Top Secret · Geheimnisvolle Faktoren

Jens Zumbrägel

Professur für Kryptographie
Fakultät für Informatik und Mathematik
Universität Passau

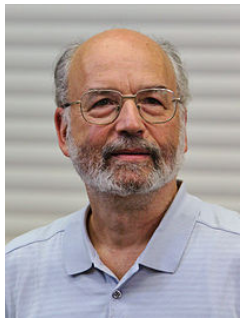
Lehrerfortbildung 2023
Universität Passau

Übersicht

1. Das RSA-Kryptosystem
   - Sicherheitsdiskussion
2. Primalitätstests
3. Das Faktorisierungsproblem
   - Überblick von Faktorisierungsalgorithmen
4. Indexkalkül
   - Quadratisches Sieb
   - Fortgeschrittenes Indexkalkül
   - DLP und Rekordberechnungen

Das RSA-Kryptosystem

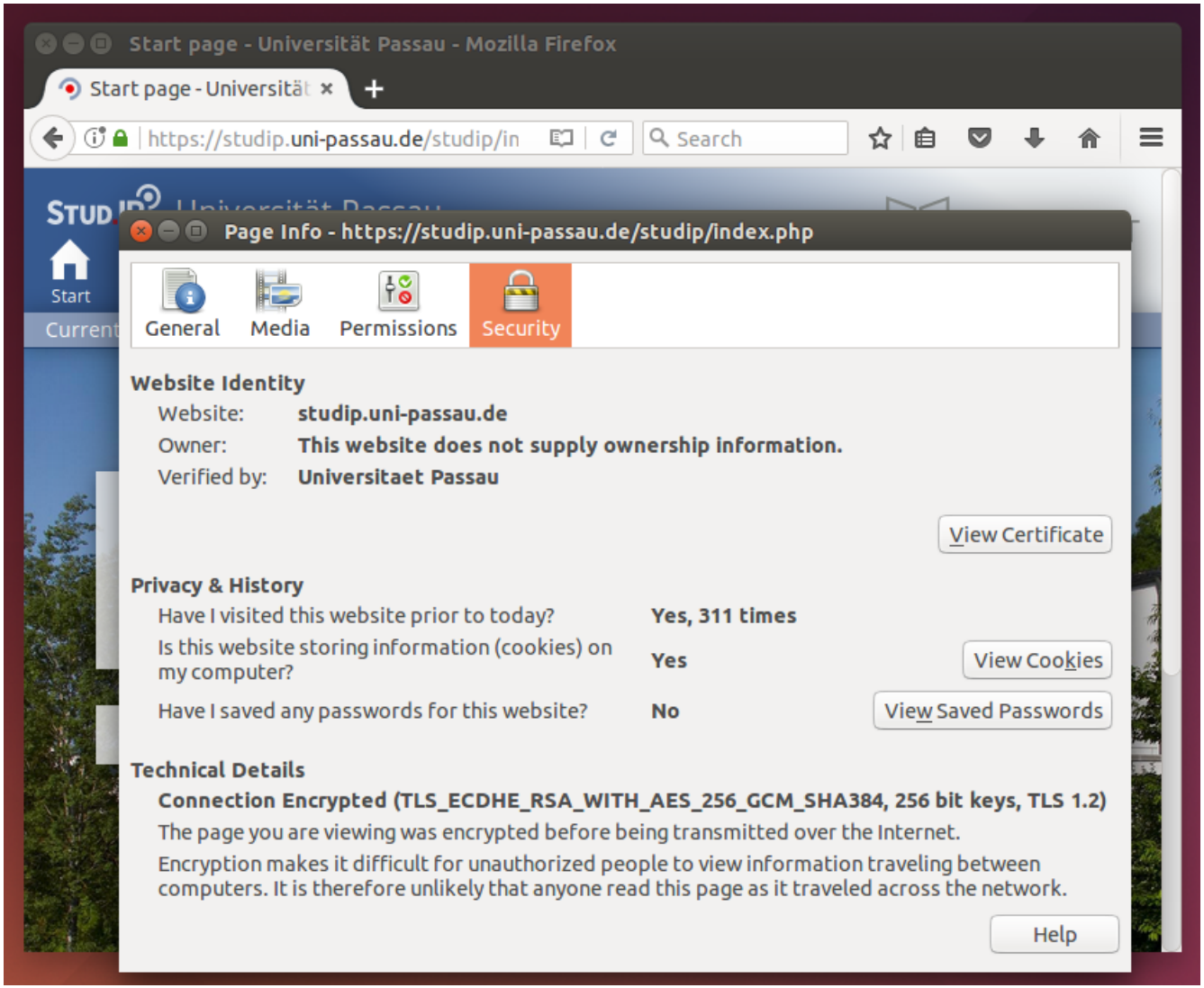
- benannt nach den Autoren Rivest, Shamir und Adleman (1978)


