Strategy of Relations Collection in Factoring RSA Modulus

Haibo Yu[1(✉)] and Guoqiang Bai[2,3]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, China
yhb13@mails.tsinghua.edu.cn
[2] Institute of Microelectronics, Tsinghua University, Beijing, China
baigq@mail.tsinghua.edu.cn
[3] Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China

Abstract. In this paper, we propose a new strategy of relations collection in factoring RSA modulus. The strategy has absolute advantage at computation efficiency with a highly parallel structure, and the reports have higher probability of being relations, since the strategy only considers small and medium primes in factor base without large primes. Besides, it is worth noting that the proposed algorithm used in strategy, only involves a multiplication, which can speed up relations collection step. Furthermore, we propose an architecture for multiplier that is based on our algorithm. Due to the inherent characters of the algorithm, our proposed architecture can perform with less registers, which makes for VLSI area optimization. Additionally, the comparison results with the published achievements show that our strategy could be a good choice for relations collection in factoring RSA modulus.

Keywords: General number field sieve · Relations collection · Prime factors · Highly parallel structure · Multiplication

1 Introduction

General Number Field Sieve (GNFS) [1,2] is the asymptotically fastest factorization algorithm so far [1], which is a threat to public-key cryptography (PKC) relied on the hardness of factoring large numbers, RSA included.

In its main step, relations collection step, pairs of integers called *relations* being collected, is dominant expensively and theoretically in GNFS, which is usually performed by sieving technology.

Many sieving methods used in GNFS have been proposed. Line sieving [1] was firstly proposed in 1993, which was applied to 423-bit factorization [3]. In the same year, Pollard described a more efficient implementation on the basis of line sieving, called lattice sieving [2], which was the main idea of the majority of the published approaches, such as 768-bit RSA modulus factorization [4]. In the following year, 1994, a combination of lattice sieving and trial division [5] was

S. Qing et al. (Eds.): ICICS 2015, LNCS 9543, pp. 199–211, 2016.
DOI: 10.1007/978-3-319-29814-6_16

adopted to speed up sieving step and collect triple and quadruple large prime relations. Most recently, a higher efficiency was obtained by using so-called continued fractions and lattice sieving in 2005, where a large parallel computers were employed to speed up sieving step. [6] suggests that PC-based implementation of GNFS for 1024-bit composites is prohibitive and ASICs design for the computationally expensive steps of GNFS is attractive. Researchers therefore have presented a couple of different hardware designs to realize sieving, such as TWINKLE [6], TWIRL [7], SHARK [8] and CAIRN2 [3].

Generally speaking, the reports from sieving or lattice sieving, have low probability to be relations. Besides, large primes are always allowed to improve speed of sieving, which results in larger sieving region and more workload. Thus in this paper, we propose a new strategy of relations collection in factoring RSA modulus. The main contributions of the paper are detailed below.

Firstly, we propose a new module of relations collection in factoring large integers, in which a highly parallel structure is adopted to decrease computation time. Besides, the module only considers small and medium primes in factor base, the reports from our module therefore are more likely to be relations.

Secondly, we present a algorithm used in relations collection module. For a fixed prime, the investigated algorithm computes a estimated value with a precomputed value, in order to tell whether input data is a integer multiple of the prime. And it is worth noting that the proposed algorithm only involves a multiplication, which can speed up relations collection step and be beneficial to hardware implementation. Moreover, adequate parameters can be chosen to guarantee the feasibility of the proposed algorithm, and meanwhile, maintain the concision of algorithm.

A third contribution is the architecture for a multiplier that is based on our algorithm. Multiplication is the core part of the proposed algorithm, which is also the only time-consuming operation in algorithm. The running time of multiplication is mainly depends on the size of input data, and is not strongly dependent in the size of the factor given. Due to the inherent characters of the algorithm, our proposed architecture can perform with less registers, which makes for VLSI area optimization.

The organization of this paper is given as follows: relevant definitions are described in Sect. 2, which do a brief look back at the relevant mathematical background. In Sect. 3, the proposed module of relations collection in factoring RSA modulus is presented, and the investigated algorithm used in strategy is introduced in Sect. 4, being the core part of the paper. The architecture of multiplier based on the proposed algorithm is presented in Sect. 5. Some comparison results are given in Sect. 6, which shows that the proposed strategy could be a good choice of relations collection. Finally, Sect. 7 concludes the paper.

