— Summary —
Evaluation of complexity of the sieving step of the general number field sieve

Thorsten Kleinjung and Arjen K. Lenstra

EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland

December 5, 2018

It will be apparent from our estimates, achieved using the best technologies available (as published in the open domain), that breaking even the least secure of commonly used public keys, namely 1024-bit RSA moduli, requires an effort of well over a million core years. A cryptanalytic effort of that magnitude is out of reach of the type of academic projects that have, over the years, helped shape the public perception of the level of security offered by public key cryptosystems. Obviously, these figures depend on the hardware and implementations that we used and will change over time.

Breaking 1536-bit RSA moduli can be estimated to be at least five order of magnitude more difficult, in accordance with theoretical estimates, and 2048-bit RSA moduli are, most likely, yet at least four orders of magnitude harder to break — but obtaining truly reliable evidence for the latter estimate requires hardware that we do not have at our disposal and do not even expect to get access to within the next decade.

Radically new insights are required to change these estimated efforts. As mentioned above innovative ideas have been unusually scarce lately and it would be fair to say that the regular, extrapolation-based rate of cryptanalytic progress as assumed for instance in [2,1], simply turned out to be overly optimistic.

References

1. A. K. Lenstra. Unbelievable security: Matching AES security using public key systems. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 67–86. Springer, Heidelberg, 2001.
2. A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

Evaluation of complexity of the sieving step of the general number field sieve

Thorsten Kleinjung and Arjen K. Lenstra

EPFL IC LACAL, Station 14, CH-1015 Lausanne, Switzerland

December 5, 2018

1 Introduction

The present introduction consists of four parts: a first part discussing technical trends for the past 10 years that are related to integer factorization; a second part discussing recent technical trends in related fields, in particular pertaining to newly emerging trends in public key cryptography; a third part consisting of a brief summary of our findings, which can also be found on the separately provided "Summary" page; and a final part describing the structure of the remainder of the report.

Technical trends for the past 10 years related to integer factorization. Looking at the developments in the field of integer factorization, at this point in time we are experiencing the longest record-less drought since general purpose integer factoring became relevant for cryptology [16,15]. Not only does the current 768-bit record [8]* date back to December 2009, also we are not aware of efforts that are underway and that would break this record in a meaningful manner. Since the 768-bit record there has been an almost total lack of new insights that would be worth an exhaustive testing effort, implying that embarking on a huge calculation just for the sake of a new record given the current state of the art would prove little more than the contributors' persistence** — while one may doubt if it is the best way to spend one's resources. Indeed, the current state of affairs does not even measure up to the bleak estimate of future cryptanalytic abilities in [2].

In particular it is noted that since the publication of [14] in 2003 there has been a total lack of new trends that could affect the sieving step of the general number field sieve method for integer factorization [10] or any other sieving based factorization method. Neither has there been, during the past 10 years, any other technical trend that could have an important effect on the general number field method for integer factorization; the effects of new findings reported in [6] and [12] must be considered to be relatively minor and the factorization related work reported in [7] has no noticeable effect on the general number field sieve.

* In this note we restrict ourselves to results that have been reported in the open literature and to computational efforts that can be supported by ordinary, open domain, academic resources.

** Maintaining interest in a huge effort is not easy: the large scale effort announced in [1], not involving factoring but elliptic curve discrete logarithms, has not been completed yet. We have not been able to ascertain if the project has been abandoned or not.

Concerning hardware trends, clock speeds and memory sizes have only developed at a very modest pace during the last decade, and have thus had little or no influence on the sieving step of the general number field sieve or its computational feasibility.

Recent technical trends pertaining to public key cryptography. This lack of progress in integer factorization does not imply that crypto-researchers relaxedly sit back, while feeling confident that information protection schemes based on 1024-bit RSA are not vulnerable to open domain, academic factorization efforts within the foreseeable future, as it used to be the case since the invention of RSA. Quite on the contrary, due to a technological development that started many years ago and that may well turn out to remain as hypothetical as it has always been — and currently still is — many crypto-practitioners are worried about the security of their current schemes and are encouraging the research community to devote its efforts to post-quantum cryptography. In particular since the United States' National Institute of Standards and Technology announced its post-quantum crypto standardization initiative, this new trend has picked up considerable steam, to the extent that interest in systems that are actually used and their cryptanalysis receives less attention than it used to get.

Obviously, the hypothetical development that we are referring to is the fear, concern, or hope that it will, some time in the future, be possible to build a quantum computer that is large enough to solve cryptanalytic problems (such as cracking 1024-bit RSA keys) that are currently out of reach for non-hypothetical, existing hardware and realistic open domain resource management. Given the considerable number of reputable scientists that occupy themselves with the many problems involved, the possibility that such a device will indeed be realized cannot be outrightly dismissed. The subject has managed to collect substantial amounts of funding from a wide variety of agencies and corporations worldwide, and progress on many different fronts is reported frequently. To an outsider it is impossible to assess how close we are getting to a quantum computer as a result of these developments, if it were not for the estimate that is not infrequently made in press releases and other publications, namely that it will be at least another decade to turn the latest progress into a working product: this was the case in the 1990s, in the 2000s, in the 2010s, and the same number is still reported right now. Extrapolation of these announced developments does not bode well for actual quantum computing anytime. A newer, related trend is the appearance of articles that express serious, (at least seemingly) well-argued doubts that cryptanalytically capable quantum devices can ever exist; it should be noted, however, that there is only a trickle of naysayer skepticism [3,4] compared to the abundance of "good news".