- das erste Public-Key Verschlüsselungssystem
- beruht eng auf dem *Zahlfaktorisierungsproblem*

Für diesen Vortrag $\mathbb{Z}_n := \mathbb{Z}/n\mathbb{Z}$ und $\mathbb{Z}_n^* := \{x \in \mathbb{Z}_n \mid \mathrm{ggT}(x, n) = 1\}$.

Beispiel Webbrowser



Das RSA-Kryptosystem

Wähle eine Zahl $n = p \cdot q$ mit verschiedenen Primzahlen p, q etwa der gleichen Größe $\log p \approx \log q$. Solch eine Zahl heißt auch RSA-Zahl.

Für die Eulersche Phi-Funktion erhalten wir

$$r := \varphi(n) = |\mathbb{Z}_n^*| = (p-1)\cdot(q-1)\,.$$

Sei die Zahl n öffentlich. Wähle $e \in \mathbb{Z}_r^*$ und sei $d := e^{-1} \in \mathbb{Z}_r^*$.
Wir nehmen e als *öffentlichen* Schlüssel und d als *private* Schlüssel.

Die Verschlüsselungs- und Entschlüsselungsfunktionen sind wie folgt:

$$f : \mathbb{Z}_n \times \mathbb{Z}_r^* \to \mathbb{Z}_n\,, \quad (m, e) \mapsto m^e$$
$$g : \mathbb{Z}_n \times \mathbb{Z}_r^* \to \mathbb{Z}_n\,, \quad (c, d) \mapsto c^d$$

Zwei wichtige Beobachtungen (1/2)

Es gilt stets $g(f(m,e),d) = m$.

Lemma

Für jede "Nachricht" $m \in \mathbb{Z}_n$ gilt $(m^e)^d = m$.

Beweis. Sei $m \in \mathbb{Z}_n$.

- Angenommen $m \in \mathbb{Z}_n^*$. Dann gilt $m^{\varphi(n)} \equiv 1$ mit Eulers Satz. Weil $ed = k\varphi(n) + 1$ erhalten wir $m^{ed} = m^{k\varphi(n)+1} \equiv m$.
- Auch für allgemeine $m \in \mathbb{Z}_n$ gilt $m^{k\varphi(n)+1} = m$, mit dem Chinesischen Restsatz (Übung).

Zwei wichtige Beobachtungen (2/2)

Lemma

Sei eine RSA-Zahl n gegeben. Dann ist es gleich schwer, den Wert $\varphi(n)$ zu erhalten, wie die Faktoren p und q zu finden.

Beweis. Sind p und q bekannt, so ist klar $\varphi(n) = (p-1)(q-1)$ gefunden.

Ist umgekehrt $\varphi(n)$ bekannt, so erhalten wir

$$n - \varphi(n) + 1 = p + q\,.$$

Schreiben wir $p = a - b$ und $q = a + b$, wobei $a = \frac{1}{2}(p+q)$ bekannt ist sowie $n = pq$, so finden wir $a^2 - n = b^2$ und somit b. □

Sicherheit des privaten Schlüssels

Sei n eine RSA-Zahl mit öffentlichem Schlüssel e und geheimem d.

Satz

Die Schwierigkeit, den privaten Schlüssel d aus dem öffentlichen e zu berechnen, ist praktisch äquivalent zum Faktorisierungsproblem für n.

Offenes Problem: Benötigt das Berechnen von m aus $c = m^e \bmod n$ notwendig die Kenntnis von d (bzw. die Faktorisierung von n)?

Weitere Bemerkungen

- Parameterwahl: $\log p \approx \log q$ mit ≥ 512 Bits, also n hat ≥ 1024 Bits.
- "Beliebte" Wahl $e = 3$ angreifbar durch eine *low encryption exponent* Attacke (Chinesischer Restsatz mit vielen Moduli).
- Weiterhin gibt es eine *low decryption exponent* Attacke.
- Verschlüsselung und Entschlüsselung können beschleunigt werden mit Chin. Restsatz durch Berechnen der Potenzen mod p, mod q separat.
- Chosen Ciphertext Attack-Sicherheit erreicht mit *Optimal Asymmetric Encryption Padding* (OAEP) Randomisierung.

