Factoring Beyond FactorDB

Husnain

The problem of distinguishing prime numbers from composite numbers and of resolving the latter into their prime factors is known to be one of the most important and useful in arithmetic. It has engaged the industry and wisdom of ancient and modern geometers to such an extent that it would be superfluous to discuss the problem at length. ... Further, the dignity of the science itself seems to require that every possible means be explored for the solution of a problem so elegant and so celebrated.

Carl Friedrich Gauss

Section 0

Preliminaries

Primes

- A **prime** is any number p such that its only divisors are 1 and p.
- Example of primes: $2, 3, 5, 7, 11, 13, \cdots$
- Any integer can be **uniquely** factored into prime numbers

Modular Arithmetic

- We say that $x \equiv y \pmod{n}$ if n cleanly divides $x - y$
- This has similar properties to $=$, i.e. we can add, subtract and multiply on both sides
- Division is different: $x^{-1} \mod n$ exists only if x and n share no divisors (this is important later on)

Section 1

Basic Algorithms

Trial Division

- As a first naive approach, we can start trying to divide numbers into N and see if they divide evenly
- Any factor d of N has a corresponding factor $N/d \implies$ we only need to check $d \leq \sqrt{N}$
- In fact, we only need to check primes $p < \sqrt{N}$

Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33
34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63	64	65	66
67	68	69	70	71	72	73	74	75	76	77
78	79	80	81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120	121

Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33
34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63	64	65	66
67	68	69	70	71	72	73	74	75	76	77
78	79	80	81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120	121

Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33
34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63	64	65	66
67	68	69	70	71	72	73	74	75	76	77
78	79	80	81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120	121

Sieve of Eratosthenes

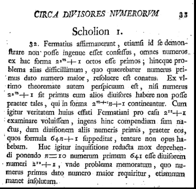
1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33
34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63	64	65	66
67	68	69	70	71	72	73	74	75	76	77
78	79	80	81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120	121

Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33
34	35	36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63	64	65	66
67	68	69	70	71	72	73	74	75	76	77
78	79	80	81	82	83	84	85	86	87	88
89	90	91	92	93	94	95	96	97	98	99
100	101	102	103	104	105	106	107	108	109	110
111	112	113	114	115	116	117	118	119	120	121

Brief Aside: Factorization Before Computers

Euler and Fermat



CIRCA DIVISORES NVMERORVM 33

Scholion 1.

32. Fermatius affirmauerant, etiamſi id ſe demonſtrare non poſſe ingenue eſſet confeſſus, omnes numeros ex hac forma $2^{2^m}+1$ ortos eſſe primos; hincque problema alias difficillimum, quo quaerebatur numerus primus dato numero maior, reſoluere eſt conatus. Ex vltimo theoremate autem perſpicuum eſt, niſi numerus $2^{2^m}+1$ ſit primus eum alios diuiſores habere non poſſe praeter tales, qui in forma $2^{m+1}n+1$ contineantur. Cum igitur veritatem huius effati Fermatiani pro caſu $2^{32}+1$ examinare voluiſſem, ingens hinc compendium ſum nactus, dum diuiſionem aliis numeris primis, praeter eos, quos formula $64n+1$ ſuppeditat, tentare non opus habebam. Huc igitur inquiſitione reducta mox deprehendi ponendo $n=10$ numerum primum 641 eſſe diuiſorem numeri $2^{32}+1$, vnde problema memoratum, quo numerus primus dato numero maior requiritur, etiamnum manet inſolutum.

Fermat had held, even though he had confessed that he frankly was unable to prove it, that all numbers of the form $2^{2^m}+1$ *are prime [...]*
And so since I wanted to examine the truth of this renowned claim of Fermat for the case of $2^{32}+1$*, I managed a huge shortening of this, by not having to try division by any prime numbers except those expressible in the form* $64n+1$*.*
And so with the problem reduced to this, I soon discovered that by setting $n=10$*, the prime number 641 is a divisor of the number* $2^{32}+1$*.*

Lehmer Sieves

She had seen a light and had stopped the whirling wheels. Again the tell-tale ray of light was located and again the number 5,283,065,753,709,209 was given as a square plus seven times another square. The machine had done its duty. These two results were all that was necessary. A few minutes computation still remained, and thus it was, while coffee was being served on one of the working tables in the laboratory the big number was broken up into the factors 59,957 and 88,114,244,437. These are the two hidden numbers which when multiplied together will give the sixteen digit number under examination. It may seem to the man in the street an odd thing to get excited about, but on this occasion

All Rome sent forth a rapturous cry,
And even the ranks of Tuscany
Could scarce forbear to cheer.

