Article

Factorization of Odd Integers with a Divisor of the Form $2^a u \pm 1$

Xingbo WANG [1,2,*]

[1] Department of Mechatronic Engineering,Foshan University,18 Jiangwanyi Road, Foshan City, PRC; xbwang@fosu.edu.cn
[2] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, PRC; dr.xbwang@qq.com

Abstract: The paper proves that an odd composite integer N can be factorized in at most $O(0.125u(log_2 N)^2)$ searching steps if N has a divisor of the form $2^a u + 1$ or $2^a u - 1$ with $a > 1$ being a positive integer and $u > 1$ being an odd integer. Theorems and corollaries are proved with detail mathematical reasoning. Algorithms to factorize the kind of odd composite integers are designed and tested by factoring certain Fermat numbers. The results in the paper are helpful to factorize the related kind of odd integers as well as some big Fermat numbers.

Keywords: Integer factorization; Fermat number; Cryptography;Algorithm

---

Factoring an odd composite number has been a research topic in number theory as well as cryptography. Although there are several algorithms to factorize a composite number, as overviewed in [1] and [2], new and efficient methods are still in need. Paper [3], a very recent publication, proved that an odd composite integer N could be factorized in $O((log_2 N)^4)$ bit operations if $N = pq$ with $q = 2^\alpha u \pm 1$ and $1 < p \le 2^\alpha + 1$ or $2^\alpha + 1 < p \le 2^{\alpha+1} - 1$, where α is a positive integer and $u \ge 1$ is an odd integer and α is a positive integer. This result might be marvelous for factoring the kind of N. However, the paper [3] did not mention the case when p is of a more general form. Hereby, this paper investigates the case $2^{\alpha+\beta} + 1 \le p \le 2^{\alpha+\beta+1} - 1$ with β being a positive integer.

The contents of this paper include five main parts: section 2 presents certain preliminaries, section 3 shows the main results including theorems, corollaries and algorithms designed for each case, section 4 presents numerical experiments on factoring certain related Fermat numbers, section 5 points out the advantage of the algorithms and the barrier to apply the algorithms as well as the possible future attempts.

1. Preliminaries

1.1. Definitions & Notations

In this whole article, symbol $\lfloor x \rfloor$ denotes the floor function, an integer function of the real number x such that $x - 1 < \lfloor x \rfloor \le x$ or equivalently $\lfloor x \rfloor \le x < \lfloor x \rfloor + 1$. Symbol $A \Rightarrow B$ means result B is derived from condition A or A can derive B out. Symbol GCD is to express the *greatest common divisor* between two integers.

1.2. Lemmas

Lemma 1(See in [4]) Properties of the floor functions with real numbers x, y and integer n
(**P1**) $\lfloor x \rfloor + \lfloor y \rfloor \le \lfloor x + y \rfloor \le \lfloor x \rfloor + \lfloor y \rfloor + 1$
(**P2**) $\lfloor x \rfloor - \lfloor y \rfloor - 1 \le \lfloor x - y \rfloor \le \lfloor x \rfloor - \lfloor y \rfloor < \lfloor x \rfloor - \lfloor y \rfloor + 1$
(**P13**) $x \le y \Rightarrow \lfloor x \rfloor \le \lfloor y \rfloor$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 March 2020 doi:10.20944/preprints202003.0174.v1

(**P**14) $\lfloor x \pm n \rfloor = \lfloor x \rfloor \pm n$
(**P**32) $n \lfloor x \rfloor \leq \lfloor nx \rfloor \leq n(\lfloor x \rfloor + 1) - 1$. Taking $n = 2$ yields $2 \lfloor x \rfloor \leq \lfloor 2x \rfloor \leq 2 \lfloor x \rfloor + 1$
Lemma 2(See in [5]) Odd integer $N > 1$ satisfies $2^{\lfloor \log_2 N \rfloor} + 1 \leqslant N \leqslant 2^{\lfloor \log_2 N \rfloor + 1} - 1$.

2. Main Results

2.1. Math Foundations

Theorem 1 Let $N = pq$ be an odd integer with p and q being odd integers bigger than 1; suppose $q = 2^\alpha u \pm 1$ and $2^{\alpha+\beta} + 1 \leqslant p \leqslant 2^{\alpha+\beta+1} - 1$ with α ,β being positive integers and $u > 1$ being an odd integer; then

$$2\alpha + \beta + 1 \leqslant \lfloor \log_2 N \rfloor$$

Proof. The conditions $u \geq 3$ and $q = 2^\alpha u + 1$ yield

$$\begin{aligned}
&N = (2^\alpha u + 1)p = 2^\alpha up + p = 2^\alpha (u + \frac{1}{2^\alpha})p \\
&\Rightarrow 2^{2\alpha+\beta}(u + \frac{1}{2^\alpha}) < N < 2^{2\alpha+\beta+1}(u + \frac{1}{2^\alpha}) \\
&\Rightarrow 2^{2\alpha+\beta} \times 3 < N < 2^{2\alpha+\beta+1}(u + \frac{1}{2^\alpha}) \\
&\Rightarrow 2\alpha + \beta + 1 < \log_2 N < 2\alpha + \beta + 1 + \log_2(u + \frac{1}{2^\alpha}) \\
&\Rightarrow 2\alpha + \beta + 1 \leqslant \lfloor \log_2 N \rfloor
\end{aligned}$$

