Zeit-Experimente zur Faktorisierung, ein Beitrag zur Didaktik der Kryptologie

Ralph-Hardo Schulz und Helmut Witten

erscheint voraussichtlich in LogIn
Version vom 29. November 2010

Bei der Übertragung von vertraulichen Nachrichten mit dem RSA-Kryptosystem (siehe z.B. [Witten_Schulz_2006-2010] oder [Schulz_2003]) werden diese durch geeignetes Potenzieren und Rechnung modulo n verschlüsselt; hierbei ist n Teil des öffentlichen Schlüssels und als Produkt von zwei geheimzuhaltenden Primzahlen p und q gewählt. Die Sicherheit der Verschlüsselung steht und fällt dabei mit der Un-Möglichkeit, n in seine beiden Primfaktoren zu zerlegen. Während das Multiplizieren zweier Zahlen (mit Langzahlarithmetik) sehr schnell geht, ist der Zeitaufwand bzw. überhaupt die Möglichkeit der Faktorisierung von der Ziffernlänge von p und q und damit von n abhängig. Es sollte also $n = p * q$ so gewählt werden, dass n nicht in praktikabler Zeit faktorisierbar ist und damit nicht der Einwegfunktions-Charakter der Verschlüsselung verloren geht. Das Bundesamt für Informationstechnik (BSI) schreibt dazu (siehe [BSI])

> "Bei asymmetrischen Verfahren sollte die Mechanismenstärke so gewählt werden, dass die Lösung der zu Grunde liegenden mathematischen Probleme einen unvertretbar großen bzw. praktisch unmöglichen Rechenaufwand erfordert (die zu wählende Mechanismenstärke hängt daher vom gegenwärtigen Stand der Algorithmik und der Rechentechnik ab)".

Da der von Experten für die Faktorisierung von 1024-Bit RSA-Moduli vorhergesagte Aufwand von circa 2 hoch 80 Operationen mit Fortschreiten der Rechentechnik allmählich in den Bereich des technisch Machbaren geriete, so das BSI, sollten für langfristige Sicherheitsanwendungen 2048-Bit RSA-Moduli eingesetzt werden. In der Praxis werden (z.B. zur Sicherung von WLAN-Netzen) zur Zeit sogar schon Module der Länge 4096 verwendet.

Als grober Gradmesser für die Grenzen der zur Zeit möglichen Zerlegbarkeit können dabei die Ergebnisse bei der Lösung der Aufgaben der sogenannten RSA-Challenge dienen (siehe [RSA]): Mit enormen Anstrengungen konnte ein Team von 13 Wissenschaftlern die Zahl RSA-768 (eine Zahl mit 768 binären bzw. 232 dezimalen Stellen) faktorisieren; nach ihren Angaben hätte das Faktorisieren auf einem herkömmlichen PC rund 2000 Jahre gedauert (s. [Kleinjung_et_al_2010]).

Auch RSA-640 (mit 193 Dezimalstellen) ist faktorisiert, hingegen nicht RSA-704 (mit 212 Dezimalstellen) und RSA-896 (mit 270 Dezimalstellen).

In dem Artikel "Primfaktorzerlegung. Experimente zum Zeitaufwand" (s. [Schulz_1996]) wurde versucht, das Phänomen des enormen Anstiegs der Rechenzeiten der Faktorisierung bei zunehmender Ziffernlänge von n mit der uns damals zur Verfügung stehenden Rechenkapazität nachzuempfinden. Im vorliegenden Artikel beschreiben wir eine Aktualisierung mit Hilfe der Software Systeme **SAGE** (s.[Sage])[1] und **CrypTool** (Version 1.4.30, hier mit CT1 bezeichnet, s. [CrypTool 1])[2] bzw. der neuen Version CrypTool 2.0 (z.Zt. 2.3631a(beta)) (s. [CrypTool 2], hier mit CT2 bezeichnet. Bei Sage werden viele hoch-qualifizierte mathematische 'open source'- Pakete kombiniert, die man z.B. online nutzen kann. CrypTool ist ebenfalls eine freie Software, die die Konzepte der Kryptographie und der Kryptoanalyse erfahrbar macht. CrypTool ist weltweit das verbreitetste Lernprogramm dieser Art.

Ziel

unserer Experimente ist es wiederum, bei der Faktorisierung von $n = p * q$ den Anstieg der Rechenzeiten in Abhängigkeit von der Ziffernlänge[3] von n mit allgemein zugänglichen (und weitgehend kostenlosen) Mitteln (online auf dem Sage Notebook oder auf eigenem PC mit CrypTool) experimentell zu untersuchen. (Dabei erreicht man natürlich nicht die eben erwähnten Spitzenleistungen wie bei RSA-768).

Vorgehen

Dazu haben wir unterschiedlich große "Semiprimzahlen" n eingegeben, also Zahlen, die jeweils Produkt zweier Primzahlen sind. Bei Sage lautet der Befehl "time factor(n)" (unter Sage selbst, nicht unter Python, s. Abbildung 1). Wenn man nicht speziellere Faktorisierungs-Verfahren (wie das Quadratische Sieb oder ECM, die

[1] Eine Kurzanleitung für Sage findet man in dem CrypTool-Skript, s. [CrypTool_2010], Seite 247 ff.

[2] Eine Dokumentation zu CrypTool findet man im CrypTool-Skript, siehe [CrypTool_2010].

[3] Für die dezimale Ziffernzahl $\ell(n)$ von $n = p * q$ gilt

$$\ell(n) = \ell(p) + \ell(q) - 1 \text{ oder } \ell(n) = \ell(p) + \ell(q).$$

Denn mit $p = p_1 * 10^{\ell(p)-1}$ und $q = q_1 * 10^{\ell(q)-1}$ ist $1 \leq p_1, q_1 < 10$ und damit
$p * q = p_1 * q_1 * 10^{[\ell(p)+\ell(q)-1]-1}$ von der Länge $\ell(p) + \ell(q) - 1$, falls $1 < p_1 * q_1 < 10$ oder $\ell(p) + \ell(q)$, falls $10 \leq p_1 * q_1 < 100$.

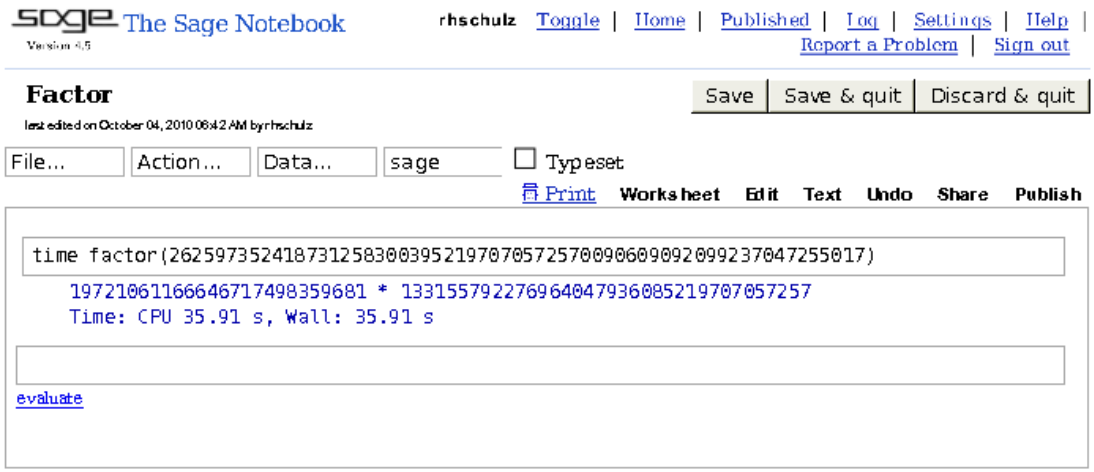

Abbildung 1: Maske beim Faktorisieren bei Sage

Elliptische Kurven Methode) verlangt, so wird das Paket von PARI aufgerufen, welches ebenfalls Sieb- und ECM-Algorithmen implementiert (lt. Beschreibung des Befehls “factor” bei Sage).