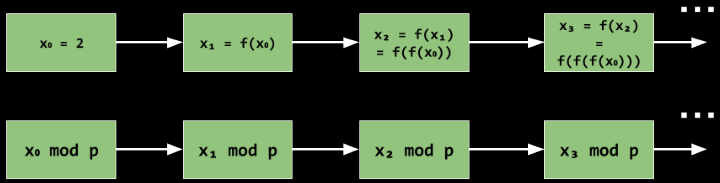
Section 2

Extracting Small Factors

Pollard Rho

Let $N = pk$, where p is small factor. Let $f(x) = p(x) \bmod N$ be any suitable polynomial, usually $p(x) = x^2 + 1$.

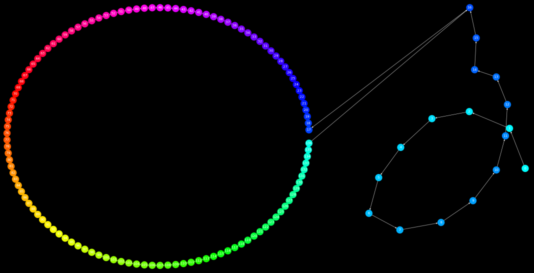
Pollard Rho



$x_0 = 2$
$x_1 = f(x_0)$
$x_2 = f(x_1) = f(f(x_0))$
$x_3 = f(x_2) = f(f(f(x_0)))$
...
$x_0 \bmod p$
$x_1 \bmod p$
$x_2 \bmod p$
$x_3 \bmod p$
...

Pollard Rho: Example

$$N = 142741636831523 = 52051 \cdot 2742341873$$



Pollard Rho: Algorithm

```
f = lambda x: (x*x + 1) % N
x = 2
y = 2
d = 1
while d == 1 or d == N:
    x = f(x)
    y = f(f(y))
    d = gcd(abs(x-y), n)
```

Pollard Rho: Application

MATHEMATICS OF COMPUTATION
VOLUME 36, NUMBER 154
APRIL 1981

Factorization of the Eighth Fermat Number

By Richard P. Brent and John M. Pollard

Abstract. We describe a Monte Carlo factorization algorithm which was used to factorize the Fermat number $F_8 = 2^{256} + 1$. Previously F_8 was known to be composite, but its factors were unknown.

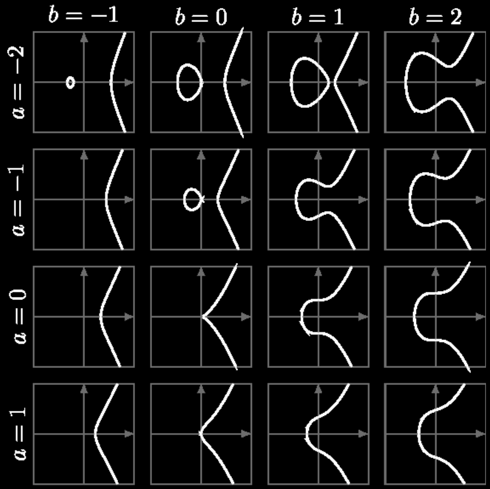
1. Introduction. Brent [1] recently proposed an improvement to Pollard's Monte Carlo factorization algorithm [4]. Both algorithms can usually find a prime factor p of a large integer in $O(p^{1/2})$ operations.

