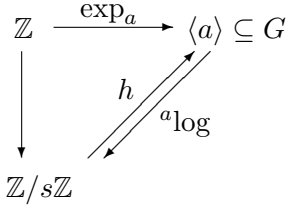
4 Der diskrete Logarithmus mit Anwendungen

4.1 Der diskrete Logarithmus

Sei G eine Gruppe (multiplikativ geschrieben) und $a \in G$ ein Element der Ordnung s (die auch ∞ sein kann). Dann ist die **Exponentialfunktion** in G zur Basis a

$$\exp_a \colon \mathbb{Z} \longrightarrow G, \quad x \mapsto a^x,$$

ein Gruppenhomomorphismus und hat die Periode s. Nach dem Homomorphiesatz ist der induzierte Homomorphismus h



ein Isomorphismus. Es gibt also eine Umkehrabbildung

$${}^a\log \colon \langle a \rangle \longrightarrow \mathbb{Z}/s\mathbb{Z}$$

auf der zyklischen Untergruppe $\langle a \rangle \subseteq G$, den **diskreten Logarithmus** in G zur Basis a, der ein Gruppenisomorphismus ist. [Der Fall $s = \infty$ passt, wenn man $\mathbb{Z}/s\mathbb{Z} = \mathbb{Z}$ setzt.]

Das wird auf die multiplikative Gruppe $\mathbb{M}_n$ angewendet: Für eine ganze Zahl $a \in \mathbb{Z}$ mit $\operatorname{ggT}(a, n) = 1$ hat die Exponentialfunktion $\bmod n$ zur Basis a,

$$\exp_a \colon \mathbb{Z} \longrightarrow \mathbb{M}_n, \quad x \mapsto a^x \bmod n,$$

die Periode $s = \operatorname{Ord} a | \lambda(n) | \varphi(n)$. Die Umkehrfunktion

$${}^a\log \colon \langle a \rangle \longrightarrow \mathbb{Z}/s\mathbb{Z}$$

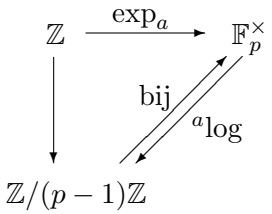
ist der diskrete Logarithmus $\bmod n$ zur Basis a.

Es ist kein effizienter Algorithmus bekannt, den diskreten Logarithmus ${}^a\log$ für große $s = \operatorname{Ord} a$ zu bestimmen, d. h., die Exponentialfunktion umzukehren – auch kein probabilistischer.

> **Informelle Definition:** Eine Funktion $f \colon M \longrightarrow N$ heißt **Einwegfunktion**, wenn für „fast alle“ Bilder $y \in N$ ein Urbild $x \in M$ mit $f(x) = y$ nicht effizient bestimmbar ist.
>
> Eine mathematisch präzise Formulierung dieser Definition lässt sich im Rahmen der Komplexitätstheorie geben, siehe später.

Diskreter-Logarithmus-Vermutung: Die Exponentialfunktion $\exp_a \bmod n$ ist für „fast alle“ Basen a eine Einwegfunktion.

Der wichtigste Spezialfall ist: Der Modul ist eine Primzahl $p \geq 3$ und $a \in [2, \ldots, p-2]$ ist ein primitives Element für p, d. h., $\operatorname{Ord} a = p - 1$.



Damit der diskrete Logarithmus praktisch nicht effizient zu berechnen ist, muss man den Primzahlmodul p etwa in der Größenordnung wie einen RSA-Modul wählen, d. h. nach dem heutigen Stand der Technik reichen 1024-Bit-Primzahlen als Modul nicht aus, 2048-Bit-Primzahlen gelten noch als für ein paar Jahre sicher.

Eine ganze Reihe unterer Schranken für die Komplexität der Berechnung des diskreten Logarithmus in verschiedenen Berechnungsmodellen enthält das – im Literaturverzeichnis der Vorlesung genannte – Buch von Shparlinski.

