Factoring estimates for a 1024-bit RSA modulus

Arjen Lenstra[1], Eran Tromer[2], Adi Shamir[2], Wil Kortsmit[3], Bruce Dodson[4], James Hughes[5], Paul Leyland[6]

[1] Citibank, N.A. and Technische Universiteit Eindhoven,
1 North Gate Road, Mendham, NJ 07945-3104, USA, arjen.lenstra@citigroup.com
[2] Department of Computer Science and Applied Mathematics,
Weizmann Institute of Science, Rehovot 76100, Israel,
{tromer,shamir}@wisdom.weizmann.ac.il
[3] Technische Universiteit Eindhoven,
P.O.Box 513, 5600 MB Eindhoven, The Netherlands, rcwil@win.tue.nl
[4] Lehigh University, Bethlehem, PA 18015-3174, USA, bad0@lehigh.edu
[5] Storage Technology Corporation,
7600 Boone Ave No, Minneapolis, MN 55428, USA, James_Hughes@StorageTek.com
[6] Microsoft Research Ltd,
7 JJ Thomson Avenue, Cambridge, CB3 0FB, UK, pleyland@microsoft.com

Abstract. We estimate the yield of the number field sieve factoring algorithm when applied to the 1024-bit composite integer RSA-1024 and the parameters as proposed in the draft version [17] of the TWIRL hardware factoring device [18]. We present the details behind the resulting improved parameter choices from [18].

Keywords: 1024-bit RSA, factorization, number field sieve, TWIRL

1 Introduction

RSA with 1024-bit moduli is widely used. It is unlikely that breaking a single 1024-bit RSA modulus will change much, just as repeatedly breaking DES had, for obvious economic reasons, limited effect on legacy applications. Nevertheless, despite the possible lack of immediate practical relevance, in cryptographic circles there is wide-spread interest in the question how hard it would be to factor a 1024-bit RSA modulus (cf. [2], [12]).

At the Asiacrypt 2002 rump session an innovative hardware device, 'TWIRL', was presented that would be able to factor 1024-bit RSA moduli at a much lower cost than before. The work reported here was inspired by that presentation and the draft of TWIRL [17]. The draft presents cost estimates for a number field sieve (NFS) factorization of a 1024-bit composite that rely on extrapolations of parameter settings used for a 512-bit NFS factorization (cf. Section 4). To our knowledge the accuracy of long range extrapolation from 512 to 1024 bit parameter selection had never been properly tested. Our goal was therefore to do a 'reality check' of the choices made in [17]. Given the many uncertainties involved in the factoring process we did not expect conclusive results but hoped to get an indication if the proposed parameters looked 'reasonable' or not. As it turned out, our results suggested that the choices were over-optimistic. Our approach was subsequently adopted by the authors of TWIRL. It allowed them to derive realistic parameters and to fine-tune the improved design [18]. The additional cost of the new choices is offset, approximately, by the greater efficiency of the new design, so that the overall cost estimates of [17] and [18] are similar. The details of the parameter settings from [18] are presented in Appendix B.

A sketch of our approach follows. We assume elementary background on the NFS (cf. Section 2). We selected the number RSA-1024 from [16] as a representative 1024-bit RSA modulus. This choice was supported by experiments that did not reveal significant differences between RSA-1024 and several other 1024-bit products of randomly selected 512-bit primes. We followed the search strategy from [13], [14], [15] to select number fields of degrees 5, 6, 7, 8, and 9 for RSA-1024, but we did not spend as much time on the search as we would have done for an actual factoring attempt. The resulting number fields can thus be regarded as somewhat worse than the number fields that would result from a more extensive search and the resulting estimates are on the pessimistic side. The better polynomial selection program of Jens Franke and Thorsten Kleinjung can handle only degree 5. It was used in Appendix B.

For all these number fields and a wide range of factor base sizes and sieving regions (including the choices made in [17]) we estimated the expected number of relations using numerical approximation of the applicable smoothness and semi-smoothness probabilities. Unfortunately, there is no a priori way to evaluate how close the resulting estimates are to the actual yield. To validate the estimates, we therefore ran extensive (semi-)smoothness tests on the actual numbers that would appear in an NFS factoring attempt, restricted to the most promising degrees and subsets of the sieving regions. We used the relatively slow test described in Section 3. This posed no problems because our object was determining the yield, not optimizing the speed. It can be seen in Section 5 that although the different methods do not produce identical results, the actual smoothness tests do inspire a high level of confidence in the numerical approximations.

Furthermore, we computed similar estimates for the multiple number field approach from [5], under the untested and possibly over-optimistic assumption that all number fields are about equally 'good' as the number fields we generated (cf. Section 6). In the same section we estimated the yield under the assumption that we are able to find much better number fields than we found, for instance by adapting the Franke/Kleinjung program to higher degrees. Corresponding actual smoothness experiments were not performed for these variations, because they involve number fields that we did not actually manage to construct.

There is nothing new to our approach and neither are the results earth-shaking. In particular we did not attempt to address the uncertainties referred to above, namely to analyse the cycle-matching behavior of relations involving large primes. We are not aware of any progress in that area. Despite the lack of innovative results, we hope that the approach presented in this paper is helpful to other researchers in this field. From that point of view our work already proved useful, as witnessed by the evolution of [17] into [18] (cf. Appendix B).

2 Number field sieve background

This section describes the parts of the number field sieve factoring algorithm which are relevant for this paper. See [10] for further details. The number of primes $\leq x$ is denoted by $\pi(x)$. An integer is *y-smooth* if all its prime factors are $\leq y$. An integer k is *(y, z, ℓ)-semi-smooth* if it is y-smooth except for at most ℓ prime factors that are $> y$ and $\leq z$ (referred to as *large primes*). If this is the maximal such ℓ, then k is *strictly (y, z, ℓ)-semi-smooth*.

Regular NFS. Let n be the number to be factored. Fix a degree d. Find an integer m (close to $n^{1/(d+1)}$), an irreducible polynomial $f \in \mathbf{Z}[X]$ of degree d such that $f(m) \equiv 0 \bmod n$, and a corresponding skewness ratio s (cf. [13], [14], [15]). This f is chosen such that the values $b^d f(a/b)$, for coprime pairs of integers (a, b) with $b > 0$, have a larger than average y-smooth factor, for small y. For integer k, let $\eta(y, k)$ denote the largest y-smooth factor of k and $\lambda(y, k) = \log(\eta(y, k))$ the natural logarithm thereof. For random integers, the expected value $E(y)$ of $\lambda(y, k)$ is known to be

$$E(y) = \sum_{p<y,\, p \text{ prime}} (\log p)/(p-1).$$

The expected value $E_f(y)$ of $\lambda(y, b^d f(a/b))$ can be determined experimentally by averaging $\lambda(y, b^d f(a/b))$ over a large random set of coprime pairs (a, b) with $b > 0$. The correction factor that measures f's advantage is defined as $t = \exp(E_f(2^{30}) - E(2^{30}))$.

Fix rational smoothness and semi-smoothness bounds $y_{\mathbf{r}}$ and $z_{\mathbf{r}}$ and algebraic ones $y_{\mathbf{a}}$ and $z_{\mathbf{a}}$, with $y_{\mathbf{r}} \leq z_{\mathbf{r}}$ and $y_{\mathbf{a}} \leq z_{\mathbf{a}}$. Fix the number of large primes on the rational side $\ell_{\mathbf{a}}$ and on the algebraic side $\ell_{\mathbf{r}}$. In the sieving step find *relations*: pairs of coprime integers (a, b) with $b > 0$ such that the *rational norm* $N_{\mathbf{r}}(a, b) = |a - bm|$ is $(y_{\mathbf{r}}, z_{\mathbf{r}}, \ell_{\mathbf{r}})$-semi-smooth and the *algebraic norm* $N_{\mathbf{a}}(a, b) = |b^d f(a/b)|$ is $(y_{\mathbf{a}}, z_{\mathbf{a}}, \ell_{\mathbf{a}})$-semi-smooth. If $N_{\mathbf{r}}(a, b)$ is $y_{\mathbf{r}}$-smooth and $N_{\mathbf{a}}(a, b)$ is $y_{\mathbf{a}}$-smooth, the relation is referred to as a full relation, otherwise it is called a partial relation. Approximately $\pi(\min(y_{\mathbf{r}}, y_{\mathbf{a}}))/d!$ full relations are free, namely one for each prime $p \leq \min(y_{\mathbf{r}}, y_{\mathbf{a}})$ such that f has d roots modulo p (cf. [10]). A non-free relation (a, b) for which $N_{\mathbf{r}}(a, b)$ is strictly $(y_{\mathbf{r}}, z_{\mathbf{r}}, L_{\mathbf{r}})$-semi-smooth and $N_{\mathbf{a}}(a, b)$ is strictly $(y_{\mathbf{a}}, z_{\mathbf{a}}, L_{\mathbf{a}})$-semi-smooth will be called an *$(L_{\mathbf{r}}, L_{\mathbf{a}})$-partial relation*. We use the standard abbreviations ***ff*** for $(0, 0)$-partial relations, ***fp*** for $(0, 1)$-partial relations, ***pf*** for $(1, 0)$-partial relations and ***pp*** for $(1, 1)$-partial relations.

For the $N_{\mathbf{r}}(a, b)$'s the sieving step involves sieving with the primes $\leq y_{\mathbf{r}}$, the *rational factor base* of cardinality $\pi(y_{\mathbf{r}})$. For the $N_{\mathbf{a}}(a, b)$'s it involves sieving with pairs (p, r) with $p \leq y_{\mathbf{a}}$ prime and $f(r) \equiv 0 \bmod p$, the *algebraic factor base* of cardinality $\approx \pi(y_{\mathbf{a}})$. Let $T(y_{\mathbf{r}}, y_{\mathbf{a}}) = \pi(y_{\mathbf{r}}) + \pi(y_{\mathbf{a}}) - \pi(\min(y_{\mathbf{r}}, y_{\mathbf{a}}))/d!$.