Für die Verwendung von CrypTool 1 (CT1) sollte man das Programmpaket (siehe [CrypTool 1]) mit Windows oder Windows-Emulator herunterladen und dann die Menüs

“Einzelverfahren/RSA-Kryptosystem/Faktorisieren einer Zahl”

benutzen[4]! In CT1 sind folgende Faktorisierungs-Algorithmen implementiert (s. Abbildung 2): Brute-Force-Methode, Algorithmus nach Brent, Pollards (p-1)- Methode, Williams (p+1)-Methode, Algorithmus nach Lenstra (ECM) und Quadratische Sieb-Methode (QS).[5]

[4]Die Online-Hilfe von CrypTool zum Faktorisieren erhält man durch den Aufruf “Hilfe” und der erwähnten Menüeinträge oder aber an jeder Stelle des Programms durch F1 !

[5]Man beachte auch hier die “Hilfe” von CT1, bei der einige Kommentare zu den Methoden stehen: So wird empfohlen, Brute-Force und Brent als Vorstufe zu verwenden, da diese Verfahren sehr schnell kleine Faktoren finden. Pollards (p-1)- bzw. Williams (p+1)-Methode führe mit großer Wahrscheinlichkeit zum Erfolg, wenn die zu faktorisierende Zahl einen Primfaktor p derart hat, dass (p-1) bzw. (p+1) nur aus kleinen Primfaktoren besteht. Und ECM sei für Faktoren mit bis zu 30 Dezimalstellen geeignet, hingegen das Quadratische Sieb zur Faktorisierung von Zahlen, die aus mehr als einem großen Faktor (mit mehr als 30 Stellen) zusammengesetzt sind.
Eine Beschreibung der Quadratischen Sieb-Methode findet man z.B. in [Wikipedia_1]; die Größenordnung der Laufzeit zum Faktorisieren von n wird dort (nach [Pomerance_1996]) unter bestimmten Voraussetzungen als $\exp(\sqrt{\ln n \cdot \ln\ln n})$ angegeben. Auch Pollards $(p-1)$-Verfahren ist in Wikipedia beschrieben (siehe [Wikipedia_3]).

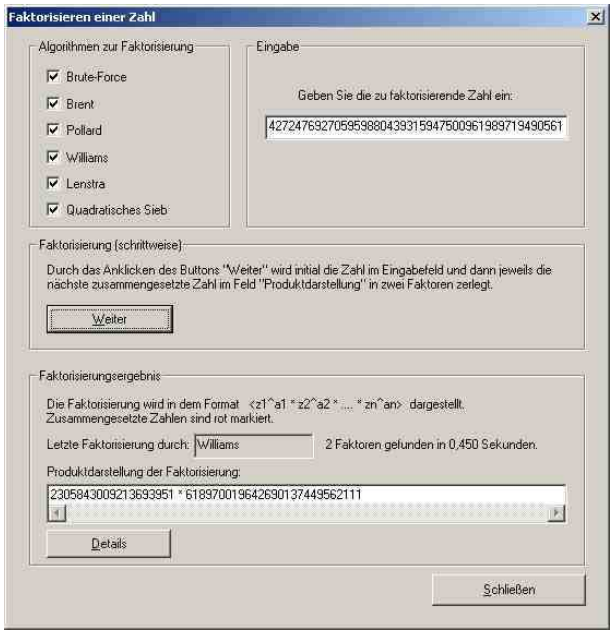

Abbildung 2:
Maske zum Faktorisieren bei CrypTool 1

Alle diese Algorithmen kann man zunächst gleichzeitig anwenden; sie werden dann nach und nach abgeschaltet, wenn sie nicht zum Erfolg führen; man kann sie aber auch einzeln starten, was wir dann im zweiten Durchgang mit den jeweils erfolgreichen Algorithmen ausgeführt haben.
Ein Problem vor dem Starten der Faktorisierung ist die Bereitstellung von geeigneten Faktoren für Semiprimzahlen, d.h. die **Gewinnung von Primzahlen geeigneter Ziffern-Anzahl** [6], was experimentell z.B. durch Faktorierung von zufällig gewählten Zahlen oder Verwendung von Mersenne-Zahlen (d.h. Zahlen der Form $2^p - 1$ mit p prim, s. z.B. [Ribenboim_1996]) gelingt. Gezielter kann man auch mit dem Befehl “p=next_prime(m);p”(bei Sage) oder “nextprime(m);” (z.B. in MAPLE) arbeiten, mit dem sich die kleinste Primzahl oberhalb m bestimmen lässt; eine Zufallsvariante stellt “p=next_prime(randint($10^{\ell-1}, 10^{\ell}$));p” dar. Bei CrypTool ist die Gewinnung der Primzahlen vorgegebener Größe noch einfacher, nämlich mit den Menüpunkten (vgl. Abbildung 3):

[6] Im RSA-System sollten p und q weder zu weit auseinander noch zu nahe beieinander sein; denn wenn einer der beiden Faktoren wesentlich kleiner ist, so ist die Wahrscheinlichkeit der Faktorisierung bei gegebener Länge $\ell(n)$ größer, ebenso falls beide Faktoren nahe beieinander liegen: Aus $p = a - x$ und $q = a + x$ folgt $b(x) := p * q + x^2 = a^2$, sodass für “laufende” x nur zu prüfen ist, ob $b(x)$ Quadrat ist (“Faktorisierungsmethode von Fermat”).

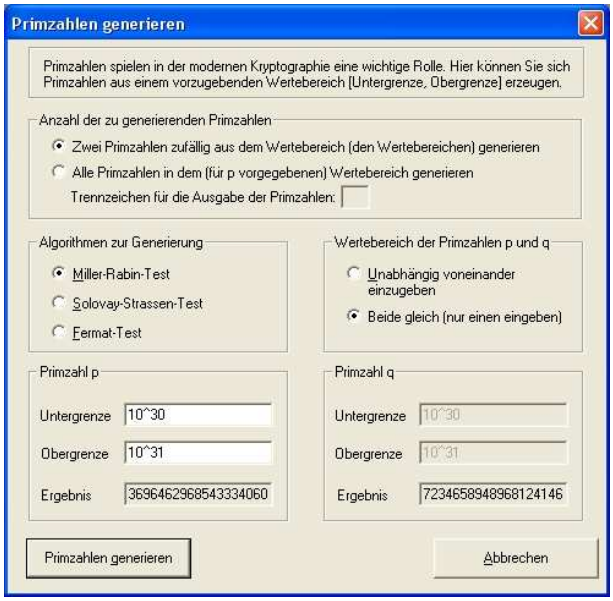

Abbildung 3:
Maske zum Generieren von Primzahlen bei CrypTool 1

"Einzelverfahren/RSA-Kryptosystem/Primzahlen generieren".
Die Semiprimzahlen erhält man dann in sehr kurzer Rechenzeit durch Produktbildung: $n = p * q$.