Primalitätstests

Wie kann man effizient testen, ob eine Zahl

- prim oder
- zusammengesetzt ist?

Der Fermattest

Ist p prim, dann gilt

$$a^p \equiv a \pmod{p}$$

für alle $a \in \mathbb{Z}$ nach Fermats Satz.

Wenn also $a \in \mathbb{Z}$ existiert mit $a^p \not\equiv a \pmod{p}$ (oder $a^{p-1} \not\equiv 1$ wenn $p \nmid a$), dann ist p nicht prim.

Dieser einfache Test kann für einen indirekten Nachweis verwendet werden, dass eine Zahl zusammengesetzt ist, ohne Faktoren explizit zu bestimmen.

Komplexität. Mittels "square-and-multiply" benötigt ein Fermattest $O(\log p)$ modulare Multiplikationen und somit (höchstens) $O((\log p)^3)$ Bit-Operationen.

Ausflug in die Fermatzahlen

Fermat vermutete 1640 dass die Zahl

$$F_n := 2^{2^n} + 1$$

für $n \in \mathbb{N}$ stets prim sei. In jedem Fall gilt, dass $2^m + 1$ nur dann prim sein kann wenn $m = 2^n$ für ein $n \in \mathbb{N}$ ist (Übung).

Example

Die ersten Fermatzahlen lauten

n	0	1	2	3	4
F_n	3	5	17	257	65 537

und bis hierhin ist die Vermutung korrekt.

Die Fermatzahl F_5

Behauptung

Die Fermatzahl $F_5 = 2^{32} + 1 = 4\,294\,967\,297$ ist zusammengesetzt.

Tatsächlich können wir für $n = F_5 - 1 = 2^{32}$ prüfen ob $a^n = a^{2^{32}} \equiv 1$ gilt, mit 32 mal Quadrieren modulo F_5.
(Zum Vergleich würde man 6542 *trial divisions* durch alle Primzahlen $\leq \sqrt{F_5}$ benötigen.)

Zum Beispiel, mit $a = 3$ erhalten wir $a^{2^{32}} \equiv 3\,029\,026\,160 \not\equiv 1$, also F_5 ist nicht prim.

Eulers Trick

Euler hat bereits 1732 gezeigt, dass F_5 zusammengesetzt ist, also Fermats Vermutung widerlegt. Er hat sogar einen Faktor gefunden, nämlich 641, durch folgende Überlegung.

Lemma

Ist p ein Primfaktor von F_n, dann gilt $p \equiv 1 \pmod{2^{n+1}}$.

Beweis. Angenommen $p \mid 2^{2^n} + 1$, also $2^{2^n} \equiv -1 \pmod{p}$. In $\mathbb{Z}_p^*$ haben wir dann $\operatorname{ord}(2) = 2^{n+1}$, und daraus folgt nach Lagrange $2^{n+1} \mid p-1$. □

Für $n = 5$ müssen also nur die Primzahlen $p \equiv 1 \pmod{64}$ geprüft werden. Tatsächlich gilt $641 \mid 2^{32} + 1$, da $2^{32} = (256^2)^2 \equiv 154^2 \equiv -1 \pmod{641}$.

Weitere Bemerkungen

- Es gibt zusammengesetzte Zahlen p für welche $a^p \equiv a \pmod{p}$ für alle $a \in \mathbb{Z}$ gilt. Diese "pseudo-Primzahlen" werden Carmichael-Zahlen genannt; die kleinste lautet $p = 561 = 3 \cdot 11 \cdot 17$.

- Für große Zahlen ist Faktorisierung ein schwierigeres Problem als ein Primalitätstest.

  So ist zurzeit bekannt, dass alle Fermatzahlen $F_5, F_6, \ldots, F_{32}$ nicht prim sind, wobei nur die Fermatzahlen bis F_{11} vollständig faktorisiert wurden (und für F_{20} und F_{24} wurde bisher kein Faktor gefunden).

