Factorization Attacks
The Problem

Trying to factor a large integer is a much harder problem than testing whether or not the number is prime. And proving that a pseudoprime is actually prime is more difficult than verifying its pseudoprimality.
The Problem

Trying to factor a large integer is a much harder problem than testing whether or not the number is prime. And proving that a pseudoprime is actually prime is more difficult than verifying its pseudoprimality.

A probable prime is an integer that satisfies a certain condition that is satisfied by all prime numbers but is not satisfied by most composite numbers. If we consider Fermat’s Little Theorem, for $n \in \mathbb{Z}$, choose $a \not\equiv (a, n) = 1$. Then if $a^{n-1} \pmod{n} \equiv 1$, $n$ is likely prime.
The Problem

Trying to factor a large integer is a much harder problem than testing whether or not the number is prime. And proving that a pseudoprime is actually prime is more difficult than verifying its pseudoprimality.

A probable prime is an integer that satisfies a certain condition that is satisfied by all prime numbers but is not satisfied by most composite numbers. If we consider Fermat’s Little Theorem, for $n \in \mathbb{Z}$, choose $a \nmid (a, n) = 1$. Then if $a^{n-1} \pmod{n} \equiv 1$, $n$ is likely prime.

But not all integers satisfying this are prime; for example, $341 = 11 \times 13$ and $2^{340} \equiv 1 \pmod{341}$. So, we say that 341 is a pseudoprime base 2.
Before Trying Complicated Techniques ...

It is usually a good idea to do a number of trivial divisions by small primes before moving onto complicated methods.
It is usually a good idea to do a number of trivial divisions by small primes before moving onto complicated methods.

It is also a good idea (when applicable) to check and see if an integer fails a pseudoprimality test before trying to factor it.
Before Trying Complicated Techniques ...

It is usually a good idea to do a number of trivial divisions by small primes before moving onto complicated methods.

It is also a good idea (when applicable) to check and see if an integer fails a pseudoprimality test before trying to factor it.

Complicated factorization techniques all have possibility of failure, which is indistinguishable from the reasonable failure to factor due to $n$ being prime. So, it is important to know that any failure in attempting factorization of $n$ is due to bad luck in the algorithm rather than $n$ being prime so that we can adjust and try again.
Before Trying Complicated Techniques ...

It is usually a good idea to do a number of trivial divisions by small primes before moving onto complicated methods.

It is also a good idea (when applicable) to check and see if an integer fails a pseudoprimality test before trying to factor it.

Complicated factorization techniques all have possibility of failure, which is indistinguishable from the reasonable failure to factor due to \( n \) being prime. So, it is important to know that any failure in attempting factorization of \( n \) is due to bad luck in the algorithm rather than \( n \) being prime so that we can adjust and try again.

Consequently, the sophisticated factorization methods are not good for testing primality. Therefore, we will only discuss the factorization of integers we know are composite.
Let $n$ be prime with $n > 2$. It follows that $n - 1$ is even and we can write it as $2^s \cdot d$, where $s$ and $d$ are positive integers with $d$ odd.
Let \( n \) be prime with \( n > 2 \). It follows that \( n - 1 \) is even and we can write it as \( 2^s \cdot d \), where \( s \) and \( d \) are positive integers with \( d \) odd.

For each \( a \) in \((\mathbb{Z}/n\mathbb{Z})\), either

\[
a^d \equiv 1 \pmod{n}
\]

or

\[
a^{2r \cdot d} \equiv -1 \pmod{n}
\]

for some \( 0 \leq r \leq s - 1 \).
Setting Up Miller-Rabin Primality Test

Let $n$ be prime with $n > 2$. It follows that $n - 1$ is even and we can write it as $2^s \cdot d$, where $s$ and $d$ are positive integers with $d$ odd.

For each $a$ in $(\mathbb{Z}/n\mathbb{Z})$, either

$$a^d \equiv 1 \pmod{n}$$

or

$$a^{2r \cdot d} \equiv -1 \pmod{n}$$

for some $0 \leq r \leq s - 1$.

To show one of these must be true, we note that in considering Fermat’s Little Theorem, if we keep taking roots of $a^{n-1}$, we will either get 1 or -1. If we get -1, the second equivalence holds and we are done. If we never get -1, then once we have taken out all powers of 2, we are left with the first equivalence.
The Miller-Rabin primality test is based on the contrapositive of this claim. So, if we can find an $a$ such that neither of the above is true, then $n$ is not prime. We call $a$ a witness for the compositeness of $n$. Otherwise $a$ is called a strong liar and $n$ is strong probable prime to base $a$. 

Example

Determine if $n = 221$ is prime.
The Miller-Rabin primality test is based on the contrapositive of this claim. So, if we can find an \( a \) such that neither of the above is true, then \( n \) is not prime. We call \( a \) a witness for the compositeness of \( n \). Otherwise \( a \) is called a strong liar and \( n \) is strong probable prime to base \( a \).

