Revisiting Fermat's Factorization Method

Gajraj Kuldeep and Rune Hylsberg Jacobsen

gkuldeep,rhj{@ece.au.dk}, Aarhus University, Denmark

Abstract. This paper addresses the problem of factoring composite numbers by introducing a novel approach to represent their prime divisors. We develop a method to efficiently identify smaller divisors based on the difference between the primes involved in forming the composite number. Building on these insights, we propose an algorithm that significantly reduces the computational complexity of factoring, requiring half as many iterations as traditional quadratic residue-based methods. The presented algorithm offers a more efficient solution for factoring composite numbers, with potential applications in fields such as cryptography and computational number theory.

1 Introduction

Integer factorization has been one of the most fascinating problems in mathematics and computer science, and it has been studied extensively for centuries. The difficulty of decomposing a large composite number into its prime factors forms the foundation of many cryptographic systems. Among these, the RSA public key algorithm stands out as one of the most well-known applications of integer factorization. RSA relies on the hardness of factorizing the product of two large prime numbers, a task which is computationally infeasible for classical computers when the numbers involved are sufficiently large.

Despite its ancient roots, integer factorization has gained renewed interest in modern times due to its implications in cryptography and security. In fact, the security of much of today's digital communication, including secure web transactions, is underpinned by the assumption that efficient methods for factoring large integers do not exist. As of now, no known classical algorithms can factor large integers efficiently, making factorization a cornerstone of public-key cryptography.

Over the years, several factorization algorithms have been developed, ranging from basic methods like trial division and Fermat's method to more advanced approaches such as the quadratic sieve and the general number field sieve (GNFS), which is currently the fastest known classical algorithm for large numbers [1, 2]. However, with the advent of quantum computing, factorization has become a topic of intense scrutiny. Quantum algorithms, particularly Shor's algorithm, offer a polynomial-time solution to the integer factorization problem, potentially threatening the security of classical cryptographic schemes[3, 4].

Existing methods, such as those based on quadratic residues, often require a significant number of iterations to achieve results. This motivates the need for

more efficient algorithms that can reduce computational effort. In this work, we seek to develop a more effective approach by representing prime divisors and identifying smaller divisors of the difference between the primes. Our aim is to design an algorithm that reduces the number of iterations required, offering a faster alternative to traditional methods.

2 Fermat's Method

Fermat's factorization method is based on the idea that any odd integer N can be represented as the difference of two squares:

$$N = x^2 - y^2 = (x - y)(x + y)$$

Where $N = PQ$ for two primes P and Q, Fermat's method seeks to express N as:

$$N = (x - y)(x + y)$$

This implies:

$$x = \frac{P + Q}{2}, \quad y = \frac{P - Q}{2}.$$

The key idea is to search for integers x and y such that $x^2 - N$ becomes a perfect square. Fermat's method works efficiently if P and Q are close to each other, i.e., when $P \approx Q$. The algorithm proceeds by starting with $x = \lceil \sqrt{N} \rceil$ and incrementing x until $x^2 - N = y^2$ for some integer y. Once such an x and y are found, the factors of N can be computed as:

$$P = x + y, \quad Q = x - y$$

3 Difference Based Method

Let N be a composite odd number such that,

$$N = PQ, \tag{1}$$

where P and Q are primes.

Without loss of generality throughout this article, it is assumed that Q is greater than P.

Since $Q > P$ and both are odd primes, we can write Q in the following way,

$$Q = P + T, \tag{2}$$

where T is an even number. By the placing the value of Q in 1, we get,

$$N = P^2 + PT, \tag{3}$$

First we demonstrate the relation with Fermat's factoring equation, i.e., $N = \left(\frac{Q+P}{2}\right)^2 - \left(\frac{Q-P}{2}\right)^2$. We can write $T = 2t$ for some positive integer t. Now, we define P using t and $\alpha \geq 0$ in the following way,

$$P = \alpha t + s, \tag{4}$$

where s is an integer, i.e., s can take both positive and negative values depending on αt. Similarly we write Q in t and s as,