The purpose of the sieving step is to find approximately $T(y_{\mathbf{r}}, y_{\mathbf{a}})$ independent cycles: sets C of relations such that $\prod_{(a,b)\in C} N_{\mathbf{r}}(a, b)$ is a square times a $y_{\mathbf{r}}$-smooth number and, simultaneously, $\prod_{(a,b)\in C} N_{\mathbf{a}}(a, b)$ is a square times a $y_{\mathbf{a}}$-smooth number. The condition on the last square is slightly more involved; see below. A full relation is a cycle of length 1. Two $(1,0)$-partial relations whose rational norms share a large prime can be combined into a cycle of length 2. Similarly, for two $(0,1)$-partial relations (a_1, b_1) and (a_2, b_2) whose algebraic norms share the large prime p, a length 2 cycle follows if the relations correspond to the same root of $f \bmod p$, i.e., if $a_1/b_1 \equiv a_2/b_2 \bmod p$. Longer cycles may be built by pairing matching rational large primes or matching algebraic large primes with corresponding roots.

The part of the (a, b)-plane where relations are sought, the sieving region, consists of a, b with $-A < a \le A$ and $0 < b \le B$ for sufficiently large $A, B > 0$ with $A/B \approx s$. The size $2AB$ of the sieving region is denoted by S. A rectangular sieving region is in general not optimal in the sense that certain carefully chosen and somewhat smaller regions may yield the same number of relations (cf. [20]). For our yield computations this is hardly a concern.

Given approximately $T(y_\mathbf{r}, y_\mathbf{a})$ independent cycles, the factorization of n follows by applying the matrix step to the cycles and the square-root step to the results of the matrix step; these final two steps are not discussed in this paper.

Cycle yield. The number of relations required to obtain $T(y_\mathbf{r}, y_\mathbf{a})$ independent cycles is determined by the matching behavior of the large primes. This behavior varies from factorization to factorization and is not yet well understood. Obviously, $T(y_\mathbf{r}, y_\mathbf{a})$ distinct (non-free) full relations suffice, but this is necessary only if the large primes cannot be paired at all — that has never occurred in practice so far. Furthermore, the behavior gets considerably more complicated if more than a single large prime is allowed in the rational and algebraic norms. This is customary in current factorizations because it leads to a considerable speedup (cf. [4]). The uncertainty about the matching behavior of the large primes is the main reason that it is currently impossible to give reliable estimates for the difficulty of factoring numbers that are much larger than the numbers we have experience with. For that reason, we mostly restrict ourselves to estimates of the sieving region that would be required to find $T(y_\mathbf{r}, y_\mathbf{a})/c$ non-free full relations for a range of $y_\mathbf{r}$ and $y_\mathbf{a}$ values and several values of $c \ge 1$. Note that, for any number of large primes per relation, $\pi(z_\mathbf{r}) + \pi(z_\mathbf{a})$ relations always suffice.

Effort required. For smoothness bounds $y_\mathbf{r}$ and $y_\mathbf{a}$, sieving region size S and assuming a traditional implementation, the sieving effort is dominated by the number of times the primes and (prime,root) pairs in the factor bases hit the sieving region. This value is approximately proportional to

$$S(\log\log(y_\mathbf{r}) + \log\log(y_\mathbf{a})).$$

Furthermore, memory for the sieve and the factor bases may be needed.

Coppersmith's multi-polynomial version. As shown in [5] an improvement of the regular NFS can be obtained by considering a set G of irreducible degree d polynomials with shared root m modulo n. In that case, a relation is a pair of coprime integers (a, b) with $b > 0$ such that $N_\mathbf{r}(a, b)$ is $(y_\mathbf{r}, z_\mathbf{r}, \ell_\mathbf{r})$-semi-smooth and $b^d g(a/b)$ is $(y_\mathbf{a}, z_\mathbf{a}, \ell_\mathbf{a})$-semi-smooth for a $g \in G$. The goal is to find $\pi(y_\mathbf{r}) + \#G(\pi(y_\mathbf{a}) - \pi(\min(y_\mathbf{r}, y_\mathbf{a}))/d!)$ cycles. First, sieving is used to find a set V of $(y_\mathbf{r}, z_\mathbf{r}, \ell_\mathbf{r})$-semi-smooth rational norms (with a and b coprime). Next, a smoothness test different from sieving is used (in [5] the elliptic curve method is suggested) to test $b^d g(a/b)$ for $(y_{\mathbf{a}}, z_{\mathbf{a}}, \ell_{\mathbf{a}})$-semi-smoothness for all $(a, b) \in V$ and all $g \in G$. The approximate runtime of the relation collection becomes proportional to

$$S \log\log(y_{\mathbf{r}}) + E(\#V)(\#G)$$

where E is a constant of proportionality that depends on the $(y_{\mathbf{a}}, z_{\mathbf{a}}, \ell_{\mathbf{a}})$-semi-smoothness test used. Its value is best determined empirically.

3 Number field sieve analysis and estimates

Let the notation be as above. This section describes the methods we used to estimate the yield of the NFS. Let $L_x[r, \alpha]$ denote any function of x that equals

$$\exp((\alpha + o(1))(\log x)^r (\log\log x)^{1-r}), \text{ for } x \to \infty,$$

where α and r are real numbers with $0 \le r \le 1$ and logarithms are natural.

Estimating smoothness and semi-smoothness probabilities. Let $\sigma_\ell(u, v)$ denote the probability that a random integer $\le x$ is strictly $(x^{1/u}, x^{1/v}, \ell)$-semi-smooth, for $x \to \infty$. In particular, $\sigma_0(u, v)$ is the probability of $x^{1/u}$-smoothness, and equals the Dickman $\rho(u)$ function (cf. [1], [6]) which is $u^{-u+o(1)}$ for $u \to \infty$ (cf. [3], [7]). Also, let $\bar{\sigma}_2(u, v, w)$ be the probability that a random integer $\le x$ is $x^{1/u}$-smooth except for exactly two prime factors $> x^{1/u}$ and $\le x^{1/v}$ whose product is $< x^{1/w}$ (note that $\sigma_2(u, v) = \bar{\sigma}_2(u, v, v/2)$). We assume that these functions give good approximations of the semi-smoothness probabilities for the finite values of x that we consider (cf. Section 5, [1], [9]).

Closed expressions for σ_ℓ are not known. Thus, for ρ and σ_1 we used the numerical approximation methods given in [1]. To compute σ_2 and $\bar{\sigma}_2$ we used a natural generalization of [9, Theorem 3.1] and performed the integration numerically using the *GNU Scientific Library.*

Asymptotic runtime. It is heuristically assumed that with respect to smoothness properties $N_{\mathbf{r}}(a, b)$ and $N_{\mathbf{a}}(a, b)$ behave independently as random integers of comparable sizes. It follows that a pair of coprime integers (a, b) leads to a full relation with probability ${u_{\mathbf{r}}}^{-u_{\mathbf{r}}+o(1)} {u_{\mathbf{a}}}^{-u_{\mathbf{a}}+o(1)}$, where $u_{\mathbf{r}} = \frac{\log(N_{\mathbf{r}}(a,b))}{\log(y_{\mathbf{r}})}$ and $u_{\mathbf{a}} = \frac{\log(N_{\mathbf{a}}(a,b))}{\log(y_{\mathbf{a}})}$. Optimization of the parameters leads to the heuristic asymptotic expected NFS runtime $L_n[1/3, (64/9)^{1/3}] \approx L_n[1/3, 1.923]$, for $n \to \infty$, $y_{\mathbf{r}}$ and $y_{\mathbf{a}}$ both equal to $L_n[1/3, (8/9)^{1/3}]$ (the 'square-root of the runtime'), and the sieving region size $S = L_n[1/3, (64/9)^{1/3}]$. The correction factor t and large primes are believed to affect these values only by a constant factor (which disappears in the $o(1)$). Coppersmith's multi-polynomial variant [5] runs, asymptotically, slightly faster in expected time $L_n[1/3, 1.902]$. These expressions provide some insight into parameter selection, but the presence of the $o(1)$ limits their practical value. See Section 4 for how they are often used in practice.

Estimating the yield using ρ and σ_ℓ. For actual yield estimates we include the correction factor t defined in Section 2. Redefine $u_{\mathbf{a}} = \frac{\log(N_{\mathbf{a}}(a,b)/t)}{\log(y_{\mathbf{a}})}$, and define $v_{\mathbf{r}} = \frac{\log(N_{\mathbf{r}}(a,b))}{\log(z_{\mathbf{r}})}$, $v_{\mathbf{a}} = \frac{\log(N_{\mathbf{a}}(a,b)/t)}{\log(z_{\mathbf{a}})}$. Then under the same assumptions as above, it follows that (a,b) forms an $(L_{\mathbf{r}}, L_{\mathbf{a}})$-partial relation with probability

$$\sigma_{L_{\mathbf{r}}}(u_{\mathbf{r}}, v_{\mathbf{r}}) \cdot \sigma_{L_{\mathbf{a}}}(u_{\mathbf{a}}, v_{\mathbf{a}}).$$

Integration of these probabilities over the sieving region gives an estimate for the total yield of $(L_{\mathbf{r}}, L_{\mathbf{a}})$-partial relations. An estimate for $\#V$ in the runtime of Coppersmith's variant is obtained by integrating the $\sigma_{L_{\mathbf{r}}}(u_{\mathbf{r}}, v_{\mathbf{r}})$ values over the sieving region. Similar integrations are used to compute candidate frequencies in Appendix B. A correction factor $6/\pi^2 \approx 0.608$ is applied to all results to account for the probability that a and b are coprime. The integrations were carried out using *Mathematica* and the *GNU Scientific Library*.

Actual smoothness tests. To get an impression of the accuracy of the above ρ and σ_1-based estimates compared to the actual NFS yield, we tested $N_{\mathbf{r}}(a,b)$ and $N_{\mathbf{a}}(a,b)$-values for smoothness for wide ranges of (a,b) pairs. Because it has never been doubted that the probability that $N_{\mathbf{r}}(a,b)$ and $N_{\mathbf{a}}(a,b)$ are smooth equals the product of the smoothness probabilities, we did not test that assumption.

We had no access to a siever that allows the range of factor base sizes we intended to test, nor to hardware on which it would be able to run efficiently. Therefore we wrote a smoothness test that uses trial division up to 2^{30} combined with the elliptic curve factoring method (ECM). The choice 2^{30} was partially inspired by our wish not to miss any semi-smooth $N_{\mathbf{r}}(a,b)$ or $N_{\mathbf{a}}(a,b)$-values that would, in theory, be found when using one of the parameter choices from [17].