The present authors are not qualified to contribute a useful opinion to the quantum debate. But it seems clear that adopting post-quantum cryptography anytime soon may not be the most urgent priority and may — or will — lead to undue risks. At the same time, being up-to-date on the security of traditional cryptosystems remains as important as it used to be, and this will be the case for many years to come. The present note contributes to the

security assessment of traditional public key cryptosystems by providing estimates for the effort required to break RSA keys of various relevant sizes.

Summary of our findings. It will be apparent from our estimates, achieved using the best technologies available (as published in the open domain), that breaking even the least secure of commonly used public keys, namely 1024-bit RSA moduli, requires an effort of well over a million core years. A cryptanalytic effort of that magnitude is out of reach of the type of academic projects that have, over the years, helped shape the public perception of the level of security offered by public key cryptosystems. Obviously, these figures depend on the hardware and implementations that we used and will change over time.

Breaking 1536-bit RSA moduli can be estimated to be at least five order of magnitude more difficult, in accordance with theoretical estimates, and 2048-bit RSA moduli are, most likely, yet at least four orders of magnitude harder to break — but obtaining truly reliable evidence for the latter estimate requires hardware that we do not have at our disposal and do not even expect to get access to within the next decade.

Radically new insights are required to change these estimated efforts. As mentioned above innovative ideas have been unusually scarce lately and it would be fair to say that the regular, extrapolation-based rate of cryptanalytic progress as assumed for instance in [11,9], simply turned out to be overly optimistic.

Report organization. This report is organized in the following manner. Section 2 contains a high level description of the best — to our knowledge — cryptanalytic tool that we have at our disposal to assess the security of RSA moduli, namely the general number field sieve method for integer factorization. Two specific steps of the number field sieve that play a prominent role for our analyses are explained at the level of details required to be able to fully understand our approach and, if required, to re-derive our results; these are the sieving step and the cofactoring step. The details of the assessment method that we used, and how we arrived at the optimal overall estimates for a successful cryptanalytic effort, are described in Section 3 for 768-, 1024-, and 1536-bit composites. The actual results of our experiments, for the three sizes just mentioned but also for the 2048-bit case, are reported in Section 4; for the 2048-bit case this includes a description of our assessment methodology. Background data are provided in the appendices.

Comments on the other steps of the number field sieve integer factorization method are explicitly not included in this report, as the other main step, the linear algebra step, would require more resources than accessible to us in terms of memory and communication networks. All other steps are less critical.

2 Review

This section contains a brief review of some basic material of the previous report [5], namely of the number field sieve [10], of its sieving step and finally of the cofactoring step, a substep of the sieving step.

2.1 The number field sieve

Let N be an integer which is neither a power of a prime nor divisible by small primes, small depending on the parameters chosen in the algorithm. The number field sieve attempts to factor N by performing the following five steps:

1. Polynomial selection
   Two coprime polynomials $f_1, f_2 \in \mathbb{Z}[x]$ are chosen such that they share a root m modulo N and such that for each of the two polynomials its coefficients are coprime. The corresponding homogeneous polynomials are denoted $F_1, F_2 \in \mathbb{Z}[x, y]$, i.e., $F_i(x, y) = f_i(\frac{x}{y}) \cdot y^{\deg(f_i)}$, $i = 1, 2$. Moreover, let $K_i = \mathbb{Q}[x]/(f_i)$, $i = 1, 2$, denote the associated number fields.
2. Collection of relations (or sieving step)
   In this step sufficiently many coprime pairs $(a, b) \in \mathbb{Z} \times \mathbb{Z}_{>0}$ are found such that $F_i(a, b)$ is L_i-smooth (i.e., splits into primes $\leq L_i$) for $i = 1, 2$. These pairs are called relations and the integers L_i are called large prime bounds or smoothness bounds.
3. Construction of the matrix
   Since the previous step might produce duplicate relations, usually as a result of the lattice sieving technique (cf. below), these duplicates are removed in a first phase. The set of unique relations gives rise to a matrix over $\mathbb{F}_2$ whose rows correspond to (most) prime ideals of norm at most L_i in the number field K_i and whose columns correspond to these unique relations; the entries in the column associated to (a, b) are the valuations modulo 2 at a prime ideal of $a - bx$ in the corresponding K_i. Preprocessing this matrix by elementary column operations and deleting zero rows or columns leads to a smaller matrix M whose columns now correspond to sets of relations.
4. Linear algebra
   Several solutions of the system of linear equations $Mv = 0$ are computed in this step.
5. Final computation
   A subspace of the vector space of solutions from the previous step is calculated such that each element of the subspace gives rise to a congruence $c_1^2 \equiv c_2^2 \pmod{N}$ which leads to a possibly non-trivial splitting of N by computing $\gcd(c_1 + c_2, N)$.