2 Preliminaries

Relations Collection Step. The entire factorization begins with polynomial selection. Two irreducible polynomial $f_{\boldsymbol{r}}, f_{\boldsymbol{a}} \in \mathbb{Z}[x]$, degree $d_{\boldsymbol{r}}$ and $d_{\boldsymbol{a}}$, respectively,

and an integer e, satisfy the condition $f_{\boldsymbol{r}}(e) \equiv f_{\boldsymbol{a}}(e) \equiv 0 \pmod{N}$. In general, $f_{\boldsymbol{r}}$ and $f_{\boldsymbol{a}}$ are called *rational* and *algebraic* polynomials, with smoothness bounds $B_{\boldsymbol{r}}$ and $B_{\boldsymbol{a}}$, respectively. A pair of coprime integers (a, b) with $b > 0$ is referred to as *relations*, such that $b^{d_{\boldsymbol{r}}} f_{\boldsymbol{r}}(a/b)$ is $B_{\boldsymbol{r}}$-smooth and $b^{d_{\boldsymbol{a}}} f_{\boldsymbol{a}}(a/b)$ is $B_{\boldsymbol{a}}$-smooth, where an integer is B-smooth if all its prime factors are at most B, as usual. A combination of *sieving*, collection of a large passel of promising relations (called *candidates*), and *cofactorization*, identification whether it really being a relation or not, is usually adopted in the relations collection step.

3 The Proposed Module of Relations Collection

Finding sufficiently many relations, namely sieving, is the major work of GNFS, which is normally about 90 % of the total computational time [9]. A primary reason for the popularity of line sieving or lattice sieving is less division operations, which is one of the computationally expensive algorithms.

In both sieve and lattice sieve, large primes are always allowed to speed up sieving step. However, they make it harder to decide whether enough relations have been found, as the criterion [1] of the number of relations needed is no longer adequate. It is worth noting that few candidates can survive in cofactorization and become relations needed in following steps. For instance, only 0.027522 % and 0.0006789 % candidates can survive in [9] and [10], respectively.

In this section, a new module of relations collection is proposed. The module only considers the small and medium primes in factor base without large primes. Besides, we try to find the expression of integer's standard factorization, in order to tell whether it being $B_{\boldsymbol{r}}$-smooth or $B_{\boldsymbol{a}}$-smooth. As a consequence, the candidates from the proposed module are more likely to be relations, which can decrease the workload of relations collection to a great extent.

Let a set of odd primes $\{p_1, p_2, \cdots, p_i, \cdots, p_k\}$ be factor base and N be input data for smoothness test, such that $N = b^{d_{\boldsymbol{r}}} f_{\boldsymbol{r}}(a/b)$ or $b^{d_{\boldsymbol{a}}} f_{\boldsymbol{a}}(a/b)$ for rational or algebraic side. The whole structure consists of k isolated units in parallel, shown in Fig. 1, where k is the number of primes in factor base and F_i is one-bit output signal (if N is an integer multiple of p_i, $F_i = 1$; otherwise, $F_i = 0$). Moreover, we assume that most of the work is spent in the first splitting and subsequent splittings are negligible, thus power of primes is not considered.

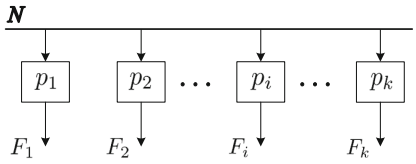

Fig. 1. High-level schema of the proposed relations collection module

All basic units run in parallel as soon as input data N arrives, and output F_i, $1 \leq i \leq k$, tells whether N is a smooth integer. The most straightforward

way, finding all prime factors in factor base of integer N, is adopted, which can decrease the influence of large primes. Besides, with synchronous parallel triggered structure, the proposed module has absolute advantage at computation efficiency. For realization of the proposed strategy, the algorithm used in basic units and hardware implementations are of great importance, which will be described in Sects. 4 and 5 in more detail, respectively.

4 Investigated Algorithm

4.1 Basic Definitions

To make the following explanations and descriptions easier, we present some notations used in this paper firstly. An n-bit integer X is denoted in radix 2 representation as $X = (X_{n-1} \cdots X_0)_2$ and $X_{n-1} \neq 0$; X_i refers to the i-th bit of X, $0 \leq i \leq n-1$; $|X|$ refers to the bit width of X in radix 2. Note that all integers discussed in this paper are represented in radix 2. Known that N is a nonnegative odd integer, after shifting all factors of two, and p is an odd prime. For now, let us assume their bit lengths are m and $m+\gamma$, respectively, i.e. $|p| = m$ and $|N| = m+\gamma$, thus we have $\gamma > 0$, $2^{m-1} \leq p < 2^m$ and $2^{m+\gamma-1} \leq N < 2^{m+\gamma}$.

