NUMBER 12 — MAY 3, 1999

RSA Laboratories' Bulletin

News and advice on data security and cryptography

An Analysis of Shamir's Factoring Device

Robert D. Silverman
RSA Laboratories, Bedford, Massachusetts

At a Eurocrypt rump session, Professor Adi Shamir of the Weizmann Institute announced the design for an unusual piece of hardware. This hardware, called "TWINKLE" (which stands for The Weizmann INstitute Key Locating Engine), is an electro-optical sieving device which will execute sieve-based factoring algorithms approximately two to three orders of magnitude as fast as a conventional fast PC. The announcement only presented a rough design, and there a number of practical difficulties involved with fabricating the device. It runs at a very high clock rate (10 GHz), must trigger LEDs at precise intervals of time, and uses wafer-scale technology. However, it is my opinion that the device is practical and could be built after some engineering effort is applied to it. Shamir estimates that the device can be fabricated (after the design process is complete) for about $5,000.

What is a sieve-based factoring algorithm?

A sieve based algorithm attempts to construct a solution to the congruence $A^2 = B^2 \bmod N$, whence GCD(A-B,N) is a factor of N. It does so by *attempting* to factor many congruences of the form $C = D \bmod N$, where there is some special relation between C and D. Each of C and D is attempted to be factored with a fixed set of prime numbers called a *factor base*. This yields congruences of the form:

$$\Pi\,(P_i^a) = \Pi\,(p_i^b) \bmod N$$

where P_i are the primes in the factor base associated with C and p_i are the primes in the factor base associated with D. These factored congruences are found by sieving all the primes in the factor base over a long sieve interval. One collects many congruences of this form (as many as there are primes in the two factor bases) then finds a set of these congruences which when multiplied together yields squares on both sides. This set is found by solving a set of linear equations mod 2. Thus, there are two parts to a sieve-based algorithm: (1) collecting the equations by sieving, and (2) solving them. The number of equations equals the sum of the sizes of the factor bases. A variation allows somewhat larger primes in the factorizations than those in the factor bases. This has the effect of greatly speeding the sieving process, but makes the number of equations one needs to solve much larger. One could choose not to use the larger primes, but then one needs a much larger factor base, once again resulting in a larger matrix.

It should be noted that sieve based algorithms can also be used to solve discrete logarithm problems as well as factor. This applies to discrete logs over finite fields, but not to elliptic curve discrete logs. Solving discrete logs takes about the same amount of time as factoring does for same-sized keys. However, the required space and time for the matrix is much larger for discrete logs. One must solve the system of equations modulo the order of the field, rather than mod 2.

What has been achieved so far with conventional hardware?

Recently, a group led by Peter Montgomery announced the factorization of RSA-140, a 465-bit number. The effort took about 200 computers, run-

Robert Silverman is a Senior Research Scientist at RSA Laboratories, and can be reached at rsilverman@rsa.com.

RSA Laboratories
A Division of RSA Data Security

ning in parallel, about 4 weeks to perform the sieving, then it took a large CRAY about 100 hours and 810 Mbytes of memory to solve the system of equations. The size of the factor bases used totaled about 1.5 million primes resulting in a system of about 4.7 million equations that needed to be solved.

How long would RSA-140 take with TWINKLE?

Each device is capable of accommodating a factor base of about 200,000 primes and a sieve interval of about 100 million. RSA-140 required a factor base of about 1.5 million, and the sieve interval is adequate, so about 7 devices would be needed. One can use a somewhat smaller factor base, but a substantially smaller one would have the effect of greatly increasing the sieving time. This set of devices would be about 1000 times faster than a single conventional computer, so the sieving could be done in about 6 days with 7 devices. The matrix would still take 4 days to solve, so the net effect would be to reduce the factorization time from about 33 days to 10 days, a factor of 3.3. This is an example of Amdahl's law which says that in a parallel algorithm the maximum amount of parallelism that can be achieved is limited by the serial parts of the algorithm. The time to solve the matrix becomes a bottleneck. Even though the matrix solution for RSA-140 required only a tiny fraction of the total CPU hours, it represented a fair fraction of the total ELAPSED time: it took about 15% of the elapsed time with conventional hardware for sieving. It would take about 40% of the elapsed time with devices. Note further that even if one could sieve *infinitely* fast, the speedup obtained would only be a factor of 8 over what was actually achieved.

How long would a 512-bit modulus take with TWINKLE?

A 512-bit modulus would take 6 to 7 times as long for the sieving and 2 to 3 times the size of the factor bases as RSA-140. The size of the matrix to be solved grows correspondingly, and the time to solve it grows by a factor of about 8. Thus, 15 to 20 devices could do the sieving in about 5-6 weeks. Doubling the number will cut sieving time in half. The matrix would take another 4 weeks and about 2 Gbytes of memory to solve. The total time would be 9-10 weeks. With the same set of conventional hardware as was used for RSA-140, the sieving would take 6 to 7 months and the matrix solving resources would remain the same.

Please note that whereas with RSA-140, solving the matrix would take 40% of the elapsed time, with a 512-bit number it would take just a bit more. This problem will get worse as the size of the numbers being factored grows.

How well will TWINKLE scale to larger numbers?