Bei CrypTool 2 (CT2) wird zum Faktorisieren eine modifizierte "msieve"-Bibliothek verwendet; Msieve verwendet unter CT2 hauptsächlich das sogenannte MPQS (Multiple polynomial quadratic sieve, s. z.B. [Wikipedia_1]). [7]
Dieses Sieb wird allerdings erst angewandt, nachdem mit einigen "einfacheren" Algorithmen versucht wurde, die Zahl möglichst schnell zu zerlegen. (Dieser Versuch dauert ungefähr 1-2 Sekunden.) Erst wenn das nicht gelingen sollte, wird das MPQS verwendet. Zu den einfacheren Algorithmen zählen hier: triviale Divisionen (durch kleine Primzahlen), Pollards ρ-Methode (s. z.B. [Wikipedia_2]), $P+-1$ und die Elliptische Kurven-Methode ECM.

Bei CT2 ist auch verteiltes paralleles Rechnen möglich; damit lässt sich (gleich-

[7] "msieve" gibt es als eigenständige Konsolenanwendung bei http://sourceforge.net/projects/msieve/ (oder über http://www.boo.net/ jasonp/qs.html). Das "msieve", das man von dieser Projektseite laden kann, unterstützt von Haus aus immer nur einen Kern. Die Unterstützung von beliebig vielen Kernen wurde von Sven Rech bei der Integration in CT2 zu der Original-Bibliothek hinzugefügt.

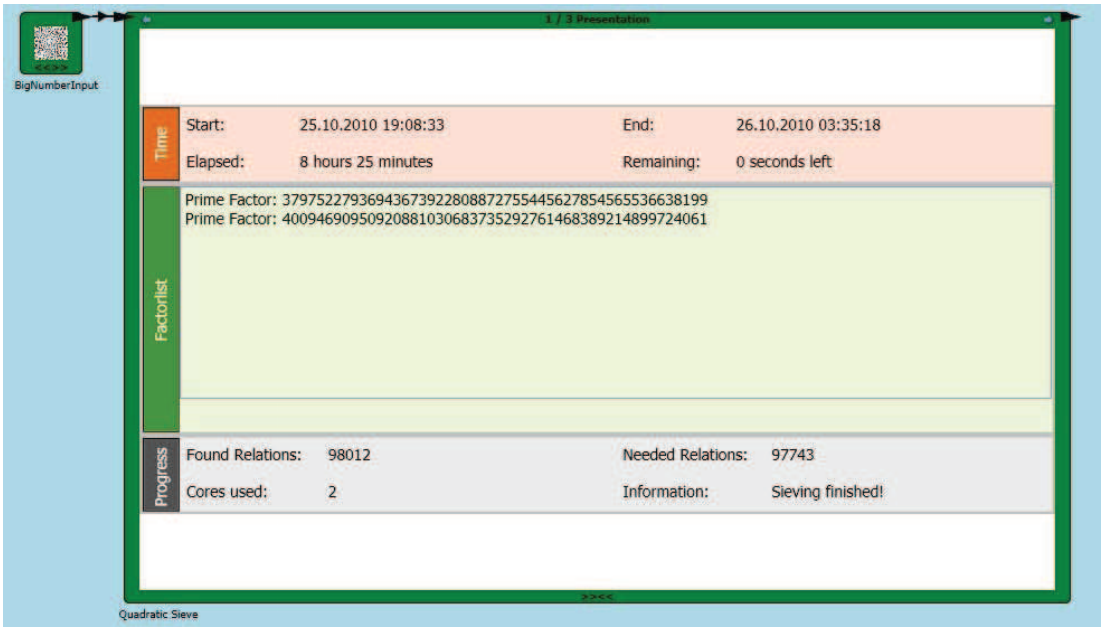

Abbildung 4:
Bildschirmphoto nach erfolgter Zerlegung von RSA-100 durch CrypTool 2

artige Rechner vorausgesetzt) die zum Faktorisieren benötigte Rechenzeit durch die Anzahl der beteiligten Rechner teilen.[8]

Ergebnis

Das Ergebnis unserer Experimente ist in den untenstehenden Tabellen (Tabelle 1 für Sage-, Tabellen 2 und 3 für CrypTool-Berechnungen) und ausführlich in der durch den LogIn-Service zu erhaltenden Tabelle 4 angegeben und[9] graphisch dargestellt, nämlich in den Abbildungen 5, 7 und 9 (Zerlegungszeit t in Abhängigkeit von der dezimalen Ziffernzahl $z = l(n)$ von n) sowie in den Abbildungen 6 , 8 und 10 (mit $\ln(t)$ als Funktion von z, also halb-logarithmisch). Man sieht bei den Abbildungen 6 und 10 deutlich und bei Abbildung 8 mit Ausreißern, dass der Anstieg der Rechenzeit bei der Faktorisierung ungefähr exponentiell erfolgt und damit sehr schnell an die Kapazitätsgrenze des verwendeten Rechners stößt.

Diskussion

[8] Weil CT2 das P2P-Framework für verteiltes Rechnen inkludiert hat, kann man ad-hoc-Netze zwischen unterschiedlichen Personen aufbauen, die ihre Rechner einer gemeinsamen Aufgabe, also z.B. dem Faktorisieren, zur Verfügung stellen.

[9] Mit Hilfe des Tools 'Create Chart' von Microsoft Office Excel

Bei der Rechnung auf Sage, die stets mit Pari ausgeführt wurde, zeigt Abbildung 6 mit halblogarithmischen Koordinaten nur geringe Abweichungen von dem (durch eine Gerade angezeigten) exponentiellen Verlauf; es stört weder, dass die Abweichungen der Längen der Faktoren zum Teil nicht ganz klein sind noch dass gelegentlich Mersenne-Primzahlen (also Primzahlen $M_p := 2^p - 1$ mit p prim) genommen wurden. Unklar ist, ob die geringfügig unterschiedlichen Rechenzeiten für ein und dieselbe Zerlegungs-Aufgabe bei gleichem Algorithmus (s. die Anmerkungen in Tabelle 1 bzw. in der über den LogIn-Service zu erhaltenden Tabelle 4) dadurch entstehen, dass der verwendete Algorithmus nicht ganz deterministisch arbeitet, oder dass dies zu den normalen Zeitabweichungen je nach Auslastung bei Multiuser/Multitasking-Betriebssystemen gehört.