Let $L_N[\alpha, c] = e^{(c+o(1))(\log N)^\alpha (\log\log N)^{1-\alpha}}$ for $\alpha, c \in \mathbb{R}$ and $0 \leq \alpha \leq 1$. If the polynomials in the first step are chosen appropriately, the large prime bounds are chosen as $L_N[\frac{1}{3}, \sqrt[3]{\frac{8}{9}}]$ and the relations (a, b) are collected for a and b bounded by $L_N[\frac{1}{3}, \sqrt[3]{\frac{8}{9}}]$ then the running time of the number field sieve is heuristically $L_N[\frac{1}{3}, \sqrt[3]{\frac{64}{9}}]$.

2.2 The sieving step

The goal of this step is to find sufficiently many coprime pairs (a, b) such that $F_i(a, b)$ is L_i-smooth for both $i = 1$ and $i = 2$. In most situations the most efficient algorithm for

carrying out this task is lattice sieving. For a prime ideal $\mathfrak{q}$ of $\mathcal{O}_{K_1}$ which is unramified, of degree 1 and does not divide the leading coefficient of f_1, the set of pairs (a, b) such that $(a - bx)$ is contained in $\mathfrak{q}$ forms a lattice of covolume q where q is the norm of $\mathfrak{q}$. For each pair (a, b) in this lattice $F_1(a, b)$ is divisible by q. In lattice sieving one chooses a range for q, e.g., an interval $[q_0, q_1]$, and examines for each prime ideal $\mathfrak{q}$ as above a region consisting of A points in the associated lattice, checking whether $F_1(a, b)$ and $F_2(a, b)$ are both smooth. The prime ideal $\mathfrak{q}$ is called a special q and the region is called the sieving region. Notice that the primality of $\mathfrak{q}$ is not needed since the above considerations also apply to ideals which are a product of prime ideals satisfying the conditions above; below such composite special q are used in the 2048-bit case.

The smoothness tests are usually carried out as follows. Let B_i, $i = 1, 2$, be positive integers with $B_i < L_i$ and $B_1 < q_0$ (or at least $B_1 < q$ for the special q under consideration), and let $|F_i(a, b)| = S_i^{(a,b)} R_i^{(a,b)}$ be the decomposition where the prime factors of $S_i^{(a,b)}$ are at most B_i and the prime factors of $R_i^{(a,b)}$ are all larger than B_i. A sieving procedure roughly analogous to the sieve of Eratosthenes allows to determine (most of) the (a, b) in the sieving region for which both $R_i^{(a,b)}$, $i = 1, 2$, are below given bounds. This is done by identifying for (almost) each degree-1 prime ideal of $\mathcal{O}_{K_1}$ resp. $\mathcal{O}_{K_2}$ with norm at most B_1 resp. B_2 the set of pairs (a, b) of the sieving region which are contained in the prime ideal, and reorganizing this information in order to estimate the size of $S_i^{(a,b)}$. The set of these prime ideals is called the factor base and the bounds B_i, $i = 1, 2$, are called the factor base bounds. In a second phase it is checked for each of these candidates whether $R_i^{(a,b)}$ is L_i-smooth for $i = 1$ and for $i = 2$; this latter phase is called the cofactoring step.

2.3 The cofactoring step

In the second phase of the sieving step, the cofactoring step, for $i = 1, 2$ integers R_i that are free of factors $\leq B_i$ have to be tested for L_i-smoothness. First, if either R_i-value is greater than L_i and cannot be proved to be composite using a probabilistic compositeness test (with base 2), then the pair (R_1, R_2) is discarded as probably not smooth. For each remaining pair (R_1, R_2) the smoothness tests are done in a two-stage process: a first stage consisting of a factoring attempt of the R_i-values for $i = 1, 2$ that are composite but not yet known to be L_i-smooth (thus larger than $B_i L_i$) and a second stage that aims to find the full factorizations of the pairs that (possibly as a result of the first stage) are known to be smooth or for which a non-trivial factor has been found (in the first stage) for each R_i that is not yet known to be L_i-smooth. It should be noted that variations have been used where the stages are intertwined.

In the first stage:

- depending on the sizes of R_1 and R_2 a particular sequence of factoring methods is applied to not already known to be smooth R_1 or R_2 (the factoring methods used are Pollard's $p - 1$ method, the elliptic curve method (ECM), and multiple polynomial quadratic sieve (MPQS); the sequence is determined as explained further below);

– if a non-trivial factor of R_i is found, further first stage R_i factoring attempts are skipped;
– if R_i is found not to be L_i-smooth (because a prime factor larger than L_i is found), then the smoothness test for the pair (R_1, R_2) fails and the pair (R_1, R_2) is discarded.