In this paper we describe a modification of Brent's algorithm which is useful when the factors are known to lie in a certain congruence class. To test its effectiveness, the algorithm was applied to the Fermat numbers $F_k = 2^{2^k} + 1$, $5 \leqslant k \leqslant 13$. The least factors of all but F_8 were known [2], and F_8 was known to be composite. The algorithm rediscovered the known factors and also found the previously unknown factor 1,238,926,361,552,897 of F_8.*

$$F_8 = 2^{2^8} + 1$$

$$= \underbrace{1238926361552897}_{\texttt{16 digits}} \cdot \underbrace{934 \cdots 321}_{\texttt{62 digits}}$$

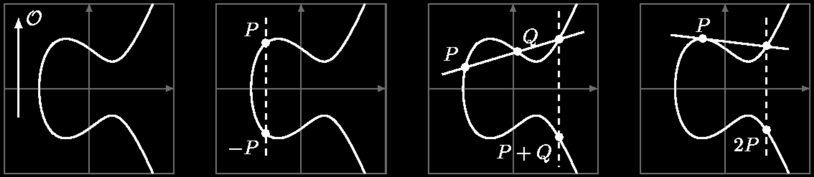
ECM: Intro



$$y^2 = x^3 + a \cdot x + b$$

Image Credit: https://www.iacr.org/authors/tikz/

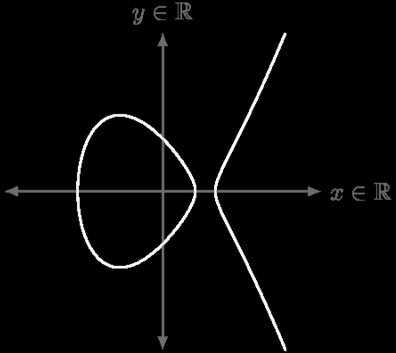
ECM: Point Addition



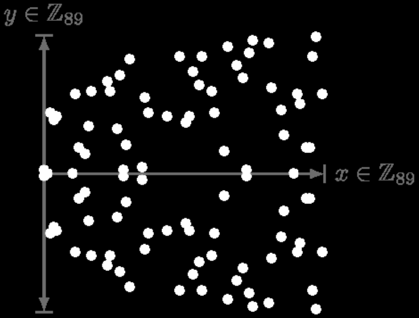
Neutral element $\mathcal{O}$ | Inverse element $-P$ | Addition $P+Q$ "Chord rule" | Doubling $P+P$ "Tangent rule"

Image Credit: https://www.iacr.org/authors/tikz/

ECM: Changing Fields



$y^2 = x^3 - 2x + 1$ over $\mathbb{R}$



$y^2 = x^3 - 2x + 1$ over $\mathbb{Z}_{89}$

Image Credit: https://www.iacr.org/authors/tikz/

ECM: A Problem and A Solution

- To calculate the sum of two points, we need to calculate multiplicative inverses.
- This is fine over $\mathbb{R}$, but this may not work $\bmod N$.
- However, this failure allows us to find a factor of N, if $x^{-1} \pmod{N}$ does not exist, then x and N share a factor

ECM: Application

FACTORIZATION OF THE TENTH AND ELEVENTH FERMAT NUMBERS

RICHARD P. BRENT

ABSTRACT. We describe the complete factorization of the tenth and eleventh Fermat numbers. The tenth Fermat number is a product of four prime factors with 8, 10, 40 and 252 decimal digits. The eleventh Fermat number is a product of five prime factors with 6, 6, 21, 22 and 564 decimal digits. We also note a new 27-decimal digit factor of the thirteenth Fermat number. This number has four known prime factors and a 2391-decimal digit composite factor. All the new factors reported here were found by the elliptic curve method (ECM). The 40-digit factor of the tenth Fermat number was found after about 140 Mflop-years of computation. We discuss aspects of the practical implementation of ECM, including the use of special-purpose hardware, and note several other large factors found recently by ECM.

$$F_{10} = 2^{2^{10}} + 1 = \underbrace{45592577}_{\texttt{8 digits}} \cdot \underbrace{6487031809}_{\texttt{8 digits}} \cdot \underbrace{465\cdots897}_{\texttt{40 digits}} \cdot \underbrace{130\cdots577}_{\texttt{257 digits}}$$

$$F_{11} = 2^{2^{11}} + 1 = \underbrace{319489}_{\texttt{6 digits}} \cdot \underbrace{974849}_{\texttt{6 digits}} \cdot \underbrace{167\cdots137}_{\texttt{21 digits}} \cdot \underbrace{356\cdots513}_{\texttt{22 digits}} \cdot \underbrace{173\cdots177}_{\texttt{564 digits}}$$

Section 3

Sieving out Larger Factors

Quick! Factor 4819.