Bei der Rechnung mit CT1 werden die größeren Ausreißer verständlich, wenn man die Verfahren mit berücksichtigt; sie sind nämlich alle bei Pollards (p-1)-Algorithmus bzw. Williams (p+1)-Verfahren aufgetreten, die, wie schon erwähnt, günstig sind, wenn die zu faktorisierende Zahl einen Primfaktor p hat, so dass $(p-1)$ bzw. $(p+1)$ nur Produkt kleiner Primfaktoren ist. So hat z.B. bei unserer Zahl der Länge 63 die Zahl p-1 nur Faktoren der Länge 1,3,4,5,6 und 13. Außerdem sind bei allen großen Abweichungen Mersenne-Primzahlen involviert (bei denen ja $M_p + 1 = 2^p$ die erwähnte Eigenschaft hat). Alle stärkeren Ausreißer waren Ausreißer nach unten, d.h. führten zu einer schnelleren Faktorisierung. Dies zeigt: Die Fälle, in denen Spezialalgorithmen eine Chance haben, sind die Ausnahme; andererseits verlangen diese Sonderfälle, dass man bei der Wahl von Primzahlen Tests durchführt, bevor man sie für RSA benutzt.

Erwartungsgemäß verringern sich die Rechenzeiten bei der Rechnung mit CT1, wenn man (im 2. Durchgang) das günstigste Verfahren fest auswählt.

Eine wesentlich größere Leistungsfähigkeit und daraus resultierende Reduktionen der Rechenzeiten zeigen sich aber bei Berechnungen mit CrypTool 2: Nicht nur wird die Hardware besser genutzt (z.B. durch Verwendung beider Kerne), sondern es stehen auch leistungsfähigere Algorithmen zur Verfügung wie das “Multiple Polynomial Quadratic Sieve” MPQS (s. [Wikipedia_1])[10].

[10]Die msieve-Bibliothek enthält die derzeit vermutlich am höchsten optimierte QS-Implementierung. Der Autor Jason P. hat viele Jahre daran gearbeitet und rein durch programmtechnische (nicht algorithmische) Optimierung immer bessere Ergebnisse erzielt.
Allerdings ist bei CrypTool noch nicht das Zahlkörpersieb (s. z.B. [Wikipedia_5]) implementiert, mit dem u.a. RSA-768 (s.o.) faktorisiert wurde.

Zwei besondere Zahlen

Von den Zerlegungen wollen wir hier kurz zwei besonders hervorheben. Die eine betrifft die 10-stellige Zahl 8616460799, die Zahl von William S. Jevon. Dieser hatte in einem 1873 erschienen Buch schon den möglichen Einwegcharakter der Produktbildung zweier großer Primzahlen – einer der Schlüsseltatsachen beim RSA-Kryptosystem – bemerkt und geschrieben[11]:

> "Kann der Leser sagen, welche zwei Zahlen miteinander multipliziert die Zahl 8616460799 ergeben? Ich denke, es ist unwahrscheinlich, dass das irgendjemand außer mir selbst je wissen wird."

Hier irrte sich Jevon gewaltig[12]: Schon 1903 konnte D. N. Lehmer die Faktoren angeben, und S.W. Golomb zeigte in einem Artikel (s. [Golomb_1996]), dass schon Zeitgenossen von Jevon mit den damaligen Hilfsmitteln die Zerlegung in wenigen Stunden, wahrscheinlich schon innerhalb einer Stunde, hätten leisten können. Auch Günter Ziegler erzählt die Geschichte von Jevons Zahl in seinem hübschen Buch "Darf ich Zahlen" (s. [Ziegler_2010], S.40f). Mit Sage oder CrypTool gelingt die Zerlegung in weniger als 0,01 Sekunden.

Die zweite Zahl, die wir hier hervorheben wollen, ist die Zahl RSA-100 aus der RSA-Challange (s. z.B. [Wikipedia_4]), nämlich

1522605027922533360535618378132637429718068114961380688657908494 5/
8012296325895289765400035069200613 9,

eine Zahl mit 100 Dezimalstellen; diese wurde schon 1991 von Arjen K. Lenstra mit dem MPQS in wenigen Tagen zerlegt. Erstaunlicher Weise benötigt CT2 nur 8 Stunden und 25 Minuten zur Zerlegung (s. Abbildung 4). Damit bietet sich durch Parallelschaltung von Rechnern die Möglichkeit, RSA-100 innerhalb einer Unterrichtsstunde zu faktorisieren.

Beachtlich ist also, welchen Fortschritt es bei Hard- und Software gegeben hat und wie sehr selbst Ron Rivest 1977 daneben lag, als er den Zeitbedarf für die Faktorisierung einer Zahl mit 125 dezimalen Stellen (selbst unter der Annahme, dass eine modulare Multiplikation in einer Nanosekunde ausgeführt werden kann) auf $4*10^{16}$ Jahre schätzte[13]; RSA-129 hielt er für praktisch unzerlegbar, was

[11] Hier aus dem Englischen übersetzt

[12] Eine Geschichte, die József Dénes (gemäß [Golomb_1996]) "ausgraben" konnte.

[13] Da man das Alter der Erde und des Sonnensystems auf $4{,}55*10^9$ Jahre schätzt (aktuell etwas niedriger, aber immer noch auf mindestens 4,44 Milliarden Jahre), wird die Diskrepanz der Zeitschätzung von Rivest zu der tatsächlich benötigten Zeit von CrypTool 2 besonders anschaulich.

Derek Atkins, Michael Graff, Arjen K. Lenstra and Paul Leylan 1994 widerlegen konnten. Auch auf privatem PC, so die Schätzung von CrypTool 2, liegt die zur Zerlegung von RSA-129 benötigte Zeit nur bei knapp 3 Monaten[14], bei Parallelisierung entsprechend weniger.
Was Rivest also nicht berücksichtigt hatte, war die Möglichkeit des Fortschritts bei der Entwicklung von Faktorisierungs-Algorithmen.

Fazit

Unsere Experimente zeigen, dass die Zerlegungszeiten von Semi-Primzahlen bei zunehmender Länge dieser Zahlen (bei gleichem Rechner und festem Algorithmus) ungefähr exponentiell ansteigen.

Infolge schnellerer Hardware und verbesserter Algorithmen kann man auf Einzelplatz-PCs Semi-Primzahlen mit einer Länge bis ca. 90 Dezimalziffern im Stundenbereich in ihre Primfaktoren zerlegen. Das aktuell schnellste Tool ist dabei CrypTool 2; im Jahr 1996 konnten wir eine 21-stellige Semiprimzahl mit 11-stelligen Faktoren in etwa 229 Sekunden zerlegen[15]; heute wird die Rechenzeit mit (abgerundeten) 0 Sekunden angegeben; und die 45-stellige Mersenne-Semiprimzahl M_{149} konnten wir damals nicht mehr in angemessener Zeit faktorisieren, heute dagegen in unter 2,5 Sekunden[16]. Teilweise (z.B. bei ca. 25-stelligen Zahlen) sind wir heute, 14 Jahre später, um 20.000 mal schneller.

