwendet. Sie stellt arithmetische Funktionen zur Verfügung, die auf beliebig grosse Zahlen angewandt werden können.

HASHTABELLE
Bei der Implementation des Algorithmus wurde die Map der Standard Tempalte Library (STL) verwendet. Diese basiert auf einen ausbalancierten Baum.

Um *m* Einträge einzufügen wird der Aufwand auf O(*m* log *m*) abgeschätzt. Da die Tabelle jedoch im Voraus zur Wahl berechnet werden kann, ist die Performance beim Einfügen für die Wahl selbst nicht so kritisch.

Während der Wahl ist es jedoch wichtig, dass die Lockups auf die Hashtabelle schnell gemacht werden können. Einen Lockup wird mit O(log *m*) abgeschätzt.

LAUFZEITMESSUNGEN
Um abschätzen zu können wie viel Zeit ein PC zur Berechnung des Wahlergebnisses benötigt, wurde der Babystep-Giantstep Algorithmus anhand zweier Parameter gemessen.

1. Es wurde gemessen wie sich verschieden grosse Gruppen auf die Laufzeit auswirken. Gemessen wurden Grössen von *N*=1024 Bit bis *N*=5120 Bit, mit jeweils 512 Bit Intervall.
2. Es wurde gemessen wie sich verschieden grosse Teilmengen auf die Laufzeit auswirken. Gemessen wurden Wahlgrössen von *w*=1E+06 bis *w*=1E+10.

Die Bitgrössen *N* beziehen sich auf die Primzahl einer Gruppe. Dazu wurde anhand eines Bitwertes die nächste Primzahl gesucht. Z.B. für die Grösse von 1024 Bit wurde die nächste Primzahl gesucht, die grösser als 2^{1024} ist. Dies ist die minimale Grösse, welche in der Kryptographie als sicher angesehen werden kann. Für die Kryptographie als sicher angesehen wird ein Wert von 2^{2048} Bit. Die Auswertung des Algorithmus soll zeigen, ob der diskrete Logarithmus bei einer sicheren Gruppengrösse, innerhalb einer Minute gefunden werden kann.

Die Wahlgrösse gibt an wie hoch der maximale Wert eines Wahlergebnisses sein kann. Eine grössere Wahlgrösse ermöglicht entweder mehr Wähler oder mehr Wahloptionen.

Die Laufzeitmessungen wurden an zwei Stellen im Algorithmus durchgeführt. Diese Stellen messen aufwändige Teile zur Berechnung des diskreten Logarithmus. Es sind dies:

1. Die Erstellung der Babystep-Tabelle mit *m* Einträgen.
2. Das Durchsuchen der Tabelle nach der passenden Giantstep-Übereinstimmung.

Es wurde die Annahme getroffen, dass ein Wahlalgorithmus auf einen Satz vorberechneter Primzahlen zugreifen kann. Diese müssen dann nicht jedes Mal neu berechnet werden. Beim gemessenen Algorithmus wurden alle Primzahlen bei der Entwicklung fest in den Code eingegeben.

Beim Einfügen in die Tabelle werden jedes Mal *m* Einträge erstellt und eingefügt. Das Einfügen beinhaltet die Berechnung der Zahl und dessen Einfüge-Operation in die Hashtabelle.

Die Giantstep Suche ist vom Wahlergebnis abhängig. Die Laufzeit dazu liegt zwischen 1 und *m* Suchabfragen. Für den Test wurden immer *m* Suchabfragen gemessen. Dies entspricht dem schlechtesten Fall. Im besten Fall könnte der berechnete Kandidat gerade in der ersten Runde gefunden werden. Als Suchabfrage ist die Berechnung eines möglichen Kandidaten und dessen Suche in der Tabelle gemeint.

Die Laufzeitmessungen wurden mit einer Intel Xeon CPU mit 2.26 GHz und 4 GB RAM gemacht. Zur Bestimmung eines Ergebnisses wurden jeweils fünf Messungen durchgeführt und dann der Mittelwert genommen.