Hint

$$4819 = 4900 - 81$$

Hint

$$4819 = 4900 - 81 = 70^2 - 9^2$$

Hint

$$4819 = 4900 - 81 = 70^2 - 9^2 = (70 + 9)(70 - 9) = 79 \cdot 61$$

Fermat's Method

This is an example of *Fermat's method* of factorization - however, for general N, it is quite hard to find integers x, y such that $x^2 - y^2 = N$ or alternatively $x^2 \equiv y^2 \pmod{N}$

Let's try to factor $N = 1791$ with this method.

$$\begin{aligned} 41^2 &\equiv 32 \pmod{N} \\ 42^2 &\equiv 115 \pmod{N} \\ 43^2 &\equiv 200 \pmod{N} \end{aligned}$$

This seems hopeless - none of the values on the right hand side are perfect squares...

Solution: Combine Congruences

Notice that $32 \cdot 200 = 80^2$. We therefore have

$$41^2 \cdot 43^2 = (41 \cdot 43)^2 \equiv 114^2 \equiv 32 \cdot 200 = 80^2 \pmod{N}$$

We therefore have that $(114 - 80)(114 + 80) = kN$ for some integer k - therefore, $\gcd(114 - 80, N) = 17$ is a nontrivial factor of N.

How to pick congruences?

- Define an integer to be *B-smooth* if none of its prime factors exceed B.
- Through *sieving* (similar to the Sieve of Eratosthenes), we can get pairs $(x_i, y_i = x_i^2 - N)$ such that $x_i^2 - N$ is B-smooth

How to pick congruences?

Let's say we have pairs $(x_1, y_1) \cdots (x_n, y_n)$. To get a successful factorization, we need to pick a subset of y_i such that $\prod_i y_i$ is a perfect square.

How to pick congruences?

- Since y_i are B-smooth, we can write $y_i = \prod_{1 \leq j \leq k} p_j^{e_j}$ where $p_1, p_2, \cdots, p_k$ are the primes below B.
- Define the *exponent vector* of $y_i = \prod_{1 \leq j \leq k} p_j^{e_j}$ to be the vector $[e_1 \cdots e_k]$

Why the exponent vector?

Let's say that we have that we have $y_a = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$ and $y_b = p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}$
We then have that

$$\begin{aligned} y_a \cdot y_b &= (p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}) \cdot (p_1^{f_1} p_2^{f_2} \cdots p_k^{f_k}) \\ &= p_1^{e_1+f_1} p_2^{e_2+f_2} \ldots p_k^{e_k+f_k} \end{aligned}$$

Reading off exponent vectors, we therefore have that *multiplying y values is equivalent to adding their exponent vectors*

And what about the perfect square?

Note that for any perfect square, we have that
$k^2 = (p_1^{e_1} p_2^{e_2} \cdots p_N^{e_N})^2 = p_1^{2e_1} p_2^{2e_2} \cdots p_N^{2e_N}$
Reading off the exponent vector, we have that *a perfect square will have an exponent vector of all even numbers*

Nobody Expects Linear Algebra

Using exponent vectors, we can reframe the problem as saying that we want a sum of some exponent vectors such that all of the entries are even - this is equivalent to the zero vector mod 2.
This can be recast as a problem of linear algebra over $\mathbb{F}_2$. Additionally, the theorems from linear algebra tell us that we need $k+1$ relationships in order to find a subset that sums to the zero vector.

Quadratic Sieve: A Recap

1. Choose a smoothness bound B (advanced math says that a good bound is $(e^{\sqrt{\ln n \ln \ln n}})^{\frac{1}{2}}$ but this can be tuned to taste), and let $\pi(B)$ be the number of primes less than B
2. Starting with $x_i = \lceil \sqrt{N} \rceil$, use sieving to find $\pi(B) + 1$ values of $(x_i, y_i = x_i^2 - N)$ such that y_i is B-smooth; generate the corresponding exponent vectors for each y_i mod 2
3. Use linear algebra to find a subset of these exponent vectors that sum to the zero vector in $\mathbb{F}_2$.
4. Use the method described above to get a factorization of N as in Fermat's method.