$$Q = (\alpha + 2)t + s. \tag{5}$$

Using 4 and 5, 1 can be written as,

$$\begin{aligned} N &= \alpha(\alpha+2)t^2 + 2(\alpha+1)ts + s^2 \\ N &= \begin{bmatrix} t & s \end{bmatrix} \begin{bmatrix} \alpha(\alpha+2) & \alpha+1 \\ \alpha+1 & 1 \end{bmatrix} \begin{bmatrix} t \\ s \end{bmatrix} = \mathbf{x}\mathbf{A}\mathbf{x}^T, \end{aligned} \tag{6}$$

where

$$\mathbf{x} = \begin{bmatrix} t & s \end{bmatrix}$$

and

$$\mathbf{A} = \begin{bmatrix} \alpha(\alpha+2) & \alpha+1 \\ \alpha+1 & 1 \end{bmatrix}.$$

It can be noted that the matrix $\mathbf{A}$ is symmetric and invertible.

Using 4, 5, and 6, it is easy to establish link with the Fermat's factoring equation. We have the following relations,

$$\frac{Q-P}{2} = t, \tag{7}$$

and

$$\frac{Q+P}{2} = (\alpha+1)t + s. \tag{8}$$

By adding t^2 to 6, We have

$$\begin{aligned} N + t^2 &= ((\alpha+1)t + s)^2, \\ N &= ((\alpha+1)t + s)^2 - t^2, \\ N &= \left(\frac{Q+P}{2}\right)^2 - \left(\frac{Q-P}{2}\right)^2. \end{aligned} \tag{9}$$

This demonstrates the relation with Fermat's factoring equation.

We can use 6 and solve for t. The solutions are given as,

$$t = \frac{-(\alpha+1)s \pm \sqrt{s^2 + \alpha(\alpha+2)N}}{\alpha(\alpha+2)}. \tag{10}$$

10 can be simplified as,

$$P = \alpha t + s = \frac{s \pm \sqrt{s^2 + \alpha(\alpha+2)N}}{(\alpha+2)}. \tag{11}$$

Similarly solving for s we get,

$$\begin{aligned} s &= -(\alpha+1)t \pm \sqrt{t^2+N}, \\ Q &= (\alpha+2)t + s = t \pm \sqrt{t^2+N} = \frac{T \pm \sqrt{T^2+4N}}{2}. \end{aligned} \tag{12}$$

T is the difference of two odd primes that makes T always even. Therefore, $\alpha = 2$ is chosen to find information about T. In this case $P = 2t + s = T + s$, and inserting the value of P in 3, we get,

$$N = s^2 + 2T^2 + 3sT \tag{13}$$

To find the factors of T and s. Let T and s be defined as,

$$\begin{aligned} T &= Gt_1 + t_0, \\ s &= Gs_1 + s_0, \end{aligned} \tag{14}$$

where G is a positive integer. By inserting these values into 13, we get,

$$N = G^2(2t_1^2 + 3t_1s_1 + s_1^2) + G(4t_1t_0 + 3t_1s_0 + 3s_1t_0 + 2s_1s_0) + (2t_0^2 + 3t_0s_0 + s_0^2),$$
$$(2t_0^2 + 3t_0s_0 + s_0^2) \equiv N((G)). \tag{15}$$

We can find the possible values of t_0 and s_0 for a given G. If there is only one possible value of t_0 or s_0 for given G then there is no ambiguity.