The simplest approach would have been to subject each successive number to be tested to trial division followed, if necessary, by the ECM. To obtain slightly greater speed, and without having to deal with the imperfections (overlooking smooth values) and inconveniences (memory requirements, resieving or trial divisions to obtain the cofactor) of sieving, the trial divisions were organized in such a way that a large consecutive range of a's could be handled reasonably efficiently, for a fixed b. For the algebraic norms this was achieved as follows (the rational norms are processed similarly). Let $[A_1, A_2]$ be a range of a-values to be processed. For all (prime,root) pairs (p,r) with $p < 2^{30}$ calculate the smallest $a_p \geq A_1$ such that $a_p \equiv br \bmod p$ (i.e., p divides $N_{\mathbf{a}}(a_p, b)$) and if $a_p \leq A_2$ insert the pair (p, a_p) in a heap that is ordered with respect to non-decreasing a_p values. Next, for $a = A_1, A_1+1, \ldots, A_2$ in succession compute $c_a = N_{\mathbf{a}}(a,b)$, remove all elements with $a_p = a$ from the top of the heap, remove all corresponding factors p from c_a, and if $a_p + p \leq A_2$ insert $(p, a_p + p)$ in the heap. Note that this can be seen as a variant of the 'largish station' design from [18]. The resulting c_a values have no factors $< 2^{30}$, are prime if $< 2^{60}$, and subjected to the ECM if composite. Due to the probabilistic nature of the ECM, factors between 2^{30} and the smoothness bound $y_{\mathbf{a}}$ (or $y_{\mathbf{r}}$) may be overlooked. With proper ECM parameter settings and reasonably sized $y_{\mathbf{a}}$ (and $y_{\mathbf{r}}$) this does not occur often. Furthermore, no relation relevant for the primary choice in [17] will be overlooked.

4 Traditional extrapolation

In this section we sketch the traditional approach to estimate the difficulty of factoring a 1024-bit RSA modulus. Let R indicate a resource required for a factorization effort. For instance, R could indicate the computing time or it could be the factor base size, or the total matrix weight, or any other aspect of the factorization for which one wants to measure the cost or size.

For each resource R let $C_R(x)$ be a function that measures, asymptotically for $x \to \infty$ and in the relevant unit, how much of R is needed to factor x. For several resources a theoretical expression for this function is known. For instance, when R measures the total expected computing time, then

$$C_R(x) \approx L_x[1/3, (64/9)^{1/3}],$$

with $L_x[,]$ as in Section 3. If R measures the factor base size the constant $(64/9)^{1/3}$ in this expression would, in theory, be halved.

Assume that $R_{n'}$ units of some resource R are known to be required (or were used) to factor some RSA modulus n'. Then $\frac{C_R(n)}{C_R(n')} R_{n'}$ is used to estimate how much of R would be required (or feasible) for the factorization of RSA modulus n. In this type of estimate it is customary to ignore all $o(1)$'s, if they occur in C_R. Based on frequent observations this is not unreasonable if $\log(n')$ and $\log(n)$ are close. For large scale extrapolations, however, omitting the $o(1)$'s may be an over-simplification that might produce misleading results.

Furthermore, even if $\log(n')$ and $\log(n)$ are close, C_R-based extrapolation for resources R that are well understood in theory, may lead to results that have no practical value. As an example, for a 512-bit factorization, e.g. RSA-155, one would recommend a factor base size that is about 2.5 times larger than for a 462-bit factorization (as RSA-140). In practice, however, the entire concept of factor base size is obscured by the use of multiple large primes and special q's: it turned out that using the same factor base size did not lead to severe performance degradation.

This particular effect that not-even-nearly-optimal factor base sizes still lead to only slightly suboptimal performance is due to the behavior around the minimum of the runtime curve as a function of the factor base size: the runtime only gradually increases for factor base sizes that are much larger or somewhat smaller than the optimum. On the other hand, it increases sharply if the factor base size gets much too small (cf. [20]). This explains the potential dangers of $o(1)$-less factor base size extrapolation: a suboptimal small choice, in the region where the curve is relatively well behaved, for the factor base for n' may extrapolate to a factor base size for n in the steep region of the curve, thereby leading to a much larger total runtime for n than anticipated; see also Section 5, Table 2.

It is not uncommon to use $n' =$ RSA-155 (a 512-bit number) as the basis for the extrapolation. In [11] the following parameters were proposed for 512-bit numbers (in the notation of Section 2), which is close to the values used for the factorization of RSA-155 (cf. [4]):

512-bit moduli: $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{24}$, sieving region of size $S = 1.6\text{E}16$ ($A = 9\text{E}9$, $B = 9\text{E}5$; we use '$v\text{E}w$' for '$v \cdot 10^w$'). According to [17] the sieving step can be done in less than ten minutes on a US$10K device.

Straightforward ($o(1)$-less) extrapolation suggests that 768 and 1024-bit moduli would require smoothness bounds that are 75 and 2700 times larger and sieving regions that are 6000 and 7.5E6 times larger, respectively: smoothness bounds approximately 2^{30} and 2^{35} and $S \approx 1\text{E}20$ and $S \approx 1.2\text{E}23$, respectively. As shown in [12] additional optimization arguments may enter into and further complicate the extrapolation. In [17] this leads to relatively small estimates for the smoothness bounds and relatively large sieving regions:

768-bit moduli: $y_{\mathbf{r}} = y_{\mathbf{a}} = 1.2\text{E}7$ ($< 2^{24}$), $S = 4.2\text{E}20$ ($A = 1.5\text{E}12$, $B = 1.5\text{E}8$). The sieving step can be done within 70 days on a US$5K device.
1024-bit moduli: $y_{\mathbf{r}} = y_{\mathbf{a}} = 2.5\text{E}8$ ($< 2^{28}$), $S = 6\text{E}23$ ($A = 5.5\text{E}13$, $B = 5.5\text{E}9$). The sieving step takes a year on a US$10M device.

Furthermore, the following is given in [17] and claimed to be an overestimate based on traditional extrapolation:

1024-bit moduli, but not using partial relations: $y_{\mathbf{r}} = y_{\mathbf{a}} = 1.5\text{E}10$ ($< 2^{34}$), $S = 6\text{E}23$. The sieving step takes a year on a US$50M device.

5 Results

Let the notation be as in Section 2. In this section we present our ρ and σ_1-based estimates for the yield of the NFS when applied to RSA-155 and RSA-768 with the parameters as suggested in [17] (and specified in Section 4) and to RSA-1024 for a wide variety of parameters, including those from [17]. Furthermore, we compare the estimates to the results of smoothness tests applied to numbers that would occur in an actual NFS factorization attempt. In Appendix B we give the corresponding estimates for RSA-1024 and the parameter choices from [18].

512-bit moduli. Let n = RSA-155, $d = 5$, f as in [4], $s = 10800$, and $t = \exp(5.3)$. Application of our ρ and σ_1-based estimates to $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{24}$, $z_{\mathbf{r}} = z_{\mathbf{a}} = 2^6 y_{\mathbf{r}} = 2^{30}$, $A = 9\text{E}9$, and $B = 9\text{E}5$ result in an estimated yield of $T(y_{\mathbf{r}}, y_{\mathbf{a}})/8.9 \approx 2.4\text{E}5$ ***ff***'s, 2.2E6 ***fp***'s, 9.1E5 ***pf***'s, and 8.1E6 ***pp***'s. Because the parameter choice was intended for the use of more than a single large prime per norm, these results look acceptable: if more than one tenth of the matrix is filled with ***ff***'s, combinations of multi-prime partial relations will certainly fill in the rest.

With $y_{\mathbf{r}} = 2^{29}$, $y_{\mathbf{a}} = 2^{30}$, and $B = 4.0\text{E}4$ the same fraction of the matrix would be filled with ***ff***'s for a sieving effort that is more than 470 times lower, but $T(y_{\mathbf{r}}, y_{\mathbf{a}})$ would be 38.4 times larger, and sieving would have required more fast RAM than was available in 1999. Because $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{24}$ is much smaller than the choice that would minimize the sieving effort, extrapolation may result in very large sieving efforts, as mentioned in Section 4. See also Table 2 below.

768-bit moduli. For n = RSA-768 we generated a fifth degree polynomial with $s \approx 26000$ and $t \approx \exp(5.3)$. To get $S = 4.2\text{E}20$, we use $B = 9\text{E}7$ and $A = sB$. With $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{24}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) = 2.1\text{E}6$, and $z_{\mathbf{r}} = z_{\mathbf{a}} = 2^{10}y_{\mathbf{r}} = 2^{34}$ we estimate a yield of fewer than 40 ***ff***'s, 1200 ***fp***'s, 500 ***pf***'s, and 2E4 ***pp***'s. It is unlikely that this is feasible, unless a substantial effort is spent on finding multi-prime partial relations. With $y_{\mathbf{r}} = 2^{29}$, $y_{\mathbf{a}} = 2^{30}$, and the same sieving region, about $T(y_{\mathbf{r}}, y_{\mathbf{a}})/16 \approx 5.2\text{E}6$ ***ff***'s can be expected. With reasonable use of partial relations this may be feasible.