4.2 Diffie-Hellman-Schlüsselaustausch

Übertragen werden soll ein Schlüssel für eine symmetrische Chiffrierung. Dazu haben Diffie und Hellman 1976 das folgende Verfahren vorgeschlagen, das auf der Exponentialfunktion in endlichen Körpern, also einer (mutmaßlichen) Einweg-Funktion beruht:

1. A (Alice) und B (Bob) einigen sich (öffentlich) über eine Primzahl p und eine zugehörige Primitivwurzel a.
2. A erzeugt eine Zufallszahl x, bildet $u = a^x \bmod p$ und sendet u an B.
3. B erzeugt eine Zufallszahl y, bildet $v = a^y \bmod p$ und sendet v an A.
4. A berechnet $k = v^x \bmod p$, und B berechnet $k = u^y \bmod p$.

Die Zahl k ist der gemeinsame geheime Schlüssel (oder dient zu dessen Bestimmung nach einem öffentlich bekannten Verfahren). Dass sowohl A als auch B den gleichen Schlüssel haben, liegt an der Gleichung

$$v^x \equiv a^{xy} \equiv u^y \pmod{p}.$$

Ein Lauscher kann nur die Zahlen p, a, u und v abfangen, die ihm nicht gestatten, k oder x oder y effizient zu berechnen.

Damit hat man also auch so etwas wie ein hybrides Verschlüsselungsverfahren; es unterscheidet sich von einem „echten“ asymmetrischen Verfahren aber insofern, als A nicht an B spontan eine Nachricht senden kann, sondern eine Synchronisation herstellen muss.

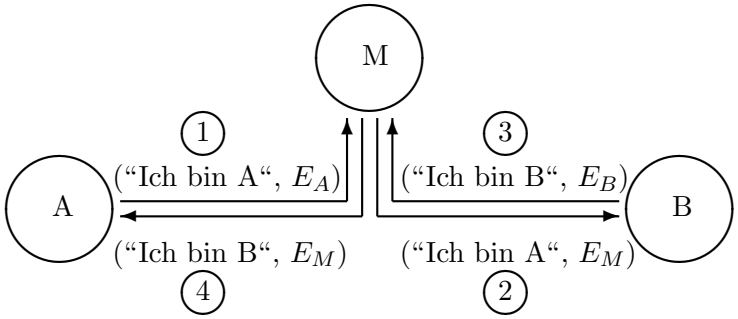
Wenn ein Angreifer effizient diskrete Logarithmen berechnen kann, kann er auch das Diffie-Hellman-Verfahren effizient brechen. Ob auch die umgekehrte Implikation gilt, ist unbekannt.

Beim britischen Geheimdienst CESC war das Verfahren schon 1974 entdeckt, aber natürlich geheim gehalten worden.

4.3 Der Mann in der Mitte

In diesem Abschnitt wird ein Kommunikationssystem mit asymmetrischer Chiffrierung betrachtet; für den DIFFIE-HELLMAN-Schlüsselaustausch funktioniert der Angriff genauso. Das Problem ist, dass ein Angreifer seinen Schlüssel in die Kommunikation einschleusen kann. Genauer:

A = Alice und B = Bob wollen miteinander kommunizieren. Dazu sendet A an B ihren öffentlichen Schlüssel E_A und B an A seinen öffentlichen Schlüssel E_B. Der Angreifer M = Mallory, der „Mann in der Mitte“, fängt diese Sendungen ab und ersetzt beide Male den abgefangenen öffentlichen Schlüssel durch seinen eigenen E_M. Dann kann M unbemerkt den ganzen Nachrichtenverkehr zwischen A und B abhören und sogar verfälschen; diese Situation ist in der folgenden Abbildung dargestellt.



Es gibt verschiedene Auswege aus dieser Bedrohung, die aber alle die asymmetrische Kryptographie verkomplizieren; der übliche Ausweg ist die Verwendung von Zertifikaten: Die öffentlichen Schlüssel aller Teilnehmer am Kommunikationsverfahren werden von einer bekannten und „vertrauenswürdigen“ Stelle – einem sogenannten Trustcenter – digital signiert. Zertifikat = öffentlicher Schlüssel mit digitaler Signatur des Trustcenters.