For given N and p, the intermediate ratio B is defined as:

$$B = \frac{N}{p}\, 2^{\alpha-\beta} \tag{1}$$

where α and β are two parameters and $\alpha > \beta$.

Lemma 1. *For given N and p, if N is an integer multiple of p, the conclusion holds: $B_{\alpha-\beta-1}, \cdots, B_1, B_0 = 0$ and $B_{\alpha-\beta} = 1$.*

Proof. See the Appendix. ■

Lemma 2. *Let $\mu = \left\lfloor \frac{N}{2^{m+\beta}} \left\lfloor \frac{2^{m+\alpha}}{p} \right\rfloor \right\rfloor$ be $(\gamma+\alpha-\beta)$-digit positive integer, with $\beta > \gamma$ and $\alpha > m+\gamma$. Then it holds*

$$\mu = \begin{cases} B-1, & \text{if } B \text{ is an integer} \\ \lfloor B \rfloor, & \text{if } B \text{ is a decimal} \end{cases} \tag{2}$$

Proof. See the Appendix. ■

Based on Lemma 1 and 2, B and μ have a kind of standard binary representations, denoted as B_s and μ_s (see Fig. 2), respectively, when N is an integer multiple of p. Moreover, we also have $\mu_s = B_s - 1$.

As we can see above, choosing adequate parameters α, β and γ is of great importance. First of all, we have $\gamma > 0$, according to the definition of γ. Besides, Lemma 2 shows that $\beta > \gamma$ and $\alpha > m+\gamma$ are necessary. More generally, parameters should meet the requirements of $\alpha > \beta > \gamma > 0$ and $\alpha > m+\gamma$.

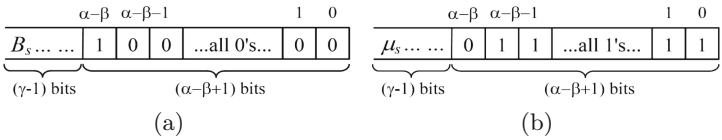

Fig. 2. The standard binary representations (a) the sets of B_s; (b) the sets of μ_s.

4.2 The Main Idea of the Algorithm

Based on Lemma 1, we conclude that $B = B_s$ is a necessary and sufficient condition for N being an integer multiple of p. Thus, the proposed algorithm is developed to test whether B is a standard representation for the purpose of determining the prime factor of an integer.

To avoid computationally expensive divisions, the algorithm uses the input N and a reciprocal of p to compute the intermediate ratio B. Rewrite (1) as

$$B = \frac{N}{2^{m+\beta}} \frac{2^{m+\alpha}}{p}$$

in which $2^{m+\alpha}/p$ is a constant and the value can be precomputed, for a fixed odd prime p. Since $2^{m+\alpha}/p$ is a decimal but not an integer, we use $\left\lfloor \frac{2^{m+\alpha}}{p} \right\rfloor$ to avoid storing floating-point values. Therefore, we focus on $\mu = \left\lfloor \frac{N}{2^{m+\beta}} \left\lfloor \frac{2^{m+\alpha}}{p} \right\rfloor \right\rfloor$, which is a estimated value of B.

If $\mu = \mu_s$, p is possibly a prime factor of N based on Lemmas 1 and 2, for the reason that $\mu = \mu_s$ is just the necessary condition. However, in most of cases μ-value can give correct result about divisibility of N and p. There is only one situation in which μ could not work, namely $B = B_{evil}$ (see Fig. 3).

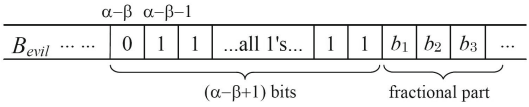

Fig. 3. Binary representation of the sets B_{evil}

With some simplifications, the algorithm computes μ-value, checks whether μ eauls μ_s or not, and outputs $F = 1$ if it does, without consideration of case B_{evil}. (If $F = 0$, p is not a prime factor of N; otherwise, p is very possible a prime factor of N.)

4.3 The Feasibility of the Proposed Algorithm

In this subsection, we show that even if a defect in the idea of testing μ-value only, the algorithm can still produce satisfactory results in relations collection.