1024-bit moduli. For n = RSA-1024 we considered degrees $d = 5, 6, 7, 8, 9$, each with corresponding integer m, d-th degree polynomial f, skewness ratio s, and correction factor t as specified in Appendix A. For each of these degrees and $S = 6\text{E}23$ the estimated yield figures are presented in the first two parts of Table 1, both for $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{28}$ and $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$. Because the skewness ratio s depends on d, the height $B = \sqrt{S/(2s)}$ and width $2A = 2sB$ of the sieving region depend on d. In the last two parts the effect is given of doubling and quadrupling B, thereby increasing S (and the sieving effort) by a factor 4 and 16, respectively (since the skewness ratio s is kept invariant). We used $z_{\mathbf{r}} = z_{\mathbf{a}} = 2^j y_{\mathbf{r}}$ for $j \in \{8, 12, 16\}$ and indicate the expected ***fp***, ***pf***, and ***pp*** yield by ***fp***$_j$, ***pf***$_j$, and ***pp***$_j$, respectively. Note that [17] does not use partial relations for $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$. It follows from Table 1 that unless multi-prime partial relations are collected

Table 1. Estimated yields for smoothness bounds from [17].

d	s	B	***ff***	***fp***$_8$	***pf***$_8$	***pp***$_8$	***fp***$_{12}$	***pf***$_{12}$	***pp***$_{12}$	***fp***$_{16}$	***pf***$_{16}$	***pp***$_{16}$
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{28}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 2.9\text{E}7$, $S = 6\text{E}23$, sieving effort 3.6E24												
5	87281.9	1.9E9	22	4.7E2	2.3E2	5.1E3	9.2E2	4.3E2	1.8E4	1.6E3	6.9E2	5.1E4
6	458.9	2.6E10	74	1.7E3	6.3E2	1.4E4	3.3E3	1.1E3	5.0E4	5.8E3	1.8E3	1.4E5
7	40.9	8.6E10	1.5E2	3.6E3	1.0E3	2.4E4	6.9E3	1.8E3	8.1E4	1.2E4	2.8E3	2.2E5
8	107.3	5.3E10	34	8.2E2	1.8E2	4.5E3	1.6E3	3.2E2	1.5E4	2.8E3	4.8E2	4.0E4
9	8.5	1.9E11	3	69	14	2.5E2	1.3E2	24	8.2E2	1.8E2	37	2.2E3
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 1.5\text{E}9$, $S = 6\text{E}23$, sieving effort 3.8E24												
5	87281.9	1.9E9	9.1E6	1.1E8	5.6E7	6.9E8	2.0E8	9.5E7	2.1E9	3.3E8	1.5E8	5.2E9
6	458.9	2.6E10	2.1E7	2.8E8	1.0E8	1.4E9	5.1E8	1.7E8	4.1E9	8.2E8	2.6E8	1.0E10
7	40.9	8.6E10	3.1E7	4.3E8	1.2E8	1.7E9	7.7E8	2.0E8	5.0E9	1.3E9	2.9E8	1.2E10
8	107.3	5.3E10	6.8E6	1.0E8	2.2E7	3.3E8	1.9E8	3.6E7	9.9E8	3.1E8	5.2E7	2.4E9
9	8.5	1.9E11	5.3E5	8.5E6	1.5E6	2.5E7	1.6E7	2.5E6	7.3E7	2.6E7	3.6E6	1.8E8
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 1.5\text{E}9$, $S = 2.4\text{E}24$, sieving effort 1.5E25												
5	87281.9	3.7E9	1.9E7	2.4E8	1.2E8	1.5E9	4.4E8	2.0E8	4.6E9	7.1E8	3.1E8	1.1E10
6	458.9	5.1E10	4.0E7	5.6E8	2.0E8	2.7E9	1.0E9	3.3E8	8.1E9	1.6E9	5.0E8	2.0E10
7	40.9	1.7E11	5.2E7	7.4E8	2.0E8	2.9E9	1.3E9	3.3E8	8.7E9	2.2E9	5.0E8	2.1E10
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 1.5\text{E}9$, $S = 9.8\text{E}24$, sieving effort 6.1E25												
5	87281.9	7.4E9	4.1E7	5.3E8	2.5E8	3.3E9	9.5E8	4.3E8	1.0E10	1.5E9	6.6E8	2.5E10
6	458.9	1.0E11	7.5E7	1.0E9	3.7E8	5.2E9	2.0E9	6.3E8	1.6E10	3.2E9	9.5E8	3.9E10
7	40.9	3.4E11	8.6E7	1.3E9	3.4E8	5.0E9	2.3E9	5.7E8	1.5E10	3.8E9	8.4E8	3.7E10

on a much wider scale than customary or practical, the choice $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{28}$, and thus the smaller choice $y_{\mathbf{r}} = y_{\mathbf{a}} = 2.5\text{E}8$ from [17], looks infeasible. Also the choice $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, and therefore the choice $y_{\mathbf{r}} = y_{\mathbf{a}} = 1.5\text{E}10$ from [17], is infeasible if, as suggested in [17], partial relations are not used and if a sieving region size S as proposed in [17] is used. To get the choice $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$ to work without partial relations, our estimates suggest that $d = 6$ with $B \approx 2.9\text{E}12$

(corresponding to $S \approx 8\text{E}27$) would suffice. This would, however, be about 13000 times more expensive than the estimate from [17]: the initial 2.6E10 b-values produce about $T(y_{\mathbf{r}}, y_{\mathbf{a}})/72$ ***ff***'s, but the performance deteriorates for larger b's so that much more than 72 times the initial effort is needed to find $T(y_{\mathbf{r}}, y_{\mathbf{a}})$ ***ff***'s. For $d = 5$ or 7 it would be 1.1 or 3.5 times more expensive, respectively.

Using partial relations is probably a more efficient way to get $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$ to work, as suggested by the last two parts of Table 1. Since there are no adequate methods yet to predict if the partial relation yield as listed, in practice augmented with partial relations with 3 or more large primes, would suffice or not, we cannot make any definite statements on the resulting cost, the practical merit of the cost estimate from [17], or the semi-smoothness bound that would be required. Note that the performance of $d = 6, 7$ deteriorates faster than for $d = 5$, as expected.

In Table 2 the effect of low smoothness bounds is illustrated. The total expected sieving effort to find $T(y_{\mathbf{r}}, y_{\mathbf{a}})/32$ ***ff***'s is listed for $d = 6$, $y_{\mathbf{r}} = 2^j$ with $j = 28, 29, \ldots, 51$ and $y_{\mathbf{a}} = 2y_{\mathbf{r}}$. The optimum 9.3E20 is achieved at $j = 47$. When j gets smaller the effort at first increases slowly and gradually, but around $j = 39$ the effort grows faster than the smoothness bounds shrink, and for smaller j the performance deteriorates rapidly.

Table 2. Sieving effort to find $T(2^j, 2^{j+1})/32$ ***ff***'s for $d = 6$.

j	effort	j	effort	j	effort	j	effort	j	effort	j	effort
28	1.5E36	32	5.6E26	36	1.7E23	40	4.8E21	44	1.2E21	48	9.6E20
29	4.7E32	33	3.7E25	37	5.2E22	41	2.9E21	45	1.0E21	49	1.0E21
30	1.4E30	34	4.2E24	38	2.0E22	42	2.0E21	46	9.4E20	50	1.2E21
31	1.7E28	35	7.2E23	39	9.1E21	43	1.5E21	47	9.3E20	51	1.4E21

We now vary d and $i_{\mathbf{r}}, i_{\mathbf{a}} \in \{25, 26, \ldots, 50\}$ and minimize the sieving effort to find $T(2^{i_{\mathbf{r}}}, 2^{i_{\mathbf{a}}})/c$ ***ff***'s, for various c's. The resulting sieving efforts with corresponding optimal smoothness bounds are listed in Table 3. It can be seen that both effort and smoothness bounds decrease with increasing c. This effect is stronger for larger d. Overall, $d = 7$ is the best choice, with $d = 6$ better than $d = 8$ for small c but vice versa for larger ones. For non-optimal smoothness bounds, however, $d = 7$ may not be the best choice, as illustrated in Table 1.

Table 3. Minimal sieving efforts to find $T(2^{i_{\mathbf{r}}}, 2^{i_{\mathbf{a}}})/c$ ***ff***'s.

	$c = 1$		$c = 8$		$c = 16$		$c = 32$		$c = 64$		$c = 128$	
d	$i_{\mathbf{r}}, i_{\mathbf{a}}$	effort	$i_{\mathbf{r}}, i_{\mathbf{a}}$	effort	$i_{\mathbf{r}}, i_{\mathbf{a}}$	effort	$i_{\mathbf{r}}, i_{\mathbf{a}}$	effort	$i_{\mathbf{r}}, i_{\mathbf{a}}$	effort	$i_{\mathbf{r}}, i_{\mathbf{a}}$	effort
6	48,49	1.6E23	47,48	7.2E21	47,48	2.6E21	47,48	9.2E20	47,48	3.3E20	46,47	1.2E20
7	47,49	9.4E22	47,49	3.5E21	46,48	1.1E21	46,47	3.5E20	45,47	1.1E20	45,46	3.6E19
8	48,50	3.7E23	47,49	1.0E22	46,48	3.0E21	46,48	8.7E20	45,47	2.5E20	45,47	7.5E19

Actual smoothness tests for RSA-1024. The accuracy of our ρ and σ_1-based estimates as derived for n = RSA-1024 was tested by applying smoothness tests (as explained in Section 3) to $N_{\mathbf{r}}(a, b)$ and $N_{\mathbf{a}}(a, b)$-values for wide ranges of

(a, b)-pairs with coprime a and b and degrees and parameters as in Appendix A. More than 100 billion values have been tested for degrees 6 and 7. No major surprises or unexpected anomalies were detected. Thus, although it may be too early to have complete confidence in the ρ and σ_1-based estimates, there is neither any reason to dismiss them.

For $d = 6$ this is illustrated in Tables 4, 5, and 6. Tables 4 and 5 contain the accumulated results of smoothness tests for $N_{\mathbf{r}}(a, b)$ and $N_{\mathbf{a}}(a, b)$-values, respectively, for more than 100 billion coprime (a, b) pairs and 176 different b values ranging from 2^9 to 2^{31}. They list the number of $(2^i, 2^j, 1)$-semi-smooth $N_{\mathbf{r}}(a, b)$ and $N_{\mathbf{a}}(a, b)$-values (for i, j ranges as specified in the tables) that were found using trial division up to 2^{30}, followed by the $(\rho + \sigma_1)$-based estimate between parentheses. Table 6 contains the accumulated results of more expensive smoothness tests for $N_{\mathbf{a}}(a, b)$-values for 5.6 million coprime (a, b) pairs and 13 different b-values ranging from 2^{14} to 2^{26}. For $34 \leq j \leq 40$ and $31 \leq i \leq j$ it lists the number of $(2^i, 2^j, 1)$-semi-smooth $N_{\mathbf{a}}(a, b)$-values, found using trial division up to 2^{30} followed by ECM, again followed by the $(\rho+\sigma_1)$-based estimate between parentheses. The fact that the estimated value is systematically somewhat higher than the actual value can be attributed to the fact that the estimated values average over all positive numbers less than some bound, whereas most values that are actually tested are close to the bound. This is partly offset by the use of asymptotic smoothness probabilities, which are somewhat smaller than the concrete probabilities (e.g., for $\rho(u_{\mathbf{r}})$ the correction term is roughly $+0.423\rho(v_{\mathbf{r}} - 1)/\log N_{\mathbf{r}}(a, b)$; cf. [1]).

