On the security of cryptographic primitives regarding technological innovations

Willi Geiselmann and Rainer Steinwandt

Institut für Algorithmen und Kognitive Systeme,
Fakultät für Informatik, Am Fasanengarten 5
Universität Karlsruhe, D-76128 Karlsruhe
{geiselma, steinwan}@ira.uka.de

Abstract: Besides exploiting structural insight into a cryptographic algorithm, cryptanalysis also tries to exploit technological innovations. Here we survey recent proposals for supporting the most time-consuming steps in the number field sieve, the best currently known (classical) algorithm for factoring large integers.

1 Introduction

Peter Shor's probabilistic algorithms for factoring integers and computing discrete logarithms on quantum computers [Sh94] provide an impressive example for the interconnections of algorithmic and technological innovations. Once quantum computers are available, these algorithms allow for a polynomial time solution of the two mentioned problems, which is in sharp contrast to the best known algorithms for "classical" hardware: For currently available computers, only algorithms with super-polynomial running time are known for these tasks.
Having in mind the relevance of the two problems for practically used cryptographic schemes, it is not surprising that the cryptographic research community also invests significant effort in attempts to find specialized hardware that speeds up the currently available "classical" algorithms. Here the main focus is not necessarily on finding hardware designs that allow for asymptotic improvements – already finding a possibility to break an at the moment widely deployed parameter choice would be very interesting.
In the sequel we survey some recent developments in the design of specialized hardware devices for supporting the so-called *number field sieve* (see [LL93]), the best currently known algorithm for factoring large integers. Rather impressive progress has been achieved here in the last three years, which causes several researchers to assume that factoring 1024-bit RSA-numbers is in principle feasible for a sufficiently large organization.

2 Factoring Integers with the Number Field Sieve

Typically, modern algorithms for factoring consist of two steps which dominate the total running time of the algorithm plus a pre- and a post-computation step. In particular, this holds for the number field sieve (NFS), which is nowadays the most common algorithm when dealing with the factorization of large integers as occurring in the public key of RSA-like cryptosystems. It should be emphasized here, that albeit not being dominating for the total running time of the algorithm, the pre- and post-computations must not be considered as trivial steps: For the NFS, here an irreducible polynomial has to be fixed, that determines the number field involved in a factorization. Choosing this parameter properly is a rather critical task, as a bad choice can drastically increase the running time of the subsequent relation collection step. Here, we do not recall the details of the individual parts of the NFS, but it is important to note that the already mentioned

- *relation collection step*, and the
- *linear algebra step*

dominate the overall running time when applying the NFS. Consequently, proposals for specialized hardware focus on these two parts. Interestingly, the view on the difficulty of the two steps has changed throughout the last years: in the linear algebra step, elements from the kernel of a large sparse matrix over GF(2) have to be determined, which traditionally involves the use of a "super"-computer with a sufficiently large amount of (fast) RAM. This is quite different from the relation collection step, which can easily be distributed onto "normal" PCs and workstations. Thus, it is tempting to believe that the linear algebra step is the more difficult one. However, by now the situation is just the other way round; in particular, manufacturing a specialized hardware for dealing with the linear algebra step of 1024 bit numbers seems much easier than doing the same for the relation collection step.

3 Supporting hardware for the linear algebra step

A major step forward in the design of specialized hardware for the linear algebra step is due to Daniel Bernstein [Be01]. His solution makes use of the well-known Wiedemann-algorithm, which we do not recall here. For the sequel it is sufficient to know, that this algorithm allows to reduce the linear algebra step to the problem of computing (long) sequences of matrix-vector-products $A{\cdot}v$, $A^2{\cdot}v$, $A^3{\cdot}v$, …, $A^k{\cdot}v$ where A is the sparse matrix over GF(2) derived in the relation collection step and v is a vector with GF(2)-entries.

3.1 Bernstein's proposal

Bernstein observed that the latter computation can be performed efficiently by means of a repeatedly applied parallel sorting algorithm running on a mesh of simple processing units. More specifically, Bernstein proposed the use of Schimmler's sorting algorithm which can sort $M^2{:=}2^{2m}$ numbers in less than $8M$ "steps" on M^2 processing units. Moreover, this algorithm requires only "local" communication and has a rather simple control logic which facilitates a hardware implementation. In dependence of the cost measure used, Bernstein's approach does even allow for an asymptotic speed-up of the number field sieve. Unfortunately, already for factoring a 512 bit RSA number, an ASIC that can handle the complete matrix needed for this size, would require a wafer area which is beyond the limits of current technology: realizing a circuit which spans more than a complete wafer is extraordinary difficult and error-prone. However, in 2002 Lenstra et al. [Le02] presented a new design for the linear algebra step which might even be able to handle the linear algebra step for 1024 bit numbers on a single wafer.