If at the end of the first stage an R_i remains that is not known to be L_i-smooth and that has not been successfully factored in the first stage, then the pair (R_1, R_2) is discarded as being too hard to factor. Otherwise, in the second stage, a combination of ECM and MPQS is attempted to factor all sufficiently small remaining composite parts of R_1 and R_2. If found to be not L_i-smooth then the pair (R_1, R_2) is discarded as not being smooth, if not fully factored, then it is discarded as being too hard to factor.

Given the sizes, the smoothness probability of the pair (R_1, R_2) can be approximated (using that the prime factors of R_i are larger than B_i for $i = 1, 2$ because the pairs (R_1, R_2) were found in the sieving step), along with the success probability and average running time of the first stage (depending on the applicable sequence). The second stage is less amenable to a precise analysis: all we can say is that in the second stage the parameters are chosen in such a way that a smooth R_i is factored with a probability of at least 0.9. As in the previous report this implies that in the second stage a smooth pair will be fully factored with probability at least 0.81.

The sequence of factoring methods to be used in the first stage is created by comparing many sequences for each relevant pair of bit-lengths of the composites R_1 and R_2. For each pair of bit-lengths the sequence is chosen for which the ratio of the approximated running time and success probability fits best the targeted number of seconds to be spent per relation. An indication for the latter is given by the parameter r, with larger r indicating that more time may be spent in the cofactoring step. For some bit-lengths the resulting sequence may be empty because the ratio is too large for all sequences considered.

3 General procedure

Since even for the smallest considered integer completing the sieving step takes a lot of time, the estimates in this report are obtained by executing a small but representative fraction of the sieving step and extrapolating the results. As in the previous report a sieving step is considered to be complete when the number of unique relations is at least $2\pi(L)$ where $L = L_1 = L_2$ is the smoothness bound and $\pi(x)$ is the number of primes up to x. Since lattice sieving usually will produce duplicate relations, i.e., relations found for several different special q-values, these have to be corrected for in the extrapolation. Below it is described how this can be done, followed by an explanation how the sieving experiments were executed and evaluated for composites of 768, 1024, and 1536 bits. The 2048-bit case is separately described in its own section.

3.1 Duplicates and weight

For a multiset of relations the weight of a relation (occurring in this multiset) is defined as $\frac{1}{n}$ where n is the multiplicity of the relation in the multiset of relations upon completion of the sieving step. Hence the number of unique relations is the sum of the weights of the relations in the multiset. Given the parameters of the sieving step it is relatively easy to compute the weight of a relation that was obtained in the sieving step. This is done by inspecting the factorizations of the polynomial values and identifying the primes p in these factorizations for which the relation could have been obtained by lattice sieving with p as the special q. For each such prime one checks whether lattice sieving would produce the relation (either by a conceptually simple but costly execution of the lattice sieve for that special q or by a more elaborate but faster check of each step in the lattice sieve, e.g., checking whether the cofactoring step would decompose the corresponding cofactor etc.). Therefore, once a sufficiently uniformly distributed fraction of the sieving step has been executed, the number of unique relations for the complete sieving step can be approximated in an efficient manner. The procedure has been tested on several data sets where a complete sieving step was carried out; the predicted results usually deviated by less than 1%.

3.2 Sieving experiments

For a single 768-bit, 1024-bit and 1536-bit integer the sieving experiments were conducted as follows. Apart from the integer, a fixed polynomial pair and a smoothness bound L (chosen as described below), the input of an experiment consists of a set of parameters $(B_1, B_2, A, r, C_{1,max}, C_{2,max}, q_{max}, l, n_q)$. The first two parameters are the factor base bounds for the two sides, as described in the previous section, and chosen as indicated below. The size of the sieving region per special q is set to A. As set forth in the previous section, the parameter r, an indication for the number of seconds to be spent per relation, is used to find optimal factoring method sequences for cofactoring, and the parameters $C_{1,max}$ and $C_{2,max}$ determine up to which bit-lengths sequences are sought. The largest 2^{c_1} for which a non-empty sequence exists for bit-lengths (c_1, c_2) is called C_1 and analogously for C_2 (thus $C_i \leq C_{i,max}$ for $i = 1, 2$). With $\ell_i = B_1 + il$ and $u_i = \ell_{i+1} - 1$, for all $i \in \mathbb{Z}_{\geq 0}$ such that $\ell_i < q_{max}$ sieving is done in the length l interval $[\ell_i, u_i]$ for n_q special q-values starting at $\frac{\ell_i + u_i}{2}$.

Given the effort involved of even doing a representative fraction of the sieving step, just a single parameter optimization is carried out per bit-length. Because after decades of factorization experiments we have never observed substantial deviations from the growth rate as expected based on the heuristic asymptotic running time, we are confident that the running time figures reported here are of the correct order of magnitude for relevant composites of comparable sizes.