Similarly, the conditions $u \geq 3$ and $q = 2^\alpha u - 1$ yield

$$\begin{aligned}
&N = (2^\alpha u - 1)p = 2^\alpha up - p = 2^\alpha (u - \frac{1}{2^\alpha})p \\
&\Rightarrow 2^{2\alpha+\beta}(u - \frac{1}{2^\alpha}) < N < 2^{2\alpha+\beta+1}(u - \frac{1}{2^\alpha}) \\
&\Rightarrow 2^{2\alpha+\beta} \times 2 < N < 2^{2\alpha+\beta+1}(u - \frac{1}{2^\alpha}) \\
&\Rightarrow 2\alpha + \beta + 1 < \log_2 N < 2\alpha + \beta + 1 + \log_2(u - \frac{1}{2^\alpha}) \\
&\Rightarrow 2\alpha + \beta + 1 \leqslant \lfloor \log_2 N \rfloor
\end{aligned}$$

□
Remark 1 During the reasoning process, the properties list in Lemma 1 are frequently applied, and so they are with the following reasoning.
Proposition 1 Let $N = pq$ be an odd integer with p and q being odd integers; suppose $q = 2^\alpha u \pm 1$ and $2^{\alpha+\beta} + 1 \leqslant p \leqslant 2^{\alpha+\beta+1} - 1$ with α ,β being positive integers and $u > 1$ being an odd integer; then

$$\alpha \leqslant \left\lfloor \frac{\log_2 N}{2} \right\rfloor - 1$$

Proof. We consider two cases: $q \geq p$ and $q < p$. For the case $q > p$, it follows

$$q \geq p \Rightarrow q \geq \sqrt{N} \geq p > 2^{\alpha+\beta} \Rightarrow \alpha + \beta < \log_2 \sqrt{N} \Rightarrow \alpha + \beta \leq \left\lfloor \frac{\log_2 N}{2} \right\rfloor$$

Since $\beta \geq 1$, it yields

$$\alpha \leq \left\lfloor \frac{\log_2 N}{2} \right\rfloor - \beta \leq \left\lfloor \frac{\log_2 N}{2} \right\rfloor - 1$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 March 2020 doi:10.20944/preprints202003.0174.v1

Now the case $q < p$. First consider $q = 2^\alpha u + 1$; then

$$\begin{aligned} & q < p \Rightarrow q \leqslant \sqrt{N} \Rightarrow 2^\alpha \leqslant \frac{\sqrt{N}-1}{u} < \frac{\sqrt{N}}{3} \\ & \Rightarrow \alpha < \frac{1}{2}\log_2 N - \log_2 3 < \frac{1}{2}\log_2 N - 1 \\ & \Rightarrow \alpha \leqslant \left\lfloor \frac{\log_2 N}{2} \right\rfloor - 1 \end{aligned}$$

Likewise, when $q = 2^\alpha u - 1$ it follows

$$\begin{aligned} & q < p \Rightarrow q \leqslant \sqrt{N} \Rightarrow 2^\alpha \leqslant \frac{\sqrt{N}+1}{u} \leqslant \frac{\sqrt{N}+1}{3} \\ & \Rightarrow \alpha \leqslant \log_2(\sqrt{N}+1) - \log_2 3 = \frac{1}{2}\log_2 N + \log_2(1+\frac{1}{\sqrt{N}}) - \log_2 3 \end{aligned}$$

Since $0 < \log_2(1+\frac{1}{\sqrt{N}}) < 1$ and $1.5 < \log_2 3 < 2$, it knows

$$\begin{aligned} & \frac{1}{2}\log_2 N - 2 < \frac{1}{2}\log_2 N + \log_2(1+\frac{1}{\sqrt{N}}) - \log_2 3 = \frac{1}{2}\log_2 N - 0.5 \\ & \Rightarrow \alpha \leqslant \left\lfloor \frac{\log_2 N}{2} \right\rfloor - 1 \end{aligned}$$

□

Remark 2 The case $q = 2^\alpha u + 1$ actually yields $\alpha \leqslant \left\lfloor \frac{\log_2 N}{2} \right\rfloor - 2$ because $\alpha < \frac{1}{2}\log_2 N - \log_2 3 \Rightarrow \alpha \leqslant \left\lfloor \frac{\log_2 N}{2} \right\rfloor - 2$.