RESULTATE

Der allgemeine Zeitrahmen für die Messungen beträgt 60 Sekunden. Die Auswertungen sollen zeigen mit welchen Grössen innerhalb der 60 Sekunden gearbeitet werden kann.

Die einzelnen Messungen wurden 5 Mal durchgeführt. Der angegebene Wert entspricht dann dem Mittelwert der Messungen.

MODELLIERUNG EINER ABSCHÄTZUNGSFUNKTION

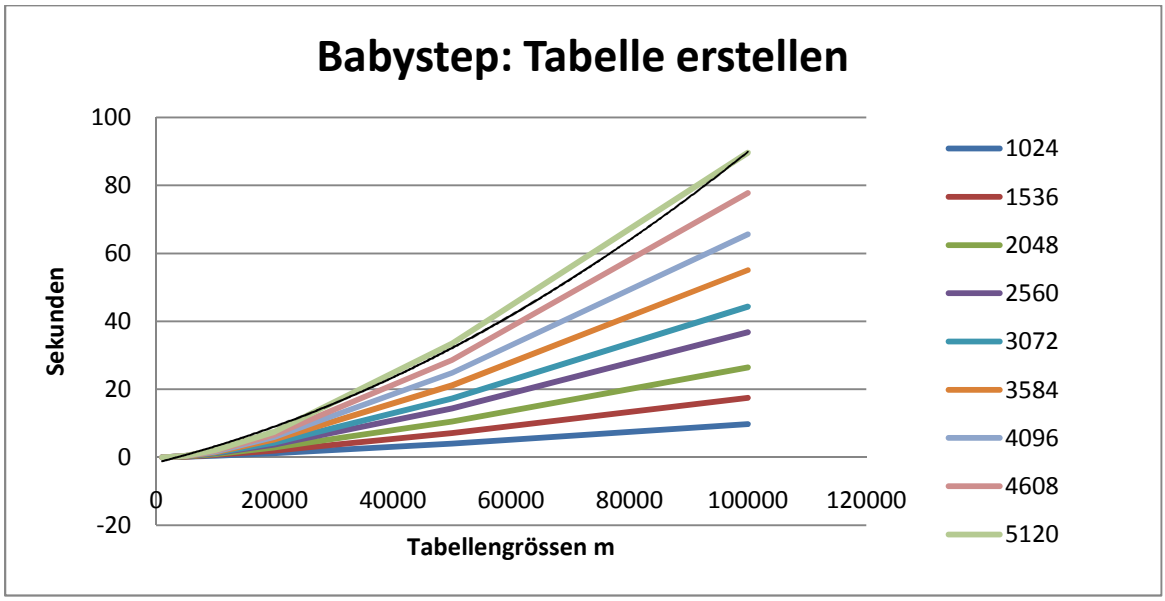
Um die Laufzeitkomplexität für beliebige Tabellengrössen *m* genauer abschätzen zu können, wird eine Funktion für die Tabellengrösse *m* und die Bitgrösse *N* modelliert.

Mit linearer Regression werden Koeffizienten anhand von gemessenen Werten berechnet. Diese Koeffizienten bilden dann die Basis um für beliebige Argumente die resultierenden Funktionswerte hervor zusagen.

Zur Überprüfung des Modells werden einige Werte mit dem Modell vorhergesagt. Danach werden diese Werte mit den gemessenen Werten verglichen. Die Differenzen zwischen den gemessenen und den vorhergesagten Werten werden betrachtet um das Modell zu bewerten. Dabei sollen sowohl für kleine Argumente, als auch für grosse Argumente die Abweichungen klein bleiben.