In consideration of case B_{evil}, N cannot be divided by p obviously, but $\mu = \mu_s$ still stands according to the definition of μ, which is called *erroneous judgement*. Let us firstly analysis the probability of erroneous judgement on the precondition of $\mu = \mu_s$.

For now, an integer N_p, being less than N and the greatest multiple of p, satisfies $N = d + N_p$, $p|N_p$ and $0 < d < p$. Let rewrite (1) once more as:

$$B = \frac{N}{p}\ 2^{\alpha-\beta} = \frac{d+N_p}{p}\ 2^{\alpha-\beta} = \frac{d}{p}\ 2^{\alpha-\beta} + \frac{N_p}{p}\ 2^{\alpha-\beta}$$

in which $d/p\ 2^{\alpha-\beta}$ and $N/p\ 2^{\alpha-\beta}$ are denoted as B_u and B_v, respectively. For more simple analysis, we assume $\alpha - \beta = 3$ for no reason (this works with other positive integers).

Because of $0 < d < p$, i.e. $0 < d/p < 1$, we have the conclusion that $d/p = b_1 2^{-1} + b_2 2^{-2} + b_3 2^{-3} + \cdots$, shown in Fig. 4.

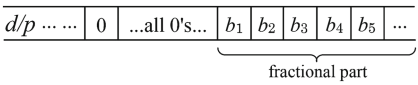

Fig. 4. Binary representation of the sets d/p

It is always true that N_p/p is an integer since N_p is an integer multiple of p. Besides B_v is obtained by left shifting $\alpha - \beta$ bits of N_p/p (3 bits, based on assumption above), which means $B_{v(\alpha-\beta-1)}, B_{v(\alpha-\beta-2)}, \cdots, B_{v0} = 0$. Similarly, after left shifting $\alpha - \beta$ bits of d/p, we obtain B_u in binary representation. To clarify the properties of the sets of B_v and B_u, we give Fig. 5.

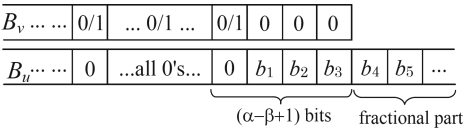

Fig. 5. Binary representation of the sets B_v and B_u with $\alpha - \beta = 3$

Based on the assumption above, $\alpha - \beta = 3$, we can get more detailed binary representations of B_v and B_u, with known condition $B_{evil} = B_u + B_v$. Because of the fact $B_{v2} = B_{v1} = B_{v0} = 0$ and $B_{evil2} = B_{evil1} = B_{evil0} = 1$, we have $b_1 = b_2 = b_3 = 1$. The range value of d/p is hence given by

$$2^{-1} + 2^{-2} + 2^{-3} < \frac{d}{p} < 1 \tag{3}$$

Furthermore, it is easy to obtain that $B_{v3} = 0$ (that is $B_{v(\alpha-\beta)} = 0$). On the basis of relation between B_v and N_p/p, the last significant bit of N_p/p

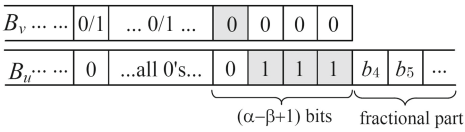

Fig. 6. Detailed binary representation of the sets B_v and B_u with $\alpha - \beta = 3$

equals to $B_{v(\alpha-\beta)}$, which give the conclusion that N_p/p is even. A more specific representations of B_v and B_u are indicated in Fig. 6

Since N_p is a arbitrary integer that is multiple of p, it is reasonable to suppose that the probability of any N_p/p being even is $1/2$. Moreover, generally assuming $\alpha - \beta = w$, the range value of d/p is given by

$$2^{-1} + 2^{-2} + \cdots + 2^{-w} < \frac{d}{p} < 1$$

The probability of erroneous judgement is obtained as follows:

$$P = \frac{1}{2}\left[1 - \left(1 - \frac{1}{2^w}\right)\right] = \frac{1}{2^{w+1}} \tag{4}$$

As it can be observed, if we want to minimize the probability of erroneous judgement we must guarantee the difference between α and β being large enough. For instance, for $\alpha - \beta = 15$, the probability of judgement result being true is 99.9985 % on the precondition of $\mu = \mu_s$. Higher success probability therefore owes to larger difference between α and β.