Theorem 2 Let $N = pq$ be an odd integer with p and q being odd integers bigger than 1; suppose $q = 2^\alpha u + 1$ and $2^{\alpha+\beta} + 1 \leq p \leq 2^{\alpha+\beta+1} - 1$ with α ,β being positive integers and $u > 1$ being an odd integer; then

$$\left\lfloor \frac{u}{4} \right\rfloor \leq \left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor \leq \left\lfloor \frac{u}{2} \right\rfloor + 1$$

Proof. See the following reasoning process.

$$\begin{aligned} & N = (2^\alpha u + 1)p = 2^\alpha up + p \\ & \Rightarrow 2^\alpha u(2^{\alpha+\beta}+1) + (2^{\alpha+\beta}+1) < N < 2^\alpha u(2^{\alpha+\beta+1}-1) + (2^{\alpha+\beta+1}-1) \\ & \Rightarrow 2^{2\alpha+\beta}u + 2^\alpha u + 2^{\alpha+\beta} < N - 1 < 2^{2\alpha+\beta+1}u - 2^\alpha u + 2^{\alpha+\beta+1} - 2 \\ & \Rightarrow 2^{2\alpha+\beta}u + 2^{\alpha+\beta} < N - 1 < 2^{2\alpha+\beta+1}u + 2^{\alpha+\beta+1} - 2 < 2^{2\alpha+\beta+1}u + 2^{\alpha+\beta+1} \\ & \Rightarrow 2^{2\alpha+\beta}(u + \tfrac{1}{2^\alpha}) < N - 1 < 2^{2\alpha+\beta+1}(u + \tfrac{1}{2^\alpha}) \\ & \Rightarrow \tfrac{1}{4}(u + \tfrac{1}{2^\alpha}) < \tfrac{N-1}{2^{2\alpha+\beta+2}} < \tfrac{1}{2}(u + \tfrac{1}{2^\alpha}) \\ & \Rightarrow \left\lfloor \tfrac{u}{4} \right\rfloor \leq \left\lfloor \tfrac{u}{4} + \tfrac{1}{2^{\alpha+2}} \right\rfloor \leq \left\lfloor \tfrac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor \leq \left\lfloor \tfrac{u}{2} + \tfrac{1}{2^{\alpha+1}} \right\rfloor \leq \left\lfloor \tfrac{u}{2} \right\rfloor + 1 \\ & \Rightarrow \left\lfloor \tfrac{u}{4} \right\rfloor \leq \left\lfloor \tfrac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor \leq \left\lfloor \tfrac{u}{2} \right\rfloor + 1 \end{aligned}$$

□

Theorem 3 Let $N = pq$ be an odd integer with p and q being odd integers bigger than 1; suppose $q = 2^\alpha u - 1$ and $2^{\alpha+\beta} + 1 \leq p \leq 2^{\alpha+\beta+1} - 1$ with α ,β being positive integers and $u > 1$ being an odd integer; then

$$\left\lfloor \frac{u}{4} \right\rfloor - 1 \leq \left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor \leq \left\lfloor \frac{u}{2} \right\rfloor$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 March 2020 doi:10.20944/preprints202003.0174.v1

Proof. See the following reasoning process.

$$
\begin{aligned}
&N = (2^{\alpha}u - 1)p = 2^{\alpha}up - p \\
&\Rightarrow 2^{\alpha}u(2^{\alpha+\beta} + 1) - (2^{\alpha+\beta+1} - 1) < N < 2^{\alpha}u(2^{\alpha+\beta+1} - 1) - (2^{\alpha+\beta} + 1) \\
&\Rightarrow 2^{2\alpha+\beta}u + 2^{\alpha}u - 2^{\alpha+\beta+1} < N - 1 < 2^{2\alpha+\beta+1}u - 2^{\alpha}u - 2^{\alpha+\beta} - 2 \\
&\Rightarrow 2^{2\alpha+\beta}u - 2^{\alpha+\beta+1} < N - 1 < 2^{2\alpha+\beta+1}u - 2^{\alpha+\beta} - 2 < 2^{2\alpha+\beta+1}u - 2^{\alpha+\beta} \\
&\Rightarrow 2^{2\alpha+\beta}(u - \tfrac{1}{2^{\alpha-1}}) < N - 1 < 2^{2\alpha+\beta+1}(u - \tfrac{1}{2^{\alpha+1}}) \\
&\Rightarrow \tfrac{1}{4}(u - \tfrac{1}{2^{\alpha-1}}) < \tfrac{N-1}{2^{2\alpha+\beta+2}} < \tfrac{1}{2}(u - \tfrac{1}{2^{\alpha+1}}) \\
&\Rightarrow \lfloor \tfrac{u}{4} \rfloor - 1 \le \left\lfloor \tfrac{u}{4} - \tfrac{1}{2^{\alpha+1}} \right\rfloor \le \left\lfloor \tfrac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor \le \left\lfloor \tfrac{u}{2} - \tfrac{1}{2^{\alpha+2}} \right\rfloor \le \lfloor \tfrac{u}{2} \rfloor \\
&\Rightarrow \lfloor \tfrac{u}{4} \rfloor - 1 \le \left\lfloor \tfrac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor \le \lfloor \tfrac{u}{2} \rfloor
\end{aligned}
$$

□

2.2. Algorithms For General Cases

Corollary 1 Let $N = pq$ be an odd integer with p and q being odd integers; suppose $q = 2^{\alpha}u + 1$ and $2^{\alpha+\beta} + 1 \le p \le 2^{\alpha+\beta+1} - 1$ with $\alpha > 1$, $\beta \ge 1$ being integers and $u > 1$ being an old integer; then N can be factorized at most $O(\frac{u}{8}(\log_2 N)^2)$ searching steps.