Die heute aktuell eingesetzten Verfahren (bei SSL etc.) nutzen aber Semi-Primzahlen mit wesentlich mehr als 200 Dezimalstellen und liegen damit noch weit außerhalb der Reichweite von Einzel-PCs.

Anerkennung:

Herrn Prof. Bernhard Esslinger (CrypTool und Universität Siegen) danken wir herzlich für wertvolle Hinweise bei der Entstehung dieses Artilkels.
Unser Dank gilt auch Herrn Sven Rech vom Lehrstuhl für Verteilte Systeme der Universität Duisburg-Essen für technische Auskünfte zu CrypTool 2, insbesondere zur Bibliothek “msieve”.

[14] Genauer: bei 83 Tagen, 18 Stunden und 55 Minuten
[15] Mit dem Programm DERIVE 2.08 auf Compaq 386/20e; siehe [Schulz_1996]!
[16] Z.B. mit dem Quadratischen Sieb in CrypTool 1

l(n)	t(n)	ln(t(n))	l(p)	l(q)		Anmerkungen
10	0		5	5		(Jevons-Zahl)
25	0,01	-4,605170	13	13		auch 0,02 sec
34	0,03	-3,506558	15	19		
39	0,16	-1,832581	20	20		
40	0,14	-1,966113	20	20		
46	0,56	-0,579818	19	27	MM	
52	3,00	1,098612	26	26		
52	3,91	1,363537	26	26		
55	6,59	1,885553	28	28		
56	5,88	1,771557	28	28		
56	8,24	2,109000	28	29		
58	6,64	1,893112	27	31		auch 6,63 sec
60	13,24	2,583243	27	33	MM	auch 13,21 und 13,42 sec
60	16,33	2,793004	27	33		auch16,28 und 16,38 sec
61	26,56	3,279406	26	36		auch 35,91
62	52,09	3,952973	31	31		
63	37,42	3,622205	31	33	M	auch 37,37 sec
65	58,39	4,067145	26	40		
66	77,63	4,351954	33	34		
68	81,48	4,400358	34	34		
69	172,6	5,150977	31	39	M	auch 171,68 und 173,63 sec
70	189,63	5,245075	35	36		
71	184,55	5,217920	30	41		
71	188,63	5,239787	31	40		
71	175,18	5,165814	36	36		
71	143,66	4,967449	33	39	MM	auch 145,31 sec
72	263,93	5,575684	36	36		
72	202,29	5,309702	36	37		
72	143,42	4,965777	36	37		
72	171,47	5,144408	33	40	M	auch 181,05 und 173,73 sec
73	327,01	5,789991	37	37		
73	339,89	5,828622	33	41	M	auch 340,75 und 339,75 sec
74	313,4	5,747480	36	39		
75	440,86	6,088727	37	39	M	
75	609,14	6,412048	36	40		
75	518,56	6,251056	37	39		
76	905,3	6,80827	37	40		auch 912,72 sec
77	1285,47	7,158880	39	39		
78	838,49	6,731603	39	40	M	
79	>1800		40	40		Abbruxch

Tabelle 1: Ergebnis-Liste (Rechnung mit Sage/Pari) (Extrakt aus Tabelle 4)
Legende: $l(n)$: dezimale Ziffernzahl von n; $t(n)$: CPU-Zeit der Faktorisierung von n in sec, M: Mersenne-Zahl; weitere Anmerkungen: alternative Zeit bei der gleichen Aufgabe

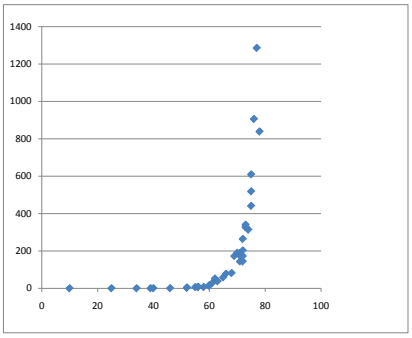

Abbildung 5:
Faktorisierungszeit t in Abhängigkeit von der Zeichenlänge von n. (Rechnung mit Sage)

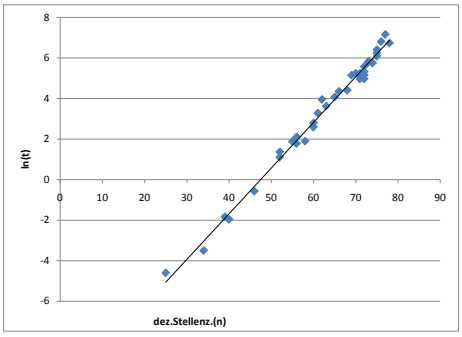

Abbildung 6:
$\ln(t)$ in Abhängigkeit von der Zeichenlänge von n. (Rechnung mit Sage)

l(n)	n (Näherungswert)	T(n)	t(n)	ln (t(n))	Methode CT1
10	8616460799	0,047	0,016	-4,135167	Pollard
25	5,01856E+24	0,25	0,016	-4,135167	Pollard
34	1,24082E+33	0,187	0,016	-4,135167	Pollard
39	8,74298E+38	4,141	0,859	-0,151986	QS
40	2,17203E+39	3,686	0,766	-0,266573	QS
46	1,42725E+45	0,047	0,031	-3,473768	Williams
52	1,60003E+51	52,921	11,702	2,4597598	QS
52	4,35604E+51	39,984	10,563	2,3573573	QS
55	9,01475E+54	107	23,734	3,1669086	QS
56	1,60003E+55	78	20,155	3,0034524	QS
56	9,01475E+55	129	31,312	3,4440014	QS
58	1,35627E+57	76	28,094	3,335556	QS
60	1,00434E+59	261	61	4,1108739	QS
60	2,67620E+59	315	76	4,3307333	QS
61	2,62597E+60	426	95	4,5538769	QS
62	2,67426E+61	475	119	4,7791235	QS
63	8,53380E+62	0,25	0,031	-3,473768	Pollard
65	5,73858E+64	807	206	5,3278762	QS
66	7,88323E+65	1673	473	6,1590954	QS
68	1,93196E+67	1766	496	6,2065759	QS
69	8,94834E+68	0,156	0,032	-3,442019	Pollard
70	8,69565E+69	3498	1045	6,9517722	QS
71	1,11111E+70	3455	801	6,6858609	QS
71	1,53041E+70		1439	7,2717037	QS
71	2,28790E+70		1235	7,1188262	QS
71	2,76070E+70		1020	6,9275579	QS

Tabelle 2:
Ergebnis-Liste (Experimente mit CrypTool 1 auf eigenem PC; s. auch Tabelle 4)
Legende: $l(n)$: dezimale Ziffernzahl von *n*; $T(n)$: CPU-Zeit der Faktorisierung von *n* in Sekunden bei allen angeschalteten Algorithmen; $t(n)$ CPU-Zeit der Faktorisierung von *n* in Sekunden bei Wahl des angegebenen Algorithmus.
Pollard: Pollards (p-1), QS: Quadratisches Sieb, W: Williams (p+1)-Verfahren