Moderne Primalitätstests

Zur Zahl n bezeichne $\mathbb{Z}_n^* := \{x \in \mathbb{Z}_n \mid \mathrm{ggT}(x, n) = 1\}$. Betrachte Mengen:

$$U_n := \{x \in \mathbb{Z}_n^* \mid x^{n-1} \equiv x \bmod n\}$$

$$V_n := \{x \in \mathbb{Z}_n^* \mid x^{(n-1)/2} \equiv \left(\tfrac{x}{n}\right) \bmod n\}$$

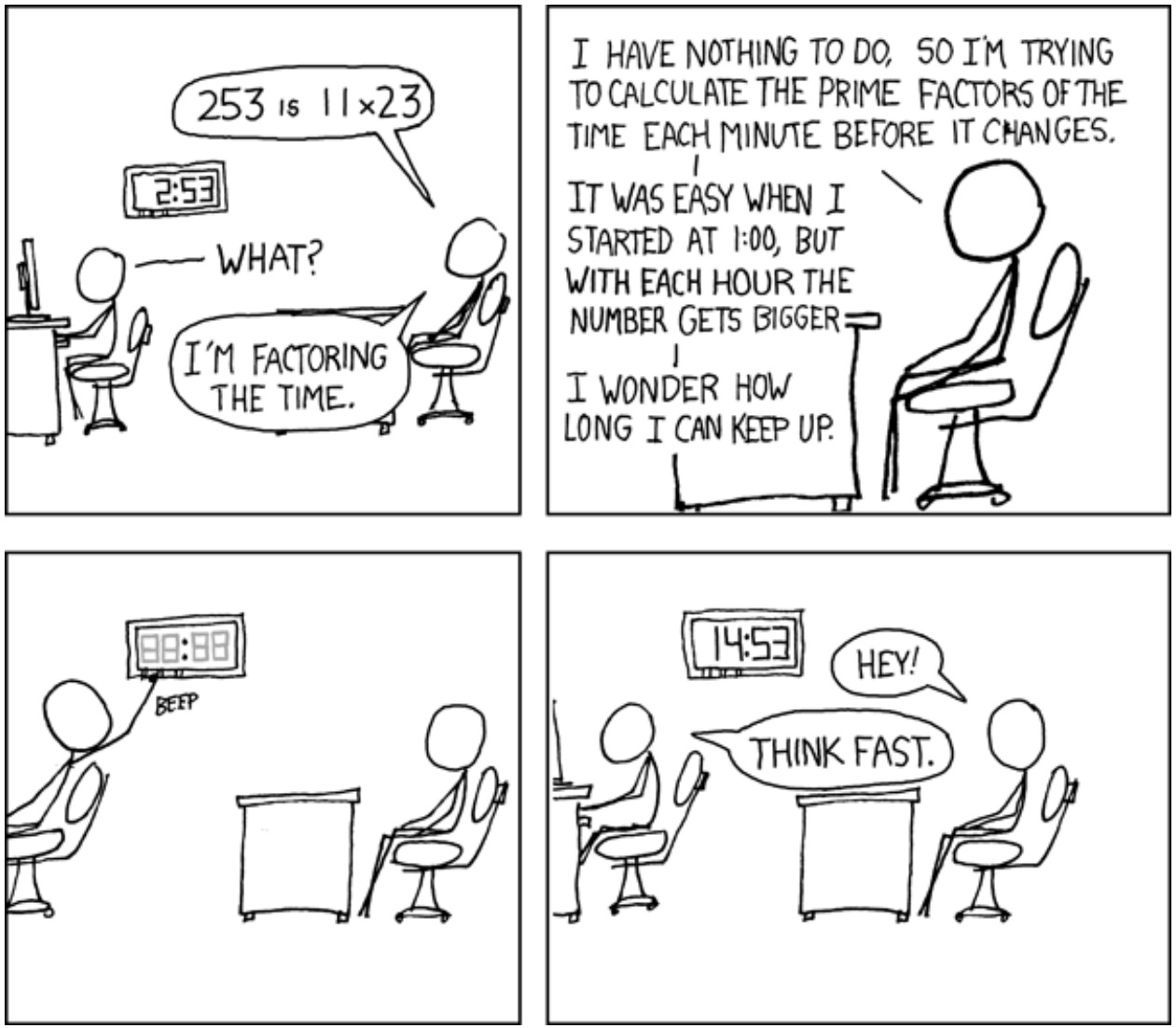
(wobei $\left(\frac{x}{n}\right)$ das *Jacobi-Symbol* ist)

$$W_n := \{x \in \mathbb{Z}_n^* \mid x^d \equiv 1 \text{ oder } x^{2^r d} \equiv -1 \text{ für ein } r < s\}$$

(wobei $n-1 = 2^s d$ mit d ungerade)

- Solovay-Strassen (1977):
  V_n ist Untergruppe, und es gilt $V_n = \mathbb{Z}_n^*$ genau dann wenn n prim ist.
- Miller-Rabin (1980):
  Es gilt $W_n \subseteq V_n \leq U_n \leq \mathbb{Z}_n^*$, und es gilt $W_n = \mathbb{Z}_n^*$ genau dann wenn p prim ist.

