IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 20, 2019, accepted November 11, 2019, date of publication November 15, 2019,
date of current version December 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2953755

The Integer Factorization Algorithm With Pisano Period

LIANGSHUN WU[1,2], H. J. CAI[1,2], AND ZEXI GONG[2,3]
[1]School of Computer Science, Wuhan University, Wuhan 430079, China
[2]Zall Smart Commerce Research Institute, Wuhan 430000, China
[3]Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115, USA

Corresponding author: H. J. Cai (hjcai@whu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61832014.

ABSTRACT Large integer factorization is one of the basic issues in number theory and is the subject of this paper. Our research shows that the Pisano period of the product of two prime numbers (or an integer multiple of it) can be derived from the two prime numbers themselves and their product, and we can therefore decompose the two prime numbers by means of the Pisano period of their product. We reduce the computational complexity of modulo operation through the ''fast Fibonacci modulo algorithm'' and design a stochastic algorithm for finding the Pisano periods of large integers. The Pisano period factorization method, which is proved to be slightly better than the quadratic sieve method and the elliptic curve method, consumes as much time as Fermat's method, the continued fractional factorization method and the Pollard p-1 method on small integer factorization cases. When factoring super-large integers, the Pisano period factorization method has shown as strong performance as subexponential complexity methods; thus, this method demonstrates a certain practicability. We suggest that this paper may provide a completely new idea in the area of integer factorization problems.

INDEX TERMS Fibonacci, integer factorization, Pisano periods, RSA.

I. INTRODUCTION

Large integer factorization is one of the basic issues in number theory. It has been studied for hundreds of years and has been a topic of great interest to mathematicians, computer scientists and cryptographers in recent decades. Integer factorization is not only the most direct attack means on the asymmetric cryptographic algorithm RSA, but also the most critical entry point for RSA security analysis. Therefore, any progress in large integer factorization will attract the attention of the cryptography community.

RSA's asymmetry comes from the invertible operation of modular exponentiation, and its security is mainly based on the practical difficulty of integer factorization. Whether the security of RSA is equivalent to integer factorization has not been proven theoretically since there is no proof that breaking RSA requires large integer factorization. For example, Wang and Yan [1] proposed a new quantum algorithm for attacking RSA based on the quantum Fourier transform and variable substitution by using the fixed point property of RSA, which does not need to factorize n but directly restores its plaintext M from the RSA ciphertext C. Many other ways of cracking RSA have been proven to be feasible, including but not limited to the chosen ciphertext attack, the public module attack, the small exponential attack, the Wiener attack, the Coppersmith theorem attack, and the side-channel attack [2]. Integer factorization is the most obvious and the most plausible means of attacking. In number theory, the problem of large integer factorization can be summarized as follows: for a known integer $N = p_1p_2$, where p_1 and p_2 are two prime numbers, how do we calculate the values of p_1 and p_2?

Common large integer factorization methods include Fermat's factorization method, continued fractional factorization method (abbreviated as CF) [3], the quadratic sieve method (abbreviated as QS) [4], Pollard's p-1 algorithm [5], the elliptic curve factorization method (abbreviated as EC) [6] and the general number field sieve method (abbreviated as GNFS) [7]. Until now, the largest factored RSA number was 768 bits long (known as RSA-768, which has 232 decimal digits) by GNFS. In 1994, Peter Shor showed that a quantum

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Chi Chen.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

computer—if one could ever be created for the purpose—would be able to factor large integers in polynomial time (with the time complexity of $o(\eta^2(\lg\eta)(\lg\lg\eta))$ for factorizing an η-bit long number), thereby breaking RSA [8]. The basic principle of Shor's algorithm is to use the quantum Fourier transform to transform the problem of large integer factorization into finding the period of a function. Some testbed quantum computing platforms are already available, such as cloud quantum computers from IBM and computers based on nuclear magnetic resonance (NMR) [9]. Geller and Zhou [10] employed Fermat numbers and eight qubits to decompose 51 and 85, the largest numbers to be factored by Shor's algorithm thus far. However, $2k+1$ qubits are required to factor k-bit integers, so the scalability of universal quantum devices limits their application on a large scale [11]. Another promising approach to integer factorization is quantum adiabatic computing (QAC), which was first proposed by Burges [12]. Jiang *et al.* [13] recently proposed a generalized quadratic unconstrained binary optimization (QUBO) model to represent the multiplication tables and was able to factor 376298. Peng *et al.* [14] (Jan. 2019) factored 1005973 with only 89 qubits, which is the largest integer factored using quantum factorization to date.

From a universal quantum computation perspective [15], existing quantum computers are not yet able to learn practical quantum factorization algorithms.

The sequence F_n satisfying $F_0=0, F_1=1,\ldots,F_{n+1}=F_n+F_{n-1}, n\in\mathbb{Z}, n\geq 1$ is called the Fibonacci sequence, where F_n denotes the nth Fibonacci number. The series obtained by taking each element of the Fibonacci sequence modulo any modulus is periodic, and these periods are known as Pisano periods [16], [17]. However, there are few studies on the relationship between Pisano periods and integer factorization. In this paper, the Pisano period of the product of two prime numbers will be studied in-depth and associated with large integer factorization.

II. PISANO PERIOD

A. GENERAL PROPERTIES

Let

$$A=\begin{bmatrix}1 & 1\\ 1 & 0\end{bmatrix}. \tag{1}$$

It is easy to show that

$$A^n=\begin{bmatrix}F_{n+1} & F_n\\ F_n & F_{n-1}\end{bmatrix}, n\geq 1, \tag{2}$$

which is the matrix representation of the Fibonacci sequence [18].

Theorem 1: For any nonnegative integer k, the following equations hold.

a) $F_0+F_1+F_2+\ldots+F_n=F_{n+2}-1$.

b) $F_n=F_{k+1}F_{n-k}+F_kF_{n-k-1}(n\geq k)$; in particular, $F_{2n+1}=F_{n+1}^2+F_n^2$.

c) $F_{n-1}F_{n+1}-F_n^2=(-1)^n$; from this we can see that the adjacent Fibonacci numbers are coprime.

d) $F_{(n+2)k}=(F_{k-1}+F_{k+1})F_{(n+1)k}-(-1)^kF_{nk}, n\geq 0$; in particular, $F_{2k}=(F_{k-1}+F_{k+1})F_k$.

Proof: From (1), we have $(I-A)^{-1}=-A$; it follows from the fact that $I+A+\ldots+A^n=(I-A)^{-1}(I-A)(I+A+\ldots+A^n)=(-A)(I-A^{n+1})=A^{n+2}-A$ that a) holds.

Equation b) can be obtained by matrix multiplication.

Since $\det A=-1$, we obtain c) from the fact that $\det A^n=(\det A)^n=(-1)^n$.