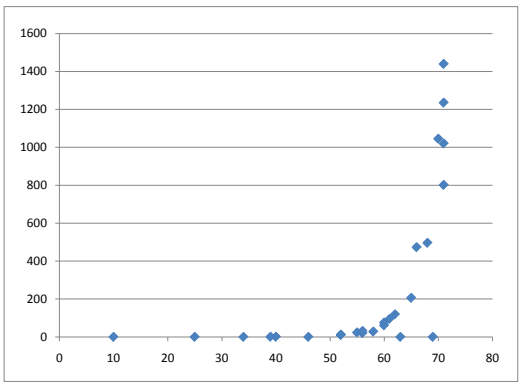

Abbildung 7:
Faktorisierungszeit *t* in Abhängigkeit von der Zeichenlänge von *n*.
(Rechnung mit CrypTool 1 auf eigenem PC)

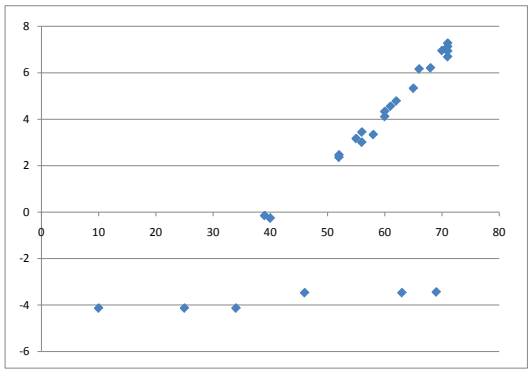

Abbildung 8:
$\ln(t)$ in Abhängigkeit von der Zeichenlänge von *n*. (Rechnung mit CrypTool 1); stärkere Ausreißer nach unten bei den Methoden von Pollard bzw. Williams.

l(n)	*n (Näherungswert)*	t(n) [sec]	ln(t(n))
10	8616460799	0	
25	5,01856E+24	0	
34	1,24082E+33	0	
39	8,74298E+38	0	
40	2,17203E+39	0	
46	1,42725E+45	0	
52	1,60003E+51	1	0
52	4,35604E+51	1	0
55	9,01475E+54	1	0
56	1,60003E+55	1	0
56	9,01475E+55	2	0,693147181
58	1,35627E+57	2	0,693147181
60	1,00434E+59	4	1,386294361
60	2,67620E+59	4	1,386294361
61	2,62597E+60	5	1,609437912
62	2,67426E+61	6	1,791759469
63	8,53380E+62	8	2,079441542
65	5,73858E+64	10	2,302585093
66	7,88323E+65	17	2,833213344
68	1,93196E+67	23	3,135494216
69	8,94834E+68	27	3,295836866
70	8,69565E+69	40	3,688879454
71	1,11111E+70	32	3,465735903
71	1,53041E+70	36	3,583518938
71	2,28790E+70	34	3,526360525
71	2,76070E+70	37	3,610917913
72	1E+71	39	3,663561646
72	1,82405E+71	55	4,007333185
72	2E+71	34	3,526360525
72	4,72154E+71	42	3,737669618
73	1,36986E+72	48	3,871201011
73	7,46308E+72	55	4,007333185
74	4,98736E+73	70	4,248495242
75	2,33070E+74	92	4,521788577
75	3,87467E+74	104	4,644390899
75	5,13083E+74	114	4,736198448
76	3,98613E+75	116	4,753590191
77	8,75598E+76	168	5,123963979
78	4,95090E+77	193	5,262690189
79	5,84788E+78	223	5,407171771
80	5,43974E+79	286	5,655991811
81	4,40514E+80	294	5,683579767
81	8,58080E+80	315	5,752572639
82	6,94879E+81	329	5,796057751
83	3,10024E+82	485	6,184148891
83	7,09936E+82	568	6,342121419
84	3,16742E+83	724	6,584791392
100	1,522605E+99	30405	10,32236235

Tabelle 3:
Ergebnis-Liste (Experimente mit CrypTool 2 auf eigenem PC; s. auch Tabelle 4)
Legende: $l(n)$: dezimale Ziffernzahl von *n*; $t(n)$ Wall-Zeit (End-Zeit minus Start-Zeit) der Faktorisierung.

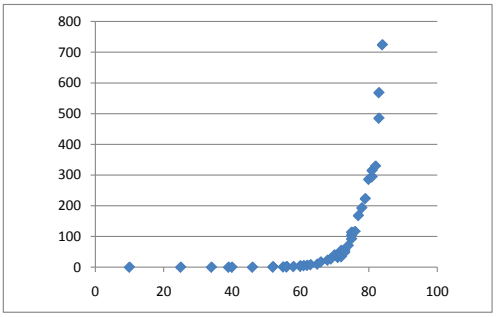

Abbildung 9:
Faktorisierungszeit t in Abhängigkeit von der Zeichenlänge von n (Rechnung mit CrypTool 2 auf eigenem PC).

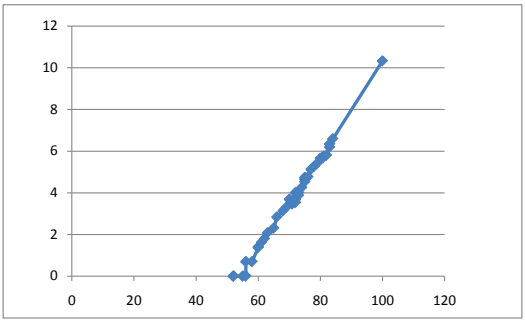

Abbildung 10:
$\ln(t)$ in Abhängigkeit von der Zeichenlänge von n (Rechnung mit CrypTool 2 auf eigenem PC).

Literatur:

[BSI] Bundesamt für Informationstechnik:IT-Grundschutz-Kataloge M 2.164
https://www.bsi.bund.de/cln_183/ContentBSI/grundschutz/kataloge/m/m02/m02164.html

[CrypTool_2010] Das CrypTool-Skript: Kryptographie, Mathematik und mehr. Hintergrundmaterial und Zusatzinformationen zum freien E-Learning - Programm CrypTool (mit Code-Beispielen zur Zahlentheorie, geschrieben in Sage). 10. Auflage. Veröffentlicht mit CrypTool-Version 1.4.30 von Bernhard Esslinger und dem CrypTool Entwickler-Team, 1998-2010, Frankfurt am Main.
http://www.cryptool.de/index.php/de/script-documentationmenu-69.html
http://www.cryptool.de/download/CrypToolScript-de.pdf

[CrypTool 1] CrypTool. Lernprogramm für Kryptographie und Kryptoanalyse
http://www.cryptool.org/index.php/de/download-topmenu-63.html

[CrypTool 2] CrypTool 2.0
http://cryptool2.vs.uni-due.de/

[Golomb_1996] GOLOMB, S.: On factoring Jevons' number. Cryptologia **20/3** (1996) p. 243-246.

[Kleinjung_et_al_2010] KLEINJUNG, TH. u.a.: Factorization of a 768-bit RSA modulus – version 1.4, 2010. http://eprint.iacr.org/2010/006.pdf

[Pomerance_1996] POMERANCE, CARL: A Tale of Two Sieves, Notices of the

AMS, 43 (1996) p. 1473-1485.
(Webversion: http://www.ams.org/notices/199612/pomerance.pdf)

[Ribenboim_1996] RIBENBOIM, P.: The new book of prime numbers. Springer V. ^{3}1996.