Das Faktorisierungsproblem



xkcd.com/247

Das Faktorisierungsproblem

Dieses berühmte Problem wird oft in Public-Key Kryptographie genutzt.

Definition

Das Faktorisierungsproblem lautet:

gegeben eine zusammengesetzte Zahl $n = p \cdot q$,
wobei p, q verschiedene Primzahlen mit $\log p \approx \log q$,

finde die Primfaktoren p, q von n.

Dies ist der schwierigste Fall des *allgemeinen* Faktorisierungsproblems, also die Primfaktorisierung $n = \prod_i p_i^{e_i}$ für allgemeine Zahl n zu finden.

Der aktuelle Faktorisierungsrekord ist eine 829 Bit RSA-Zahl (250 Stellen) in 2020, unter Verwendung von etwa 31 Million core-Stunden.

Überblick von Faktorisierungsalgorithmen

1. **Trial division.**
   Teste für alle Primzahlen p bis zu einer Schranke, ob diese die Zahl n teilen; wenn ja, ersetze n durch n/p.

2. **Pollards $p-1$-Methode.**
   Funktioniert gut, wenn ein Primfaktor p glattes $p-1$ besitzt.
   Sei angenommen $p-1 \mid k := \prod_{q^e \leq B} q^e$. Dann gilt $a^k \equiv 1 \bmod p$.
   Wenn $a^k \not\equiv 1 \bmod n$ finde Faktor durch $\mathrm{ggT}(a^k - 1, n)$.

3. **Elliptische Kurven.**
   Erweiterung von Pollards $p-1$-Methode, die gut für jedes n funktioniert; sie findet Primfaktoren p in Zeit $L_p(\frac{1}{2})$.

Kongruente Quadrate

4. **Kongruente Quadrate.**
   Finde ganze Zahlen x, y so dass

$$x^2 \equiv y^2 \mod n\,.$$

   In diesem Fall haben wir $n \mid x^2 - y^2 = (x-y)(x+y)$.
   Mit Wahrscheinlichkeit $\approx \frac{1}{2}$ (sofern x, y zufällig unabhängig) haben wir $x \not\equiv \pm y$, in welchem Fall $\mathrm{ggT}(x \pm y, n)$ die Faktoren liefern.

   Solche kongruenten Quadrate können mit Indexkalkülmethoden gefunden werden, wie dem *quadratischen Sieb*.

Ansatz

Sei $m := \lfloor \sqrt{n} \rfloor$ und betrachte $f := (X+m)^2 - n$.

Sei $\mathcal{S} := \{s \text{ prim} \mid s \leq B\} \cup \{-1\}$ die *Faktorbasis*.
Suche nach $|x| \leq C$ so dass $f(x)$ B-glatt ist, also über S faktorisiert.

Beispiel. Sei $n = 7429$, also $m = 86$ und $f := (X+86)^2 - 7429$.
Wähle $S := \{-1, 2, 3, 5, 7\}$. Dann erhalten wir

$$f(-3) = 83^2 - 7429 = -540 = (-1) \cdot 2^3 \cdot 3^2 \cdot 5$$
$$f(1) = 87^2 - 7429 = 140 = 2^2 \cdot 5 \cdot 7$$
$$f(2) = 88^2 - 7429 = 315 = 3^2 \cdot 5 \cdot 7$$

Wir kombinieren die rechten Seiten um ein Qudrat zu erzeugen:

$$f(1) \cdot f(2) \equiv (87 \cdot 88)^2 \equiv (2 \cdot 3 \cdot 5 \cdot 7)^2 \bmod n$$

ergibt das kongruente Quadrat $227^2 \equiv 210^2 \bmod n$.