3.3 Evaluation

The data collected during the sieving experiment is evaluated in the following manner. For each interval $[\ell_i, u_i]$ the weight of the discovered relations is computed. Then the number of special q-values in $[\ell_i, u_i]$ is estimated using the prime number theorem, and the weight as well as the sieving time are scaled by $\frac{\pi(u_i)-\pi(\ell_i)}{n_q}$. The sums of the weights and the sieving times over all intervals give approximations of the expected number of unique relations and the expected total sieving time. If the estimated number of unique relations is greater than the number of prime ideals, the evaluation is repeated with fewer intervals, i.e., a smaller q_{max}-value: due to the smaller range of special q-values the weights have to be recomputed. The number of intervals is decreased as long as the estimated number of unique relations is at least $2\pi(L)$. With n the final estimated number of unique relations, the combined total length of all intervals and the total sieving time in core years are scaled by the factor $\frac{n}{2\pi(L)}$ (which will be close to 1) resulting in the values that are below referred to as q_{len} and T.

4 Parameters and results of the sieving experiments

As in the previous report, the target numbers RSA768, RSA1024, RSA1536, and RSA2048 were taken from the list of RSA challenge numbers [13].

4.1 Sieving experiments for the 768-bit integer RSA768

For the experiments the polynomial pair from the actual factorization (dating back to 2009) was used, with smoothness bound $L = 2^{37}$ (the value originally planned to be used to the factorization and for which the cofactoring step was optimized, even though the larger value 2^{40} was used because it was deemed to be somewhat advantageous). It is considered unlikely that these choices are far from optimal.

Many experiments with varying memory requirements were conducted to assess their effect on the running time of the sieving step. The memory requirements of the lattice siever are approximately proportional to the size of the total factor base which can be proportionally approximated by $\pi(B_1) + \pi(B_2)$, which in turn is roughly proportionally approximated by $B_1 + B_2$. Thus for several values of the sum $B_1 + B_2$, distanced by a factor of roughly two, sieving experiments were carried out for several values of B_1, A and r, varying the latter in a reasonable region in order to find a nearly optimal set of parameters for each sum $B_1 + B_2$ considered. The best parameters and the results are listed in the table below, with the bold row indicating the parameter choices resulting in the lowest estimated running time (reported in the table in core years).

B_1	B_2	A	r	C_1	C_2	q_{len}	T
$1.4e9$	$600e6$	2^{32}	2	2^{141}	2^{141}	$2.99e9$	572.57
$900e6$	**$100e6$**	**2^{32}**	**2**	**2^{141}**	**2^{160}**	**$3.49e9$**	**561.99**
$400e6$	$100e6$	2^{31}	2	2^{141}	2^{160}	$7.86e9$	645.01
$180e6$	$20e6$	2^{31}	1	2^{139}	2^{160}	$16.8e9$	924.19
$90e6$	$10e6$	2^{30}	2	2^{160}	2^{160}	$32.5e9$	1237.48
$40e6$	$10e6$	2^{30}	1	2^{160}	2^{160}	$69.2e9$	2066.41
$14e6$	$6e6$	2^{30}	2	2^{160}	2^{160}	$76.3e9$	2737.34

Most of the time 24 sieving experiments were carried out simultaneously on the 24 cores of the machine used for the experiments.

Since RSA768 has already been factored, a comparison is in order. This is not entirely straightforward due to a number of unavoidable differences: the original computation was carried out on a wide variety of machines with differing memory sizes so that several sets of parameters had to be used; the smoothness bound was larger, namely 2^{40}; and more sieving was done to somewhat alleviate the subsequent parts of the computation. However, limiting the comparison to the parameter set for the largest memory machines while scaling for the clock rates, and taking into account the oversieving, the figures are reasonably close.

4.2 Sieving experiments for the 1024-bit integer RSA1024

For this integer a modest effort was spent on choosing a polynomial pair and the smoothness bound was set to $L = 2^{42}$ (compared to 2^{37} for 768-bit a growth by approximately the square root of the anticipated running time increase; see also below right after the table).

The experiments were carried out as for the 768-bit case, resulting in the table below where the best parameters are listed for each considered sum $B_1 + B_2$, and with the lowest estimated running time in bold. Unlike the 768-bit case each sieving experiment used all 24 cores: for the larger factor base sizes – and thus memory requirements – there was no choice; for the smaller factor base sizes reducing the number of cores per sieving experiment would have been an option, but for the sake of comparison we chose not to do so.

B_1	B_2	A	r	C_1	C_2	q_{len}	T
$62e9$	$2e9$	2^{37}	200	2^{200}	2^{192}	$131e9$	$1.68e6$
$30e9$	**$2e9$**	**2^{37}**	**200**	**2^{199}**	**2^{192}**	**$139e9$**	**$1.52e6$**
$14e9$	$2e9$	2^{37}	200	2^{199}	2^{192}	$159e9$	$1.57e6$
$7e9$	$1e9$	2^{35}	200	2^{199}	2^{192}	$718e9$	$1.77e6$
$3.5e9$	$500e6$	2^{34}	200	2^{226}	2^{192}	$1.6e12$	$2.26e6$
$1.8e9$	$200e6$	2^{35}	200	2^{227}	2^{192}	$1.09e12$	$2.57e6$
$900e6$	$100e6$	2^{35}	200	2^{228}	2^{192}	$1.44e12$	$3.17e6$
$450e6$	$50e6$	2^{35}	200	2^{229}	2^{192}	$2.08e12$	$4.28e6$
$225e6$	$25e6$	2^{35}	200	2^{244}	2^{192}	$3.05e12$	$6.9e6$
$110e5$	$18e6$	2^{36}	200	2^{249}	2^{192}	$2.5e12$	$12.6e6$