Merkregel. *Ein Schlüsselaustausch kann vor dem Mann in der Mitte nur sicher sein, wenn die Partner sich gegenseitig authentisieren können.*

Übungsaufgabe. Welche Information könnte man beim DIFFIE-HELLMAN-Verfahren als Zertifikat verwenden?

4.4 Geheime Kommunikation ohne Schlüsselaustausch

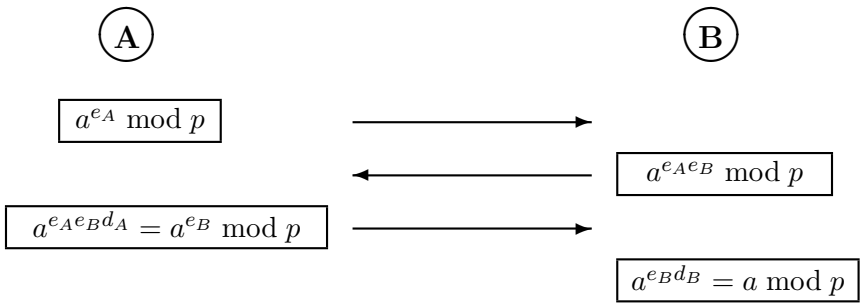
Es ist auch möglich, vertraulich zu kommunizieren, ohne vorher Schlüssel vereinbart zu haben – der Angriff durch den Mann in der Mitte ist hier aber ebenfalls möglich.

Die Idee lässt sich an einem Bild aus dem Alltagsleben verdeutlichen:

- Alice legt die Nachricht in eine Kiste und verschließt sie mit einem Vorhängeschloss, zu dem nur sie den Schlüssel hat. Sie schickt die Kiste an Bob.
- Bob kann die Kiste natürlich nicht öffnen. Statt dessen verschließt er sie mit einem weiteren Vorhängeschloss, zu dem nur er den Schlüssel hat. Er schickt die doppelt verschlossene Kiste an Alice zurück.
- Alice entfernt ihr Vorhängeschloss und schickt die nunmehr nur noch mit Bobs Schloss verschlossene Kiste wieder an Bob.
- Bob entfernt sein Schloss, öffnet die Kiste und liest die Nachricht.

Dieses kryptographische Protokoll heißt Massey-Omura-Schema oder Shamirs No-Key-Algorithmus. Es lässt sich mit Hilfe der diskreten Exponentialfunktion umsetzen; seine Sicherheit beruht auf der Komplexität des diskreten Logarithmus:

Vorgegeben ist eine öffentlich bekannte, allgemein gültige (große) Primzahl p. Alice und Bob wählen je ein Paar von Exponenten d und e mit $ed \equiv 1 \pmod{p-1}$, also $a^{de} \equiv a \pmod{p}$ für alle ganzen Zahlen $a \in \mathbb{Z}$. Jeder hält seine *beiden* Exponenten geheim. Das Protokoll, mit dem Alice nun eine Nachricht a an Bob schickt, sieht so aus:



Ein Angreifer, der diskrete Logarithmen berechnen könnte, könnte aus den erlauschten Geheimtexten $a^{e_A} \bmod p$ und $a^{e_A e_B} \bmod p$ den Exponenten e_B und daraus durch Kongruenzdivision auch d_B berechnen. Ebenso könnte er aus $a^{e_A e_B} \bmod p$ und $a^{e_A e_B d_A} = a^{e_B} \bmod p$ den Exponenten d_A und dann auch e_A berechnen.

Ein anderer Angriff ist nicht bekannt.

4.5 ElGamal-Chiffrierung – Idee

Die ElGamal-Chiffre ist ein asymmetrisches Verfahren – genauer gesagt ein hybrides –, das ebenfalls auf der Komplexität des diskreten Logarithmus beruht.

Eine Primzahl p und ein $g \in [2 \dots p-2]$ sind öffentlich festgelegt; g sollte von hoher Ordnung, am besten primitives Element sein.