Lineare-Algebra-Schritt

Angenommen wir haben $x_1, \ldots, x_k$ gefunden mit

$$f(x_i) \equiv \prod_{s \in \mathcal{S}} s^{e_{i,s}} \mod n$$

B-glatt. Suche nach Teilmenge $J \subseteq \{1, \ldots, k\}$ so dass die Summen $\sum_{j \in J} e_{j,s} = 2f_s$ gerade sind. Dann haben wir ein kongruentes Quadrat

$$\prod_{j \in J} f(x_j) \equiv \Big(\prod_{s \in \mathcal{S}} s^{f_s} \Big)^2 \mod n\,.$$

Beispiel. Im vorigen Beispiel sind die Exponenten $e_{i,s}$:

x_i	-1	2	3	5	7
-3	1	2	3	1	0
1	0	2	0	1	1
2	0	2	0	1	1

Betrachtet man die Zeilen modulo 2, so kann mit linearer Algebra über $\mathbb{Z}_2 = \{0, 1\}$ gewünschte nicht-triviale Linearkombination gefunden werden.

Sieben

Naiv: Teste alle $f(x)$ an den Stellen $x = 0, \pm 1, \pm 2, \ldots$ auf B-Glattheit.

Clever: Betrachte ein *Siebe-Intervall* $\{-C, \ldots, C\}$, und berechne $f(x)$ für alle $|x| \leq C$. Für alle $p \in \mathcal{S}$ teile $f(x)$ durch p so oft wie möglich:

- Finde Lösungen $y \in \mathbb{Z}_p$ für $f(y) \equiv 0 \bmod p$, wenn sie existieren
- dann $p \mid f(x)$ wenn immer $x = \ldots, y-2p, y-p, y, y+p, y+2p, \ldots$.

Beispiel. Betrachte wieder $n = 7429$ und sei $C = 3$.
Dann $f \bmod 2 = X^2 + 1$, $f \bmod 3 = X^2 + X$, $f \bmod 5 = (X+1)^2 + 1$ und $f \bmod 7 = (X+2)^2 - 2$. Wir erhalten folgende Siebetabelle:

x	-3	-2	-1	0	1	2	3
$f(x)$	-540	-373	-204	-33	140	315	492
2	-135		-51		35		123
3	-5		-17	-11		35	41
5	-1				7	7	
7					1	1	

Fortgeschrittenes Indexkalkül

L-Notation: Für $\alpha \in [0,1]$ und $c \in \mathbb{R}_{>0}$ sei

$$L_n(\alpha, c) := \exp\left((c+o(1))(\log n)^{\alpha}(\log\log n)^{1-\alpha}\right).$$

Ist die Konstante c nicht spezifiziert, so schreiben wir $L_n(\alpha)$.
Also bedeutet $\alpha = 0$ polynomiell und $\alpha = 1$ exponentiell in $\log n$.
Im Fall $0 < \alpha < 1$ spricht man von *subexponentieller* Komplexität.

Das quadratische Sieb hat eine Laufzeit von $L_n(\frac{1}{2}, c) = \mathcal{L}_n(c)$.

Dies kann jedoch weiter verbessert werden!
Die grundlegende Idee von $L(\frac{1}{3})$-Algorithmen ist, Relationen auf eine Weise zu generieren so dass die Elemente auf beiden Seiten sich zufällig verhalten; sie müssen dann gleichzeitig glatt sein, sind aber von beträchtlich "kleinerer" Größe.

Weitere Bemerkungen

- $L(\frac{1}{3})$-Algorithms für das *diskrete Logarithmusproblem* (DLP)
  - Das Zahlkörpersieb kann auch für das DLP in $\mathbb{Z}_p^*$ verwendet werden.
  - Für das DLP in $\mathbb{F}_{q^m}^*$ (mit q "klein") kann der Algorithmus adaptiert werden, dies ist das sogenannte *Funktionenkörpersieb*.
- Quasi-polynomielle Algorithmen!
  - Für endliche Körper kleiner Charakteristik (wie $\mathbb{F}_{2^m}^*$) wurde das Funktionenkörpersieb drastisch verbessert. Man kann so einen *quasi-polynomiellen* $m^{O(\log m)}$ Algorithmus, was in $L(0+o(1))$ ist.

Diskretes Logarithmusproblem

Sei p eine Primzahl.

$a \bmod p$ sei der Rest (in $\{0, \ldots, p-1\}$) bei Division der Zahl a durch p.