As for the 768-bit case, the parameters reported in the above table for the 1024-bit case are not affected by suboptimal hardware restrictions and thus optimal (given values of the sum $B_1 + B_2$), with the relatively small exception (compared to hardware-caused suboptimal choices encountered for larger bit-lengths, cf. below) that the former case could be done single-threaded, whereas for the present one multi-threading had to be used. This makes it possible and worthwhile to compare the present 1024-bit results to an extrapolation of the earlier 768-bit case. As usual the extrapolation is done by setting to 0 the $o(1)$-term in the definition of the function $L_N[\alpha, c]$ and calculating the ratio for $N = 2^{1024}$ and $N = 2^{768}$ for $\alpha = \frac{1}{3}$ and $c = \sqrt[3]{\frac{8}{9}}$ or $\sqrt[3]{\frac{64}{9}}$. This leads to ratios of 35 for the B_1 and B_2-values and of 1221 for the size of the total sieving region (i.e., A times the number of special q) and for the running time.

At first glance the experiments and the extrapolation seem to match quite well regarding the factor base bounds with ratios of 32 versus 35. However, the rows above and below the optimal ones suggest that the optimal $B_1 + B_2$ in the 768-bit case is slightly above $1e9$ and in the 1024-bit case slightly below $32e9$. Nevertheless, the deviation seems to be less than a factor of 2. Regarding the total sieving area the ratio obtained from the experiments is about 1100 which is very close to the extrapolation. Only with respect to the running time the experiment is off by a factor of 2.2, taking more than twice the time suggested by the extrapolation. Comparing in more detail the sieving runs reveals two issues that explain this effect. Firstly, in the 1024-bit case sieving tasks used 24 threads whereas they were single-threaded in the 768-bit case; one might expect that a similar run on a one core machine would reduce the factor of 2.2 to less than 2. Secondly, B_1 and B_2 are below 2^{32} in the 768-bit case, making it possible to use faster code and less memory.

4.3 Sieving experiments for the 1536-bit integer RSA1536

For this integer the polynomial pair and smoothness bound $L = 2^{50}$ were taken from the previous report.

Extrapolating (as described above) the 1024-bit factor base bounds suggests that for the 1536-bit case the optimal value for $B_1 + B_2$ would be in the range between $5e12$ and $20e12$, thus requiring several tens of terabytes of memory. The 256GB of memory available on the machine to be used imposes a largest possible (and suboptimal) value of $114e9$ for $B_1 + B_2$, so only that value was used, with lower ones even worse and thus not considered. Moreover, since sieving for a single special q over a region of size $A = 2^{45}$ takes more than a day on the 24-core machine, the parameters B_1, A and r were not selected by extensive experiments but as follows. For several values of B_1 and r, lattice sieving for one special q was done for a relatively small value of A such as 2^{37} or 2^{38}, and the expected number of relations (cf. previous report or 2048-bit case below) was used to choose the best values for B_1 and r. These values were fixed and A was increased until the expected maximal special q would be well below L.

This procedure led to the following parameters and results (with, as usual, r in seconds and T in core years).

B_1	B_2	A	r	C_1	C_2	q_{len}	T
$110e9$	$4e9$	2^{45}	200000	2^{320}	2^{240}	$277e12$	$0.94e12$

With the exception of the polynomial selection the data in the 1024-bit case are on relatively solid ground so that one might try to extrapolate it to the 1536-bit case. A computation as in the 1024-bit case gives a ratio of about 300 for the factor base bounds and a ratio of about 10^5 both for the size of the total sieving region and for the running time. Since the extrapolated factor base bounds are a hundred times bigger than those used in the experiment one might expect a considerable improvement by using bounds in the region that would follow from the extrapolation. Unfortunately, such experiments would require a machine with a main memory on the order of about 25TB which was not at our disposal. However, one might use the data for varying factor base bounds in the 768- and 1024-bit cases for the following speculative extrapolation. Namely, in these cases comparing the data for the best value of $B_1 + B_2$ to a value near $\frac{1}{100}$ of it, one gets a ratio near 5 both for the size of the total sieving region and for the running time. Thus if $B_1 = 11e12$, $B_2 = 400e9$ would be used for the 1536-bit case it is reasonable to predict that both the total sieving region and the running time are about 5 times smaller than the figures for the actual experiment reported in the above table. This would fit the extrapolation using $L_N[\alpha, c]$ quite well.

4.4 Sieving experiments for the 2048-bit integer RSA2048

Again, for this integer the polynomial pair and smoothness bound $L = 2^{57}$ were taken from the previous report.