Finally, the proposed algorithm is given in Algorithm 1. The main idea of the algorithm is computing a estimated μ-value to approach B-value in order to decrease computational complexity. The algorithm computes $\mu = \left\lfloor \frac{N}{2^{m+\beta}} \left\lfloor \frac{2^{m+\alpha}}{p} \right\rfloor \right\rfloor$ for an input N and a precomputed reciprocal of p. If $\mu \neq \mu_s$, we have that p is not a prime factor of N; otherwise, it is highly possible to get that p is a prime factor of N.

Algorithm 1. The proposed algorithm

Input: p, with $|p| = m$; N, with $|N| = m + \gamma$; precomputing $P_{re} = \left\lfloor \frac{2^{m+\alpha}}{p} \right\rfloor$;
Output: F, being the state of divisibility relation;
1: $temp \Leftarrow N \cdot P_{re}$;
2: $\mu \Leftarrow \left\lfloor \frac{temp}{2^{m+\beta}} \right\rfloor$;
3: **if** $\mu \neq \mu_s$ **then**
4: $F \Leftarrow 0$;
5: **else**
6: $F \Leftarrow 1$;
7: **end if**
8: **return** F;

In Algorithm 1, *temp* is an intermediate variable to store product of N times P_{re}, and μ-value is obtained by right shifting $m+\beta$ bits of *temp*. Since $m+\beta$ is a constant, we do not need shifting operation (step 2 in Algorithm 1), but just compare fixed partial bits of *temp*, namely $temp[m+\alpha : m+\beta]$, with μ_s. As a consequence, the critical path only contains one multiplication.

In relations collection of GNFS, it is worth noting that input data N has similar bit width for different pairs in sieving region. Therefore, parameters in basic units are not much different from each other. Let m_{max} and γ_{max} denote as the maximum bit width of prime and the maximum distance between $|N|$ and $|p|$. Based on assumptions above and parameters conditions, we assume $\alpha = m_{max} + \gamma_{max} + 1$, recorded as α_{max}. Note that a larger α-value means wider binary multiplier, thus we choose $\alpha_{max} = m_{max} + \gamma_{max} + 1$. Moreover, β is determined according to the desired success probability, denoted as β_{max}. It is obvious that α_{max} and β_{maxe} satisfy all basic units' requirements.

5 Hardware Implementation and Synthesis Results

The effectiveness of the proposed strategy lies in an efficient area-time hardware implementation. In particular, the multiplier plays a crucial role, which is also the concern of hardware implementation. To optimize the multiplier, one obviously needs to minimize the latency and/or maximize the throughput [11]. Of course, speed is not the only criterion of interest. VLSI area and cost also influence and limit the designs. Because of the amount of primes in factor base, we focus on the less area design of multipliers. Therefore, a economical design, using as few registers as possible, is proposed in this section.

It is obvious that one of the operands, i.e. P_{re}, being fixed and precomputed, is a constant, which can be wrote in ROM or FLASH during manufacture of the ASICs. Besides, to save register resources, the multiplier receives the input data N in synchronous serial mode from lower to higher bits. Consequently, one-bit register is required to store input bit of N and no register is needed for P_{re}.

A conventional binary multiplier produces an output whose number of bit is sum of the input operands. However, due to the inherent characters of the proposed algorithm, the minimum $m+\beta$ bits of the product is unnecessary and can be discarded for multiplier optimization in terms of area. Under assumption above ($\alpha = m + \gamma + 1$), we have bit widths of P_{re} and *temp* are $\alpha + 1$ and $m + \gamma + \alpha + 2$, respectively. Figure 7 gives a more direct comprehension of the product *temp*, in which shadow areas are essential bits to compare with μ_s.

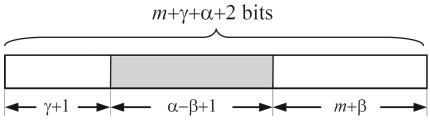

Fig. 7. Schematic diagram of the product *temp*'s component

For instance, setting $m = 24$, $\gamma = 96$, $\alpha = 121$ and $\beta = 100$, we have $|temp| = m + \gamma + \alpha + 2 = 243$ and the minimum 124 bits of $temp$ are useless. Therefore, we propose a specific multiplier design based on above phenomenon, which can save up to 50 % register resources.