Quadratic Sieve: Application

9686	9613	7546	2206
1477	1409	2225	4355
8829	0575	9991	1245
7431	9874	6951	2093
0816	2982	2514	5708
3569	3147	6622	8839
8962	8013	3919	9055
1829	9451	5781	5154

A ciphertext challenge worth $100

$$N = \underbrace{114\cdots541}_{\text{129 digits}}$$

Contrast this with the difficulty of finding the two prime factors of a 125-or 126-digit number obtained by multiplying two 63-digit primes. If the best algorithm known and the fastest of today's computers were used, Rivest estimates that the running time required would be about 40 quadrillion years!

Quadratic Sieve: Application

THE MAGIC WORDS ARE SQUEAMISH OSSIFRAGE

Extended Abstract

Derek Atkins[1], Michael Graff[2], Arjen K. Lenstra[3], Paul C. Leyland[4]

[1] 12 Rindge Avenue, Cambridge, MA 02140, U. S. A.
E-mail: warlord@mit.edu
[2] Iowa State University, 215 Durham Center, Ames, IA 50010-2120, U. S. A.
E-mail: explorer@iastate.edu
[3] MRE-2Q334, Bellcore, 445 South Street, Morristown, NJ 07960, U. S. A.
E-mail: lenstra@bellcore.com
[4] Oxford University Computing Services, 13 Banbury Road, Oxford, OX2 6NN, U. K.
E-mail: pcl@ox.ac.uk

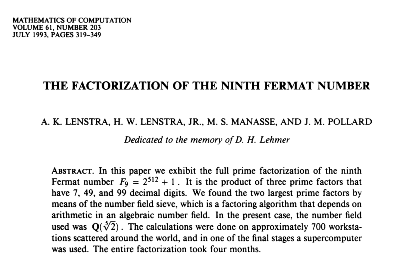
Abstract. We describe the computation which resulted in the title of this paper. Furthermore, we give an analysis of the data collected during this computation. From these data, we derive the important observation that in the final stages, the progress of the double large prime variation of the quadratic sieve integer factoring algorithm can more effectively be approximated by a quartic function of the time spent, than by the more familiar quadratic function. We also present, as an update to [15], some of our experiences with the management of a large computation distributed over the Internet. Based on this experience, we give some realistic estimates of the current readily available computational power of the Internet. We conclude that commonly-used 512-bit RSA moduli are vulnerable to any organization prepared to spend a few million dollars and to wait a few months.

$$N = \underbrace{349\cdots577}_{64\ digits}\cdot\underbrace{327\cdots533}_{65\ digits}$$

An Improvement

Note that the candidates that we pick in the sieving step are of the order $O(\sqrt{N})$.
If we could reduce the size of these candidates, then they would have a higher probability of being smooth — therefore decreasing the runtime.

The Number Field Sieve


MATHEMATICS OF COMPUTATION
VOLUME 61, NUMBER 203
JULY 1993, PAGES 319–349

THE FACTORIZATION OF THE NINTH FERMAT NUMBER

A. K. LENSTRA, H. W. LENSTRA, JR., M. S. MANASSE, AND J. M. POLLARD

Dedicated to the memory of D. H. Lehmer

ABSTRACT. In this paper we exhibit the full prime factorization of the ninth Fermat number $F_9 = 2^{512} + 1$. It is the product of three prime factors that have 7, 49, and 99 decimal digits. We found the two largest prime factors by means of the number field sieve, which is a factoring algorithm that depends on arithmetic in an algebraic number field. In the present case, the number field used was $\mathbf{Q}(\sqrt[5]{2})$. The calculations were done on approximately 700 workstations scattered around the world, and in one of the final stages a supercomputer was used. The entire factorization took four months.