Table 4. Actual and estimated number of $(2^i, 2^j, 1)$-semi-smooth $N_{\mathbf{r}}(a, b)$'s for $d = 6$.

j	i						
	24	25	26	27	28	29	30
24	2.4E3(2.7E3)						
25	4.9E3(5.5E3)	6.3E3(7.0E3)					
26	7.7E3(8.6E3)	1.2E4(1.4E4)	1.5E4(1.7E4)				
27	1.1E4(1.2E4)	1.9E4(2.1E4)	2.8E4(3.1E4)	3.4E4(3.7E4)			
28	1.4E4(1.6E4)	2.6E4(2.9E4)	4.3E4(4.7E4)	6.1E4(6.7E4)	7.1E4(7.7E4)		
29	1.8E4(2.0E4)	3.4E4(3.7E4)	5.8E4(6.4E4)	8.9E4(9.8E4)	1.2E5(1.3E5)	1.4E5(1.5E5)	
30	2.2E4(2.4E4)	4.2E4(4.7E4)	7.5E5(8.2E4)	1.2E5(1.3E5)	1.8E5(1.9E5)	2.3E5(2.5E5)	2.5E5(2.7E5)
31	2.6E4(2.9E4)	5.1E4(5.7E4)	9.3E4(1.0E5)	1.5E5(1.7E5)	2.4E5(2.6E5)	3.3E5(3.6E5)	3.8E5(4.1E5)
32	3.1E4(3.4E4)	6.1E4(6.8E4)	1.1E5(1.2E5)	1.9E5(2.1E5)	3.0E5(3.2E5)	4.3E5(4.7E5)	5.1E5(5.5E5)
33	3.6E4(4.0E4)	7.2E4(8.0E4)	1.3E5(1.5E5)	2.3E5(2.5E5)	3.7E5(4.0E5)	5.5E5(5.9E5)	6.5E5(7.1E5)
34	4.2E4(4.7E4)	8.4E4(9.3E4)	1.6E5(1.7E5)	2.7E5(3.0E5)	4.4E5(4.8E5)	6.7E5(7.2E5)	8.0E5(8.7E5)
35	4.8E4(5.4E4)	9.7E4(1.1E5)	1.8E5(2.0E5)	3.2E5(3.5E5)	5.2E5(5.6E5)	8.0E5(8.6E5)	9.7E5(1.0E6)
36	5.5E4(6.1E4)	1.1E5(1.2E5)	2.1E5(2.3E5)	3.6E5(4.0E5)	6.0E5(6.5E5)	9.3E5(1.0E6)	1.1E6(1.2E6)
37	6.3E4(7.0E4)	1.3E5(1.4E5)	2.4E5(2.6E5)	4.2E5(4.6E5)	6.9E9(7.5E5)	1.1E6(1.2E6)	1.3E6(1.4E6)
38	7.1E4(7.9E4)	1.4E5(1.6E5)	2.7E5(3.0E5)	4.7E5(5.2E5)	7.8E5(8.5E5)	1.2E6(1.3E6)	1.5E6(1.6E6)
39	8.1E4(9.0E4)	1.6E5(1.8E5)	3.0E5(3.3E5)	5.3E5(5.8E5)	8.9E5(9.7E5)	1.4E6(1.5E6)	1.7E6(1.9E6)
40	9.1E4(1.0E5)	1.8E5(2.0E5)	3.4E5(3.8E5)	6.0E5(6.6E5)	1.0E6(1.1E6)	1.6E6(1.7E6)	1.9E6(2.1E6)

For $d = 7$ we found comparable results. Because of the asymptotic nature of the estimates, it may be expected that they become even more accurate for the larger b's that may occur in practice (cf. Table 1).

Table 5. Actual and estimated number of $(2^i, 2^j, 1)$-semi-smooth $N_{\mathbf{a}}(a,b)$'s for $d = 6$.

j	i						
	24	25	26	27	28	29	30
28	0(0.15)	0(0.41)	0(0.96)	0(1.85)	0(2.53)		
29	0(0.19)	1(0.54)	1(1.36)	1(2.94)	1(5.32)	1(7.01)	
30	0(0.23)	1(0.69)	1(1.80)	1(4.14)	1(8.34)	5(14.18)	10(16.87)
31	0(0.29)	1(0.86)	2(2.28)	2(5.45)	5(11.63)	17(21.94)	24(28.52)
32	1(0.34)	2(1.04)	3(2.81)	3(6.88)	8(15.21)	27(30.34)	40(41.10)
33	1(0.41)	2(1.24)	5(3.40)	5(8.45)	12(19.11)	39(39.44)	58(54.70)
34	1(0.48)	2(1.47)	5(4.05)	5(10.17)	15(23.36)	49(49.31)	70(69.41)
35	1(0.56)	2(1.72)	6(4.76)	7(12.05)	21(28.00)	60(60.01)	82(85.33)
36	1(0.65)	2(2.00)	7(5.55)	10(14.12)	27(33.05)	71(71.63)	97(102.57)
37	1(0.75)	2(2.30)	8(6.42)	11(16.39)	31(38.58)	82(84.26)	111(121.26)
38	2(0.86)	3(2.65)	9(7.38)	12(18.88)	36(44.61)	95(97.98)	132(141.52)
39	2(0.99)	3(3.03)	10(8.45)	14(21.62)	41(51.20)	106(112.90)	148(163.51)
40	2(1.13)	3(3.46)	11(9.62)	19(24.63)	47(58.41)	115(129.13)	163(187.36)

Table 6. Actual and estimated number of $(2^i, 2^j, 1)$-semi-smooth $N_{\mathbf{a}}(a,b)$'s for $d = 6$.

j	i									
	31	32	33	34	35	36	37	38	39	40
34	0(0.30)	0(0.49)	0(0.70)	0(0.82)						
35	0(0.39)	0(0.66)	0(1.03)	1(1.41)	1(1.62)					
36	0(0.48)	0(0.85)	0(1.38)	1(2.05)	1(2.73)	1(3.08)				
37	0(0.58)	0(1.05)	0(1.75)	2(2.72)	2(3.90)	2(5.03)	2(5.60)			
38	0(0.69)	0(1.26)	0(2.15)	2(3.44)	3(5.14)	4(7.11)	5(8.95)	5(9.84)		
39	1(0.81)	1(1.49)	1(2.58)	3(4.21)	4(6.46)	8(9.30)	9(12.48)	12(15.36)	13(16.72)	
40	1(0.93)	1(1.74)	1(3.04)	4(5.02)	6(7.86)	12(11.62)	15(16.20)	18(21.16)	21(25.51)	23(27.52)

6 More or better polynomials?

Estimating the performance of Coppersmith's variant. We estimated the yield and performance of Coppersmith's multi-polynomial version of the NFS by assuming that for any degree d we can find a set G of any reasonable cardinality consisting of degree d polynomials with a shared root m modulo n and with skewness ratios and correction factors comparable to those in Appendix A. Table 7 lists some estimates for $d = 6, 7$ and $\#G = 6$ that can be compared to the estimates in Table 1. The dimension of the matrix increases 7/2-fold and the yield improves by a factor 6. The ***fp*** and ***pp*** yield increase may not be that effective, since large primes match only if they occur in the norm of the same polynomial. The relation collection effort changes from sieving effort 3.8E24 to sieving effort 1.9E24 plus a number of semi-smoothness tests (indicated by 'ECM effort') involving a constant of proportionality E measuring the relative performance compared to sieving.

The practical implications are as yet unclear. For current implementations E would be too large to make the multi-polynomial version competitive, but an entirely different picture may emerge for dedicated non-sieving hardware smoothness tests. Also, our choices $d = 6, 7$ and $\#G = 6$ were not meant to optimize anything, they are just for illustrative purposes to facilitate comparison with the regular NFS data in Table 1. Clearly, this subject deserves further study.

Table 7. Estimated yields for smoothness bounds from [17] with 6 polynomials.

d	s	B	ff	fp_8	pf_8	pp_8	ECM effort	fp_{12}	pf_{12}	pp_{12}	ECM effort
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, goal $\approx$ 5.3E9, S = 6E23, sieving effort 1.9E24											
6	458.9	2.6E10	1.3E8	1.7E9	6.2E8	8.2E9	E5.6E20	3.0E9	1.0E9	2.5E10	E8.7E20
7	40.9	8.6E10	1.8E8	2.6E9	7.1E8	9.9E9	E3.6E21	4.6E9	1.2E9	3.0E10	E5.4E21

The effect of much better polynomials. In an actual factorization attempt considerably more time would be spent to find good polynomials. So, in practice, we may expect correction factors t that are larger than the ones given in Appendix A for polynomials which may have smaller coefficients. An example of such a polynomial is given in Appendix B. This effect can be approximated by applying our estimates to the same f and m values but with incorrect (too large) correction factors t. In Table 8 the results are given if t is replaced by t^3 for $d = 6, 7$, with parameters as in Table 1 (i.e., mostly as in [17]). With the current state of the art of polynomial selection methods it is unlikely that such large correction factors can be found in practice. Thus, the figures in Table 8 are probably too optimistic. Compared to Table 1 the yield improves by a factor about 3: a relatively small effect that does not have an impact on the observations made in Section 5 about $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{28}$ and $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$. For $d = 6$ and $y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$ not using partial relations (and correction factor t^3) would require B = 9.4E11 with corresponding S = 8.2E26. This is about 1300 times more expensive than the estimate from [17]. We conclude that our limited polynomial search did not lead to overly poor estimates.