The following notation is used in our discussion of multiplication design:

(a) X Multiplicand $X_{k-1}X_{k-2}\cdots X_1X_0$
(b) Y Multiplier $Y_{s-1}Y_{s-2}\cdots Y_1Y_0$
(c) MP Product $MP_{k+s-1}MP_{k+s-2}\cdots MP_1MP_0$

The cumulative product MP is stored in shift register. Because we assume N (regarded as the multiplier) is received in serial mode, the next bit of the multiplier Y (being the role of N) to be considered is always available at a one-bit register and is used to select 0 or X for the addition. Multiplication can be done by adding a successive number (0 or X) to cumulative partial product (initialized to 0) and shifting the cumulative partial product by one bit.

In order to introduce our approach with the help of an example, let us consider $Y = (1011)_2$ and $X = (1101)_2$. Figure 8 depicts the detail of shift register in the multiplication.

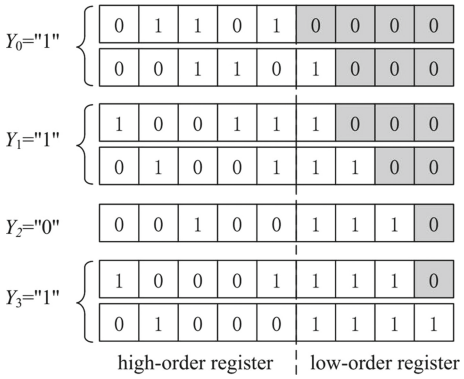

Fig. 8. Behavior of shift register in the 4×4 multiplication

Note that the cumulative partial product and initialed "0" (shadow areas bits) share the same register, keeping the total number of bits at $k + s + 1$. When all initialed "0" (s bits in total) shift from register, the multiplication is done. Besides, results of accumulation are only assigned to high-order register, and lower bits of the cumulative partial product is stored in low-order register with initialed "0". If what we need is only the maximum 5 bits of the full product, the high-order register is adequate to store cumulative partial product without the use of low-order register.

For the proposed algorithm, the last $m + \beta$ bits of full product are useless. Besides, it is obvious that $m + \beta > m + \gamma$, where $m + \gamma$ is the bit width of N (called multiplier in multiplication). With less-registers multipler, $\alpha + 2$ bits

wide is adequate for register *temp*, instead of $m + \gamma + \alpha + 2$. Thus there are almost $(\alpha + 2)$-bit registers are used, without consideration of control portion.

Since the optimization goal of the multiplier design is minimum area, there are some limitations of speed of multiplication. Because of serial input and bit-at-a-time, the running time mainly depends on the size of N.

The proposed algorithm has been described by Verilog HDL and then synthesized with Synopsys Design Compiler 2013. The design library for compiling is SMIC 130 nm CMOS cell library. For $\gamma = 24$ and $m + \gamma = 128$, total area is 1680 gates; and for $\gamma = 31$ and $m + \gamma = 140$, total area is 1830 gates, in which area is measured by the number of NAND gate with two inputs.

6 Estimates of Factoring

A significant improvement in the design of the relations collection module has a great impact on the entire large integer factorization. In this section, we briefly estimate the costs and running time of a special hardware as ASICs. This special hardware could be produced as single ICs, ready for the use in large circuits.

423-Bit. In 2007, [3] reported implementational and experimental results of dedicated sieving device "CAIRN 2" with factoring a 423-bit integer. 30 days were required for the sieving step, in which line sieving (or the pipelined sieving) was used instead of lattice sieving. Since sieving region parameters $H_a = 2.3 \cdot 10^9$ and $H_b = 3 \cdot 10^4$, we have sieving region is $S = 2H_aH_b = 1.38 \cdot 10^{14}$. Besides, relations been found are up to 128-bit.

The running time is mainly depends on the size of input data N for the proposed strategy, therefore including the initialization and postprocessing, a input integer requires approximately $133T$, where T is the system clock cycle. The proposed strategy requires the following effort. On a single 2.2 GHz (as the same frequency as [3]) IC, relations collection would take about 98 days. In other words, 19.6 days are required for relations collection on five chips in parallel.

768-Bit. 768-bit RSA modulus was factored in 2009 [4]. The most cumbersome step, sieving, took almost two years on many hundreds of 2.2 GHz AMD Opteron processors. Input integers were up to 2^{140} and 2^{110} for algebraic and rational smoothness tests, respectively. Moreover, $2 \cdot 10^{18}$ coprime pairs should be considered based on sieving experiments.

With the same operation frequency, we assume $140T$ is required for an input data (or a coprime pair) on average, based on the proposed less-registers multipliers. For one single chip, $2 \cdot 10^{18}$ inputs can be handled within 4027 years.