Proof. First, it follows when $\alpha > 1$ and $\beta \ge 1$

$$
\begin{aligned}
&N = 2^{\alpha}up + p \Rightarrow u = \tfrac{N-p}{2^{\alpha}p} = \tfrac{N}{2^{\alpha}p} - \tfrac{1}{2^{\alpha}} \\
&\Rightarrow \tfrac{N}{2^{2\alpha+\beta+1}} - \tfrac{1}{2^{\alpha}} < u < \tfrac{N}{2^{2\alpha+\beta}} - \tfrac{1}{2^{\alpha}} \\
&\Rightarrow \tfrac{N-1}{2^{2\alpha+\beta+1}} + \tfrac{1}{2^{2\alpha+\beta+1}} - \tfrac{1}{2^{\alpha}} < u < \tfrac{N-1}{2^{2\alpha+\beta}} + \tfrac{1}{2^{2\alpha+\beta}} - \tfrac{1}{2^{\alpha}} \\
&\Rightarrow \left\lfloor \tfrac{N-1}{2^{2\alpha+\beta+1}} \right\rfloor - 1 \le u \le \left\lfloor \tfrac{N-1}{2^{2\alpha+\beta}} \right\rfloor
\end{aligned}
$$

Let

$$
n_u = \frac{\left\lfloor \frac{N-1}{2^{2\alpha+\beta}} \right\rfloor - \left(\left\lfloor \frac{N-1}{2^{2\alpha+\beta+1}} \right\rfloor - 1 \right)}{2} + 1
$$

Then n_u is the number of the odd integers that u possibly takes. Since

$$
\left\lfloor \frac{N-1}{2^{2\alpha+\beta+1}} \right\rfloor \le \left\lfloor \frac{N-1}{2^{2\alpha+\beta}} \right\rfloor - \left\lfloor \frac{N-1}{2^{2\alpha+\beta+1}} \right\rfloor \le \left\lfloor \frac{N-1}{2^{2\alpha+\beta+1}} \right\rfloor + 1
$$

it holds

$$
\frac{1}{2}\left\lfloor \frac{N-1}{2^{2\alpha+\beta+1}} \right\rfloor + \frac{1}{2} + 1 \le n_u \le \frac{1}{2}\left\lfloor \frac{N-1}{2^{2\alpha+\beta+1}} \right\rfloor + 1 + 1
$$

and thus

$$
\left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor + 1 - \frac{1}{2} < n_u < \left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor + 2 + 1
$$

That is

$$
\left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor + 1 \le n_u \le \left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor + 2
$$

By Theorem 2,

$$
\left\lfloor \frac{u}{4} \right\rfloor \le \left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor \le \left\lfloor \frac{u}{2} \right\rfloor + 1
$$

It results in

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 March 2020 doi:10.20944/preprints202003.0174.v1

$$\left\lfloor \frac{u}{4} \right\rfloor + 1 \leq n_u \leq \left\lfloor \frac{u}{2} \right\rfloor + 3$$

This finsihes proving that there are at most $\lfloor \frac{u}{4} \rfloor + 2$ searching steps to find a u_0 that fits $q = 2^{\alpha}u_0 + 1$ around $\left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor$.

Now consider the ranges of α and β in algorithm design. By Theorem 1 and Proposition 1,

$$2\alpha + \beta + 1 \leq \lfloor \log_2 N \rfloor$$

and

$$\alpha \leq \left\lfloor \frac{\log_2 N}{2} \right\rfloor - 1$$

It seems that $\lfloor \log_2 N \rfloor$ might be taken to be the maximal value of $2\alpha + \beta + 1$. However this does not work because by Lemma 2

$$2^{\lfloor \log_2 N \rfloor} + 1 \leq N \leq 2^{\lfloor \log_2 N \rfloor + 1} - 1$$

which leads to

$$\left\lfloor \frac{N-1}{2^{\lfloor \log_2 N \rfloor}} \right\rfloor = 1, 2 \leq \left\lfloor \frac{N-1}{2^{\lfloor \log_2 N \rfloor - 1}} \right\rfloor \leq 3, 4 \leq \left\lfloor \frac{N-1}{2^{\lfloor \log_2 N \rfloor - 2}} \right\rfloor \leq 7$$

and

$$2^{\chi} \leq \left\lfloor \frac{N-1}{2^{\lfloor \log_2 N \rfloor - \chi}} \right\rfloor \leq 2^{\chi+1} - 1$$

Accordingly, it is suitable to take $\lfloor \log_2 N \rfloor - 2$ to be the maximal value of $2\alpha + \beta + 1$ and then the condition that odd integer $u > 1$ can be ensured. Consequently, an algorithm is designed as Algorithm 1.