Algorithm 1 Find Special Pairs of t_0 and s_0 for a G

```
ite ← 1
t0Spece ← [ ]
s0Spece ← [ ]
NmodG ← N((G))
for t0 ← 0 to G − 1 do
    for s0 ← 0 to G − 1 do
        if 2 · t0^2 + 3 · t0 · s0 + s0^2((G)) == NmodG then
            t0Spece[ite] ← t0
            s0Spece[ite] ← s0
            ite ← ite + 1
        end if
    end for
end for
```

Let's take $L = \left\lfloor \sqrt{N} \right\rfloor$. We have assumed that $P < Q$, therefore we have $P < L < Q$. Let $a, b \in \mathbf{Z}$ and P, Q, and N are defined as,

$$\begin{aligned} P &= L - a, \\ Q &= L + b, \\ N &= L^2 + L(b-a) - ab. \end{aligned} \tag{16}$$

Let $b - a = r$, and 16 for N can be written as,

$$N = L^2 + Lr - a^2 - ar. \tag{17}$$

Fermat's factoring equation, $N = \left(\frac{Q+P}{2}\right)^2 - \left(\frac{Q-P}{2}\right)^2$ have $Q+P$ and $Q-P$. Using Eq. 16 $Q + P = 2L + r$ and using 5 and 4, we get $Q - P = T$.

To find the factors of r and a. Let r and a are defined as,

$$\begin{aligned} r &= Gr_1 + r_0, \\ a &= Ga_1 + a_0, \end{aligned} \tag{18}$$

where G is a positive number. By inserting these values into 17, we get,

$$(L^2 + Lr_0 - a_0^2 - a_0 r_0) \equiv N((G)). \tag{19}$$

Algorithm 1 can be used to find the values of r_0 and a_0 by changing only the if condition with given in 19.

The focus is on T and r but values s and a are also studied. We list the some interesting values of t_0 and r_0 where they are unique for different RSA numbers for a G [5]. First, we take RSA-250 so that the values can be verified. The t_0 and r_0 for different values of G are given in Table 1.

Table 1. Values of t_0 and r_0 for RSA-250 number

$G = 2$	$G = 3$	$G = 4$	$G = 6$	$G = 8$	$G = 12$	$G = 24$
$t_0 = 0$	$t_0 = 0$	$t_0 = 0$	$t_0 = 0$	$t_0 = 0$	$t_0 = 0$	$t_0 = 0$
$r_0 = 0$	$r_0 = \{0, 1\}$	$r_0 = 0$	$r_0 = \{0, 4\}$	$r_0 = \{0, 4\}$	$r_0 = \{0, 4\}$	$r_0 = \{0, 4, 12, 16\}$

We can write $T = 24t_1$ and $r = 4r_1$ and form additional constraints on P and Q as $Q - P = 0((24))$ and $Q + P = 2((4))$, and Q and P can be written in linear form as,

$$\begin{aligned} Q &= L + 12t_1 + 2r_1 \\ P &= L - 12t_1 + 2r_1 \end{aligned} \tag{20}$$

Similarly, P and Q can also be written as,

$$\begin{aligned} P &= 24p_1 + a, \\ Q &= 24p_1 + 24t_1 + a, \end{aligned} \tag{21}$$

where $a \in \{1, 5, 7, 11, 13, 17, 19, 23\}$ for some positive integer p_1.

We present the values of RSA numbers RSA-160 and RSA-270 in Table 2 and 3.

Table 2. Values of t_0 and r_0 for RSA-260 number

$G = 2$	$G = 3$	$G = 4$	$G = 6$	$G = 8$	$G = 12$	$G = 24$
$t_0 = 0$	$t_0 = 0$	$t_0 = 2$	$t_0 = 0$	$t_0 = \{2, 6\}$	$t_0 = 6$	$t_0 = \{18, 6\}$
$r_0 = 0$	$r_0 = \{1, 2\}$	$r_0 = 2$	$r_0 = \{2, 4\}$	$r_0 = 6$	$r_0 = \{2, 10\}$	$r_0 = \{14, 22\}$

Table 3. Values of t_0 and r_0 for RSA-270 number

$G = 2$	$G = 3$	$G = 4$	$G = 6$	$G = 8$	$G = 12$	$G = 24$
$t_0 = 0$	$t_0 = 0$	$t_0 = 2$	$t_0 = 0$	$t_0 = \{2, 6\}$	$t_0 = 6$	$t_0 = \{18, 6\}$
$r_0 = 0$	$r_0 = \{0, 1\}$	$r_0 = 0$	$r_0 = \{0, 4\}$	$r_0 = 0$	$r_0 = \{0, 4\}$	$r_0 = \{0, 16\}$