3.2 An improvement of Lenstra, Shamir, Tomlinson, and Tromer

In contrast to Bernstein's approach, the design presented in [Le02] relies on the use of a new – and theoretically still unexplored – routing algorithm, so-called *clockwise transposition routing*. Similarly, as in Schimmler's algorithm, only local compare-exchange operations and a very simple control logic are required. Further on, the new hardware proposal makes use of nowadays available technology that allows to mix special storage processes for very small DRAM with processes for ASICs. This trick is crucial for obtaining a circuit that both fits on a single standard 300 mm silicon wafer and can store a (sparse) representation of the matrix needed for the linear algebra step in the case of numbers with up to 1024 bit.
Of course, manufacturing a circuit that spans a full wafer is technologically still challenging, but it is more feasible than realizing a circuit that spans several wafers. Fortunately, in 2003 it was shown that even the use of a single wafer-sized circuit can be avoided through a suitable "partitioning" of the matrix derived in the relation collection.

3.3 The partitioning of Geiselmann and Steinwandt

The matrix-vector multiplications occurring in Wiedemann's algorithm can be split into multiplications of submatrices with subvectors. When exploiting this idea for a hardware implementation, the question of combining the intermediate results efficiently arises, but as explained in [GS03a], this problem can be solved. In summary, one can derive a design for a network of small chips which can complete the linear algebra step for 1024 bit numbers within a few days. Consequently, the linear algebra step for 1024 bit numbers is at the moment considered as in principle doable. Due to the distribution onto small chips in [GS03a], one may even think of increasing the matrix size beyond the estimates for the 1024 bit case. In particular, as already said in [Le02] one can *"conclude that from a practical standpoint, the security of RSA relies exclusively on the hardness of the relation collection step of the number field sieve."*

4 Supporting hardware for the relation collection step

Relation collection is done by *sieving*: Simplifying, one can say that here long intervals of integers are scanned for values where a specific integer-valued function has many "small" prime factors. Based on the use of expensive high-end technology, in 1999 Adi Shamir presented a design for an opto-electronic device which could do the sieving extremely fast [Sh99]; even the New York Times reported on this invention.

4.1 An opto-electronic proposal: TWINKLE

In the TWINKLE device, the sieving interval is represented by the time (number of clock cycles) used by the device, and each processor takes care of one prime number p. Namely, it counts to p, reports a "hit", counts to p again, etc. To be able to handle all the reports, Shamir proposed the use of LEDs and a photo detector to collect the light. This raises some non-trivial technical questions, but, e.g., through the use of high-end GaAs technology, all these obstacles can theoretically be overcome. Nevertheless, a practical implementation of a TWINKLE device has not been reported yet.

4.2 Silicon-based proposals: Mesh-based sieving and TWIRL

In [GS02] it is shown how Schimmler's sorting algorithm can be used to implement a sieving device for 512 bit numbers on a single silicon wafer. The idea is to split the sieving interval into subintervals which are handled individually. Unfortunately, handling small primes is quite troublesome in this design, and it is unclear how to scale the device to larger numbers. However, recent results [GS03b] show that a related design based on a routing mesh (that uses a compact *factor base* encoding in DRAM) allows for a significant speed-up and might allow for realizing the relation collection for 768-bit numbers on silicon chips that can be manufactured with current technology.
The fastest currently known design is *The Weizmann Institute Relation Locator* [ST03] which stores the factor bases in DRAM and can in principle handle 1024 bit numbers. However, for 1024 bit numbers, in the proposed form the largest involved chip fills a complete 300 mm wafer, which makes manufacturing the device rather challenging.

5 Conclusion

Significant progress in the design of hardware supporting the NFS has been achieved, and although the factorization of a 768 bit number has not been reported yet, by means of the proposed designs it might well be doable in reasonable time.

References

[Be01] Bernstein, D. J.: Circuits for integer factorization: a proposal. Manuscript, 2001. http://cr.yp.to/papers.html
[GS02] Geiselmann, W. and Steinwandt, R.: A Dedicated Sieving Hardware. Proc. of PKC '03, vol. 2567 of LNCS, pp. 256 – 266. Springer, 2002.
[GS03a] Geiselmann, W. and Steinwandt, R.: Hardware to Solve Sparse Systems of Linear Equations Over GF(2). To appear in Proc. of CHES '03, LNCS. Springer, 2003.
[GS03b] Geiselmann, W. and Steinwandt, R.: Yet Another Sieving Device. Submitted , 2003.
[Le02] Lenstra, A. K., Shamir A., Tomlinson J., and Tromer, E.: Analysis of Bernstein's Factorization Circuit. Proc. of ASIACRYPT 2002, vol. 2501 of LNCS, pp. 1 – 26. Springer, 2002.
[LL93] Lenstra, A. K. and Lenstra, H. W.: The development of the number field sieve, vol. 1554 of Lecture Notes in Mathematics. Springer, 1993.
[Sh94] Shor, P. W.: Algorithms for quantum computation: Discrete logarithms and factoring. Proc. of the 35th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, pp. 124 – 134, 1994.
[Sh99] Shamir, A.: Factoring Large Numbers with the TWINKLE Device. Proceedings of CHES '99, vol. 1717 of LNCS, pp. 2 – 12. Springer, 1999.
[ST03] Shamir, A. and Tromer, E.: Factoring Large Numbers with the TWIRL Device. To appear in Proc. of CRYPTO '03. Springer, 2003.