Table 8. Estimated yields for smoothness bounds from [17] with correction factor t^3.

d	s	B	ff	fp_8	pf_8	pp_8	fp_{12}	pf_{12}	pp_{12}	fp_{16}	pf_{16}	pp_{16}
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{28}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx$ 2.9E7, S = 6E23, sieving effort 3.6E24												
6	458.9	2.6E10	2.6E2	5.8E3	2.2E3	4.9E4	1.1E4	4.0E3	1.7E5	2.0E4	6.3E3	4.7E5
7	40.9	8.6E10	6.8E2	1.5E4	4.6E3	1.0E5	2.9E4	8.0E3	3.5E5	5.1E4	1.2E4	9.5E5
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx$ 1.5E9, S = 6E23, sieving effort 3.8E24												
6	458.9	2.6E10	5.7E7	7.2E8	2.8E8	3.5E9	1.3E9	4.9E8	1.1E10	2.1E9	7.3E8	2.6E10
7	40.9	8.6E10	9.9E7	1.3E9	3.9E8	5.1E9	2.4E9	6.4E8	1.5E10	3.9E9	9.4E8	3.7E10
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx$ 1.5E9, S = 2.4E24, sieving effort 1.5E25												
6	458.9	5.1E10	1.1E8	1.4E9	5.3E8	6.9E9	2.5E9	8.9E8	2.1E10	4.1E9	1.3E9	5.1E10
7	40.9	1.7E11	1.7E8	2.3E9	6.6E8	9.1E9	4.2E9	1.1E9	2.7E10	6.8E9	1.6E9	6.5E10
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx$ 1.5E9, S = 9.8E24, sieving effort 6.1E25												
6	458.9	1.0E11	2.0E8	2.8E9	1.0E9	1.4E10	4.9E9	1.7E9	4.1E10	8.0E9	2.6E9	1.0E11
7	40.9	3.4E11	2.8E8	4.0E9	1.1E9	1.6E10	7.3E9	1.9E9	4.8E10	1.2E10	2.8E9	1.2E11

7 Conclusion

We applied numerical methods to estimate the yield of the NFS when applied to the 1024-bit RSA modulus RSA-1024, and tested the accuracy of our results using actual smoothness tests. Our methods and results were taken into account in the updated version [18] of the draft version of TWIRL [17] and are presented in Appendix B. Accurate estimates of the difficulty of factoring 1024-bit RSA moduli require a better understanding of the large prime matching behavior than is available today. Continued large factorization efforts may prove helpful.

Our results suggest that effective smoothness bounds for RSA-1024 are larger than the ones proposed in [17]. Larger smoothness bounds stress the importance of the alternative cost measure proposed in [2] and of approaches to smoothness testing that avoid sieving and storage of the complete factor bases. TWINKLE and TWIRL (cf. [19], [18]) both require processing elements or storage for essentially the complete factor bases and time for the sieving. Such designs may eventually be surpassed by, say, a carefully designed ECM-based smoothness test as proposed in [2], because the latter allows a better trade-off between space and time. This does not disqualify TWIRL for the sizes proposed in [18], but indicates that in the long term the approach from [2] may be more promising.

Acknowledgment. We thank Mike Szydlo for useful discussions, and for sharing his observations about [17]. We are grateful to Thorsten Kleinjung and Jens Franke for their polynomial selection program and subsequent discussions.

References

1. E. Bach, R. Peralta, *Asymptotic semi-smoothness probabilities*, University of Wisconsin, Technical report #1115, October 1992
2. D.J. Bernstein, *Circuits for integer factorization: a proposal*, manuscript, November 2001; available at `http://cr.yp.to/papers.html#nfscircuit`
3. E.R. Canfield, P. Erdös, C. Pomerance, *On a problem of Oppenheim concerning "Factorisatio Numerorum"*, J. Number Theory **17** (1983) 1–28
4. S. Cavallar, B. Dodson, A.K. Lenstra, W. Lioen, P.L. Montgomery, B. Murphy, H.J.J. te Riele, et al., *Factorization of a 512-bit RSA modulus*, Proceedings Eurocrypt 2000, LNCS 1807, Springer-Verlag 2000, 1–17
5. D. Coppersmith, *Modifications to the number field sieve*, Journal of Cryptology **6** (1993) 169–180
6. R. Crandall, C. Pomerance, *Prime numbers*, Springer-Verlag, 2001
7. N.G. De Bruijn, *On the number of positive integers $\leq x$ and free of prime factors $> y$, II*, Indag. Math. **38** (1966) 239–247
8. International Technology Roadmap for Semiconductors 2002 Update, `http://public.itrs.net/`
9. R. Lambert, *Computational aspects of discrete logarithms*, Ph.D. thesis, University of Waterloo, 1996.
10. A.K. Lenstra, H.W. Lenstra, Jr., (eds.), *The development of the number field sieve*, Lecture Notes in Math. **1554**, Springer-Verlag 1993
11. A.K. Lenstra, A. Shamir, *Analysis and optimization of the TWINKLE factoring device*, Proceedings Eurocrypt 2000, LNCS 1807, Springer-Verlag 2000, 35–52
12. A.K. Lenstra, A. Shamir, J. Tomlinson, E. Tromer, *Analysis of Bernstein's factorization circuit*, Proceedings Asiacrypt 2002, LNCS 2501, Springer-Verlag 2002, 1–26
13. P.L. Montgomery, B. Murphy, *Improved polynomial selection for the number field sieve*, extended abstract for the conference on the mathematics of public-key cryptography, June 13-17, 1999, The Fields institute, Toronto, Ontario, Canada

14. B. Murphy, *Modelling the yield of the number field sieve polynomials*, Proceedings ANTS-III, LNCS 1423, Springer-Verlag, 1998, 137–151
15. B. Murphy, *Polynomial selection for the number field sieve integer factorisation algorithm*, PhD thesis, The Australian National University, July 1999
16. RSA Challenge Administrator, see `http://www.rsasecurity.com/rsalabs/challenges/factoring/index.html`
17. A. Shamir, E. Tromer, *Factoring large numbers with the TWIRL device (preliminary draft)*, February 4, 2003; available at `www.wisdom.weizmann.ac.il/~tromer/papers/twirl-20030208.ps.gz`
18. A. Shamir, E. Tromer, *Factoring large numbers with the TWIRL device*, Proceedings Crypto 2003, LNCS 2729, Springer-Verlag 2003, 1–26
19. A. Shamir, *Factoring large numbers with the TWINKLE device*, Proceedings CHES'99, LNCS 1717, Springer-Verlag, 1999
20. R.D. Silverman, *Optimal parameterization of SNFS*, Manuscript, 2002

A Polynomials for RSA-1024

Let the notation be as in Section 2. RSA-1024 $= 135\ldots563$ is a 1024-bit number whose 309 decimal digits can be found in [16]. For $d = 5, 6, 7, 8, 9$ we present the value of m, the skewness ratio s, the correction factor t, and the d-th degree polynomial f. For all d we have that $f(m) =$ RSA-1024 and the number of free relations behaves as estimated in Section 2.

$d = 5$**:** $m = 4016606149940576776127592250520584531962067322396239426984 8,$

$$\begin{aligned}
d = 5\text{:}\quad & m = 40166061499405767761275922505205845319620673223962394269848, \\
& s = 87281.9,\ t = \exp(4.71), \\
& f(X) = 1291966090228800X^5 - 640923572655549773652421X^4 \\
&\qquad + 22084609569698872827347541432045436154518749958885X^3 \\
&\qquad + 395968894120701874630226095753546547718334332711719805X^2 \\
&\qquad - 969659739570663862858360422925321994203407742793583219578 26X \\
&\qquad - 4149238485198657863882627412883817567549615187136520422680871493. \\
d = 6\text{:}\quad & m = 629042860635589902725572332002739171597034508 8070,\ s = 458.857,\ t = \exp(3.10), \\
& f(X) = 2180047385355840X^6 - 3142872579455569636X^5 \\
&\qquad - 1254155662796860036208992514969847001569768X^4 \\
&\qquad - 1234618459668212931188535497431179367033899 9X^3 \\
&\qquad + 326853630498301587526877377811152784944999520522X^2 \\
&\qquad + 4609395911122979440239635705733809071478223546768X \\
&\qquad - 110746927687582599679550175816747063649255199965909 97. \\
d = 7\text{:}\quad & m = 1039002975678183603195246439069164254585 85,\ s = 40.9082,\ t = \exp(3.66), \\
& f(X) = 1033308066924956844000X^7 - 1607550115434903530384 79X^6 \\
&\qquad - 1953036272361510565766762963004277 51X^5 \\
&\qquad - 6732299766097047296232233142462051885 7X^4 \\
&\qquad + 8528866874226821944413384946675849792 83X^3 \\
&\qquad + 1222612473873462051375075541601552132234 49X^2 \\
&\qquad - 9410422625986284574258926092966248452782 18X \\
&\qquad - 3880671209559044857530412651862712063732543 2. \\
d = 8\text{:}\quad & m = 1364850538695913738402818687041215458,\ s = 107.255,\ t = \exp(5.13), \\
& f(X) = 1121673850908090480 0X^8 + 412696396286148938585 9X^7 \\
&\qquad - 1175791917822439782941507504635X^6 + 2996639999067533888196133035298645X^5 \\
&\qquad + 208240147656019048048262524877102283X^4 \\
&\qquad - 273577029261398618678576092511528878 73X^3 \\
&\qquad - 342483409910020774289672696011470992653 5X^2 \\
&\qquad - 129575387126478114914365102382831882192 29X \\
&\qquad + 8733287829967486818441309661955398847347705. \\
d = 9\text{:}\quad & m = 131071707154406288685947736054548 8,\ s = 8.51584,\ t = \exp(3.89), \\
& f(X) = 11829510000X^9 - 323042712742X^8 - 22960091664443611251501443 10X^7 \\
&\qquad - 17667833832765445702215975840307X^6 + 1047509842434615097951397998479 08X^5 \\
&\qquad + 684082899341824778960200186325064X^4 - 855848613284815182617841442493 8636X^3 \\
&\qquad + 3230171878199466794643608399114487 4X^2 - 4211883730221892830363726045151563 8X \\
&\qquad - 1293558869408225281960437545569172565.
\end{aligned}$$

B The parameter settings from [18]

This appendix provides analysis of the NFS parameters used in the revised TWIRL design [18]. It follows the approach of Section 3, extended to produce estimates for the frequency of intermediate candidates.