As in the 1536-bit case the only meaningful choice for the factor base bounds is to choose them as large as possible while fitting into the available memory. It follows that the same factor base bounds were chosen as in the 1536-bit case. Choosing the other parameters revealed an obstacle, namely that a similar approach as in the 1536-bit case would lead to a sieving region size A far above 2^{50}, perhaps even above 2^{55}. Thus sieving for a single special q would take months or even years. Therefore an approach similar to the one in the previous report was used. More precisely, the special q was chosen as $q = q_1 q_2$ with $q_i \in \mathcal{Q}_i$ where $\mathcal{Q}_1$ is the set of special q in the interval $[110e9, 300e9]$ and $\mathcal{Q}_2$ is the set of special q in the interval $[300e9, 625e9]$.

Another obstacle was that it takes on average several core years to produce one relation, so that modest computing resources would by far not allow to obtain sufficiently many relations for an estimate of the total number of relations and the number of duplicates. Therefore the number of relations was estimated by analyzing the smoothness probabilities of the cofactors (cf. previous report; essentially for each pair of bit-lengths (c_1, c_2) a

probability was precalculated that the cofactoring strategy associated to these bit-lengths would report both cofactors being L-smooth, and these probabilities are summed for all pairs of cofactors).

The procedure for estimating the percentage of duplicates described in the previous report was also applied here. Essentially, approximations for the number of relations n_{rel} and for the number of duplicates n_{dup} were calculated. Several simplifications were used which heuristically rather increase than decrease the quotient $\frac{n_{dup}}{n_{rel}}$. For the choice of the range and shape of special q as above the formulae of the previous report become

$$n_{rel} = \sum_{q_1 \in \mathcal{Q}_1} \sum_{q_2 \in \mathcal{Q}_2} A\rho\left(\frac{\log g_1(q_1 q_2 A) - \log q_1 q_2}{\log L}\right)\rho\left(\frac{\log g_2(q_1 q_2 A)}{\log L}\right)$$

and

$$\begin{aligned} n_{dup} = & \frac{1}{2}\sum_{q_1 \in \mathcal{Q}_1}\sum_{q_2 \in \mathcal{Q}_2}\sum_{r_2 \in \mathcal{Q}_2}\frac{A}{\max(q_2, r_2)}\rho\left(\frac{\log g_1(q_1 \min(q_2, r_2)A) - \log q_1 q_2 r_2}{\log L}\right) \times \\ & \times\rho\left(\frac{\log g_2(q_1 \min(q_2, r_2)A)}{\log L}\right) \\ & + \frac{1}{2}\sum_{q_1 \in \mathcal{Q}_1}\sum_{r_1 \in \mathcal{Q}_1}\sum_{q_2 \in \mathcal{Q}_2}\frac{A}{\max(q_1, r_1)}\rho\left(\frac{\log g_1(\min(q_1, r_1)q_2 A) - \log q_1 r_1 q_2}{\log L}\right) \times \\ & \times\rho\left(\frac{\log g_2(\min(q_1, r_1)q_2 A)}{\log L}\right), \end{aligned}$$

where g_1 and g_2 are functions approximating the polynomial values, as defined in the previous report.

Using the choices and tools above the parameters and results are as follows.

B_1	B_2	A	r	C_1	C_2	T
$110e9$	$4e9$	2^{39}	$800e6$	2^{390}	2^{280}	$128e15$

In this case most extrapolations would be mere speculation. As in the 1536-bit case one might expect to reduce the running time by increasing the factor base bounds considerably. This would also imply that one can use prime special q which would facilitate comparisons to the smaller cases.

References

1. D. V. Bailey, B. Baldwin, L. Batina, D. J. Bernstein, P. Birkner, J. W. Bos, G. van Damme, G. de Meulenaer, J. Fan, T. Güneysu, F. Gurkaynak, T. Kleinjung, T. Lange, N. Mentens, C. Paar, F. Regazzoni, P. Schwabe, and L. Uhsadel. The Certicom challenges ECC2-X. Special-purpose Hardware for Attacking Cryptographic Systems – SHARCS 2009, 2009. `http://www.hyperelliptic.org/tanja/SHARCS/record2.pdf`.
2. J. W. Bos, M. E. Kaihara, T. Kleinjung, A. K. Lenstra, and P. L. Montgomery. On the security of 1024-bit RSA and 160-bit elliptic curve cryptography. Cryptology ePrint Archive, Report 2009/389, 2009. `http://eprint.iacr.org/`.
3. G. Kalai. How quantum computers can fail. `http://arxiv.org/abs/quant-ph/0607021v3`, 2008.
4. G. Kalai. The quantum computer puzzle. `http://www.ams.org/journals/notices/201605/rnoti-p508.pdf`, 2016.
5. T. Kleinjung. Evaluation of complexity of mathematical algorithms. `http://www.cryptrec.go.jp/exreport/cryptrec-ex-0601-2006.pdf`, 2006.
6. T. Kleinjung. Polynomial selection, presented at the CADO workshop. See `http://cado.gforge.inria.fr/workshop/slides/kleinjung.pdf`, 2008.
7. T. Kleinjung. Quadratic sieving. *Mathematics of Computation*, 85:1861–1873, 2016.
8. T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, and P. Zimmermann. Factorization of a 768-bit RSA modulus. In T. Rabin, editor, *Crypto 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 333–350. Springer, Heidelberg, 2010.
9. A. K. Lenstra. Unbelievable security: Matching AES security using public key systems. In C. Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 67–86. Springer, Heidelberg, 2001.
10. A. K. Lenstra and H. W. Lenstra Jr. *The Development of the Number Field Sieve*, volume 1554 of *Lecture Notes in Mathematics*. Springer-Verlag, 1993.
11. A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.
12. A. Miele, J. W. Bos, T. Kleinjung, and A. K. Lenstra. Cofactorization on graphics processing units. In L. Batina and M. Robshaw, editors, *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2014.
13. RSA the security division of EMC. The RSA challenge numbers. Formerly on `http://www.rsa.com/rsalabs/node.asp?id=2093`, now on `http://en.wikipedia.org/wiki/RSA_numbers`.
14. A. Shamir and E. Tromer. Factoring large number with the TWIRL device. In D. Boneh, editor, *Crypto 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 1–26. Springer, Heidelberg, 2003.
15. P. Zimmermann. Integer factoring records. `https://members.loria.fr/PZimmermann/records/factor.html`, 2013.
16. P. Zimmermann. Previous records. `https://members.loria.fr/PZimmermann/records/factor-previous.html#general`, 2013.