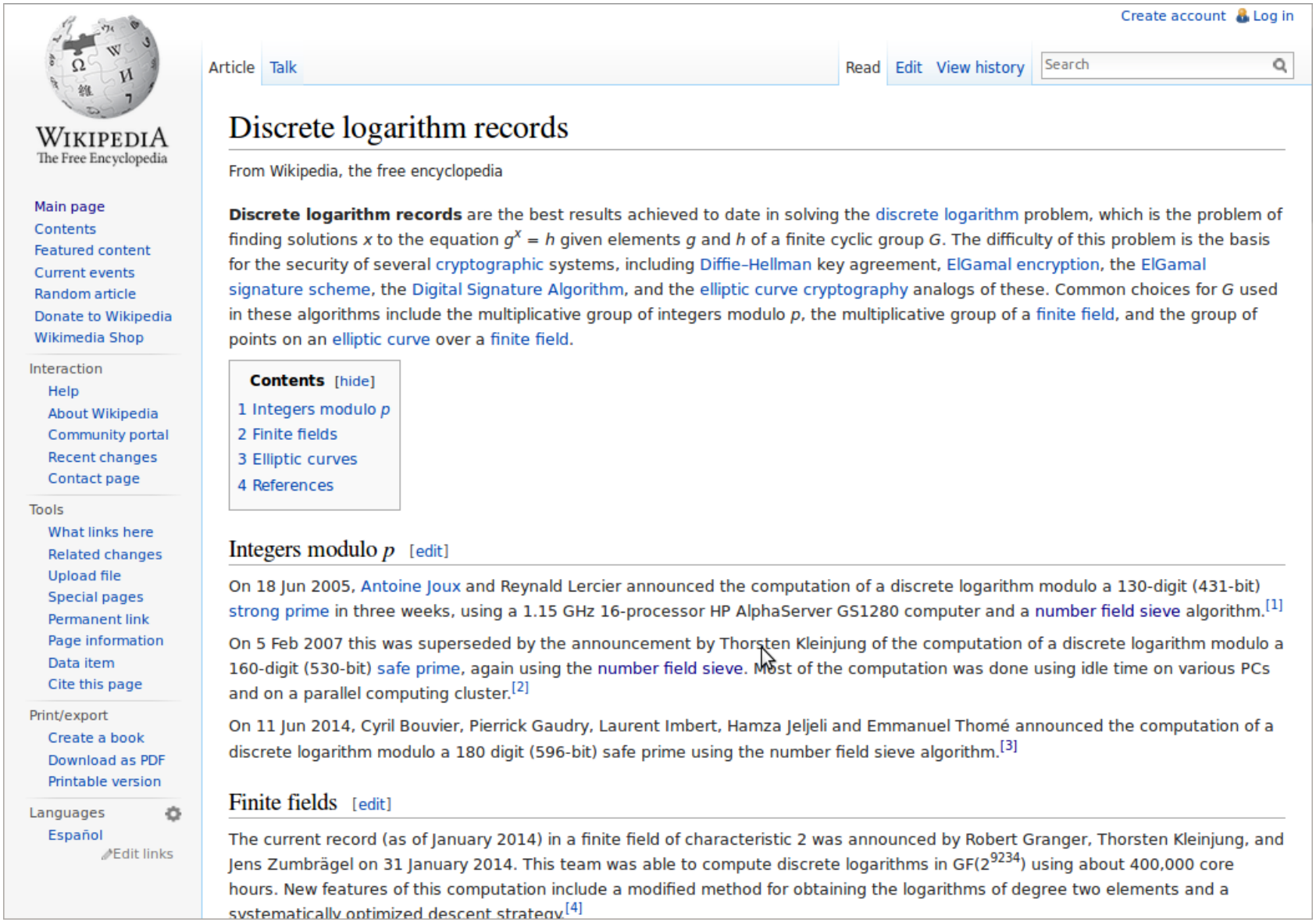
Eine *Primitivwurzel* modulo p ist eine Zahl g,
so dass $g^x \bmod p$ für $x \in \mathbb{N}$ alle Reste in $\{1, \ldots, p-1\}$ durchläuft.

Beispiel: Sei $p = 13$ und $g = 2$. Für $2^x \bmod 13$ erhalten wir

x	0	1	2	3	4	5	6	7	8	9	10	11
$2^x \bmod 13$	1	2	4	8	3	6	12	11	9	5	10	7

Das diskrete Logarithmusproblem ist, zu einer gegebenen Zahl b eine Zahl x zu finden mit $g^x \equiv b \mod p$.