Polynomials. We used the NFS polynomial selection program of Jens Franke and Thorsten Kleinjung, which contains several improvements on the strategy of [13][14][15] which was used to obtain the polynomials of Section 3 and Appendix A. We employed several Pentium 1.7GHz computers, for a total CPU time of about 20 days. However, most of this time was spent on experimentation with search parameters; the conclusions can be reused for other composites, so future experiments would require just a few hours. We observe that with this polynomial selection program there is a lot of flexibility in the search parameters: at a small cost in yield, one can obtain polynomials with much larger or much smaller skew, trade root properties for size properties, etc. Appendix B.2 of [18] gives the best polynomial we found for RSA-1024, which is as follows:

$d = 5$: $m = 2626198687912508027781823689041581872398105941296246738850076103682306196740$
$5529250615451338729866356088714618385497519816050227824324506707482059371105472385$0
$5700273957561400114202031348071179037320617188128273668251667044346501282228160838$7
$1694092824691383112595203927698431049857937444948214372729619704 86,$

$s = 1991935.4$, $t = \exp(6.33)$,

$$
\begin{aligned}
f(X) = {} & 171930489423634514340101141808 0X^5 \\
& - 699197348886660586107407418604363447 1X^4 \\
& + 27086030483569532894050974257851346649521314X^3 \\
& + 469375840526685745028867918355365522774102423590 42X^2 \\
& - 101070294842572111371781458850696845877706899545394501384X \\
& - 2266691593949094057861752467704537118912890989971656039843413 6,
\end{aligned}
$$

$$
\begin{aligned}
g(X) = {} & 9387723083702630698457136747702 7X \\
& - 379348954964250275136910457556396371742114833244516283 65.
\end{aligned}
$$

Here the rational-side polynomial g is non-monic; thus we redefine $N_{\mathbf{r}}(a,b) = |b \cdot g(a/b)|$. Table 9 estimates the yield of this polynomial using the parameter sets from [17] that were considered in Section 5. A comparison with Table 1 shows that this polynomial has much higher yield; indeed, both its size properties and its root properties are better (cf. [15]). Throughout this appendix we shall use this polynomial, except where noted otherwise.

Table 9. Estimated yields with [18]'s RSA-1024 polynomial and [17]'s parameters.

d	B	ff	fp_8	pf_8	pp_8	fp_{12}	pf_{12}	pp_{12}	fp_{16}	pf_{16}	pp_{16}
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{28}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 2.9\text{E}7$, $S = 6\text{E}23$, sieving effort 3.6E24											
5	3.88E8	9.9E2	2.0E4	9.7E3	2.0E5	3.8E4	1.8E4	6.8E5	6.6E4	2.8E4	1.9E6
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 1.5\text{E}9$, $S = 6\text{E}23$, sieving effort 3.8E24											
5	3.88E8	1.8E8	2.1E9	1.0E9	1.2E10	3.7E9	1.7E9	3.5E10	5.9E9	2.6E9	8.6E10
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 1.5\text{E}9$, $S = 2.4\text{E}24$, sieving effort 1.5E25											
5	3.88E8	3.8E8	4.5E9	2.2E9	2.5E10	8.1E9	3.7E9	7.7E10	1.3E10	5.6E9	1.9E11
$y_{\mathbf{r}} = y_{\mathbf{a}} = 2^{34}$, $T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 1.5\text{E}9$, $S = 9.8\text{E}24$, sieving effort 6.1E25											
5	3.88E8	8.2E8	9.9E9	4.7E9	5.7E10	1.8E10	8.0E9	1.7E11	2.9E10	1.2E10	4.2E11

Note that Section 5 gives strong indication that $d = 5$ is suboptimal, but the program we used is limited to $d = 5$. One can expect that an adaptation of the improved algorithm to $d = 6$ or $d = 7$ will yield even better results. In this light, the parameters of [18] merely imply an upper bound on cost; further improvement is likely to be possible.

Yield. To increase yield, [18] uses higher smoothness bounds than [17]: $y_\mathbf{r} = 3.5\text{E}9$, $y_\mathbf{a} = 2.6\text{E}10$, $z_\mathbf{r} = 4.0\text{E}11$, $z_\mathbf{a} = 6.0\text{E}11$. This has a dramatic effect, suggesting that the choice from [17] indeed resides on the steep region of the run-time curve (cf. Section 4). Also, the number of allowed large primes is increased to $\ell_\mathbf{r} = \ell_\mathbf{a} = 2$. Conversely, the sieving region size is reduced to $S = 3.0\text{E}23$. Table 10 gives the corresponding estimates of yield, as well as the number of intermediate candidates (see below). Note that [18] uses different notation: there R, H, B_R and B_A stand for our $2A$, B, $y_\mathbf{r}$ and $y_\mathbf{a}$, respectively.

Table 10. RSA-1024 parameters and estimates for [18].

$y_\mathbf{r} = 3.5\text{E}9$, $y_\mathbf{a} = 2.6\text{E}10$, $z_\mathbf{r} = 4.0\text{E}11$, $z_\mathbf{a} = 6.0\text{E}11$, $T(y_\mathbf{r}, y_\mathbf{a}) \approx 1.3\text{E}9$, $S = 3.0\text{E}23$
$d = 5$, $s = 1991935.4$, $B = 2.7\text{E}8$

yield of $(L_\mathbf{a}, Lr)$-partial relations									
$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$	Total
5.6E7	3.0E8	6.7E8	3.1E8	1.7E9	3.8E9	6.6E8	3.5E9	7.9E9	1.9E10

#PRS	#PBS	#PPT	#RCF	#RSS	#ACF	avg($N_\mathbf{r}(a,b)$)	avg($N_\mathbf{a}(a,b)$)
1.1E20	5.0E12	6.2E10	4.9E10	3.4E10	2.7E10	5.2E63	3.1E103

Ultimately we are interested in the number of cycles among the relations found. Alas, the dependence of the number of cycles on the number (and type) of relations is poorly understood (cf. Section 2). As noted, $\pi(z_\mathbf{r})+\pi(z_\mathbf{a})$ relations always suffice, and in past experiments the number of relations collected was always somewhat lower. Here, the estimated number of relations is $0.49\cdot(\pi(z_\mathbf{r})+\pi(z_\mathbf{a}))$. Using $\ell_\mathbf{a}, \ell_\mathbf{r} > 2$, as in the aforementioned experiments, would further increase the relation yield. Note that there are $T(y_\mathbf{r}, y_\mathbf{a})/23.2$ ***ff***'s, which seems very reasonable.

It is worth observing that while the most 'fertile' area of the sieving region is close to the origin, the relation yield of the sieving region is not yet 'dried out': for example, doubling S to 6E23 increases the number of relations significantly, to 2.8E10. The practical significance is that if someone builds a TWIRL device with hard-wired smoothness bounds and (for whatever reason) does not find enough relations using the above parameters, recovery may be possible simply by increasing S, i.e., by sieving for a longer time using the same hardware.

Candidates. For integer k, let $\mu(y,k) = k/\eta(y,k)$ denote the non-y-smooth cofactor of k. Sieving per se (i.e., the task handled by TWIRL) merely identifies the pairs (a,b) for which $\mu(y_\mathbf{r}, N_\mathbf{r}(a,b)) \le {z_\mathbf{r}}^{\ell_\mathbf{r}}$ and $\mu(y_\mathbf{a}, N_\mathbf{a}(a,b)) \le {z_\mathbf{a}}^{\ell_\mathbf{a}}$. For $\ell_\mathbf{a} = \ell_\mathbf{r} = 2$, not all such pairs form relations. Thus subsequent filtering is applied, and it should be verified that its cost is manageable. Also, in the "cascaded sieves" variant employed by the revised TWIRL design, the algebraic-side sieve handles only the pairs (a,b) that passed the rational sieve, and it should be verified that the latter are sufficiently infrequent (cf. [18, A.6]; this is crucial for achieving the high parallelism factor of 32768 inspected pairs per clock cycle).

Thus, we estimate the number of candidates at the relevant points in the algorithm by writing down the appropriate probability, integrating it over the sieving region and multiplying the result by the correction factor $6/\pi^2$ (cf. Section 3).

The types of candidates are listed below; the results of the integrations are given in Table 10. In the following, let k_1, k_2 ($k_1 \geq k_2$) denote the two largest prime factors of $N_{\mathbf{r}}(a,b)$, and let κ_1, κ_2 ($\kappa_1 \geq \kappa_2$) denote the two largest prime factors of $N_{\mathbf{a}}(a,b)$.

Pass rational sieve (PRS): The pairs that pass the rational sieve are those that fulfill $\mu(y_{\mathbf{r}}, N_{\mathbf{r}}(a,b)) \leq {z_{\mathbf{r}}}^2$. Noting that ${z_{\mathbf{r}}}^2 < {z_{\mathbf{a}}}^3$, we get that the above is equivalent to the following: $(k_1, k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1 \leq {z_{\mathbf{r}}}^2 \wedge k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1, k_2 \wedge k_1 k_2 \leq {z_{\mathbf{r}}}^2)$. Accordingly, the probability that (a,b) fulfills this can be estimated by $\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}/2) + \bar{\sigma}_2(u_{\mathbf{r}}, v_{\mathbf{r}}/2, v_{\mathbf{r}}/2)$.

Pass both sieves (PBS): the probability that a pair (a,b) passes both sieves is obtained by multiplying the above by the analogous expression for the algebraic side: $(\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}/2) + \bar{\sigma}_2(u_{\mathbf{r}}, v_{\mathbf{r}}/2, v_{\mathbf{r}}/2)) \cdot (\rho(u_{\mathbf{a}}) + \sigma_1(u_{\mathbf{a}}, v_{\mathbf{a}}/2) + \bar{\sigma}_2(u_{\mathbf{a}}, v_{\mathbf{a}}/2, v_{\mathbf{a}}/2))$.

Pass primality testing (PPT): For pairs that passed both sieves, the smooth factors are divided out to obtain $\mu(y_{\mathbf{r}}, N_{\mathbf{r}}(a,b))$ and $\mu(y_{\mathbf{a}}, N_{\mathbf{a}}(a,b))$ (note that most prime factors smaller than $y_{\mathbf{r}}$ or $y_{\mathbf{a}}$ are reported by TWIRL). If $\mu(y_{\mathbf{r}}, N_{\mathbf{r}}(a,b))$ is prime and $> z_{\mathbf{r}}$, or $\mu(y_{\mathbf{r}}, N_{\mathbf{a}}(a,b))$ is prime and $> z_{\mathbf{a}}$, then the pair is discarded. A pair (a,b) reaches and survives this test iff $(k_1, k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1 \leq z_{\mathbf{r}} \wedge k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1, k_2 \wedge k_1 k_2 \leq {z_{\mathbf{r}}}^2)$ and analogously for the algebraic side. The probability that this holds is estimated by $(\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}) + \bar{\sigma}_2(u_{\mathbf{r}}, v_{\mathbf{r}}/2, v_{\mathbf{r}}/2)) \cdot (\rho(u_{\mathbf{a}}) + \sigma_1(u_{\mathbf{a}}, v_{\mathbf{a}}) + \bar{\sigma}_2(u_{\mathbf{a}}, v_{\mathbf{a}}/2, v_{\mathbf{a}}/2))$.