Let $M=A^k$. By the Cayley-Hamilton theorem, the characteristic polynomial of M equals 0, i.e., $\varphi(M)=M^2-T\cdot M+\Delta\cdot I=0$, where $T=tr(M)$, $\Delta=\det M$. Therefore

$$M^2=T\cdot M-\Delta\cdot I. \tag{3}$$

Multiplying by M^n on both sides of the equation, we have

$$M^{n+2}=T\cdot M^{n+1}-\Delta\cdot M^n. \tag{4}$$

Replacing M with A^k, we have

$$A^{(n+2)k}=T\cdot A^{(n+1)k}-\Delta\cdot A^{nk}, \tag{5}$$

that means that

$$\begin{bmatrix}F_{(n+2)k+1} & F_{(n+2)k}\\ F_{(n+2)k} & F_{(n+2)k-1}\end{bmatrix}=T\begin{bmatrix}F_{(n+1)k+1} & F_{(n+1)k}\\ F_{(n+1)k} & F_{(n+1)k-1}\end{bmatrix}-\Delta\begin{bmatrix}F_{nk+1} & F_{nk}\\ F_{nk} & F_{nk-1}\end{bmatrix}. \tag{6}$$

By equating the cells in the matrix, we have

$$F_{(n+2)k}=T\cdot F_{(n+1)k}-\Delta\cdot F_{nk}. \tag{7}$$

Because $T=tr(A^k)=F_{k-1}+F_{k+1}$ and $\Delta=\det A^k=(\det A)^k=(-1)^k$, d) holds naturally.

The nth Pisano period, written $\pi(n)$, is the period with which the sequence of Fibonacci numbers taken modulo n repeats. For example, the sequence of Fibonacci numbers modulo 3 begins: 0, 1, 1, 2, 0, 2, 2, 1, 0, 1, 1, 2, 0, 2, 2, 1, 0, 1, 1, 2, 0, 2, 2, 1, 0, ...This sequence has period 8, so $\pi(3)=8$. For convenience, we refer to the remainders of the Fibonacci sequence modulo an integer $m(m\geq 1)$ as the "Pisano sequence", denoted $\{F_n(m)\}$, i.e., the nth term of the sequence is $F_n(m)$, where $F_n(m)$ means $F_n \pmod m$.

Define $d(m)=\min\{n|n\geq 1, m\mid F_n\}$.

Theorem 2: $m\mid F_n \Leftrightarrow d(m)\mid n$

Proof: Since $F_0=0$, suppose $m\mid F_{kd(m)}$ and $m\mid F_{(k+1)d(m)}$. Then obviously $m\mid F_{(k+2)d(m)}$ (Let the first subscript k be $k+1$; then, $F_{(k+2)d(m)}=F_{((k+1)+1)d(m)}$, and thus, $m\mid F_{(k+2)d(m)}$ holds inevitably). Let $n=kd(m)$; by mathematical induction, we have $d(m)\mid n\Rightarrow m\mid F_n$.

If instead $m\mid F_n$ but $d(m)\nmid n$, then $n=ad(m)+r$, where a,r are integers satisfying $a\geq 0, 0<r<d(m)$. Obviously $m|F_{ad(m)}$ holds since we have proved $d(m)\mid n\Rightarrow m\mid F_n$, and from Theorem 1 b), $0\equiv F_n=F_{ad(m)}F_{r+1}+F_{ad(m)-1}F_r\equiv F_{ad(m)-1}F_r \pmod m$. From Theorem 1 c) we get $gcd(F_{ad(m)},F_{ad(m)-1})=1$; thus $gcd(m,F_{ad(m)-1})=1$ and therefore $m\mid F_r$. This contradicts the definition of $d(m)$, hence $m\mid F_n\Rightarrow d(m)\mid n$.

It is not difficult to deduce the following corollaries from the above theorem.

Corollaries: For positive integers n_1, n_2, m_1, m_2, we have
a) $n_1 \mid n_2 \Leftrightarrow F_{n_1} \mid F_{n_2}$.
b) $m_1 \mid m_2 \Rightarrow d(m_1) | d(m_2)$.
c) When $gcd(m_1, m_2) = 1$, $d(m_1 m_2) = lcm(d(m_1), d(m_2))$.
d) When $m = p_1^{\alpha_1} \dots p_s^{\alpha_s}$ is the standard decomposition of m, $d(m) = lcm\left(d(p_1^{\alpha_1}), \dots, d(p_s^{\alpha_s})\right)$.

Theorem 3: $d(m) \mid \pi(m)$ and the ratio $r = \pi(m)/d(m)$ satisfies $r = \min\left\{s | s \geq 1, F_{sd(m)-1} \equiv 1 \pmod m\right\}$.

Proof: Since $F_{\pi(m)}(m) = F_0(m) = 0$, then $m \mid F_{\pi(m)}$, thus $d(m) \mid \pi(m)$ follows from Theorem 2.

Assume $t = F_{d(m)-1}(m)$. Then, we have $t = F_{d(m)+1}(m)$ and $t = F_{d(m)+2}(m)$. Thus, $t = F_{d(m)+k}(m) = tF_k(m) \pmod m$, where $0 \leq k \leq d(m) - 1$. Let $k = d(m) - 1$; then $tF_{d(m)-1}(m) \equiv t^2 \pmod m$ and $F_{2d(m)+1}(m) \equiv t^2 \pmod m$. Repeat this process to get $F_{rd(m)+1}(m) = F_{rd(m)+2}(m) \equiv t^r \equiv 1 \pmod m$.

Therefore, $F_{rd(m)+n}(m) = F_n(m) (n \geq 0)$, and Theorem 3 is proven.

The significance of Theorem 3 is that it fully reveals the structure of the Pisano sequence. Still using the notation $t = F_{d(m)+1}(m)$, the Pisano sequence within one period consists of the following r segments:

$$\begin{cases} 0, 1, 1, F_3, \dots, F_{d(m)-1} \equiv t \pmod m \\ 0, t, t, tF_3, \dots, tF_{d(m)-1} \equiv t^2 \pmod m \\ \dots \\ 0, t^{r-1}, t^{r-1}, t^{r-1}F_3, \dots, t^{r-1}F_{d(m)-1} \equiv t^r \pmod m \end{cases} \tag{8}$$

It can be seen that the subsequence, $\{F_{kd(m)+j}(m)\}, k \geq 0, 0 \leq j \leq d(m) - 1$, of the Pisano sequence is a geometric series modulo m for a fixed j, with $F_j(m)$ as the first term and t as the common ratio: $F_{kd(m)+j}(m) \equiv F_j(m)t^{k-1} \pmod m$. Therefore, the sequence $\{F_n(m)\}$ is a rearrangement of the items from the $d(m)$ geometric series modulo m. Specifically, these series own $F_0(m), F_1(m), \dots, F_{d(m)-1}(m)$ as their first terms with the same common ratio t.

Definition: For the integer sequence $\{\omega_n\}$, an integer m greater than 1, an integer s greater than 0, a nonnegative integer n_0 and an integer c, if $gcd(m, c) = 1$, $n \geq n_0$, $\omega_{n+s} \equiv c\omega_n \pmod m$, then the minimum positive integer s satisfying the above equation is called the **constraint period** of $\{\omega_n \pmod m\}$, the corresponding n_0 is called the preliminary constraint period, and c is called a **multiplier** [19].

From the above definition, we can see that $\{F_n(m)\}$ is a purely constrained periodic sequence with constraint period $d(m)$ and multiplier $c = t$.

Next, we give the values of r in Theorem 3.