Erstellen der Babystep Tabelle

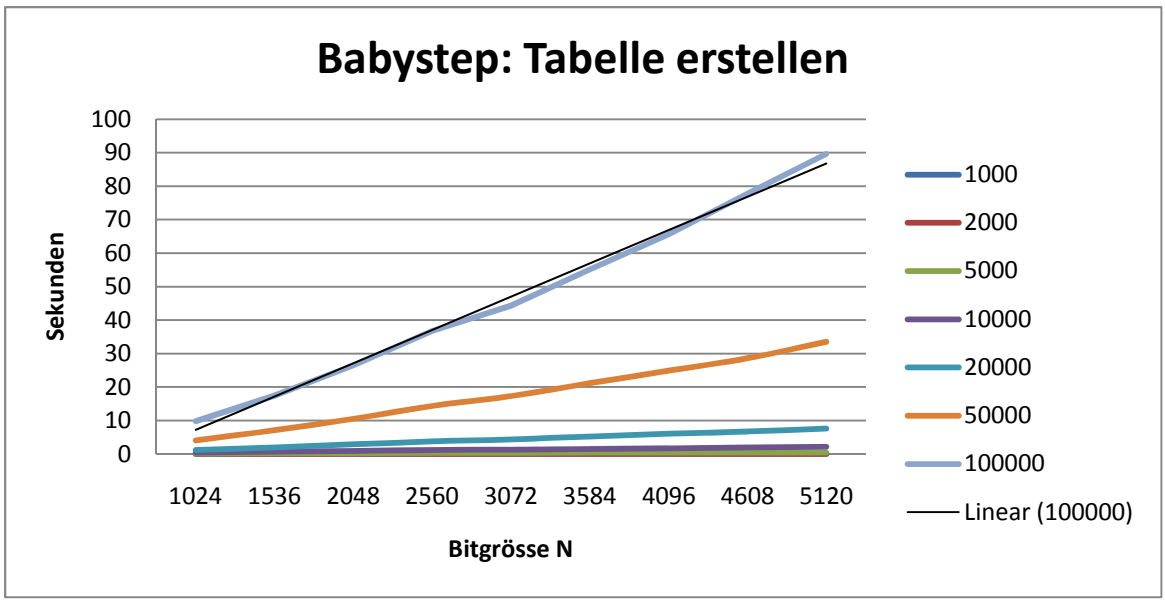
Die Erstellung der Tabelle kann vor der Wahl gemacht werden. Die Einträge werden in die oben beschriebene Hashtabelle eingefügt.



- Die Datenserien entsprechen verschiedenen Bitgrössen *N*.
- Die x-Achse zeigt die Tabellengrössen *m*.
- Die y-Achse zeigt die gemessenen Sekunden.
- Eine quadratische Trendlinie wurde eingefügt.

Mit zunehmender Tabellengrösse steigt die Laufzeit zur Erstellung der Tabelle quadratisch an. Obwohl für das Einfügen in die Tabelle eine Laufzeit von O(log(m)) angegeben wurde, scheint sich das Erstellen der Tabelle quadratisch zu verhalten.

Als Faustregel gilt: Eine Vergrösserung der Tabellengrösse wirkt sich quadratisch auf die Laufzeit zur Erstellung der Babystep Tabelle aus.



- Die Datenserien entsprechen verschiedenen Tabellengrössen *m*.
- Die x-Achse zeigt die Bitgrössen *N*.
- Die y-Achse zeigt die gemessenen Sekunden.
- Eine lineare Trendlinie wurde eingefügt.

Bei zunehmenden Bitgrössen steigt jeweils die Anzahl der Zeichen welche pro Tabelleneintrag gespeichert werden müssen. Dies kann in einer linearen Zunahme der Datenserien beobachtet werden.