Wikipedia



Create account Log in

Article Talk | Read Edit View history | Search

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikimedia Shop

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Data item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Español
Edit links

Discrete logarithm records

From Wikipedia, the free encyclopedia

Discrete logarithm records are the best results achieved to date in solving the discrete logarithm problem, which is the problem of finding solutions x to the equation $g^x = h$ given elements g and h of a finite cyclic group G. The difficulty of this problem is the basis for the security of several cryptographic systems, including Diffie–Hellman key agreement, ElGamal encryption, the ElGamal signature scheme, the Digital Signature Algorithm, and the elliptic curve cryptography analogs of these. Common choices for G used in these algorithms include the multiplicative group of integers modulo p, the multiplicative group of a finite field, and the group of points on an elliptic curve over a finite field.

Contents [hide]
1 Integers modulo p
2 Finite fields
3 Elliptic curves
4 References

Integers modulo p [edit]

On 18 Jun 2005, Antoine Joux and Reynald Lercier announced the computation of a discrete logarithm modulo a 130-digit (431-bit) strong prime in three weeks, using a 1.15 GHz 16-processor HP AlphaServer GS1280 computer and a number field sieve algorithm.[1]

On 5 Feb 2007 this was superseded by the announcement by Thorsten Kleinjung of the computation of a discrete logarithm modulo a 160-digit (530-bit) safe prime, again using the number field sieve. Most of the computation was done using idle time on various PCs and on a parallel computing cluster.[2]

On 11 Jun 2014, Cyril Bouvier, Pierrick Gaudry, Laurent Imbert, Hamza Jeljeli and Emmanuel Thomé announced the computation of a discrete logarithm modulo a 180 digit (596-bit) safe prime using the number field sieve algorithm.[3]

Finite fields [edit]

The current record (as of January 2014) in a finite field of characteristic 2 was announced by Robert Granger, Thorsten Kleinjung, and Jens Zumbrägel on 31 January 2014. This team was able to compute discrete logarithms in GF(2^{9234}) using about 400,000 core hours. New features of this computation include a modified method for obtaining the logarithms of degree two elements and a systematically optimized descent strategy.[4]

DLP-Meilensteine in endlichen Körpern

DLP in $\mathbb{F}_q$, wobei $q = p^\ell$ und $m = \log_2 q$.

bitlength m	char p	who/when	complexity
127	2	Coppersmith 1984	$L(1/3\,,\,1.526..1.587)$
401	2	Gordon, McCurley 1992	$L(1/3\,,\,1.526..1.587)$
n/a	small	Adleman 1994	$L(1/3\,,\,1.923)$
427	large	Weber, Denny 1998	$L(1/3\,,\,1.526)$
521	2	Joux, Lercier 2001	$L(1/3\,,\,1.526)$
607	2	Thomé 2001	$L(1/3\,,\,1.526..1.587)$
613	2	Joux, Lercier 2005	$L(1/3\,,\,1.526)$
556	medium	Joux, Lercier 2006	$L(1/3\,,\,1.442)$
676	3	Hayashi et al. 2010	$L(1/3\,,\,1.442)$
923	3	Hayashi et al. 2012	$L(1/3\,,\,1.442)$

DLP-Meilensteine in endlichen Körpern

DLP in $\mathbb{F}_q$, wobei $q = p^\ell$ und $m = \log_2 q$.

bitlength m	char p	who/when	complexity
1175	medium	Joux 24 Dec 2012	$L(1/3\,,\,1.260)$
1425	medium	Joux 6 Jan 2013	$L(1/3\,,\,1.260)$
1778	2	Joux 11 Feb 2013	$L(1/4+o(1))$
1971	2	GGMZ 19 Feb 2013	$L(1/3\,,\,0.763)$
4080	2	Joux 22 Mar 2013	$L(1/4+o(1))$
6120	2	GGMZ 11 Apr 2013	$L(1/4)$
6168	2	Joux 21 Mai 2013	$L(1/4+o(1))$
n/a	small	BGJT 18 Jun 2013	$L(0+o(1))$
4404	2	GKZ 30 Jan 2014	$L(1/4+o(1))$
9234	2	GKZ 31 Jan 2014	$L(1/4+o(1))$
n/a	small	GKZ 06 Jul 2015	$L(0+o(1))$
30750	2	GKLWZ 28 Mai 2019	$L(0+o(1))$