Theorem 4: Suppose $\varepsilon = (1 + \sqrt{5})/2$, $\bar{\varepsilon} = (1 - \sqrt{5})/2$, and $F_n{}'(n \geq 0)$ are integers satisfying $F_n{}' = \varepsilon^n + \bar{\varepsilon}^n$. Then the values of r in Theorem 3 are

$$r = \begin{cases} 4, & 2 \nmid d(m) \\ 1, & 2 \parallel d(m) \text{ and } m | F'_{d(m)/2} \\ 2, & 4 \mid d(m). \end{cases} \tag{9}$$

Proof: For a nonnegative integer n and positive integer s,

$$\begin{aligned} F_s F'_{n-s} &= \frac{(\varepsilon^s - \bar{\varepsilon}^s)(\varepsilon^{n-s} + \bar{\varepsilon}^{n-s})}{\varepsilon - \bar{\varepsilon}} \\ &= \frac{(\varepsilon^n - \bar{\varepsilon}^n) - (\varepsilon\bar{\varepsilon})^s(\varepsilon^{n-2s} + \bar{\varepsilon}^{n-2s})}{\varepsilon - \bar{\varepsilon}} \end{aligned} \tag{10}$$

From (10), we have

$$F_n \equiv (-1)^s F_{n-2s} \pmod{F_s F'_{n-s}}. \tag{11}$$

Let $s = 2d(m)$. Since $m \mid F_{d(m)}$, $F_n \equiv F_{n+4d(m)} \pmod m$ follows from (11), so $\pi(m) \mid 4d(m)$, i.e., $r \mid 4$. When $2 \nmid d(m)$, let $s = d(m)$; then from (11) we know that there is a n for which $F_n \not\equiv F_{n+2d(m)} \pmod m$, so $r > 2$ and $r = 4$. When $2 \mid d(m)$, let $s = d(m)$ then it follows from (11) that $r \mid 2$. Let $s = d(m)/2$. Then, (12) follows from (11):

$$F_n - (-1)^{d(m)/2} F_{n-d(m)} = F_{d(m)/2} F'_{n-d(m)/2}. \tag{12}$$

According to the definition of $d(m)$, $m \nmid F_{d(m)/2}$ is known. It follows from (12) that when $4 \mid d(m)$, $r > 1$ and $r = 2$. When $2 \parallel d(m)$, then (as in the argument above using (12)) we have

$$F_n - F_{n-d(m)} = F_{d(m)/2} F'_{n-d(m)/2}. \tag{13}$$

Therefore, if $m | F'_{d(m)/2}$, then $r = 1$ holds due to (13); otherwise $r = 2$.

B. PISANO PERIOD OF A PRIME NUMBER

We know that for the prime number p, $p \mid F_{p-(\frac{p}{5})}$, where $(\frac{p}{5})$ denotes the Legendre symbol [20], i.e., $(\frac{p}{5})$ is defined as

$$(\frac{p}{5}) = \begin{cases} 0, & p \equiv 0 (mod\ 5) \\ 1, & p \not\equiv 0 (mod\ 5), \exists x \in \mathbb{Z}, x^2 \equiv p (mod\ 5) \\ -1, & p \not\equiv 0 (mod\ 5), \nexists x \in \mathbb{Z}, x^2 \equiv p (mod\ 5). \end{cases}$$

Since $d(p) = \min\{n | n \geq 1, p \mid F_n\}$, we obtain $d(p) \mid p - (\frac{p}{5})$ from Theorem 2. Theorem 3 shows that the Pisano period is r times the constrained period, i.e., $\pi(p) = rd(p)$. Therefore,

$$\pi(p)/r \mid p - (\frac{p}{5}). \tag{14}$$

i.e.,

$$p - (\frac{p}{5}) = k \cdot \pi(p)/r. \tag{15}$$

and hence

$$p - (\frac{p}{5}) = (k/r) \cdot \pi(p). \tag{16}$$

where k is a positive integer not less than 1, i.e., $k \in \mathbb{Z}^+$. Theorem 4 tells us that r has three possible values: 4, 1, 2.

C. PISANO PERIOD OF THE PRODUCT OF TWO PRIME NUMBERS

Let N be the product of two prime numbers p_1 and p_2, $p_1 \neq p_2$.

Theorem 5: If $N = p_1p_2$, $p_1 \neq p_2$, then $\pi(N) = lcm(\pi(p_1), \pi(p_2))$,where $\pi(N)$ is the Pisano period of N, $\pi(p_1)$ is the Pisano period of p_1 and $\pi(p_2)$ is the Pisano period of (p_2).

Proof: According to the definition of the Pisano period, let $t = \pi(N)$. Then, we have

$$F_{n+t} \equiv F_n \pmod{N}, N = p_1p_2. \tag{17}$$

Therefore,

$$F_{n+t} \equiv F_n \pmod{p_1p_2} \equiv F_n \pmod{p_1 \text{ and } p_2}. \tag{18}$$

That means $\pi(p_1) \mid t$ and $\pi(p_2) \mid t$. Let s denote their least common multiple $lcm(\pi(p_1), \pi(p_2))$; then, we have $s \mid t$.

Conversely, if $F_{n+s} \equiv F_n \pmod{p_1 \text{ and } p_2}$, then since p_1 and p_2 are coprime, $F_{n+s} \equiv F_n \pmod{p_1p_2}$, i.e., $t \mid s$. Therefore, $t = s$. Li *et al.* [21] noted that the Pisano period of N is a multiple of 4 and is equal to two times its constraint period, that is, $\pi(N) = 2d(N)$. We give Theorem 6 without proof.

Theorem 6: If p_1 and p_2 are odd prime numbers and $N = p_1p_2$, then $r = 2$, i.e., $\pi(N) = 2d(N)$ and $4 \mid \pi(N)$.

Combined with Theorem 3, Theorem 6 essentially indicates that the Pisano period of p_1p_2 consists of the following two parts:

$$\begin{cases} 0, & 1, 1, F_3, \ldots, F_{d(N)-1} \\ 0, & t, t, tF_3, \ldots, tF_{d(N)-1} \end{cases} \pmod{N}. \tag{19}$$

where $t = F_{d(N)-1} \pmod{N}$.

We now present an example to confirm this conclusion. Let $p_1 = 17$, $p_2 = 19$, $N = 17 \times 19 = 323$. Then, the Pisano sequence $\{F_n(N)\}_{n\geq 0}$ is { 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 54, 287, 18, 305, 0, 305, 305, 287, 269, 233, 179, 89, 268, 34, 302, 13, 315, 5, 320, 2, 322, 1,… }. We find that the Pisano period $\pi(323) = 36$, the constraint period $d(323) = 18$, and $\pi(323) = 2 \cdot d(323)$, i.e., $r = 2$, $4 \mid \pi$ and the multiplier $t = 305$.

III. INTEGER FACTORIZATION ALGORITHM USING THE PISANO PERIOD

From Theorem 5, we know that $\pi(N) = lcm(\pi(p_1), \pi(p_2))$. Let $\pi_1 = \pi(p_1)$, $\pi_2 = \pi(p_2)$, $\pi = \pi(N)$; then, $\pi_1\pi_2 = gcd(\pi_1, \pi_2) \cdot \pi$. Let $k = gcd(\pi_1, \pi_2)$.

Using (14), we have $\pi_1 = (r_1/k_1) \cdot \left(p_1 - (\frac{p_1}{5})\right)$, $\pi_2 = (r_2/k_2) \cdot \left(p_2 - (\frac{p_2}{5})\right)$. Then

$$(kk_1k_2/r_1r_2) \cdot \pi = \left(p_1 - (\frac{p_1}{5})\right)\left(p_2 - (\frac{p_2}{5})\right). \tag{20}$$

where k,k_1 and k_2 are positive integers not less than 1 and the set of values of r_1,r_2 is {4,1,2}.