Modell für die Erstellung der Babystep Tabelle

Das Babystep Modell wurde anhand des Quellcodes aus Listing 1 entworfen. Zur Erstellung der Babystep Tabelle werden *m* Multiplikationen und Modulo Operationen und *m* Tabellen Einfüge-Operationen verwendet. Für die mathematischen Operationen wurde die Laufzeit im Zusammenhang mit der Bitgrösse der Zahlen auf $O(N^2)$ abgeschätzt. Für das Einfügen in die Tabelle wurde eine Laufzeit von O(log(m)) abgeschätzt. Daraus wurde das folgende Modell geschrieben:

$$f(m,N) = k_0 * mN^2 + k_1 * m\log_2(m) \tag{10}$$

Anhand der gemessenen Resultate wurde das Modell verändert um möglichst gute Vorhersagen für unbekannte Tabellengrössen zu erzielen. Die Resultate wurden aus den Tabellengrössen von 10‘000 bis 100‘000 und den Bitgrössen von 1024 bis 5120 erstellt. Zur Überprüfung des Modells wurden Testresultate mit Tabellengrössen von 1‘000‘000 und 2‘000‘000 mit einer Bitgrösse von 1024 erstellt. Das Modell wurde dann angepasst, damit die Testergebnisse möglichst genau durch das Modell hervor gesagt wurden. Daraus ist aus (10) folgendes Modell entstanden:

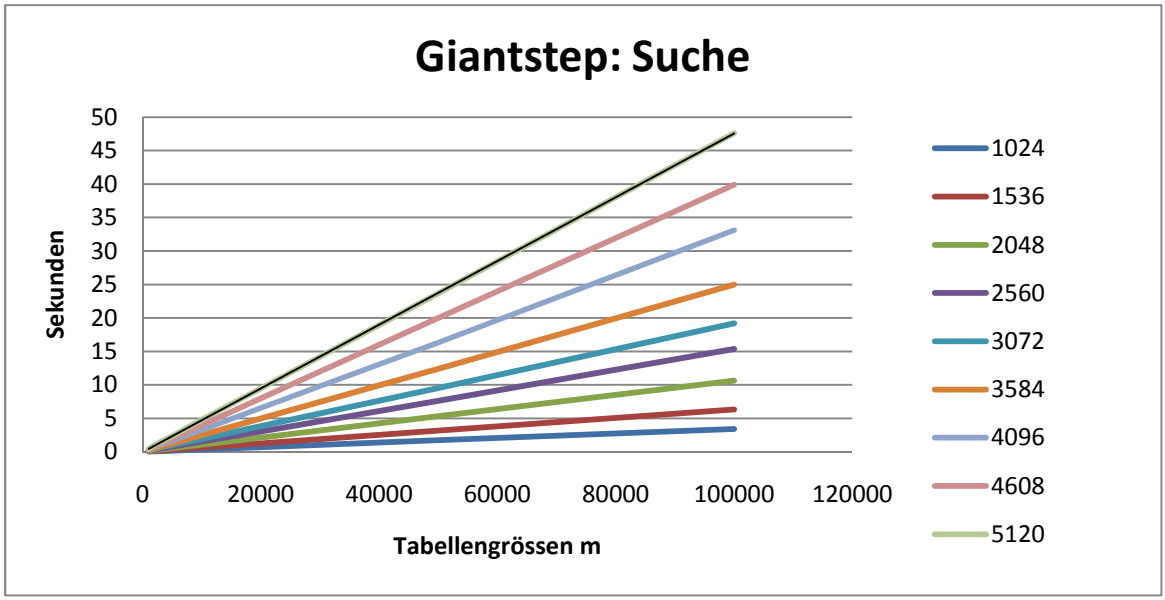
$$f(m,N) = k_0 * mN^2 + k_1 * m\log_2(mN) + k_2 * N^2 + k_3 * N \tag{11}$$

Für die Koeffizienten wurden folgende Werte ermittelt:
k_0: 3.4761E-11, k_1: 4.18802E-06, k_2: -5.57641E-07, k_3: 0.000474924

Für eine Tabellengrösse von 2‘000‘000 und einer Bitgrösse von 1024 wird nach dem Babystep Modell (11) eine Laufzeit von 5.5 Minuten vorhergesagt.
Für eine Tabellengrössen von 100‘000‘000 und einer Bitgrösse von 1024 wird eine Laufzeit von 5.2 Stunden vorhergesagt.