Algorithm 1 (Search Procedure 1)

```
Comment:Factorization by Calculating GCD
Input Parameters: N;
Begin
Calculate b = ⌊log2 N⌋ − 2, a = ⌊log2 N / 2⌋ − 2;
for i = b downto 3 do
  for u = l to r step 2 do
    for j = 2 to a do
      Calculate l = ⌊(N−1)/2^i⌋ − 1, r = ⌊(N−1)/2^(i−1)⌋;
      if l mod 2 = 0 then
        l = l − 1;
      end if
      if r mod 2 = 0 then
        r = r + 1;
      end if
      Calculate g = gcd(N, 2^j u + 1);
      if g > 1 then
        return g;
      end if
    end for
  end for
end for
End
```

Obviously, the number of total searching steps is at most $O(\frac{u}{4} \cdot \frac{\lfloor \log_2 N \rfloor}{2} \cdot \lfloor \log_2 N \rfloor) = O(\frac{u}{8} \lfloor \log_2 N \rfloor^2)$.

□

Remark 3 When u is relatively larger, we can choose a smaller b in the procedure by $2^{\chi} \leq \left\lfloor \frac{N-1}{2^{\lfloor \log_2 N \rfloor - \chi}} \right\rfloor \leq 2^{\chi+1} - 1$. For example, if we can first predict that $u > 2^{\chi}$, we can choose $b = \lfloor \log_2 N \rfloor - \chi$. By such means, a lot of computing time is saved.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 March 2020 doi:10.20944/preprints202003.0174.v1

Corollary 2 Let $N = pq$ be an odd integer with p and q being odd integers; suppose $q = 2^{\alpha}u - 1$ and $2^{\alpha+\beta} + 1 \leq p \leq 2^{\alpha+\beta+1} - 1$ with $\alpha > 1$, $\beta \geq 1$ being integers and $u > 1$ being an old integer; then N can be factorized at most $O(\frac{u}{8}(\log_2 N)^2)$ searching steps.

Proof. First, it yields when $\alpha > 1$ and $\beta \geq 1$

$$\begin{aligned} & N = 2^{\alpha}up - p \Rightarrow u = \frac{N+p}{2^{\alpha}p} = \frac{N}{2^{\alpha}p} + \frac{1}{2^{\alpha}} \\ & \Rightarrow \frac{N}{2^{2\alpha+\beta+1}} + \frac{1}{2^{\alpha}} < u < \frac{N}{2^{2\alpha+\beta}} + \frac{1}{2^{\alpha}} \\ & \Rightarrow \frac{N-1}{2^{2\alpha+\beta+1}} + \frac{1}{2^{2\alpha+\beta+1}} + \frac{1}{2^{\alpha}} < u < \frac{N-1}{2^{2\alpha+\beta}} + \frac{1}{2^{2\alpha+\beta}} + \frac{1}{2^{\alpha}} \\ & \Rightarrow \left\lfloor \frac{N-1}{2^{2\alpha+\beta+1}} \right\rfloor \leq u \leq \left\lfloor \frac{N-1}{2^{2\alpha+\beta}} \right\rfloor + 1 \end{aligned}$$

Let

$$n_u = \frac{\left\lfloor \frac{N-1}{2^{2\alpha+\beta}} \right\rfloor + 1 - \left\lfloor \frac{N-1}{2^{2\alpha+\beta+1}} \right\rfloor}{2} + 1$$

Then referring to the proof of Corollary 1, it knows

$$\left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor + 1 \leq n_u \leq \left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor + 2$$

By Theorem 3

$$\left\lfloor \frac{u}{4} \right\rfloor - 1 \leq \left\lfloor \frac{N-1}{2^{2\alpha+\beta+2}} \right\rfloor \leq \left\lfloor \frac{u}{2} \right\rfloor$$

Next is referring to the proof of Corollary 1. Accordingly, an algorithm is designed as Algorithm 2.

Algorithm 2 (Search Procedure 2)

```
Comment:Factorization by Calculating GCD
Input Parameters: N;
Begin
Calculate b = ⌊log2 N⌋ − 2, a = ⌊(log2 N)/2⌋ − 1;
for i = b downto 3 do
   for u = l to r step 2 do
      for j = 2 to a do
         Calculate l = ⌊(N−1)/2^i⌋, r = ⌊(N−1)/2^(i−1)⌋ + 1;
         if l mod 2 = 0 then
            l = l − 1;
         end if
         if r mod 2 = 0 then
            r = r + 1;
         end if
         Calculate g = gcd(N, 2^j u − 1);
         if g > 1 then
            return g;
         end if
      end for
   end for
end for
End
```

Referring to the analysis of Corollary 1, it knows Corollary 2 holds.