Enter the *number field sieve* - the currently fastest known algorithm known to factor numbers. It finds perfect squares in so called *number fields* - in the case of the factorization of F_9, the number field was $\mathbb{Q}[\sqrt[5]{2}]$

Number Fields

Any element of $\mathbb{Q}[\sqrt[5]{2}]$ can be written as $a + b \cdot 2^{1/5} + c \cdot 2^{2/5} + d \cdot 2^{3/5} + e \cdot 2^{4/5}$ for rational a, b, c, d

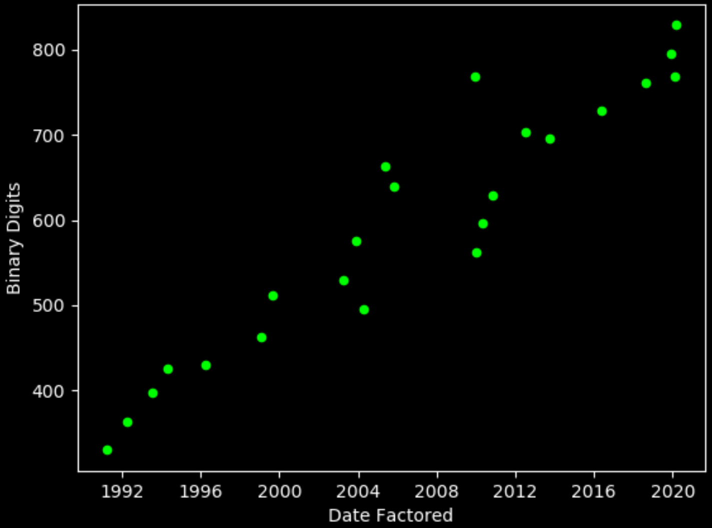
Multiplication can be defined naturally - more internals of how this works is beyond the scope of this talk.

</math>

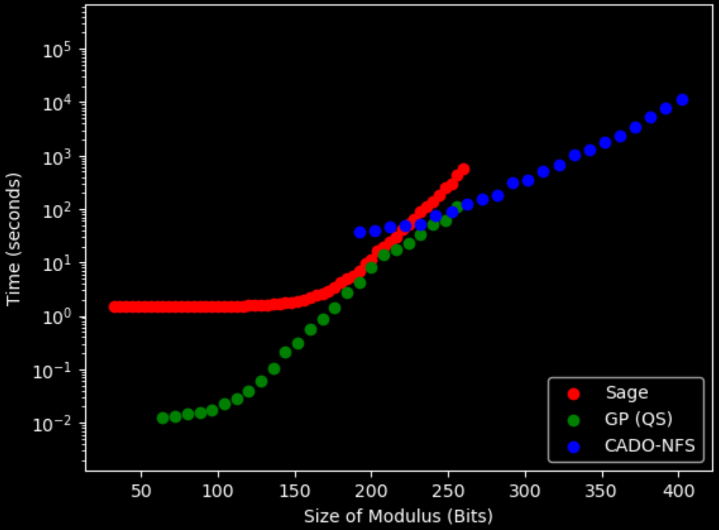
Section 4

Applications

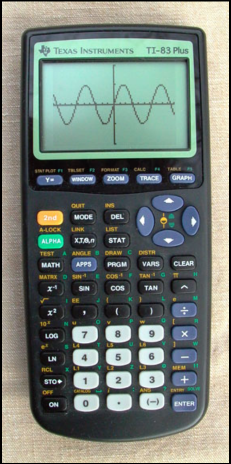
Academic Factoring Efforts



Advances in Hardware

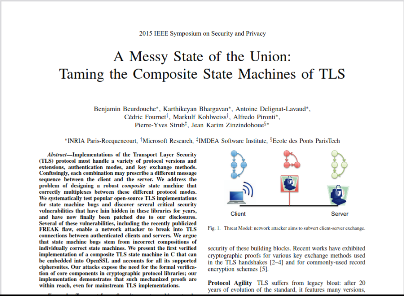


TI Signing Controversy



- In order to prevent modification, the operating system on the TI-84 was verified with a 512-bit RSA key
- This was factored in 2009 over 73 days
- Led to some backlash by TI