[RSA] RSA Laboratories, http://www.rsa.com/rsalabs/node.asp?id=2093

[Sage] SAGE Notebook Vers. 4.5. http://www.sagemath.org/

[Schulz_1996] SCHULZ, R.-H.: Primfaktorzerlegung. Experimente zum Zeitaufwand. LogIn **16** (1996), p. 22–26.

[Schulz_2003] SCHULZ, R.-H.: Codierungsheorie. Eine Einführung. (Kapitel IV). Vieweg V., ^{2}2003.

[Wikipedia_1] WIKIPEDIA: Quadratisches Sieb
http://de.wikipedia.org/wiki/Quadratisches_Sieb

[Wikipedia_2] WIKIPEDIA: Pollard-Rho-Methode
http://de.wikipedia.org/wiki/Pollard-Rho-Methode

[Wikipedia_3] WIKIPEDIA: Pollard-(p-1)-Methode
http://de.wikipedia.org/wiki/Pollard-p-1-Methode

[Wikipedia_4] WIKIPEDIA: RSA-100
http://en.wikipedia.org/wiki/RSA_numbers#RSA-100

[Wikipedia_5] WIKIPEDIA: Zahlkörpersieb
http://de.wikipedia.org/wiki/Zahlkörpersieb

[Witten_Schulz_2006-2010] WITTEN,H.& SCHULZ, R.-H.: RSA & Co. in der Schule. LogIn, Neue Folge 1 (2006) H. 140 p. 45-54, Neue Folge 2 (2006) H. 143, p. 50-58, Neue Folge 3 (2008) H. 152, p. 60-70, Neue Folge 4 (2010) H. 163/164, p. 97-103.

[Ziegler_2010] ZIEGLER, G.M.: Darf ich Zahlen? Geschichten aus der Mathematik. Piper Verlag, München ^{3}2010.

Die Internetadressen wurde zuletzt am 29.November 2011 getestet.

Adresse der Autoren:

Prof. Dr. Ralph-Hardo Schulz
c/o Inst. f. Mathematik der FU Berlin
Arnimallee 3
14195 Berlin
E-Mail: rhschulz@zedat.fu-berlin.de

StudDir a.D. Helmut Witten
Brandenburgische Straße 23
10707 Berlin
E-Mail: helmut@witten-berlin.de

Folgende Tabelle ist über den LogIn-Service zu erhalten:
Tabelle 4: Ausführlichere Aufstellung der behandelten Zerlegungen

Legende:
S: Rechnung mit Sage (Web-Interface);
CT1a: Rechnung mit CrypTool 1 auf eigenem PC
(2 GByte Hauptspeicher, einer Intel Core 2 CPU und 2,4 GHertz Taktfrequenz);
CT1b: Rechnung mit CrypTool 1 auf eigenem PC bei Wahl des (günstigeren) Algorithmus;
CT2: Rechnung mit CrypTool 2 auf eigenem PC.
P: Pollards (p-1)-Verfahren; **QS**: Quadratisches Sieb; **W**: Williams (p+1)-Verfahren

$l(n)$	t(n) [sec]	$n = p * q$
10	0,00 S 0,047 CT1a (P) 0,016 CT1b 0 CT2	8616460799 (Jevons-Zahl) =89681 * 96079
25	0,01 S; 0,02 S 0,250 CT1a (P) 0,016 CT1b 0 CT2	5018557517866741394046901 =1481124532001 * 3388342714901 (Faktor von $2^{500}+1$ bzw. von $10^{40}+13$)
34	0,03 S 0,187 CT1a (P) 0,016 CT1b 0 CT2	1240819002867598361280210027487189 =538119463428139 * 2305843009213693951
39	0,16 S 4,141 CT1a (QS) 0,859 CT1b 0 CT2	87429758973907681855541983050986592355 1 = 21213434569876541053 * 41214334569876541067
40	0,14 S 3,686 CT1a (QS) 0,766 CT1b 0 CT2	217202907621186987017305473775282155 1483 = 46600033003000345873 * 46610033003024346571
46	0,56 S 0,047 CT1a (W) 0,031 CT1b 0 CT2	14272476927059598804393159475009619897194905 61 = 2305843009213693951 * 6189700196426901374495621 11 = $(2^{61}-1)*(2^{89}-1) = M_{61} * M_{89}$
52	3,00 S 52,921 CT1a (QS) 11,702 CT1b 1 CT2	160002664011088900002765815025218400000011952508308 7 =400003330000000000000345691 * 400003330000000000000345757

52	3,91 S 39,984 CT1a (QS) 10,563 CT1b 1 CT2	435604395611088900004563500624818800000011952093529 9 =660003330000000000000345703 *660003330000000000000345733
55	6,59 S 107 CT1a (QS) 23,734 CT1b 1 CT2	90147462353748940773847017687455622139930000131851048513 =300245670000000012345612111 1 * 300245670000000123456003338 3
56	5,88 S 78 CT1a (QS) 20,155 CT1b 1 CT2	160002664011088900000027653430212890000000000119484292209 =400003330000000000000345661 * 400003330000000000000345669
56	8,24 S 129 CT1a (QS) 31,312 CT1b 2 CT2	90147462353748907413429523448366026513988880401089574443= 300245670000000012345612111 1*300245670000000012345600000 13
58	6,64 S; 6,63 S 76 CT1a (QS) 28,094 CT1b 2 CT2	135627354233541693362006500711567604569222738541000763975 1= 257878038725783152726710839*525936039003934785796425018 9809
60	13,24 S 13,21 S; 13,42 S 261 CT1a (QS) 61 CT1b 4 CT2	100433627766186892221372630609062766858404681029709092356097 =618970019642690137449562111 * 162259276829213363391578010288127 $= (2^{89} - 1) * (2^{107} - 1) = M_{89} * M_{107}$
60	16,33 S 16,38 S; 16,28 S 315 CT1a (QS) 303; 309 CT1a (QS) 76 CT1b 4 CT2	267619860332297401900866279382024711037566640405902751004511 =618970019642690137449562111 * 432363203127002885506543172618401
61	26,56 S; 35,91 S 426 CT1a (QS) 95 CT1b 5 CT2	26259735241873125830039521970705725700906090920992370472550 17 =19721061166646717498359681* 133155792276964047936085219707057257 (Faktor von $10^{38} + 7$)
62	52,09 S 475 CT1a (QS) 119 CT1b 6 CT2	267426488949013093414928041917346305986801160286423952752520 69 =369646296854333406037717721877 7* 723465894896812414632199608919 7