□

Theorem 4 Let $N = pq$ be an odd integer with p and q being odd integers; suppose $q = 2^{\alpha}u \pm 1$ and $2^{\alpha+\beta} + 1 \leq p \leq 2^{\alpha+\beta+1} - 1$ with $\alpha > 1$, $\beta \geq 1$ being integers and $u > 1$ being an odd integer; then N can be factorized in at most $O(0.25u(log_2 N)^2)$

Proof. By Theorem 2, Theorem 3, Algorithm 1 and Algorithm 2 are incorporated into the following Algorithm 3 that can factorize N provided that the divisor q is of the form $2^{\alpha}u + 1$ or $2^{\alpha}u - 1$ with $u > 1$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 March 2020 doi:10.20944/preprints202003.0174.v1

Algorithm 3 (Search Procedure 3)

Comment:Factorization by Calculating GCD
Input Parameters: N;
Begin
Calculate $b = \lfloor \log_2 N \rfloor - 2$, $a = \left\lfloor \frac{\log_2 N}{2} \right\rfloor - 1$;
for $i = b$ downto 3 **do**
  for $u = l$ to r step 2 **do**
    for $j = 2$ to a **do**
      Calculate $l = \left\lfloor \frac{N-1}{2^i} \right\rfloor - 1, r = \left\lfloor \frac{N-1}{2^{i-1}} \right\rfloor + 1$;
      if $l \mod 2 = 0$ **then**
        $l = l - 1$;
      end if
      if $r \mod 2 = 0$ **then**
        $r = r + 1$;
      end if
      Calculate $g = gcd(N, 2^j u + 1)$;
      if $g > 1$ **then**
        return g;
      end if
      Calculate $g = gcd(N, 2^j u - 1)$;
      if $g > 1$ **then**
        return g;
      end if
    end for
  end for
end for
End

□

2.3. Algorithms For Factoring Fermat Numbers

It is known that, every prime divisor of the Fermat number F_m is of the form $2^n \cdot u + 1$ with $n - m \geq 2$. Assume a Fermat number has two divisors $p = 2^{m+2+i}k + 1$ and $q = 2^{m+2+j}l + 1$ with i, j being nonnegative integers, $k > 1$ and $l > 1$ being odd integers (Note: Paper [3] proved that the case $k = 1$ or $l = 1$ was easy to factorize.); it follows

$$\begin{aligned} N = pq &= (2^{m+2+i}k + 1)(2^{m+2+j}l + 1) \\ &= 2^{2m+4+i+j}kl + 2^{m+3}(2^i k + 2^j l) + 1 \\ &\Rightarrow \frac{N-1}{2^{2m+4}} = 2^{i+j}kl + \frac{2^i k + 2^j l}{2^{m+1}} > 9 \\ &\Rightarrow 2^{m+2} < \frac{\sqrt{N-1}}{3} \\ &\Rightarrow n = m + 2 \leq \left\lfloor \frac{\log_2(N-1)}{2} - \log_2 3 \right\rfloor = \left\lfloor \frac{\log_2(N-1)}{2} \right\rfloor - 2 \end{aligned}$$

This is a theoretical upper bound for n. However, from practical point of view, referring to Prof. Wilfrid Keller's research [6], we take $m + 20$ to be the maximal limit of n and design the Algorithm 4.
Remark 4 The parameter χ in the procedure is related with the range of u and is used to limit the upper bound of b, as mentioned in Remark 3. Referring to [6], it knows that, for a Fermat number, the range of u can be predicted with an upper bound L_n . This can reduce a lot of search time.

3. Numerical Experiments

We test the designed algorithms in Maple software on a HP personal computer with E5450 CPU and Windows XP OS.The source codes of the program are presented in the Appendix Section. Due to the limitation of the computer, we factorize some Fermat numbers with relatively small u. The computation results are list in Table 1. It can see that, one divisor is obtained within one second in most cases. The experiments show that, the algorithms are acceptably fast.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 March 2020 doi:10.20944/preprints202003.0174.v1

Algorithm 4 (Factoring Fermate Numbers)

Comment:Factorization by Calculating GCD
Input Parameters: N, χ;
Begin
Calculate $n = \lfloor \log_2 \log_2 (N-1) \rfloor$,
$b = \lfloor \log_2 (N-1) \rfloor - \chi$, $a = n + 20$;
for $i = b$ downto 3 **do**
  for $u = l$ to r step 2 **do**
    for $j = 2$ to a **do**
      Calculate $l = \left\lfloor \frac{N-1}{2^i} \right\rfloor - 1, r = \left\lfloor \frac{N-1}{2^{i-1}} \right\rfloor$;
      if $l \mod 2 = 0$ **then**
        $l = l - 1$;
      end if
      if $r \mod 2 = 0$ **then**
        $r = r + 1$;
      end if
      Calculate $g = gcd(N, 2^j u + 1)$;
      if $g > 1$ **then**
        return g;
      end if
    end for
  end for
end for
End

Table 1. Factorization of Some Fermat Numbers

Item	χ	Found Divisor	α	β	u	Searching Steps	Searching time
F5	2	641	7	30	5	3	<5ms
F6	3	274177	8	54	1071	50496	30ms
F9	3	2424833	16	507	37	1462	5ms
F10	3	45592577	12	1011	11131	403468	140ms
F11	3	319489	13	2043	39	1378	5ms
F12	3	114689	14	4093	7	2	<5ms
F15	3	1214251009	21	32759	579	26348	1.2s
F16	3	825753601	19	65526	1575	51261	5.5s
F17	25	31065037602817	19	131047	59251857	12848715	1500s
F18	3	13631489	20	262141	13	5	<5ms
F19	3	70525124609	21	524273	33629	1606849	1560s
F21	19	4485296422913	23	2097133	534689	5203	9s
F23	2	167772161	25	8388606	5	3	<5ms