Let $\pi' = (kk_1k_2/r_1r_2) \cdot \pi$. Then (16) becomes

$$\pi' = \left(p_1 - (\frac{p_1}{5})\right)\left(p_2 - (\frac{p_2}{5})\right). \tag{21}$$

Only when the prime number p is equal to 5 does $(\frac{p}{5}) = 0$. When $p \neq 5$, it is easy to see that $(\frac{p}{5}) = 1 \Leftrightarrow p \pmod{10} \equiv \{1, 9\}$, $(\frac{p}{5}) = -1 \Leftrightarrow p \pmod{10} \equiv \{3, 7\}$.

When $p_1 \pmod{10} \equiv \{1, 9\}$, $p_2 \pmod{10} \equiv \{1, 9\}$(we denote this case $\{1, 1\}$), when $p_1 \pmod{10} \equiv \{3, 7\}$, $p_2 \pmod{10} \equiv \{1, 9\}$(we denote this case $\{-1, 1\}$), when $p_1 \pmod{10} \equiv \{1, 9\}$, $p_2 \pmod{10} \equiv \{3, 7\}$(we denote this case $\{1, -1\}$), or when $p_1 \pmod{10} \equiv \{3, 7\}$, $p_2 \pmod{10} \equiv \{3, 7\}$(we denote this case $\{-1, -1\}$), then

$$\pi' = p_1p_2 - p_1 - p_2 + 1, when \quad \{1, 1\}, \tag{22a}$$
$$\pi' = p_1p_2 + p_1 - p_2 - 1, when \quad \{-1, 1\}, \tag{22b}$$
$$\pi' = p_1p_2 - p_1 + p_2 - 1, when \quad \{1, -1\}, \tag{22c}$$
$$\pi' = p_1p_2 + p_1 + p_2 + 1, when \quad \{-1, -1\}, \tag{22d}$$
$$N = p_1p_2. \tag{22e}$$

The problem of integer factorization is as follows: Given the product N of two prime numbers p_1, p_2, how can the values of p_1 and p_2 be calculated from N?

We design the following algorithm:

Step 1: find π'.

Step 2: determine the values of p_1 by (22a)(22e), (22b)(22e), (22c)(22e) and (22d)(22e); then, we obtain:

$$p_1 = \left(N - \pi' + 1 \pm \sqrt{(N - \pi' + 1)^2 - 4N}\right)/2, \tag{23a}$$
$$p_1 = \left(N - \pi' - 1 \pm \sqrt{(N - \pi' - 1)^2 + 4N}\right)/2, \tag{23b}$$
$$p_1 = \left(\pi' - N + 1 \pm \sqrt{(N - \pi' - 1)^2 + 4N}\right)/2, \tag{23c}$$
$$p_1 = \left(\pi' - N - 1 \pm \sqrt{(N - \pi' + 1)^2 - 4N}\right)/2. \tag{23d}$$

Calculate (23a) (23b) (23c) (23d) in turn; then, p_2 is calculated by $p_2 = N/p_1$. If the values of p_1 and p_2 are both integers, output p_1 and p_2.

We refer to this algorithm as the "Pisano period factorization algorithm" (abbreviated as PP). The pseudocode of PP is represented as follows:

Algorithm 1 Pisano Period Factorization Algorithm

Input: π';
Output: p_1, p_2;
1: **function** PisanoPeriodFactorization(π')
2: calculate p_1 by (23a) (23b) (23c) (23d), $p_2 = N/p_1$;
3: **if** p_1, p_2 are both integers **then**
4: **return** p_1, p_2;
5: **end if**
6: **end function**

For example, suppose $N = 256961$, $\pi' = 258132$. We wish to find the values of p_1 and p_2.

Solution: By calculating (23a) (23b) (23c) (23d) successively, we find that only the value of p_1, that is obtained from (23d) is an integer 293; p_2 equals 877 in this circumstance.

IV. PISANO PERIOD SEARCHING ALGORITHM

In section III, it is noted that if the Pisano period of the product of two prime numbers is known, we can easily factor the product. In this section, we will discuss how to find the Pisano period.

A. FAST FIBONACCI MODULO ALGORITHM

First, we need to study how to obtain the nth item of the Fibonacci sequence as quickly as possible. There are three algorithms: the recursive algorithm, the loop algorithm, and the fast doubling algorithm [22]. The time complexities of these three algorithms are $o(\phi^n)$, $o(n)$, $o(\log n)$, i.e., the complexity successively decreases, with the recursive algorithm being the slowest and the fast doubling algorithm being the fastest.

$$F_{2k} = F_k\left(2F_{k+1} - F_k\right). \tag{24}$$

$$F_{2k+1} = F_{k+1}^2 + F_k^2. \tag{25}$$

Equation (24) is actually Theorem 1 d), and (25) is Theorem 1 b).

To find the value of the nth item of the Fibonacci sequence modulo an integer d, we can derive a fast modulo algorithm. By (24) and (25), we have

$$F_{2k}(mod\ d) = [F_k(mod\ d)(2F_{k+1} - F_k)(mod\ d)] \times (mod\ d), \tag{26}$$

$$F_{2k+1}(mod\ d) = [F_k^2(mod\ d) + F_{k+1}^2(mod\ d)] \times (mod\ d). \tag{27}$$

Using (26) and (27), we can design a fast algorithm for finding the nth item of the Fibonacci sequence modulo an integer d. The algorithmic logic can be expressed as the following sequence of steps.

Step 1: determine whether n is even; if yes, execute Step 2; if not, execute Step 3.

Step 2: calculate $F_n(mod\ d) = [F_{n/2}(mod\ d)(2F_{n/2+1} - F_{n/2}(mod\ d))](mod\ d)$.

Step 3: calculate $F_n(mod\ d) = [F_{n/2+1}^2(mod\ d) + F_{n/2}^2(mod\ d)](mod\ d)$.

Repeat steps 1, 2 and 3 recursively.

This method converts the operation of the nth item of the Fibonacci sequence modulo an integer d into the operation of the $n/2$th(or $(n/2+1)$th) item. This constant "take half" approach reduces the complexity of the modulo operation to $o(\log n)$.

For convenience, this algorithm is called the "fast Fibonacci modulo algorithm" in this paper. To facilitate understanding and implementation the algorithm, we give the pseudocode of "fast Fibonacci modulo algorithm" below.

Algorithm 2 Fast Fibonacci Modulo Algorithm

Input: index of Fibonacci sequence, n; modulus, m;
Output: results with an even/odd sequence index;

```
function FibonacciModulo(n,m)
    a, b = FibonacciModulo(⌊n/2⌋,m);
    c = [a(mod m) · (2b − a)(mod m)](mod m);
    d = [a^2(mod m) + b^2(mod m)](mod m);
    if n(mod 2) == 0 then
        return c, d;
    else
        return d, c + d;
    end if
end function
```

B. PISANO PERIOD SEARCHING ALGORITHM FOR LARGE INTEGERS

Bauer has given a computer algorithm and provided the program for finding the Pisano period of prime numbers [23], but there is currently no general method for finding the Pisano period of the product of two prime numbers or even arbitrary integers [19].