63	37.42 S; 37,37 S 0,250 CT1a (P) 0,031 CT1b 8 CT2	85338001347199411325839928623044853896918013840619601452909774 3 =525936039003934785796425018980 9 * 162259276829213363391578010288127 (d.h. $2^{107}-1$, also M_{107})
65	58,39 S 807 CT1a (QS) 206 CT1b 10 CT2	57385826280610186778877649465961234589685554490827652749633422527 =19721061166646717498359681 (Faktor von $10^{40}+1$) *29098751733331711114799692271346095319 67
66	77,63 S 1673 CT1a (QS) 473 CT1b 17 CT2	78832273426322211822826376368648683034583218555514031443850974593 7 =31006309784041052354154070353316 9 * 254245906640903081460388487745347 3 (Faktoren von $10^{38}+35$ bzw. $10^{38}+40$)
68	81.48 S 1766 CT1a (QS) 496 CT1b 23 CT2	19319597769065583697597909403141892104992276797595838342261048259737 =254245906640903081460388487745347 3 * 75987841945288753799392097264437 69 (Faktor von $10^{38}+40$)
69	172,6 S 171,68; 173,63 S 0,156 CT1a (P) 0,032 CT1b 27 CT2	89483380100640969930403928995838432203196801831859555699450597393614 3 =525936039003934785796425018980 9 * 17014118346046923173168730371588410572 7 (d.h. $2^{127}-1$, also M_{127})
70	189,63 S 3498 CT1a (QS) 1045 CT1b 40 CT2	86956521739130434782608695652173919304347826086956521739130434782608 77 =434782608695652173913043478260869 59 * 2000000000000000000000000000000003
71	184,55 S 3455 CT1a (QS) 801 CT1b 32 CT2	11111111111111111111111111111111111111111111111111111111111111111111111111111 = 241573142393627673576957439049 * 459948113478868463102217288952230343018 39
71	188,63 S 1439 CT1b 36 CT2	15304082226587361771824035950068730477497807548910700719111280403124303 = 525936039003934785796425018980 9 * 290987517333317111147996922713460953196 7 (Faktor von $10^{70}+5$)
71	175,18 S 1235 CT1b 34 CT2	22879002109443994490736292045886126632759985540470666831395481854663427 = 133155792276964047936085219707057257 * 171821305841924398625429553264604811 (Faktor von $10^{38}+2$)
71	143,66 S 145,31 S 3455 CT1a (QS) 1020 CT1b 37 CT 2	27606985387162255149739023449107931668458716142620601169954803000803329 =162259276829213363391578010288127 * 17014118346046923173168730371588410572 7 $=(2^{107}-1)*(2^{127}-1)=M_{107}*M_{127}$
72	263,93 S 39 CT2	10000000000000000000000000000000000037000000000000000000000000000000/ 00033 =2000000000000000000000000000000003 * 5000000000000000000000000000000011

72	202,29 S 55 CT2	18240519489995075059737701329733870826842748481476752457910001276836/ 3643 =133155792276964047936085219707057257 * 13698630136986301369863013698630136 99 (Faktor von $10^{38}+27$)
72	143,42 S 34 CT2	20000000000000000000000000000000001640000000000000000000000000000000/ 0201 =200000000000000000000000000000003 * 10000000000000000000000000000000067
72	171,47 S 181,05 S 171,91 S 173,73 S 42 CT2	472154241288322308621049626625163020582457972462095711822828862 8/ 7055809 =162259276829213363391578010288127 (d.h. $2^{107}-1$, also M_{107}) * 290987517333317111147996922713460953196 7
73	327,01 S 48 CT2	13698630136986301369863013698630137907808219178082191780821917808 21/ 917833 =100000000000000000000000000000000067 * 136986301369863013698630136986301369 9
73	339,89 S 340,75 S 339,75 S 55 CT2	7463084827204216028562220613553566489786789143432464634026782164675/ 965553 =162259276829213363391578010288127 (d.h. $2^{107}-1$, also M_{107}) * 459948113478868463102217288952230343018 39
74	313,40 S 70 CT2	4987358169131969861275321306209230112106771529895163323045597391567 8/ 019001 =133155792276964047936085219707057257 * 3745505985018109365817766300963131813 93 (Faktoren von $10^{38}+7$ bzw. $M_{257}=2^{257}-1$, vgl. [Ribenboim_1996] p. 168)
75	440,86 S 92 CT2	2330701143294099064817634297477864462642987938826393048870624274017 6/ 5354173 =136986301369863013698630136986301369 9 (Faktor von $10^{38}+27$) * 17014118346046923173168730371588410572 7 (d.h. $2^{127}-1$, also M_{127})
75	609,14 S 104 CT2	3874667341322464862157082859034100575460307938925861488420236812116 4/ 0834519 =133155792276964047936085219707057257 * 29098751733331711114799692271346095319 67 (Faktor von $10^{70}+5$)
75	518,56 S 114 CT2	5130830116463163514818857946524838102659296734184780176398351919534 3/ 0902707 =136986301369863013698630136986301369 9 * 37455059850181093658177663009631318 93 (Faktoren von $10^{38}+2$ bzw. M_{257})
76	905,30 S 912,72 S 116 CT2	3986130374429001522575300311143300729798173009315008493287111905926 49/ 9415933 =136986301369863013698630136986301369 9 * 29098751733331711114799692271346095319 67 (Faktoren von $10^{38}+27$ bzw. $10^{70}+5$)

77	1285,47 S 168 CT2	875598231275019823323891500458316806110256088887157927543056097 91/ 773510137213 = 212345678900000123456789000012345679119 * 4123456789000001234567890000123456790 27
78	838.49 S 193 CT2	495089605713143771548462573328813745169387022923139667420765802 44/ 4525814275009 =17014118346046923173168730371588410572 7 * 2909875173333171111479969227134609531967 (d.h. $2^{127} - 1$, also M_{127}.)
79	>1800 S Abbruch 223 CT2	584788329623583940070231333191948056784763452642325227565404920 00/ 00243098047477 434567801230000000012340000000123400024 1* 1345678000000000000123400000000000000197
80	286 CT2	543973742830070965981629032502471123696871465755630038272186924 66/ 797770535826811 =571720419698479484948828529839261986710 1* 95146810239340081340814798948974525187 11
81	294 CT2	440513807262414775846225762676093175761492328560198902854490047 10/ 7277398468733833 =571720419698479484948828529839261986710 1 * 77050563891829862040383849143369089669533
81	315 CT2	858080132311600557111244048957535267132914923775127382142274082 07/ 7493303918481489 =901848553990528420678897441195966498781 99 * 95146810239340081340814798948974525187 11
82	329 CT2	694879396300015829675025167390667167305548752777623530323515959 8/ 066255373611211067 = 901848553990528420678897441195966498781 99 * 77050563891829862040383849143369089669533
83	485 CT2	310024388933209374787804948211670326510072728325795765175416035 5/ 8569409309564222957 = 901848553990528420678897441195966498781 99 * 34376546656465047309066025814940496125964 3
83	568 CT2	709936327171757693036118882578583928299480011459305867614006288 4/ 9295821871112154981 = 921390177193805770004500281483529205745257 * 77050563891829862040383849143369089669533
84	724 CT2	316742124151114612292655573385203295645088864594733709020775251 60/ 8265071587292763251 = 921390177193805770004500281483529205745257 * 34376546656465047309066025814940496125964 3
100	30405 CT2	152260502792253336053561837813263742971806811496138068865790849/ 458012296325895289765400035069200 6139 **RSA-100** =379752279369436739228088727554456278545655366381 99 * 40094690950920881030683735292761468389214899724061