FREAK Attack



2015 IEEE Symposium on Security and Privacy

A Messy State of the Union: Taming the Composite State Machines of TLS

Benjamin Beurdouche*, Karthikeyan Bhargavan*, Antoine Delignat-Lavaud*, Cédric Fournet†, Markulf Kohlweiss†, Alfredo Pironti*, Pierre-Yves Strub‡, Jean Karim Zinzindohoue§*

*INRIA Paris-Rocquencourt, †Microsoft Research, ‡IMDEA Software Institute, §Ecole des Ponts ParisTech

Abstract—**Implementations of the Transport Layer Security (TLS) protocol must handle a variety of protocol versions and extensions, authentication modes, and key exchange methods. Confusingly, each combination may prescribe a different message sequence between the client and the server. We address the problem of designing a robust *composite* state machine that correctly multiplexes between these different protocol modes. We systematically test popular open-source TLS implementations for state machine bugs and discover several critical security vulnerabilities that have lain hidden in these libraries for years, and have now finally been patched due to our disclosures. Several of these vulnerabilities, including the recently publicized FREAK flaw, enable a network attacker to break into TLS connections between authenticated clients and servers. We argue that state machine bugs stem from incorrect compositions of individually correct state machines. We present the first verified implementation of a composite TLS state machine in C that can be embedded into OpenSSL and accounts for all its supported ciphersuites. Our attacks expose the need for the formal verification of core components in cryptographic protocol libraries; our implementation demonstrates that such mechanized proofs are within reach, even for mainstream TLS implementations.**

Client

Server

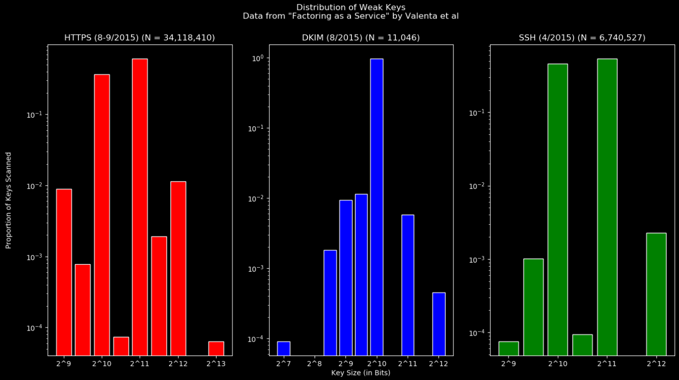
Fig. 1. Threat Model: network attacker aims to subvert client-server exchange.

security of these building blocks. Recent works have exhibited cryptographic proofs for various key exchange methods used in the TLS handshakes [2–4] and for commonly-used record encryption schemes [5].

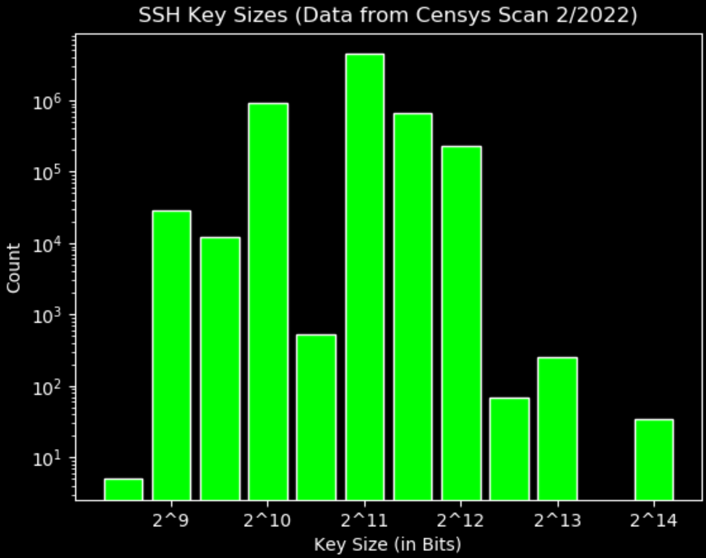
Protocol Agility TLS suffers from legacy bloat: after 20 years of evolution of the standard, it features many versions,

- Published in 2015
- Man in the middle attack found in TLS protocol that forces the use of RSA moduli of 512 bits
- Affected many mobile and desktop browsers