Equations (23a),(23b),(23c) and (23d) tell us that $\pi' = N \pm p_2 \pm p_1 \pm 1$, π' is an integral multiple of π (or constraint period, $d(m)$). Therefore, π' is also the period of the Pisano sequence $\{F_n(N)\}_{n\geq 0}$. The value of π' is approximately in the range of $[N - \max\{p_1, p_2\}, N + \max\{p_1, p_2\}]$, which means that the searching range for π' is determined by the magnitude of the larger prime number.

As a stochastic algorithm, Shor's algorithm increases the factorization success rate by increasing the number of experiments. Following this algorithm, random numbers were adopted to apply to the design of the searching algorithm for the Pisano period.

Therefore, it can be seen that if a number r is randomly selected within the searching range, the algorithm should calculate $F_r(N)$; if $F_r(N)$ appears in a previous location s, we can determine a possible period, $r - s$. If verified wrong, indicating that there is a repetition in one period, then the algorithm searches again.

This is a bit like a "birthday paradox"; in our intuition, the probability of a value in the searching range colliding with one in the sorting range is small, but in fact, the probability of such a collision is larger than expected. We will analyze the probability of a collision or hit in subsection C.

Therefore, an algorithm is designed as follows:

Step 1: determine the searching range. Estimate the difference in the decimal digits of p_1 and p_2. If the difference is x and N has θ decimal digits in total, then the magnitude of $\max\{p_1, p_2\}$ is $(\theta + x)/2$ and the searching range is $\left[N - 10^{(\theta+x)/2}, N + 10^{(\theta+x)/2}\right]$.

Step 2: determine the sorting range $[0, k]$. Calculate $F_i(N)$ for all $i \in [0, k]$ and sort them. Maintain a correspondence table of $F_i(N)$ and i.

Step 3: generate a random number r in $[N - 10^{(\theta+x)/2}, N + 10^{(\theta+x)/2}]$, calculate $F_r(N)$ and search $F_r(N)$ in the sorted sequence by the binary search method. If there is a hit, find the location s by the correspondence table of $F_i(N)$ and i. Then, the difference in the two locations, $r - s$, is considered to be π', which is the period we're looking for.

Verify whether the result is correct by checking whether $F_{r-s} \pmod N = 0$. If not, go back to Step 3; otherwise further check whether $F_{r-s+1} \pmod N = 1$. If yes, $\pi' = r - s$; if not, then π' is a constraint period, i.e., $\pi' = 2(r - s)$.

Algorithm 3 Period Searching Algorithm

Input: Digit difference, x; large integer, N; sort length, s;
Output: the Pisano period, π';

```
function PeriodSearch(x,N,x)
    search area = [N − 10^((θ+x)/2), N + 10^((θ+x)/2)];
    sort the first s residuals;
    repeat
        generate r, compute F_r(modN);
        if F_r(modN) hit the sorted residuals then
            π' = r − postion_hit;
        end if
    until π' is verified to be a period
    return π';
end function
```

For example, take $p_1 = 7$, $p_2 = 11$, $N = 77$; assuming the difference between the decimal digits is 1, the searching range should be $[77 - 10, 77 + 10] = [67, 87]$. After the first 30 residuals are ranked, the rank will be 0, 1, 1, 1, 2, 2, 3, 5, 8, 12, 12, 13, 13, 14, 21, 23, 27, 32, 34, 41, 43, 55, 57, 63, 66, 67, 68, 69 and 71, and the corresponding original positions will be 0, 1, 2, 22, 3, 13, 4, 5, 6, 11, 21, 7, 23, 24, 8, 19, 25, 28, 9, 26, 18, 10, 17, 16, 20, 12, 27, 14 and 15. Let r be a randomly chosen number in [67, 87]; for example, when $r = 86$, we have $F_{86} \pmod{77} = 8$. After we search this ranked list 4 times by using the binary search method, the original position of the hit residual is 6, so the period may be $\pi' = 86 - 6 = 80$ (as shown in Figure 1). We find that this period is correct because $F_{80} \pmod{77} = 0$, $F_{81} \pmod{77} = 1$.

C. THE ALGORITHM COMPLEXITY ANALYSIS

As explained above, the time complexity of the fast Fibonacci modulo algorithm is $o\left(\log_2 N\right)$.

Suppose that the length of the sorting area is N_1, sorted by heap sorting or quick sorting method. Thus, the time complexity is $complexity_{sort} = o(N_1 \log_2 N_1) \cdot o\left(\log_2 N_1\right)$, where $o(N_1 \log_2 N_1)$ denotes the complexity of the sorting operation and $o\left(\log_2 N_1\right)$ denotes the complexity of the modulo operation.

Assuming that the length of the search area is N_2, it takes $\sqrt{N_2}$ times on average to traverse any number within the range by using a random method [24]. Assuming that the distribution of the sequence $F_n(N)$ is sufficiently discrete, there are $\pi(N)$ possible values of $F_n(N)$ (which means that there is almost no coincidence within a period). We wish to determine the probability of a collision after $\sqrt{N_2}$ search times in the searching area.

This actually is a "birthday paradox" problem. It can be transformed into the following interesting small example:

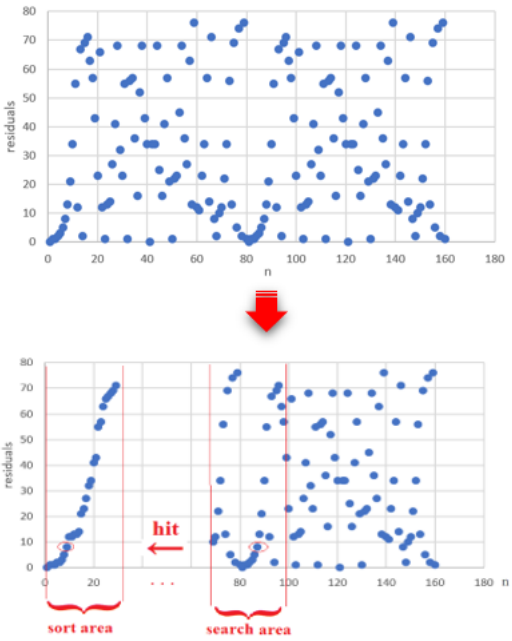

FIGURE 1. A diagram of the Pisano period searching algorithm. For $N = 77$, rank the residuals of the first 30 Fibonacci numbers modulo N. Mark [67, 87] as the search area, generate the random number 86 in [67, 87], and calculate $F_{86} \pmod{77} = 8$. By using the binary search method, we find $F_6 \pmod{77} = 8$. Therefore, the Pisano period is $86 - 6 = 80$.

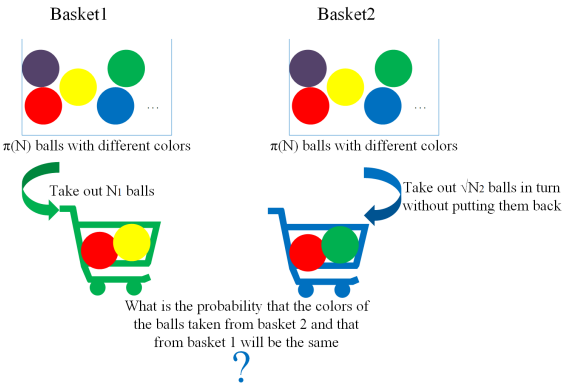

FIGURE 2. The probabilistic problem of the Pisano period searching algorithm is actually a "birthday paradox" problem and can be transformed into the following interesting small example. Both baskets have $\pi(N)$ balls of different colors, N_1 balls from basket 1 are taken out first and then $\sqrt{N_2}$ balls from the basket 2 are taken out in turn without putting them back. What is the probability of color collision of the balls taken from basket 2 and that from basket 1?.