4. Conclusion, Discussion & Future Work

4.1. Conclusion

Factoring big integers is both a challenge to and an exploitation of human intelligence. We know that there are many odd integers that have divisors of the form $2^\alpha u \pm 1$ and factorization of such numbers is still a hard problem in the world. By investigating the intrinsic structures of such odd number, this paper shows that factorization of them is highly u-related: a small u leads to a rapid factorization. The results in this paper theoretically can factorize Fermat Numbers that have a big n. For example, referring to [6], when n is bigger than 100001, $u < 20000$. Then it is easy to find a divisor of F_m.

4.2. Discussion

Seeing from the numerical experiments in section 4, one might be wonder why not to choose some big Fermat numbers to be experimental samples. This involves with a very upset situation: we have no way to compute a bigger Fermat number with a daily-work computer even with the help of the tools dealing with the big integer operations. Theoretically, a daily-work computer can be either 32bit or 64bit word length. Consequently the biggest positive integer it can deal with is at

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 March 2020 doi:10.20944/preprints202003.0174.v1

most $2^{64}-1 = 18446744073709551615$. Now there are software tools that declaim themselves to have the capability in big number computing, e.g., GMP library (The GNU Multiple Precision Arithmetic Library)[7], NTL (A Library for doing Number Theory)[8], MIRACL(Multi-precision Integer and Rational Arithmetic Cryptographic Library)[9] and so on. They surely do well work for a popular big integer but none of them can work for a 'big' Fermat number. I make such conclusion based on my own C/C++ programming experiences. Take GMP, which is said to be 'arithmetic without limitations', as an example; let's show what our barrier is. Seen from the user's manual, the GMP library provides a function *mpz_mul_2exp(mpz_t &rop, mpz_t op, unsigned long i)* to perform the following operation

$$rop = op \times 2^i$$

With this function and under the condition $1 \leq i \leq 2^w - 1$, where w is the word-length of the computer system, we can obtain a Fermat Number by defining a function SetFermatNumber as follows (Note: the number before each sentence is for later explanations)

```
void SetFermatNumber(mpz_t &rop, unsigned long k)
{
(1) mpz_t t; unsigned long s;
(2) mpz_init(t); mpz_set_ui(t,1);    //initialize t and set t=1
(3) s=1«k;                           //s = 2^k
(4) mpz_mul_2exp(rop, t, s);         //rop = 1 × 2^s = 2^(2^k)
(5) mpz_add_ui(rop, rop, 1);         //rop = rop + 1 ⇒ 2^(2^k) + 1
(6) mpz_clear(t)                     //release t
}
```

This function works well when $1 \leq k \leq 2^w - 1$. Otherwise it does not work because the sentence (3) is only available for unsigned long integers and the third parameter s in sentence (4) must be an unsigned long integer. We wish we could have gotten a function like this

$$mpz\_mul\_2expz(mpz\_t\ \&\ rop, mpz\_t\ op, mpz\_t\ z)$$

with which a real big Fermat number could be calculated.

Consequently, we have no way to obtain a computable Fermat number F_m with $m > 2^w - 1$. Hope new tools can solve this problem.

4.3. Future Work

Corollaries 1 and 2 do provide an approach to factorize odd integers with a divisor of the form $2^\alpha u \pm 1$. However, seeing the reasoning process of this paper, we have to have more researches on the speeding-up process for a big u, for example, the F7 has $q = 116503103764643 \cdot 2^9 + 1$ with a big $u = 116503103764643$. This remains the future work. Actually, in our experiments, we have found that, changing the parameters χ will lead to different searching steps. For example, when factoring F5, it took more than 340 steps when $\chi = 3$ while it merely took 3 steps when $\chi = 2$. Another example is factoring F21 that has a divisor 4485296422913. When taking $\chi = 19$ according to Remark 2 it took 5203 steps (9 seconds) to finish the work while it took 3675387 steps (over 30 minutes) when $\chi = 5$. These phenomena indicate that a speed-up attempt is available.One of our future tasks is trying to solve the problem. And I also hope more young join and succeed.

Acknowledgments: The research is supported by the Open Project Program of the State Key Lab of CAD&CG(Grant No. A2002), Foshan University and by Foshan Bureau of Science and Technology under project that constructs Guangdong Engineering Center of Information Security for Intelligent Manufacturing System.

Conflicts of Interest: The authors declare no conflict of interest.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 March 2020 doi:10.20944/preprints202003.0174.v1

Terminologies

Maple is the mathematical software in which one can program using Maple scripts to test algorithms.