Weak Keys were Prevalent



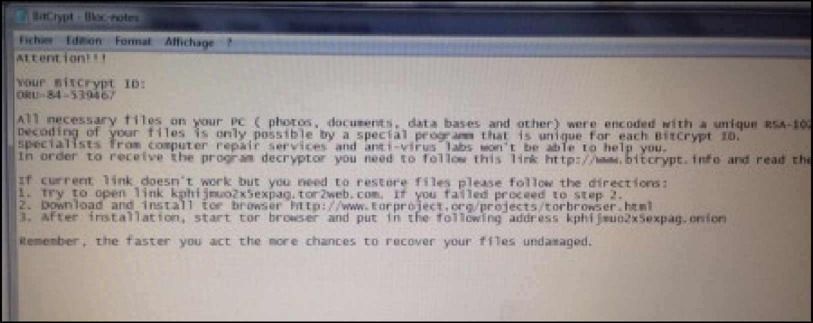
...but Not So Much Anymore



Things Can Still Go Wrong

1024 bit key $\longrightarrow$ 512 bit key due to encoding issues

Things Can Still Go Wrong



- claims to use RSA-1024
- $\underbrace{\text{128 byte modulus}}_{\text{not factorable}} \neq \underbrace{\text{128 digit modulus}}_{\text{factorable}}$

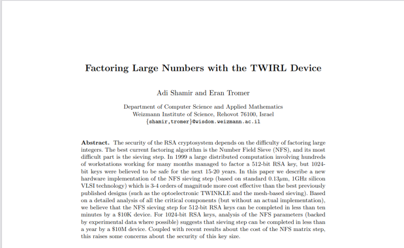
Original Blog Post

Section 5

The Future?

512 bit keys are clearly broken, what about 1024?

TWIRL



Factoring Large Numbers with the TWIRL Device

Adi Shamir and Eran Tromer

Department of Computer Science and Applied Mathematics
Weizmann Institute of Science, Rehovot 76100, Israel
{shamir,tromer}@wisdom.weizmann.ac.il

Abstract. The security of the RSA cryptosystem depends on the difficulty of factoring large integers. The best current factoring algorithm is the Number Field Sieve (NFS), and its most difficult part is the sieving step. In 1999 a large distributed computation involving hundreds of workstations working for many months managed to factor a 512-bit RSA key, but 1024-bit keys were believed to be safe for the next 15-20 years. In this paper we describe a new hardware implementation of the NFS sieving step (based on standard 0.13μm, 1GHz silicon VLSI technology) which is 3-4 orders of magnitude more cost effective than the best previously published designs (such as the optoelectronic TWINKLE and the mesh-based sieving). Based on a detailed analysis of all the critical components (but without an actual implementation), we believe that the NFS sieving step for 512-bit RSA keys can be completed in less than ten minutes by a $10K device. For 1024-bit RSA keys, analysis of the NFS parameters (backed by experimental data where possible) suggests that sieving step can be completed in less than a year by a $10M device. Coupled with recent results about the cost of the NFS matrix step, this raises some concerns about the security of this key size.

- Theoretical device created by Shamir (the S in RSA) and Tromer in *2003*.
- Can do sieving step of NFS for 1024-bit modulus for $10 million over 1 year.
- Linear algebra and remaining steps take less effort.

Enter Quantum

There exists an algorithm that can run on quantum computers that can factor numbers in polynomial time

- **Shor's algorithm**

Enter Quantum

Briefly, the algorithm entails using *quantum magic* to find the period of the sequence $a, a^2, a^3, \cdots, a^k (\pmod N)$ to then find a non-trivial square root mod N, similar to the sieving algorithms we saw earlier.

Enter Quantum

However, due to the the size of quantum computers needed to implement Shor's algorithm, at least now, it remains impractical to use for integers larger than a few thousand

See here for more details

Takeaways

- Factoring is hard - we use this to hide messages
- RSA can be vulnerable - not only too short keys, but other methods not described here! (Bad primes = pwnage)
- Use ECC (elliptic curves) if possible - more efficient, still vulnerable
- Quantum will break crypto soon^{TM}