Fermat's method has been improved using the quadratic residues. The basic idea is to focus the search for factors by leveraging congruences modulo small numbers, m, which helps reduce the number of trial steps, i.e, try only values of x when $x^2 - N = a((m))$ is quadratic residue to that number, where a is quadratic residue to m. For example the small number is $m = 24$ and we take RSA-250. For this number we have $N = 1((24))$ and $x^2((24))$ should be 1 and $x = 24x_1 + c$, where $c \in 1, 5, 7, 11, 13, 17, 19, 23$. Therefore, there are 8 possible values of c for each try. On other hand if $(2L + r)^2 - 4N$ is checked for perfect square then only 4 tries are needed as can be observed from Table 1. Furthermore, one can write $T^2 = (2L + r)^2 - 4N$.

We demonstrate the advantage of the way of representing the primes in r and T for the RSA numbers. Again we take take the same numbers RSA-250, and RSA-260, and RSA-270 and compare with quadratic residue based method in Fig. 1, 2, 3. Simulations are shown for $G = 100$ to $G = 199$ for all figures. We observe the following either the number of tries reduce by half or remain same as compared to the quadratic residue based methods.

Now we propose an algorithm to factor N based on G. Suppose we have found all the possible values of r_0s for a large G and it is represented as $\mathbf{R}$. The factoring algorithm is given as Algorithm 2.

A large value of G should be chosen in a way that the possible values of r_0 is small. For different value of N, the exercise to find large G with small possible values of r_0 has to be done again, because the value of G is depended on N.

4 Conclusion

In this paper, we present a method to represent the prime divisors of a composite number. Additionally, we propose an approach to identify the smaller divisors

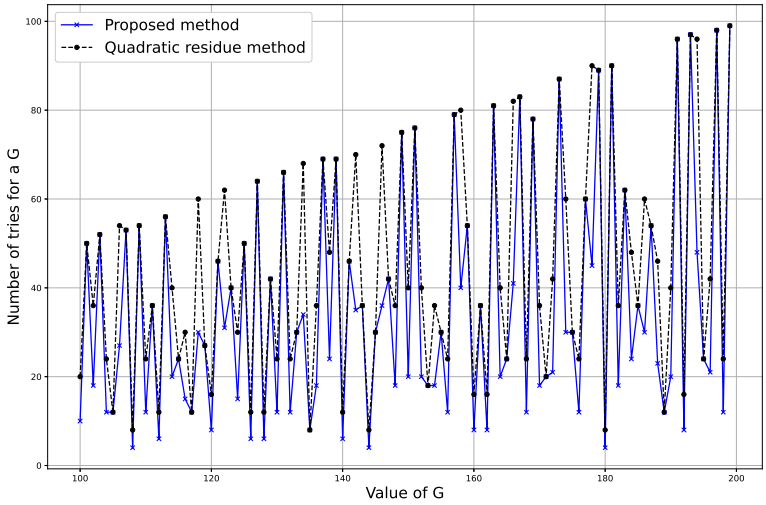

Fig. 1. Comparison between improvements using Quadratic residue method and proposed method for RSA-250

Algorithm 2 Factoring algorithm based on the G

1: $r_1 \leftarrow 0$
2: **while** $r_0 \in \mathbf{R}$ **do**
3: **if** $(2L + Gr_1 + r_0)^2 - 4N$ is square **then**
4: $f^2 \leftarrow (2L + Gr_1 + r_0)^2 - 4N$
5: $P \leftarrow gcd(f + 2L + Gr_1 + r_0, N)$
6: $Q \leftarrow gcd(-f + 2L + Gr_1 + r_0, N)$
7: **end if**
8: **end while**
9: $r_1 \leftarrow r_1 + 1$
10: Goto step 2.

of the difference between the primes forming the composite number. Finally, we introduce an algorithm for factoring composite numbers, demonstrating that it requires half the iterations compared to the quadratic residue-based method.