The estimable result above is limited by VLSI area. Now, let us consider the situation with the optimization goal being speed and throughput. Based on many achievements of fast multipliers, multiplication on the critical path can be performed in one clock cycle with pipeline design. Thus we assume one input pair can be handle in three clock cycles (initialization, multiplication and postprocessing in total). Holding the conditions above, relations collection would take 86.56 years (in other words, 90 chips could handle the problem of relations

collection within one year). A good trade-off between speed and area is necessary because of many limitations in real systems.

1024-Bit. Factoring 1024-bit RSA modulus would be about a thousand times harder than 768-bit [4], and it is still a challenge for factorization [12]. SHARK [8] was a parallelized lattice sieving device for factoring 1024-bit numbers, which consisted of 2300 identical isolated machines sieving in parallel at a clock frequency of 1 GHz. Cost estimates of SHARK were as following. One machine took 20 seconds per special q, one year was therefore necessary for 2300 machines to handle $3.7 \cdot 10^9$ special q.

There are $5.36 \cdot 10^8$ input data for each special q and each input data in sieving region is at most 125-bit. We assume $130T$ is required for each input data at 1 GHz. If the optimization goal is less area, running time of relations collection is 8188 years with one chip, which means 8188 identical chips in parallel can finish the work within one year. On the other hand, if we pursue high speed, with $3T$ for one input data, one chip can perform within 189 years (189 chips can in parallel can handle relations collection within one year) at 1 GHz.

Summary of Comparison. Note that all estimates mentioned above are given under the same workload with associated work. Since the reports from the proposed relations collection module are highly possible to be relations, compared with sieving or lattice sieving, the proposed strategy can result in smaller sieving region and less workload, with the same amount of relations. Comparison results presented above shows that our strategy could be a good choice for relations collection in factoring RSA modulus.

7 Conclusion

This work presents a new strategy of relations collection in factoring RSA modulus. The motivation is to speed up relations collection step with highly parallel structure. The aim is also to decrease the sieving region and workload since the proposed strategy only considers small and medium primes in factor base. The investigated algorithm used in strategy only involves a multiplication, which is suitable for high-speed applications. And the architecture based on the proposed algorithm perform multiplication with less area. Additionally, we estimate the costs of factoring 423-bit, 768-bit and 1024-bit, respectively, and compare them with the published results. The comparison results show that our strategy could be a good choice for relations collection in factorization. Moreover, the reports from relation collection module, based on the proposed strategy, have higher probability of being relations needed in following steps. Our strategy therefore can decrease the whole workload and computational time to a great extent.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (Grants 61472208 and U1135004), and by the National Key Basic Research Program of China (Grant 2013CB338004). The authors would like to thank the editor and reviewers for their comments.

A Proof of Lemmas

Proof of Lemma 1. If N is divisible by p, then $A = B/2^{\alpha-\beta}$ is a positive integer, which leads to the least $\alpha - \beta$ significant bits of B are fixed to be all 0's. Therefore we have $B_{\alpha-\beta-1}, \cdots, B_1, B_0 = 0$.

On the other hand, let us suppose $B_{\alpha-\beta} = 0$. Because A is obtained by right shifting $(\alpha-\beta)$ bits of B in binary representation, we have $A_0 = B_{\alpha-\beta} = 0$, which means A is even. Since $N = p \cdot A$, we can get that N is an even number, which contradicts with the known condition. According to the proof by contradiction, this shows that $B_{\alpha-\beta} = 1$, while the other $\gamma - 1$ bits can be randomly chosen. ■

Proof of Lemma 2. Given parameter $\varepsilon = \frac{N}{2^{m+\beta}} \left\lfloor \frac{2^{m+\alpha}}{p} \right\rfloor$, we may write:

$$\varepsilon - 1 < \mu \leq \varepsilon \tag{A-1}$$

$$\frac{N}{2^{m+\beta}} \left(\frac{2^{m+\alpha}}{p} - 1 \right) < \varepsilon < \frac{N}{2^{m+\beta}} \frac{2^{m+\alpha}}{p} \tag{A-2}$$

Since $2^{m+\alpha}$ cannot be divided by p, it is always true that $\left\lfloor \frac{2^{m+\alpha}}{p} \right\rfloor < \frac{2^{m+\alpha}}{p}$, which implies ε is strictly less than B. As $N < 2^{m+\gamma}$, let us rewrite (A-2):

$$B - 2^{\gamma-\beta} < \varepsilon < B \tag{A-3}$$

According to (A-1) and (A-3), we have $B - 1 - 2^{\gamma-\beta} < \mu < B$. The B-value must satisfy the one of the following two cases.