> p und g können für alle Teilnehmer gemeinsam gelten, können aber auch individuell verschieden sein.

Jeder Teilnehmer wählt sich eine zufällige Zahl

$$d \in [2 \dots p-2]$$

als privaten Schlüssel und bildet daraus

$$e = g^d \bmod p$$

als zugehörigen öffentlichen Schlüssel. Die Bestimmung von d aus e bedeutet gerade die Berechnung eines diskreten Logarithmus.

Wie soll man nun eine Nachricht a so transformieren, dass sie nur mit Kenntnis von d wiederzugewinnen ist? Die naive Idee, $e^a = g^{da} \bmod p$ zu schicken, nützt nichts – auch der Empfänger kann trotz Kenntnis von d die Nachricht a nicht entschlüsseln. Auch $r = g^a \bmod p$ zu schicken, nützt nichts – der Empfänger kann nur $r^d = e^a \bmod p$ bestimmen, aber nicht a.

Eine bessere Idee ist, erst einen Schlüssel zu erzeugen, mit dem dann ein hybrides Verfahren durchgeführt wird:

- Alice wählt ein zufälliges $k \in [2 \dots p-2]$. Als Schlüssel soll $K = e^k \bmod p$ mit dem öffentlichen Schlüssel von Bob verwendet werden; dieses kann Alice berechnen.
- Damit auch Bob den Schlüssel K bestimmen kann, schickt Alice mit ihrer Nachricht die *Schlüssel-Information* $r = g^k \bmod p$ mit.
- Bob kann daraus $r^d = g^{kd} = e^k = K \bmod p$ mit Hilfe seines privaten Schlüssels d berechnen.

Als symmetrische Chiffe beim Hybridverfahren wird die Verschiebechiffre auf $\mathbb{F}_p^\times$ genommen, wobei K als Einmalschlüssel dient – d. h., für jeden Klartextblock ist ein neuer Schlüssel K zu erzeugen und die entsprechende Schlüssel-Information mitzuschicken. Dadurch ist die Länge des Geheimtexts gerade das Doppelte der Länge des Klartexts.

Die Verschlüsselung läuft dann nach Bildung des Schlüssels und der Schlüssel-Information so weiter ab:

- Alice berechnet den Geheimtext $c = Ka \bmod p$ und schickt diesen zusammen mit r an Bob.

Die Entschlüsselung ist dann, nachdem Bob den Schlüssel K wie oben beschrieben berechnet hat:

- $a = K^{-1}c \bmod p$ durch Kongruenzdivision.

4.6 Berechnung des diskreten Logarithmus

Der klassische Algorithmus zur Berechnung des diskreten Logarithmus ist der **Index-Calculus** von ADLEMAN – „Index“ war übrigens die Bezeichnung, die GAUSS für den diskreten Logarithmus verwendet hatte.

Sei $p \geq 3$ eine Primzahl und a eine Primitivwurzel für p.

Der naive, wie üblich durch exponentiell wachsenden Aufwand bestrafte, Algorithmus zur Bestimmung von $\log_a y$ für $y \in \mathbb{F}_p^\times$ besteht darin, der Reihe nach $a, a^2, a^3, \ldots$ zu berechnen, bis x mit $a^x = y$ gefunden ist. Im Mittel sind $\frac{p}{2} - 1$ Versuche nötig, im schlechtesten Fall $p - 2$, wenn man den trivialen Fall $y = 1$ außen vor lässt.

Vorbereitung

Dieser Schritt muss zu gegebenen p und a nur einmal ausgeführt werden.

Seien $p_1 = 2$, $p_2 = 3$, …, p_k die ersten k Primzahlen – wie k gewählt wird, wird noch festgelegt.