Giantstep Suche

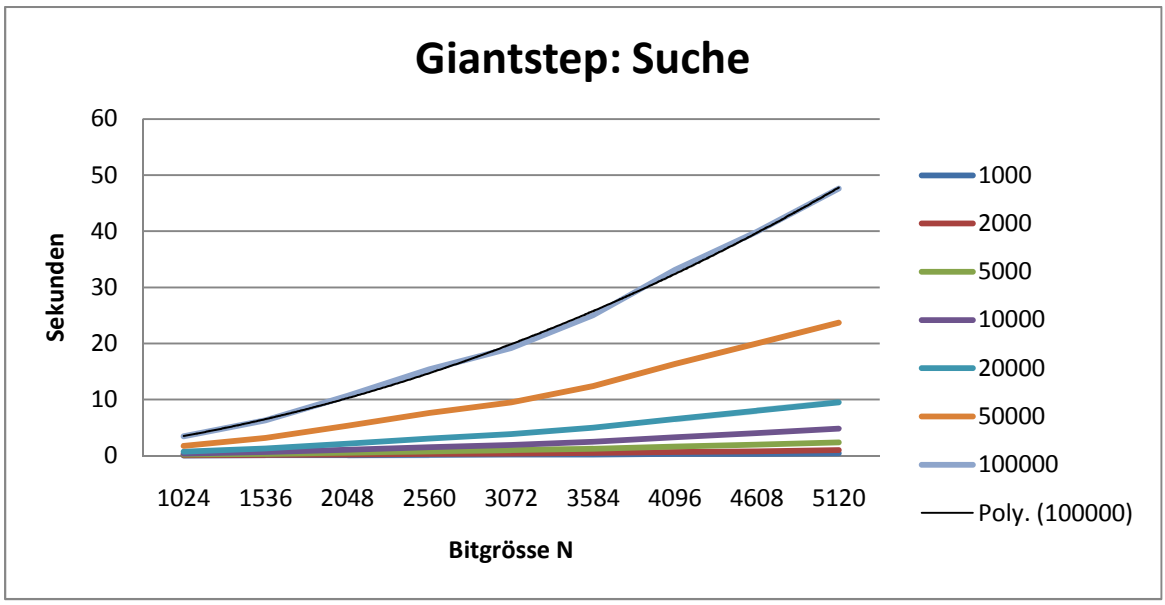
Diese Messung entspricht dem Aufwand, welcher nach der Wahlabgabe für die Berechnung des Wahlergebnisses benötigt wird. Für die Messung der Suche wurden immer alle Giantsteps berechnet. Somit zeigen diese Messungen wie lange man im schlechtesten Fall auf das Resultat der Wahl warten müsste (bezieht sich nur auf die Bestimmung des diskreten Logarithmus).



- Die Datenserien entsprechen verschiedenen Bitgrössen *N*.
- Die x-Achse zeigt die Tabellengrössen *m*.
- Die y-Achse zeigt die gemessenen Sekunden.
- Eine lineare Trendlinie wurde eingefügt.

Mit zunehmender Tabellengrösse steigt die Laufzeit beim Suchen in der Tabelle linear an. Obwohl für das Suchen in die Tabelle eine Laufzeit von O(log(m)) angegeben wurde, zeigt die Auswertung der Resultate ein lineares verhalten.

Als Faustregel gilt: Eine Vergrösserung der Tabellengrösse wirkt sich linear auf die Laufzeit der Giantstep Suche aus.



- Die Datenserien entsprechen verschiedenen Tabellengrössen *m*.
- Die x-Achse zeigt die Bitgrössen *N*.
- Die y-Achse zeigt die gemessenen Sekunden.

- Eine quadratische Trendlinie wurde eingefügt.