5 Appendix A: Polynomial pairs

The polynomial pairs used in the experiments are listed below.

RSA-768:

This is the polynomial pair used in the actual factorization from 2009.

$$\begin{aligned} f_1 = {} & 265482057982680x^6 \\ & +1276509360768321888x^5 \\ & -5006815697800138351796828x^4 \\ & -4647785447172785427177267745 0x^3 \\ & +652543726193598939710966737189 4785x^2 \\ & -1818577935208859435672601886243 4803054x \\ & -2775652667915438819952161997138 01103343120 \end{aligned}$$

$$\begin{aligned} f_2 = {} & 346610035504925018514458 29x \\ & -1291187456580021223163547791574810881 \end{aligned}$$

RSA-1024:

$$\begin{aligned} f_1 = {} & 1000000001873592720x^6 \\ & -744644915849236144164660453 2x^5 \\ & -285544812934197384909755587555432 2x^4 \\ & -1038266541898906695192760860355886451581 5x^3 \\ & +1160268202215224022080002268640209131773457498 7x^2 \\ & +5577186790441970080185754971249399812968739265217915x \\ & +5527093890728070889669810334617963187464494162068365563 75 \end{aligned}$$

$$\begin{aligned} f_2 = {} & 50793101154086383x \\ & -2265120061143968874534624738373239732665962281812 \end{aligned}$$

RSA-1536:

This is the same polynomial pair as in the previous report.

$$
\begin{aligned}
f_1 = {} & 1000000000000000037811142945142568000x^7 \\
& -437143653502715952188541072955314515995561 44x^6 \\
& +303287546715162928320354682357442025586744759016 16x^5 \\
& -254963269480634915973438988235710497517495938866607909 0x^4 \\
& -743476621449070705426118043952107910813296873947226569967 6x^3 \\
& +269410316763704652496947389047029781532137807319675783818683 53x^2 \\
& +458682187400127339305595842561659927477563851565667757607476407 86x \\
& -488677948824510084414242643357352735730882412646241428043715801060 80
\end{aligned}
$$

$$
\begin{aligned}
f_2 = {} & 2053344690892016453093412763105761229054174 73x \\
& -785657129056763494515069161451393163540266862505483850255032 1
\end{aligned}
$$

RSA-2048:

This is the same polynomial pair as in the previous report (there is a typo in the previous report resulting in the omission of the last two digits of the x^2-coefficient of f_1).

$$
\begin{aligned}
f_1 = {} & 1000000000000000000000000000000000000733381474245981560 00x^7 \\
& +83958826335886062384725839675320017089308353422946100747992593 11x^6 \\
& -16243238465871826179486629347349327664597575878170309578532569786195 86x^5 \\
& -16297922667449801701018887260891713344933788688216935280929872819397554 24x^4 \\
& +33842219052921654312275628323473891052030445427581764495124943311088430533903x^3 \\
& +97169126397510065827694983318278224750173367414888014525694202469140892668968 07x^2 \\
& -12765890572198873811369455129056067788190946112995069419519970488320653903292035425 2x \\
& +70780743244066711466692928940077321828000766324763392134444939614936295115875125481344
\end{aligned}
$$

$$
\begin{aligned}
f_2 = {} & 411234126017228656931071529595260552676 29x \\
& -11411240546504417908329289807168838099957582124631681260838032078098068397983607 1
\end{aligned}
$$

6 Appendix B: Technical data of the PC

Below are listed some technical data of the PCs used for the sieving experiments.

CPU	2× Intel Xeon E5-2680 v3 (Haswell)
Cores (per CPU)	12
Clock rate	2.5 GHz
Cache size (per CPU)	L2: 12 × 256 kB, L3: 30 MB
Memory	16 × 16 GB DDR4-2133