Für einen zufällig gewählten Exponenten r könnte es sein, dass $a^r \bmod p$ – als ganze Zahl $\in \mathbb{Z}$ betrachtet – nur Primfaktoren in $\{p_1, \ldots, p_k\}$ hat. Nach h solchen Glücksfällen hat man in $\mathbb{Z}$, also erst recht in $\mathbb{F}_p$, ein System von h Gleichungen

$$\begin{aligned} a^{r_1} \bmod p &= p_1^{\alpha_{11}} \cdots p_k^{\alpha_{1k}}, \\ &\vdots \\ a^{r_h} \bmod p &= p_1^{\alpha_{h1}} \cdots p_k^{\alpha_{hk}}. \end{aligned}$$

Daraus entsteht ein lineares Gleichungssystem für die k unbekannten Größen $\log_a p_i$ über dem Ring $\mathbb{Z}/(p-1)\mathbb{Z}$:

$$\begin{aligned} r_1 &= \alpha_{11} \cdot \log_a p_1 + \cdots + \alpha_{1k} \cdot \log_a p_k, \\ &\vdots \\ r_h &= \alpha_{h1} \cdot \log_a p_1 + \cdots + \alpha_{hk} \cdot \log_a p_k. \end{aligned}$$

Zu dessen Auflösung gibt es, wie wir aus Kapitel I wissen, effiziente Algorithmen; ist h groß genug – mindestens $h \geq k$ –, lassen sich $\log_a p_1$, …, $\log_a p_k$ bestimmen.

Die Suche nach den „Glücksfällen“ ergibt also einen probabilistischen Algorithmus.

Berechnung

Sei $y \in \mathbb{F}_p^\times$ gegeben; gesucht ist $\log_a y$.

Für einen zufällig gewählten Exponenten s könnte in $\mathbb{Z}$

$$y \cdot a^s \bmod p = p_1^{\beta_1} \cdots p_k^{\beta_k}$$

sein. Dann ist

$$\log_a y = \beta_1 \cdot \log_a p_1 + \cdots + \beta_k \cdot \log_a p_k - s$$

ganz leicht berechnet.

Die Berechnung des diskreten Logarithmus für beliebige Elemente ist daher auf die Berechnung für die Elemente der **Faktorbasis** $(p_1, \ldots, p_k)$ *reduziert.* Auch diese Reduktion ist probabilistisch.

Varianten

Aus dem vorgestellten Ansatz werden verschiedene Algorithmen konstruiert, die unterschiedlich schnell sind. Sie unterscheiden sich in der Wahl der Faktorbasis – diese kann an y adaptiert sein und muss nicht aus der lückenlosen Folge der ersten Primzahlen bestehen – sowie in der Strategie für die Wahl der Exponenten r und s.

Die schnellste Variante verwendet ein Zahlkörpersieb, wie es auch bei der Faktorisierung großer Zahlen eingesetzt wird, und kommt auf einen Aufwand

$$\approx e^{c \cdot \sqrt[3]{\log p \cdot (\log \log p)^2}},$$

wie er auch bei der Faktorisierung einer gleichlangen zusammengesetzten Zahl benötigt wird. Nach dem aktuellen Stand sind also 1024-Bit-Primzahlmoduln nur ganz kurzfristig als sicher für kryptographische Anwendungen anzusehen, mittelfristig braucht man mindestens 2048 Bit.

Als Kuriosum sei angemerkt, dass das „Secure NFS“ von SUN noch in den 90er Jahren eine Primzahl der Länge 192 Bits (58 Dezimalstellen) verwendete.

Spezielle Primzahlen

Es gibt Argumente, p speziell, also als die größere Zahl eines Germain-Paares zu wählen – also $p = 2q + 1$ mit q prim:

1. Es gibt Algorithmen, die besonders schnell sind, wenn $p - 1$ nur kleine Primfaktoren hat. Dieses Argument wird heute nicht mehr für stichhaltig angesehen, da die Chance, aus Versehen eine soche „schlechte“ Primzahl zu erwischen, extrem gering ist; außerdem ist das Zahlkörpersieb so schnell, dass die speziellen Algorithmen kaum noch einen Vorteil bringen.

2. Die Bestimmung eines primitiven Elements ist sehr viel leichter, siehe Abschnitt A.10 im Anhang.