Basket 1 has $\pi(N)$ balls of different colors. First, take out N_1 balls.

Basket 2 also has $\pi(N)$ balls of different colors. Take out $\sqrt{N_2}$ balls in turn without putting them back. What is the probability that the colors of the balls taken from basket 2 and that the colors of the balls from basket 1 will be the same? (Figure 2 illustrates this problem visually.)

The answer is as follows:

$$1 - C_{\pi(N)-N_1}^{\sqrt{N_2}} / C_{\pi(N)}^{\sqrt{N_2}}. \tag{28}$$

This is just the probability of a collision (or hit) after $\sqrt{N_2}$ search times in searching area of PP algorithm.

It is known from the theory of probability analysis that the average time complexity of hitting the sorted residuals sequence is $complexity_{search} = o(\sqrt{N_2}/N_1) \cdot o(\log_2 N_1) \cdot o(\log_2 N)$, where $o(\log_2 N_1)$ represents the time complexity of a binary search, $o(\log_2 N)$ represents the complexity of a modulo operation, and N is the large integer to be decomposed.

Therefore, the total time complexity of searching the Pisano period is $complexity_{sort} + complexity_{search} = o(N_1 \log_2 N_1) \cdot o(\log_2 N_1) + o(\sqrt{N_2}/N_1) \cdot o(\log_2 N_1) \cdot o(\log_2 N)$, which is related to three factors: the length of the sorting range N_1, the length of the searching range N_2 and the larger integer N.

Essentially, the length of the search range is determined by the decimal digit difference x between the two factors to be decomposed, i.e., $N_2 = 2 \cdot 10^{\lg N+x}$. If we assume the length of the sorting range $N_1 = N^{1/6}$ (other alternatives are certainly also permitted), then the computational complexity is as follows: $o(N^{1/6} \log_2 N^{1/6}) \cdot o(\log_2 N^{1/6}) + o(\sqrt{2 \cdot 10^{\lg N+x}}/N^{1/6}) \cdot o(\log_2 N^{1/6}) \cdot o(\log_2 N)$.

V. INTEGER FACTORIZATION EXPERIMENTS

A. COMPLEXITY COMPARISON ANALYSIS

Generally, the principle of integer factorization methods is to find an integer pair so that $x^2 \equiv y^2 \pmod N$ holds, calculate $gcd(x - y, N)$ and decompose N with a probability greater than 1/2. Methods based on this idea include the continued fractional factorization method(CF), the quadratic sieve method(QS) and the general number field sieve method (GNFS) [24].

Let $L = exp(\sqrt{\log N \log\log N})$; then, the computational complexity of L is subexponential. It has been proven that the computational complexity of CF is $L^{1+o(1)}$ ($o(1)$ for an infinitesimal quantity); that of QS is $L^{1+o(1)}$; that of EC is $L = exp((1 + o(1))\sqrt{\log N \log\log N}) = L^{1+o(1)}$; and that of GNFS is $L_N\left(1/3, (64/9)^{1/3+o(1)}\right)$, where $L_N(a, b) = exp\left(b(\log N)^a(\log N)^{1-a}\right)$ [24].

In the GNFS derivation process, the actual time consumed largely depends on the selection of polynomials, the sieving method, the solution of the linear equations, the solution of algebraic square root and the selection of some parameters [25]. In the following comparative study, we give up the comparison with this method. In addition, the implementation of Shor's algorithm relies on the invention of quantum computers and related simulation platforms; the following comparative study does not consider Shor's algorithm for the time being. Table 1 shows the computational complexity of some factorization algorithms. CF, QS, EC, GNFS and PP are all of subexponential time complexity.

B. INTEGER FACTORIZATION EXPERIMENTS

The process of integer factorization based on the Pisano period is as follows:

TABLE 1. Complexity of Some Factorization Algorithms. Suppose N represents the integer to be factored, it can be seen that PP algorithm, like other algorithms (except Fermat's), owns the subexponential complexity.

Algorithm	Complexity[a]
Fermat's	$o\left(\log\sqrt{N}\right)$
QS	$o\left(exp\left((1+o(1))\sqrt{N\log N\log\log N}\right)\right)$
EC	$o\left(exp\left((1+o(1))\sqrt{N\log N\log\log N}\right)\right)$
GNFS	$o\left(exp\left((64/9)^{1/3+o(1)}(\log N)^{1/3}(\log\log N)^2/3\right)\right)$
PP	$o(N^{1/6}\log_2 N^{1/6}) \cdot o(\log_2 N^{1/6}) +$ $o(\sqrt{2 \cdot 10^{\lg N+x}}/N^{1/6}) \cdot o(\log_2 N^{1/6}) \cdot o(\log_2 N)$

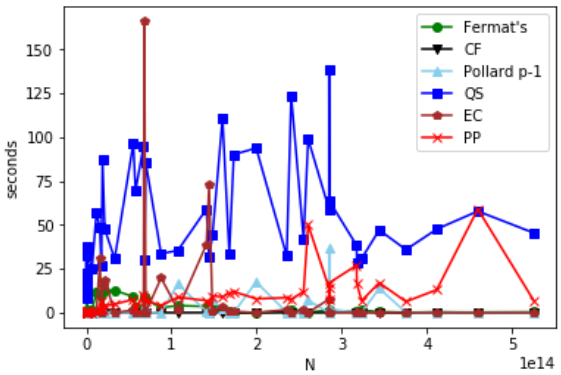

FIGURE 3. For small values of N, in which all the integers to be factored do not exceed 16 decimal digits (≤ 50Bit), the PP algorithm does not perform worse than the baseline algorithms.

Phase 1: Find the period π' by Algorithm 3.

Phase 2: Get p_1,p_2 by Algorithm 1.

In addition to obtaining codes for CF, the Pollard p-1 method, QS and EC on Github (see the acknowledgement section), we also implement Fermat's algorithm and the PP Algorithm proposed in this paper with programming. The experimental environment is a single-core machine with an Intel Xeon E5-2600v4 processor running Ubuntu Linux 14.04 LTS.

1) SMALL INTEGER CASES

Through the Miller-Rabin method, we randomly generate two prime numbers such that the difference between the decimal digits does not exceed 2 and the sum of their decimal digits does not exceed 16 ($\leq$ 50Bit), and then take N as the product of the two. We factor N using the Fermat's, CF, Pollard p-1, QS, EC algorithms and using the PP method proposed in this paper. The parameters of the PP method are as follows: the sort length $N_1 = \max\left\{10000, N^{1/6}\right\}$ and the estimated decimal digit difference
$x = 2$.