A 768 bit number will take about 6000 times as long to sieve as a 512-bit number and will require a factor base which is about 80 times large. The length of the sieve interval would also increase by a factor of about 80. Thus, while about 1200 devicess could accommodate the factor base, they would have to be redesigned to accommodate a much longer sieve interval. Such a set of machines would still take 6000 months to do the sieving. One can, of course, reduce this time by adding more hardware. The memory needed to hold the matrix would be about 64 Gbytes and would take about 24,000 times as long to solve.

A 1024-bit number is the minimum size recommended today by a variety of standards (ANSI X9.31, X9.44, X9.30, X9.42). Such a number would take 6 to 7 million times as long to do the sieving as a 512-bit number. The size of the factor base would grow by a factor of about 2500, and the length of the sieve interval would also grow by about 2500. Thus, while about 45,000 devices could accommodate the factor base, they would again have to be redesigned to accommodate much longer sieve intervals. Such a set would still take 6 to 7 million months (500,000 years) to do the sieving.

The memory required to hold the matrix would grow to 5 to 10 Terabytes and the disk storage to hold all the factored relations would be in the Petabyte range. Solving the matrix would take "about" 65 million times as long as with RSA-512. These are rough estimates, of course, and can be off by an order of magnitude either way.

What are the prospects for using a smaller factor base?

The Number Field Sieve finds its successfully factored congruences by sieving over the norms of two sets of integers. These norms are represented by polynomials. As the algorithm progresses, the coefficients of the polynomials become larger, and the rate at which one finds successful congruences drops dramatically. Most of the successes come very early in the running of the algorithm. If one uses a suboptimally sized factor base, the 'early' polynomials do not yield enough successes for the algorithm to succeed at all. One can try sieving more polynomials, and with a faster sieve device this can readily be

done. However, the yield rate can drop so dramatically that no additional amount of sieving can make up for the too-small factor base.

Keysize	Total Time	Factor Base	Sieve Memory	Matrix Memory
428	$5.5 * 10^{17}$	600K	24Mbytes	128M
465	$2.5 * 10^{18}$	1.2M	64Mbytes	825Mbytes
512	$1.7 * 10^{19}$	3M	128Mbytes	2 Gbytes
768	$1.1 * 10^{23}$	240M	10Gbytes	160Gbytes
1024	$1.3 * 10^{26}$	7.5G	256Gbytes	10Tbytes

The situation is different if one uses the Quadratic Sieve. For this algorithm all polynomials are 'equal', and one can use a sub-optimal factor base. However, for large numbers, QS is much less efficient than NFS. At 512-bits, QS is about 4 times slower than NFS. Thus, to do 512-bit numbers with devices, QS should be the algorithm of choice, rather than NFS. However, for 1024-bit numbers, QS is slower than NFS by a factor of about 4.5 million. That's a lot. And the factor base will still be too large to manage, even for QS.

What are the prospects for speeding the matrix solution?

Unlike the sieving phase, solving the matrix does not parallelize easily. The reason is that while the sieving units can run independently, a parallel matrix solver would require the processors to communicate frequently and both bandwidth and communication latency would become a bottleneck. One could try reducing the size of the factor bases, but too great a reduction would have the effect of *vastly* increasing the sieving time. Dealing with the problems of matrix storage and matrix solution time seems to require some completely new ideas.

Key Size Comparison

The table below gives, for different RSA key sizes, the amount of time required by the Number Field Sieve to break the key (expressed in total number of arithmetic operations), the size of the required factor base, the amount of memory, per machine, to do the sieving, and the final matrix memory.

The time column in the table below is useful for comparison purposes. It would be difficult to give a meaningful elapsed time, since elapsed time depends on the number of machines available. Further, as the numbers grow, the devices would need to grow in size as well. RSA-140 (465 bits) will take 6 days with 7 devices, plus the time to solve the matrix. This will require about $2.5 * 10^{18}$ arithmetic operations in total. A 1024-bit key will be 52 million times harder in time, and about 7200 times harder in terms of space.

The data for numbers up to 512-bits may be taken as accurate. The estimates for 768 bits and higher can easily be off by an order of magnitude.

Conclusion

The idea presented by Dr. Shamir is a nice theoretical advance, but until it can be implemented and the matrix difficulties resolved it will not be a threat to even 768-bit RSA keys, let alone 1024.

References:

1. A.K. Lenstra & H.W. Lenstra (eds), The Development of the Number Field Sieve, Springer-Verlag Lecture Notes in Mathematics #1554
2. Robert D. Silverman, The Multiple Polynomial Quadratic Sieve, Mathematics of Computation, vol. 48, 1987, pp. 329-340
3. H. teRiele, Factorization of RSA-140, Internet announcement in sci.crypt and sci.math, 2/4/99
4. R.M. Huizing, An Implementation of the Number Field Sieve, CWI Report NM-R9511, July 1995

For more information on this and other recent security developments, contact RSA Laboratories at one of the addresses below.

RSA Laboratories
20 Crosby Drive
Bedford, MA 01730 USA
781/687-7000
781/687-7213 (fax)
rsa-labs@rsa.com
http://www.rsa.com/rsalabs/

Copyright © 1999 RSA Laboratories, a division of RSA Data Security, Inc., a Security Dynamics Company. All rights reserved.