References

1. J. Mckee: Speeding Fermat's factoring method, Math. Comp. 68 (1999), 1729-1737.
2. W. R. Alford and C. Pomerance: Implementing the self initializing quadratic sieve on a distributed network, Number Theoretic and Algebraic Methods in Computer

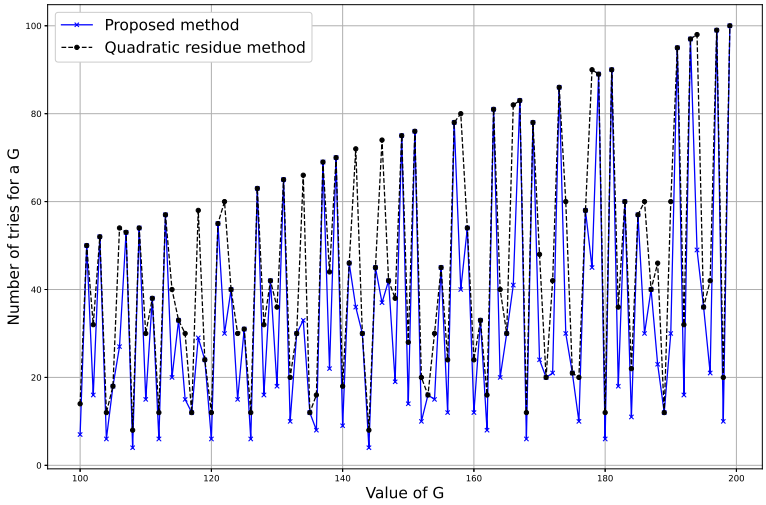

Fig. 2. Comparison between improvements using Quadratic residue method and proposed method for RSA-260

Science (Moscow) (A. van der Poorten, I. Shparlinski, and H. G. Zimmer, eds.), 1993, pp. 163–174.
3. P. Shor: Algorithms for quantum computation: discrete logarithms and factoring, Proceedings of the Thirty-Fifth Annual Symposium on the Foundations of Computer Science, 1994, pp. 124–134.
4. Jr., S., S., Wagstaff: The Joy of Factoring. Student Mathematical Library, vol. 68 Providence, Rhode Island: Amer. Math. Soc.
5. RSA Numbers, https://en.wikipedia.org/wiki/RSA_numbers, last accessed 2024/9/10.

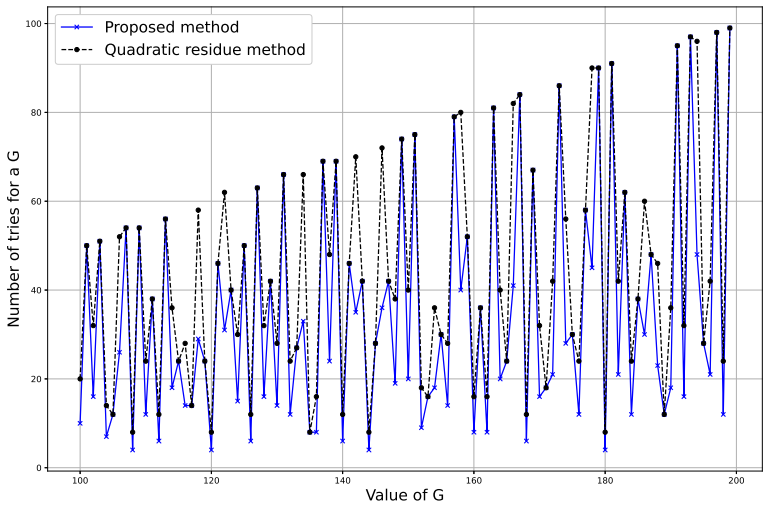

Fig. 3. Comparison between improvements using Quadratic residue method and proposed method for RSA-270