According to Figure 3, the time consumption of the PP method is equivalent to that of the Fermat's, CF, and Pollard p-1 algorithms and slightly better than the QS and EC methods in the small integer factorization cases, in which all

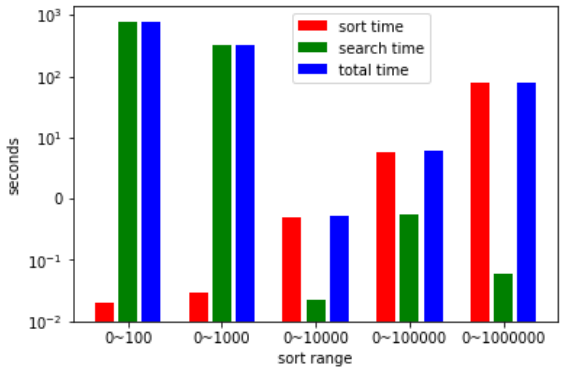

FIGURE 4. The sorting time, searching time and total time taken by the PP algorithm to factor $N = 525220163614031$ when setting different sort ranges. Total time = sort time + search time. It can been seen that as the sorting time increases, the searching time decreases accordingly, and an appropriate choice of sorting range will minimize the total time consumed. The best sorting range is [0, 1000] in this case.

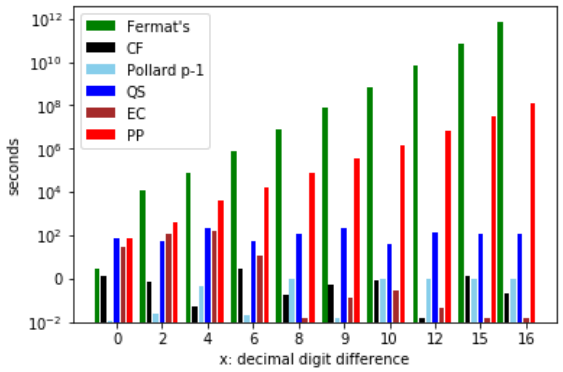

FIGURE 5. A comparison of time consumption for different decimal digit differences. The horizontal axis represents the decimal digit difference of the two prime factors, and the vertical axis represents the time-consuming seconds. Obviously, only Fermat's algorithm and our PP algorithm will consume more time linearly with x, suggesting that these two algorithms are suitable for the factorization of integers with little difference between their two prime factors.

the integers to be factored do not exceed 16 decimal digits ($\leq$ 50 Bit).

2) SORT LENGTH

For $N = 525,220,163,614,031$ (15 decimal digits long), set the sort length to 100, 1000, 10000, 100000 and 1000000. The factorization time is shown in Figure 4. It can be seen that as the sorting time increases, the searching time decreases accordingly; the total time consumption is the superposition of the sorting time and the searching time, and selecting the appropriate sort length will minimize the factorization time. In this case, when the sorting range is set to [0, 1000], the factorization time is reduced to a minimum of 0.51 seconds.

3) DECIMAL DIGIT DIFFERENCE

The decimal digit difference is the difference between the decimal digits of two prime factors; for example, the decimal digit difference between 17 and 12561229 is 6.

The Miller-Rabin method is used to generate two prime numbers whose decimal digit difference is 0 to 16 and whose decimal digit sum is 20. We take the product of these prime numbers as N. Factorization is performed using the Fermat's, CF, Pollard p-1, QS, EC and PP algorithms, where the sort length of PP is fixed to 100000.

It can be seen from Figure 5 that only Fermat's algorithm and PP algorithm will increase linearly with x; for almost every same x, PP algorithm always performs much better than Fermat's algorithm (with less time consumed); With the increase of the decimal digit difference, the performance gap between this two algorithms will expand exponentially. The time consumption of other factorization algorithms do not show certain regularity to the variation of the decimal digit difference.

Therefore, both Fermat's method and the PP method are suitable for the factorization of integers with little difference between their two prime factors, but the sensitivity of the PP method to the difference in the two prime factors is much less than that of Fermat's method. However, the CF, Pollard p-1, QS and EC methods show good robustness for decimal digit differences.

TABLE 2. Memory Usage of PP Algorithm. For RSA-100 (100 decimal digits, approximately 330 bits), the memory usage exceeds 50 GB. If we continued to factor larger integers, the computational support of super-large storage computers would need to be provided.

Decimal digits	Memory Usage[a]
20	7
59	1813
100	524589

[a]Units: MB.

4) MEMORY USAGE

Performing the CF, QS and some other algorithms will cause a large memory footprint when factoring large integers that a PC cannot provide. Since the PP algorithm is designed to maintain a sorted array, the array will occupy too much memory to allow storage of that array when using this algorithm.

TABLE 2 gives the memory consumption of the PP algorithm when decimal digits are 20, 59 (RSA-59) and 100 (RSA-100, approximately 330 bits). We can see that when the factorization of 100-decimal-digit-long integers is implemented, the memory usage exceeds 50 GB. Due to hardware constraints, we did not continue the experiment; to do so, the computational support of super-large storage computers would be needed.

5) SUPER-LARGE INTEGER CASES

We successfully factored RSA-100 (100 decimal digits, approximately 330 bits) with the PP method, which took 47,983 seconds (approximately 13.3 hours), this was slightly better than the QS method, which took 54,271 seconds (approximately 15 hours). That is, when dealing with super-large integers, the PP algorithm has shown as strong performance as subexponential complexity algorithms such as the QS method.

6) MULTITHREAD/MULTIPROCESS PARALLEL ACCELERATION

For multicore processors, the process that randomly generates a number r in the searching range and checks whether $F_r \pmod N$ hits the sorted list can be multithread/multiprocess parallel accelerated. The specific method is as follows: create a multithread/multiprocess; each child thread/process searches for the Pisano period separately; if any of the child threads/processes successfully finds the Pisano period, other child threads/processes will be killed; then, return to the main thread. Next, the main thread performs Algorithm 1 to obtain p_1,p_2. It is worth noting that all threads/processes must share memory (i.e., share a sorted array).

When decomposing RSA-100 as described above, we use 24 threads in parallel to speed up the Pisano period searching process.

VI. CONCLUSION

The research in this paper attempts to provide a new idea concerning integer factorization, which has shown some practicability and given a strong performance. However, our research is far from over. For example, we studied the general distribution of the Pisano sequence in a minimum period but did not fully utilize these characteristics in the design of the period searching algorithm; inspired by the random method of Shor's algorithm, we adopted the random numbers to speed up our algorithm, but there is no exponential improvement in efficiency. In addition, the problems of memory consumption, optimization of the sorting algorithm and acceleration of the modulo operation are still worthy of further study.

APPENDIX

Access for our code can be found here: https://github.com/wuliangshun/IntegerFactorizationWithPisanoPeriod.

ACKNOWLEDGMENT

Thanks to ''moonlightelite'', ''dmatlack'', ''pkruk'' and ''martingkelly'' from GitHub.