Mit zunehmender Bitgrösse wächst die Grösse der Zahlen. Dies wirkt sich auf die Berechnung des Giantstep Kandidaten und auf den Lockup in der Hashtabelle aus. Dabei wächst mit zunehmender Bitgrösse der Aufwand quadratisch. Zum Vergleich wurden entsprechende Trendlinien eingefügt.

Modell für die Giantstep Suche

Ein Giantstep Modell wurde anhand des Quellcodes aus Listing 1 entworfen. Zur Berechnung der Such-Schritte werden höchstens *m* Multiplikationen und Modulo Operationen benötigt. Die Laufzeit einer Operation hängt von der Bitgrösse der Zahlen ab. Eine Operation wird mit $O(N^2)$ abgeschätzt. Für das Suchen in der Tabelle wird eine Laufzeit von O(log(m)) abgeschätzt. Das daraus entstandene Modell entspricht Babystep Modell (10).

Auch hier wurde das Modell angepasst um für unbekannte Tabellengrössen möglichst gute Vorhersagen zu treffen. Es wurden die Resultate, die bei der Suche in der Tabelle gemessen wurden verwendet. Aus (10) ist somit folgendes Modell entstanden:

$$f(m,N) = k_0 * mN^2 + k_1 * m\log_2(mN) \qquad (12)$$

Für die Koeffizienten wurden folgende Werte ermittelt:
k_0: 1.73E-11, k_1: 1.03226E-06

Für eine Tabellengrösse von 2‘000‘000 und einer Bitgrösse von 1024 wird nach dem Giantstep Modell (12) eine Laufzeit von 1.6 Minuten vorhergesagt.
Für eine Teilmengengrösse von 100‘000‘000 und einer Bitgrösse von 1024 wird eine Laufzeit von 1.5 Stunden vorhergesagt.

Speicherbedarf für die Hashtabelle

Für die Abschätzung des Speicherbedarfs wird nur mit der benötigte Platz für die Key-Value-Paare gerechnet. Der Platzbedarf für Referenzen wird vernachlässigt.

$$Speicherplatzbedarf = 2 * AnzahlEinträge * (KeyGrösse + Valuegrösse)$$

Da die verwendete Hashtabelle auf einem binären Baum basiert, wird die Anzahl der Einträge verdoppelt. Im Binärbaum werden die Einträge in externen Knoten gespeichert. Um die Einträge schnell auffinden zu können werden nochmals etwa gleichviele interne Knoten benötigt.

Bei einer Tabellengrösse von *m*=100‘000 werden 100‘000 Einträge in der Hashtabelle gespeichert. Wenn bei einer Gruppengrösse von 2^{5120}, ein Key-Value-Paar 5120Bit + 32 Bit Speicherplatz benötigt, brauchen alle Einträge ungefähr 124 MB.

Parallele Programmierung

Durch Parallelisierung könnte die Berechnung des Babystep-Giantstep-Algorithmus weiter beschleunigt werden.

Die Hashtabelle könnte durch mehrere parallele Entitäten wie Threads oder Prozesse erstellt werden. Dabei wird die Teilmenge für welche die Einträge erstellt werden weiter unterteilt. Jede Entität erstellt somit einen Teil der Einträge. Es muss allerdings darauf geachtet werden, dass die Hashtabelle paralleles Einfügen zulässt ohne zu blockieren.

Auch die Giantstep Suche kann parallelisiert werden. Die zu durchsuchende Teilmenge wird wieder auf die parallelen Entitäten aufgeteilt. Jede Entität sucht einen Teil ab. Einer dieser Entitäten liefert dann das Ergebnis. Je mehr Entitäten parallel suchen können desto mehr lässt sich die Laufzeit reduzieren. Die Laufzeit für die parallele Suche beträgt dann $O\left(\frac{\sqrt{n}}{t}\right)$, wobei t die Anzahl paralleler Entitäten darstellt.

Weitere Algorithmen zum diskreten Logarithmus

Neben dem Babystep-Giantstep-Algorithmus gibt es noch weitere Algorithmen zur Berechnung des diskreten Logarithmus. Informationen dazu können z.B. aus [Buc03] entnommen werden.