Case 1—B is an integer: If $0 < 2^{\gamma-\beta} < 1$ $(\beta > \gamma)$, we have $\mu = B - 1 = \lfloor B - 1 \rfloor$, thus (2) is satisfied, as it can be observed in Fig. 9(a).

Case 2—B is not an integer: If the fractional part of B is greater than $2^{\gamma-\beta}$, we have $B - 1 - 2^{\gamma-\beta} > \lfloor B - 1 \rfloor$. That is $\mu = \lfloor B \rfloor$, as shown in Fig. 9(b). By the definition of $B = 2^{\alpha-\beta}N/p$, the minimum fractional part of B is $2^{-m+\alpha-\beta}$, as a consequence $2^{\gamma-\beta} < 2^{-m+\alpha-\beta}$ $(\alpha > m + \gamma)$ is required.

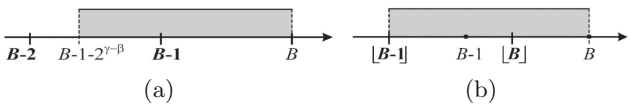

Fig. 9. The value range of μ (a) for B being an integer; (b) for B being a decimal.

Just as a slight clarification, the short thick lines give the position of integers; the dash lines stand for the open intervals; and the shadow areas indicate the value range of μ in Fig. 9. ■

References

1. Bernstein, D.J., Lenstra, A.K.: A general number field sieve implementation. In: Lenstra, A.K., Lenstra, H.W. (eds.) The Development of the Number Field Sieve. Lecture Notes in Mathematics, vol. 1554, pp. 103–126. Springer, Heidelberg (1993)
2. Pollard, J.M.: The lattice sieve. In: Lenstra, A.K., Lenstra, H.W. (eds.) The Development of the Number Field Sieve. Lecture Notes in Mathematics, vol. 1554, pp. 43–49. Springer, Heidelberg (1993)
3. Izu, T., Kogure, J., Shimoyama, T.: CAIRN 2: an FPGA implementation of the sieving step in the number field sieve method. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 364–377. Springer, Heidelberg (2007)
4. Kleinjung, T., Aoki, K., Franke, J., Lenstra, A.K., Thomé, E., Bos, J.W., Gaudry, P., Kruppa, A., Montgomery, P.L., Osvik, D.A., te Riele, H., Timofeev, A., Zimmermann, P.: Factorization of a 768-bit RSA modulus. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 333–350. Springer, Heidelberg (2010)
5. Golliver, R.A., Lenstra, A.K., McCurley, K.S.: Lattice sieving and trial division. In: Huang, M.-D.A., Adleman, L.M. (eds.) ANTS 1994. LNCS, vol. 877, pp. 18–27. Springer, Heidelberg (1994)
6. Shamir, A.: Factoring large numbers with the TWINKLE device. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 2–12. Springer, Heidelberg (1999)
7. Shamir, A., Tromer, E.: Factoring large numbers with the TWIRL device. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 1–26. Springer, Heidelberg (2003)
8. Franke, J., Kleinjung, T., Paar, C., Pelzl, J., Priplata, C., Stahlke, C.: SHARK: a realizable special hardware sieving device for factoring 1024-bit integers. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 119–130. Springer, Heidelberg (2005)
9. Miele, A., Bos, J.W., Kleinjung, T., Lenstra, A.K.: Cofactorization on graphics processing units. In: Batina, L., Robshaw, M. (eds.) CHES 2014. LNCS, vol. 8731, pp. 335–352. Springer, Heidelberg (2014)
10. Kleinjung, T.: Cofactorisation strategies for the number field sieve and an estimate for the sieving step for factoring 1024 bit integers. In: Proceedings of SHARCS, vol. 6 (2006)
11. Behrooz, P.: Computer Arithmetic: Algorithms and Hardware Designs, vol. 19. Oxford University Press, Oxford (2000)
12. Lenstra, A.K., Tromer, E., Shamir, A., Kortsmit, W., Dodson, B., Hughes, J., Leyland, P.: Factoring estimates for a 1024-bit RSA modulus. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 55–74. Springer, Heidelberg (2003)