REFERENCES

[1] Y. H. Wang and S. Y. Yan, ''New quantum algorithm for breaking RSA,'' *Comput. Sci.*, vol. 43, no. 4, pp. 24–29, Jan. 2016, doi: 10.11896/j.issn.1002-137X.2016.4.004.
[2] M. Guo, ''RSA algorithm and its challenges,'' *J. Henan Mech. Electr. Eng. College*, vol. 17, no. 2, pp. 29–31, 2009.
[3] N. Koblitz, *A Course in Number Theory and Cryptography*. Springer-Verlag, 2009.
[4] C. Pomerance, ''The quadratic sieve factoring algorithm,'' in *Advances Cryptology*. (Lecture Notes in Computer Science), vol. 209, 1984, no. 4, pp. 169–182.
[5] J. M. Pollard, ''Theorems on factorization and primality testing,'' *Math. Proc. Cambridge Philos. Soc.*, vol. 76, no. 3, pp. 521–528, 1974.
[6] H. W. Lenstra, ''Factoring integers with elliptic curves,'' *Ann. Math.*, vol. 126, no. 3, pp. 649–673, Nov. 1987.
[7] A. K. Lenstra and H. W. Lenstra, *The Development of the Number Field Sieve*. Springer, 1993.
[8] P. W. Shor, ''Algorithms for quantum computation: Discrete logarithms and factoring,'' in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, Santa Fe, NM, USA, Nov. 1994, pp. 124–134, doi: 10.1109/SFCS.1994.365700.
[9] S.-J. Wei, T. Xin, and G.-L. Long, ''Efficient universal quantum channel simulation in IBM's cloud quantum computer,'' in *Science China Physics, Mechanics & Astronomy*, 2018.
[10] M. R. Geller and Z. Zhou, ''Factoring 51 and 85 with 8 qubits,'' *Sci. Rep.*, vol. 3, no. 3, 2013, Art. no. 3023, doi: 10.1038/srep03023.
[11] C. Gidney, ''Factoring with n+2 clean qubits and n-1 dirty qubits,'' Jun. 2018, *arXiv:1706.07884*. [Online]. Available: https://arxiv.org/abs/1706.07884
[12] C. J. C. Burges, ''Factoring as optimization,'' Tech. Rep. MSR-TR-2002-83, Aug. 2002.
[13] S. X. Jiang, K. A. Britt, A. J. McCaskey, T. S. Humble, and S. Kais, ''Quantum annealing for prime factorization,'' *Sci. Rep.*, vol. 8, Dec. 2018, Art. no. 17667, [Online]. Available: https://www.nature.com/articles/s41598-018-36058-z.pdf
[14] W. Peng, B. Wang, F. Hu, Y. Wang, X. Fang, X. Chen, and C. Wang, ''Factoring larger integers with fewer qubits via quantum annealing with optimized parameters,'' *Sci. China-Phys. Mech. Astron*, vol. 62, no. 6, Jun. 2019, Art. no. 60311, doi: 10.1007/s11433-018-9307-1.
[15] A. Cho, ''Muon's magnetism could point to new physics,'' *Science*, vol. 359, no. 6374, p. 381, 2018, doi: 10.1126/science.359.6374.381.
[16] D. D. Wall, ''Fibonacci series modulo *m*,'' *Amer. Math. Monthly*, vol. 67, no. 6, pp. 525–532, 1960, doi: 10.1080/00029890.1960.11989541.
[17] M. H. Yuan, ''The Periodicity of Fibonacci modulus sequence,'' *Math. Pract. Theory*, vol. 3, no. 1, pp. 119–121, 2007.
[18] D. G. Zhu, ''Fibonacci numbers and $GL(2, F_p)$,'' *J. Central China Normal Univ. (Nat. Sci.)*, vol. 23, no. 3, pp. 327–329, Sep. 1989, doi: 10.19603/j.cnki.1000-1190.
[19] C. Z. Zhou, P. Z. Yuan, and G. N. Xiao, *Fibonacci-Lucas Sequences Its Application*, 2nd ed. Harbin, China: Harbin Institute of Technology Press, 2016, ch. 3, sec. 4, pp. 227–234.
[20] D. Andrica, V. Crişan, and F. Al-Thukair, ''On Fibonacci and Lucas sequences modulo a prime and primality testing,'' *Arab J. Math. Sci.*, vol. 24, no. 1, pp. 9–15, 2018, doi: 10.1016/j.ajmsc.2017.06.002.
[21] Y. J. Li, ''Inherent relationship between the mode periodicities of cat map and Fibonacci series,'' *J. Comput. Appl.*, vol. 30, no. 4, pp. 1027–1032, 2010, doi: 1001-9081(2010)04-1026-04.
[22] L. Wu and H. Cai, ''The periodicity of fibonacci sequence modulo a prime and its application in primality testing,'' in *Proc. 2nd Int. Conf. Comput. Sci. Appl. Eng.*, Hohhot, China, Oct. 2018, p. 152, doi: 10.1145/3207677.3278049.
[23] F. L. Bauer, ''Efficient solution of a non-monotonic inverse problem,'' in *Beauty is Our Business*. New York, NY, USA: Springer-Verlag, 1990, pp. 19–20.
[24] D. Y. Pei and Y. F. Zhu, *Algorithmic Number Theory*, 2nd ed. Beijing, China: Science Press, 2015.
[25] W. K. Chen, *Linear Networks and Systems*. Belmont, CA, USA: Wadsworth, 1993, pp. 123–135.

LIANGSHUN WU received the B.S. degree in computer science from the School of Information Science and Engineering, Central South University, Hunan, China, in 2014, the M.S. degree in finance from Wuhan University, Hubei, China, in 2017, where he is currently pursuing the Ph.D. degree in software engineering.

From 2017 to 2018, he was engaged in research work with the Software Engineering Department, School of Computer Science, Wuhan University. From 2018 to 2019, he served as a Senior Research Intern with the Zall Smart Commerce Research Institute. His research interests include cryptography, network security, and embedded systems. He was certified as a Network Engineer by Qualification Certificate of Computer and Software Technology Proficiency, China, and an Embedded System Engineer by NCRE, China.

H. J. CAI received the B.S. degree from the University of Science and Technology of China, in 1984, the M.S. degree from the Space Science and Applied Research Center, CAS, China, in 1989, and the Ph.D. degree from the University of Alaska at Fairbanks, USA, in 1995, all in space physics.

From 1986 to 1992, he was a Research Assistant with the Space Science and Applied Research Center, CAS. From 1992 to 1996, he was a Research Assistant with the Geophysical Institute, University of Alaska Fairbanks. From 1996 to 1999, he held a postdoctoral position with the Department of Physics and Astrophysics, University of Iowa, USA. From 1999 to 2005, he was with eNet Trade, Independent Realty Capital Corporation and the Bestprice Group Inc., USA. Since 2005, he has been a Professor with the International School of Software and School of Computer Science, Wuhan University, China. He is currently a Full Professor and a Ph.D. Advisor with the School of Computer Science, Wuhan University, the Executive Director of the Zall Research Institute of Smart Commerce, a Visiting Researcher with the Center for Multimedia Technologies, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, an Expert of the Global FinTech Lab, and the Co-Founder and the Chief Scientist of everiToken Public Blockchain. He is the first author of the book *Before the Rise of Machines: The Beginning of the Consciousness and the Human Intelligence*, which received the Wu Wenjun Artificial Intelligence Science and Technology Award, China, in 2017. He has more than 100 articles. He holds four authorized patents. He has published the article, Agent-Based Simulation on Wealth Transfer and Accumulation in an Agricultural Society, the IEEE International Conference on Management and Service Science, in 2009 (MASS 2009). His research interests include blockchain technologies, artificial intelligence, and financial engineering. He is a Vice-President of the China Communications Industry Association Professional Committee of Blockchain (CCIAPCB) and an Expert Committee Member of the China AI and Big Data Committee of 100.

ZEXI GONG received the B.S. degree in electrical engineering from the Wuhan Institute of Technology, Wuhan, China, in 2017. He is currently pursuing the M.S. degree in electrical and computer engineering from Northeastern University, Boston, USA.

From 2015 to 2016, he was a Research Assistant with the Institute of Electrical, Wuhan Institute of Technology, Wuhan. His research interests include the efficiency of signal processing and computer vision.