Rational cofactor factorizations (RCF): For pairs that survived primality testing, if the cofactor $\mu(y_{\mathbf{r}}, N_{\mathbf{r}}(a,b))$ is composite then it needs to be factored and tested for z_r-smoothness. The size of the cofactor to be factored is bounded by ${z_{\mathbf{r}}}^2$. This step is reached and the factorization is performed if $(y_{\mathbf{r}} < k_1, k_2 \wedge k_1 k_2 \leq {z_{\mathbf{r}}}^2)$ and $(\kappa_1, \kappa_2 < y_{\mathbf{a}}) \vee (y_{\mathbf{a}} < \kappa_1 \leq z_{\mathbf{a}} \wedge \kappa_2 < y_{\mathbf{a}}) \vee (y_{\mathbf{a}} < \kappa_1, \kappa_2 \wedge \kappa_1 \kappa_2 \leq {z_{\mathbf{a}}}^2)$. The probability that this holds is estimated by $\bar{\sigma}_2(u_{\mathbf{r}}, v_{\mathbf{r}}/2, v_{\mathbf{r}}/2) \cdot (\rho(u_{\mathbf{a}}) + \sigma_1(u_{\mathbf{a}}, v_{\mathbf{a}}) + \bar{\sigma}_2(u_{\mathbf{a}}, v_{\mathbf{a}}/2, v_{\mathbf{a}}/2))$.

Rational semi-smooth (RSS): A pair reaches the rational cofactor factorization step and passes (or skips) it if indeed $N_{\mathbf{r}}(a,b)$ is $(y_{\mathbf{r}}, z_{\mathbf{r}}, \ell_{\mathbf{r}})$-smooth and (a,b) passed the algebraic sieve. For this to happen, the condition on the rational side is $(k_1, k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1 \leq z_{\mathbf{r}} \wedge k_2 < y_{\mathbf{r}}) \vee (y_{\mathbf{r}} < k_1, k_2 \leq z_{\mathbf{r}})$, and the condition on the algebraic side is as in the previous step. Thus the probability is estimated by $(\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}) + \sigma_2(u_{\mathbf{r}}, v_{\mathbf{r}})) \cdot (\rho(u_{\mathbf{a}}) + \sigma_1(u_{\mathbf{a}}, v_{\mathbf{a}}) + \bar{\sigma}_2(u_{\mathbf{a}}, v_{\mathbf{a}}/2, v_{\mathbf{a}}/2))$.

Algebraic cofactor factorizations (ACF): For pairs that passed all of the above, if the cofactor $\mu(y_{\mathbf{a}}, N_{\mathbf{a}}(a,b))$ is composite then it needs to be factored and tested for $z_{\mathbf{a}}$-smoothness. This step is reached and the factorization is performed iff $(y_{\mathbf{a}} < \kappa_1, \kappa_2 \wedge \kappa_1 \kappa_2 \leq {z_{\mathbf{a}}}^2)$ and also the rational-side condition of the previous step holds. The corresponding probability is estimated by $(\rho(u_{\mathbf{r}}) + \sigma_1(u_{\mathbf{r}}, v_{\mathbf{r}}) + \sigma_2(u_{\mathbf{r}}, v_{\mathbf{r}})) \cdot \bar{\sigma}_2(u_{\mathbf{a}}, v_{\mathbf{a}}/2, v_{\mathbf{a}}/2)$.

Relations (Total): A pair that passes all of the above forms a relation; the probability of this occurring is estimated by $(\rho(u_\mathbf{r})+\sigma_1(u_\mathbf{r},v_\mathbf{r})+\sigma_2(u_\mathbf{r},v_\mathbf{r}))\cdot(\rho(u_\mathbf{a})+\sigma_1(u_\mathbf{a},v_\mathbf{a})+\sigma_2(u_\mathbf{a},v_\mathbf{a}))$.

The above describes one plausible ordering of the filtering steps; other variations are possible (e.g., performing the algebraic cofactor factorization before the rational cofactor factorization, or even before the rational primality testing).

Cost of cofactor factorization. As indicated above, we expect to perform about #RCF + #ACF = 7.7E10 factorizations of integers whose size is at most $\max(z_\mathbf{r},z_\mathbf{a})^2 = 3.6\text{E}23$. Such factorizations require under 30ms on average using a modern CPU. Thus, the cofactor factorization can be completed in 1 year (i.e., in parallel to the operation of the TWIRL device) using about 74 barebones PCs. This cost is negligible compared to the cost of TWIRL, and in large volumes custom hardware would reduce it further.

Optimality and effect of technological progress. The revised TWIRL parameters were essentially determined by practical concerns. Most crucially, they employ the largest value of $y_\mathbf{a}$ for which the algebraic-side TWIRL device still fits on single silicon wafer. Theoretically, this $y_\mathbf{a}$ is suboptimal; it would be beneficial to increase it. Such increase will become possible when progress in chip manufacturing technology allows fitting larger circuits into a single wafer, either by increasing the wafer size or by decreasing the feature size. Thus, for the foreseeable future we may expect the cost of TWIRL to decrease more than linearly as a function of the relevant technological parameters, i.e., faster than naively implied by Moore's law.

For a concrete example, one may consider an implementation of TWIRL using 90nm process technology, which is expected to be widely deployed during 2004. Compared to the 130nm process technology considered in [18], we may assume a reduction in area by a factor of 2 and an increase in speed by a factor of 2, for a total cost reduction by a factor of 4 (cf. [8]). Table 11 presents two appropriate NFS parameter sets. The first set is about as plausible as the one in Table 10; the cost of such a TWIRL implementation is roughly 1.1M US$×year (predicted analogously to [18]) — considerably lower than 2.5M US$×year one may expect.

The second parameter set in Table 11 shows the effect of improved technology on yield, when keeping the cost constant at 10M US$×year (i.e., the same as in [18]). Here, the estimated number of relations is $1.95\cdot(\pi(z_\mathbf{r})+\pi(z_\mathbf{a}))$, which is nearly twice the trivially sufficient number. Also, there are $T(y_\mathbf{r},y_\mathbf{a})/3.6$ ***ff***'s, which is much more than in any recent factoring experiment. Thus, we may conclude that using 90nm technology, a budget of 10M US$×year per factorization (in large quantities) leaves an ample safety margin — arguably, more than enough to account for estimation errors, relations that are lost due to approximations in the sieving process, and sub-optimal cycles-finding algorithms.

Table 11. RSA-1024 parameter sets for TWIRL with 90nm process technology.

$y_{\mathbf{r}} = 1.2\text{E}10,\ y_{\mathbf{a}} = 5.5\text{E}10,\ z_{\mathbf{r}} = 8.0\text{E}11,\ z_{\mathbf{a}} = 1.0\text{E}12,\ T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 2.9\text{E}9,\ S = 8.0\text{E}22$
$d = 5,\ s = 1991935.4,\ B = 1.4\text{E}8$

yield of $(L_{\mathbf{a}}, Lr)$-partial relations									
$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$	Total
2.2E8	9.8E8	1.8E9	9.2E8	4.0E9	7.5E9	1.4E9	6.1E9	1.1E10	3.4E10

#PRS	#PBS	#PPT	#RCF	#RSS	#ACF	avg$(N_{\mathbf{r}}(a,b))$	avg$(N_{\mathbf{a}}(a,b))$
6.3E19	1.1E13	9.8E10	7.2E10	5.9E10	4.5E10	2.7E63	1.1E102

$y_{\mathbf{r}} = 1.2\text{E}10,\ y_{\mathbf{a}} = 5.5\text{E}10,\ z_{\mathbf{r}} = 9.0\text{E}11,\ z_{\mathbf{a}} = 1.2\text{E}12,\ T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 2.9\text{E}9,\ S = 7.3\text{E}23$
$d = 5,\ s = 1991935.4,\ B = 4.3\text{E}8$

yield of $(L_{\mathbf{a}}, Lr)$-partial relations									
$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$	Total
7.9E8	3.9E9	7.9E9	3.4E9	1.7E10	3.4E10	5.4E9	2.7E10	5.5E10	1.5E11

#PRS	#PBS	#PPT	#RCF	#RSS	#ACF	avg$(N_{\mathbf{r}}(a,b))$	avg$(N_{\mathbf{a}}(a,b))$
5.2E20	4.6E13	4.6E11	3.4E11	2.7E11	2.1E11	8.1E63	2.8E104

Parameter settings for 768-bit composites. For RSA-768, [18] uses the following polynomial, obtained by the same method as above:

$$
\begin{aligned}
\mathbf{d = 5:}\ \ m = {} & 2980427354552256959621694668022969720925142335553136586170340190386865951921\\
& 42458430585097389943648179813292845509402284357573098406890616147678906706078002760\\
& 825484610584689826591183386558993533887364961255454143572139671622998845,\\
& s = 1905116.1,\ t = \exp(3.78),\\
& f(X) = 44572350495893220X^5\\
& \qquad + 1421806894351742986806319X^4\\
& \qquad - 1319092270736482290377229028413X^3\\
& \qquad - 4549121160536728229635596952173101064X^2\\
& \qquad + 6062531470679201843447146909871507448641523X\\
& \qquad - 1814356642608474735992878928235210850251713945286,\\
& g(X) = 669580586761796376057918067X - 7730028528962337116069068686542066657037329.
\end{aligned}
$$

The parameter choice and yield estimates using this polynomial are given in Table 12.

Table 12. RSA-768 parameters and estimates for [18].

$y_{\mathbf{r}} = 1.0\text{E}8,\ y_{\mathbf{a}} = 1.0\text{E}9,\ z_{\mathbf{r}} = 2.0\text{E}10,\ z_{\mathbf{a}} = 3.0\text{E}10,\ T(y_{\mathbf{r}}, y_{\mathbf{a}}) \approx 5.7\text{E}7,\ S = 3.0\text{E}20$
$d = 5,\ s = 1905116.1,\ B = 8.9\text{E}6$

yield of $(L_{\mathbf{a}}, Lr)$-partial relations									
$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$	Total
3.5E6	2.2E7	5.5E7	2.5E7	1.5E8	3.9E8	6.2E7	3.8E8	9.7E8	2.1E9

#PRS	#PBS	#PPT	#RCF	#RSS	#ACF	avg$(N_{\mathbf{r}}(a,b))$	avg$(N_{\mathbf{a}}(a,b))$
5.3E17	3.4E11	7.5E9	6.3E9	3.9E9	3.2E9	3.4E49	7.1E82