Appendix A Maple Program and Running Results

Here is the Maple program and its running results.

```
FactFermat := proc (N, x) # Maple program from Algorith 4
  local a, b, i, j, u, l, s, r, g, d, rm, c, Z;
  c := 1; # a counter to count searching steps
  Z := N-1+0.1e-1;
  b := floor(log[2](Z))-x;
  s := floor(log[2](log[2](Z)));
  a := s+15;
  for i from b by -1 to 3 do
    l := floor((N − 1)/2^i) − 1;
    r := floor((N − 1)/2^(i−1));
    if 'mod'(l, 2) = 0 then l := l-1 end if;
    if 'mod'(r, 2) = 0 then r := r+1 end if;
      for j from s+2 to a do
        for u from l by 2 to r do
        c := c+1;
        d := 2^j × u + 1;
        g := gcd(N, d);
        if 1 < g then do
          lprint(i, j, u, d, g, c);
          return g;
          end do
        end if
        end do
      end do
    end do
end proc
```

The following screenshot (see Figure A1) shows the testing results to factorize Fermat numbers.

The meanings of the output data are as follows:
The first one is the calcualted α
The second one is the calcualted β
The third one is the calcualted u
The fourth one is the calcualted $2^{\alpha}u + 1$
The fifth one is the calcualted divisor
The last one is the number of searching steps.

It can see that, F_5 takes 3 steps, F_{12} takes 2 steps,F_{18} takes 5 steps,F_{23} takes 3 steps,F_9 takes 319 steps, F_{11} takes 230 steps, F_6,F_{15},F_{16} and F_{21} take less than 10 thousand steps. F_{17} takes the longest time

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 11 March 2020 doi:10.20944/preprints202003.0174.v1

30, 7, 5, 641, 641, 3

641

$FactFermat\left(2^{2^6}+1,\ 3\right)$

54, 8, 1071, 274177, 274177, 7334

274177

$FactFermat\left(2^{2^9}+1,\ 3\right)$

507, 16, 37, 2424833, 2424833, 319

2424833

$FactFermat\left(2^{2^{10}}+1,\ 3\right)$

1011, 12, 11131, 45592577, 45592577, 59040

45592577

$FactFermat\left(2^{2^{11}}+1,\ 3\right)$

2043, 13, 39, 319489, 319489, 230

319489

$FactFermat\left(2^{2^{12}}+1,\ 3\right)$

4093, 14, 7, 114689, 114689, 2

114689

$FactFermat\left(2^{2^{15}}+1,\ 3\right)$

32759, 21, 579, 1214251009, 1214251009, 4764

1214251009

$FactFermat\left(2^{2^{16}}+1,\ 3\right)$

65526, 19, 1575, 825753601, 825753601, 8100

825753601

$FactFermat\left(2^{2^{17}}+1,\ 25\right)$

131047, 19, 59251857, 31065037602817, 31065037602817, 12848715

31065037602817

$FactFermat\left(2^{2^{18}}+1,\ 3\right)$

262141, 20, 13, 13631489, 13631489, 5

13631489

$FactFermat\left(2^{2^{19}}+1,\ 3\right)$

524273, 21, 33629, 70525124609, 70525124609, 230089

70525124609

$FactFermat\left(2^{2^{21}}+1,\ 19\right)$

2097133, 23, 534689, 4485296422913, 4485296422913, 5203

4485296422913

$FactFermat\left(2^{2^{23}}+1,\ 2\right)$

8388606, 25, 5, 167772161, 167772161, 3

167772161

Figure A1. Test Results

 doi:10.20944/preprints202003.0174.v1

References

1. Sonal Sarnaik, Dinesh Gadekarand Umesh Gaikwad, An overview to Integer factorization and RSA in Cryptography. *International Journal for Advance Research in Engineering and Thechnology*, **2014**,*2(9)*, 21-26.
2. Kharate Sunita Phulachand, An overview of Cryptography. *Innovation in IT*, **2016**, *3(1)*, 8-11.
3. Xingbo WANG, Fast Approach to Factorize Odd Integers with Special Divisors. *Journal of Mathematics and Statistics*, **2020**, *16(1)*, 40-51.https://thescipub.com/abstract/10.3844/ofsp.12920
4. X. WANG, Brief Summary of Frequently-Used Properties of the Floor Function: Updated 2018 . *IOSR Journal of Mathematics*, **2019**,*15(1)*, 30-33.
5. X. WANG, T_3 Tree and Its Traits in Understanding Integers. *Advances in Pure Mathematics*, **2018**,*8(5)*, 494-507.
6. Wilfrid Keller, Prime factors $k \cdot 2^n + 1$ of Fermat numbers F_m and complete factoring status, *http://www.prothsearch.com/fermat.html*, 2020.
7. GMP library, The GNU Multiple Precision Arithmetic Library. *https:// gmplib.org/* , 2020.
8. Victor Shoup, NTL: A Library for doing Number Theory. *https://www. shoup.net/ntl/*, 2020.
9. MIRACL , MIRACL Cryptographic SDK .*https://github.com/miracl/ MIRACL*, 2020

Sample Availability: Source codes of C/C++ programs with GMP big number library are available from the author. Readers of insterests may ask for them.