Der Pollard-Rho-Algorithmus

Dieser Algorithmus benötigt ebenfalls $O(\sqrt{n})$ Gruppenoperationen, der Speicherplatzbedarf ist jedoch konstant. Der Pollard-Rho-Algorithmus ist also vom benötigten Speicherplatz her effizienter.

Der Pohlig-Hellman-Algorithmus

Bei diesem Algorithmus handelt es sich um eine Variante des Babystep-Giantstep-Algorithmus. Dabei wird versucht die Gruppengrösse zu reduzieren. Die Laufzeit und der Speicherplatzbedarf reduziert sich dann auf $O(\sqrt{n_p})$, wobei n_p der grösste Primfaktor von n ist.

Index-Calculus-Algorithmus

Dabei handelt es sich um die effizienteste Variante zur Berechnung des diskreten Logarithmus. Er lässt sich jedoch nicht auf beliebige Gruppen anwenden.

Schlussfolgerungen

Der Babystep-Giantstep-Algorithmus bietet eine einfache, aber effiziente Implementation des diskreten Logarithmus. Wenn er beim E-Voting zum Einsatz kommt, wo nur eine Teilmenge der Gruppe durchsucht werden muss, kann er für Wahlen mit einer Wahlgrösse von bis zu $4*10^{12}$ eingesetzt werden.

Der Babystep-Giantstep-Algorithmus weist zwei laufzeitintensive Stellen auf.

1. Das Erstellen der Babystep-Tabelle.
2. Die Giantstep-Suche.

Dabei muss nur die Giantstep-Suche während der Wahl gemacht werden. Die Babystep-Tabelle kann vor der Wahl erstellt werden.

Die Resultate zeigen, dass zur Erstellung der Babystep Tabelle, bei einer Tabellengrössen von 100‘000 und einer Bitgrösse von 5120 Bit, etwa 80 Sekunden Rechenzeit benötigt wird. Um einen Eintrag bei der Giantstep-Suche in der Tabelle zu finden, wird im schlechtesten Fall etwa eine Minute Rechenzeit benötigt. Dies entspricht einer Wahlgrösse von 10^{10}. Eine aus den Resultaten abgeleitete Faustregel besagt, dass sich bei der Giantstep-Suche eine Vergrösserung der Tabellengrösse linear auf die Laufzeit auswirkt.

Ein Modell zur Berechnung der Laufzeit für beliebig grosse Tabellengrössen wurde aus den gemessenen Werten entwickelt. Das Modell sagt, dass für eine Tabellengrösse von 2‘000‘000 und einer Bitgrösse von 1024 Bit, 5.5 Minuten zur Erstellung der Tabelle und 1.6 Minuten zur Giantstep-Suche benötigt werden. Dies entspricht einer Wahlgrösse von $4*10^{12}$.

Somit kann der Babystep-Giantstep-Algorithmus für E-Voting Anwendungen zum Einsatz kommen, die $4*10^{12}$ Ja-Nein-Stimmen innerhalb von 2 Minuten zählen müssen.

Sowohl der Babystep, als auch der Giantstep können parallel implementiert werden. Dadurch kann die Laufzeit weiter reduziert werden.

LITERATUR

[Buc03] J. Buchmann, Einführung in die Kryptographie, ISBN: 3-540-40508-9

[GMP91] GNU Multiple Precision Arithmetic Library, http://gmplib.org
Timestamp: 12.06.2011

[WEl11] Wikipedia, 2011, http://de.wikipedia.org/wiki/Elgamal-Kryptosystem
Timestamp: 12.06.2011

[WPa11] Wikipedia, 2011, http://de.wikipedia.org/wiki/Paillier-Kryptosystem
Timestamp: 12.06.2011

[Sha71] D. Shanks, Class number, a theory of factorization and genera. In Proc. Symp. Pure Math. 20, 1969, pages 415–440. AMS, Providence, R.I., 1971.