Example

Determine if \( n = 221 \) is prime.
The Miller-Rabin Primality Test

We can write 220 as \(2^2 \cdot 55\), so we have \(s = 2\) and \(d = 55\). We now need a random integer \(a\) such that \(a < 221\). We choose \(a = 174\).
We can write 220 as $2^2 \cdot 55$, so we have $s = 2$ and $d = 55$. We now need a random integer $a$ such that $a < 221$. We choose $a = 174$.

$$174^{2^{0 \cdot 55}} \pmod{221} \equiv 47$$
The Miller-Rabin Primality Test

We can write 220 as $2^2 \cdot 55$, so we have $s = 2$ and $d = 55$. We now need a random integer $a$ such that $a < 221$. We choose $a = 174$.

\[
174^{2^0 \cdot 55} \pmod{221} \equiv 47
\]

\[
174^{2^1 \cdot 55} \pmod{221} \equiv 220 = n - 1
\]
The Miller-Rabin Primality Test

We can write 220 as $2^2 \cdot 55$, so we have $s = 2$ and $d = 55$. We now need a random integer $a$ such that $a < 221$. We choose $a = 174$.

$$174^{2^0 \cdot 55} \pmod{221} \equiv 47$$
$$174^{2^1 \cdot 55} \pmod{221} \equiv 220 = n - 1$$

Since we get $n - 1$, either 221 is prime or 174 is a strong liar. In this case, we want to try another $a$. 
The Miller-Rabin Primality Test

We can write 220 as $2^2 \cdot 55$, so we have $s = 2$ and $d = 55$. We now need a random integer $a$ such that $a < 221$. We choose $a = 174$.

$$174^{2 \cdot 55} \pmod{221} \equiv 47$$
$$174^{2^1 \cdot 55} \pmod{221} \equiv 220 = n - 1$$

Since we get $n - 1$, either 221 is prime or 174 is a strong liar. In this case, we want to try another $a$. Try $a = 137$. 
The Miller-Rabin Primality Test

We can write 220 as $2^2 \cdot 55$, so we have $s = 2$ and $d = 55$. We now need a random integer $a$ such that $a < 221$. We choose $a = 174$.

$$174^{2^{0 \cdot 55}} \pmod{221} \equiv 47$$
$$174^{2^{1 \cdot 55}} \pmod{221} \equiv 220 = n - 1$$

Since we get $n - 1$, either 221 is prime or 174 is a strong liar. In this case, we want to try another $a$. Try $a = 137$.

$$137^{2^{0 \cdot 55}} \pmod{221} \equiv 188$$
The Miller-Rabin Primality Test

We can write 220 as $2^2 \cdot 55$, so we have $s = 2$ and $d = 55$. We now need a random integer $a$ such that $a < 221$. We choose $a = 174$.

\[
174^{2^0 \cdot 55} \pmod{221} \equiv 47
\]
\[
174^{2^1 \cdot 55} \pmod{221} \equiv 220 = n - 1
\]

Since we get $n - 1$, either 221 is prime or 174 is a strong liar. In this case, we want to try another $a$. Try $a = 137$.

\[
137^{2^0 \cdot 55} \pmod{221} \equiv 188
\]
\[
137^{2^1 \cdot 55} \pmod{221} \equiv 205
\]
We can write 220 as $2^2 \cdot 55$, so we have $s = 2$ and $d = 55$. We now need a random integer $a$ such that $a < 221$. We choose $a = 174$.

$$174^{20 \cdot 55} \pmod{221} \equiv 47$$
$$174^{21 \cdot 55} \pmod{221} \equiv 220 = n - 1$$

Since we get $n - 1$, either 221 is prime or 174 is a strong liar. In this case, we want to try another $a$. Try $a = 137$.

$$137^{20 \cdot 55} \pmod{221} \equiv 188$$
$$137^{21 \cdot 55} \pmod{221} \equiv 205$$

So, 137 is a witness to the compositeness of 221.
Pollard’s Rho Method

Pollard’s Rho method was introduced by John Pollard in the mid-70’s. This method quickly finds relatively small factors of composite numbers. It is a very simple factorization method which already runs several times faster than trial division for numbers whose smallest prime factor is about $1,000,000$. Further, it has the practical virtue that if a number has a small prime factor, then the method finds such a factor faster than it would find a large factor. On the other hand, it is probabilistic, so we must be alert to occurrences of ‘bad cases’. Finally, it has another disadvantage in that it is difficult to prove that it works as well as it does. From an experimental viewpoint, this may be of no consequence, but in applications where correctness really matters, this could be seen as a serious flaw.
Pollard’s Rho method was introduced by John Pollard in the mid-70’s. This method quickly finds relatively small factors of composite numbers. It is a very simple factorization method which already runs several times faster than trial division for numbers whose smallest prime factor is about 1,000,000.

Further, it has the practical virtue that if a number has a small prime factor, then the method finds such a factor faster than it would find a large factor.
Pollard’s Rho Method

Pollard’s Rho method was introduced by John Pollard in the mid-70’s. This method quickly finds relatively small factors of composite numbers. It is a very simple factorization method which already runs several times faster than trial division for numbers whose smallest prime factor is about 1,000,000.

Further, it has the practical virtue that if a number has a small prime factor, then the method finds such a factor faster than it would find a large factor.

On the other hand, it is probabilistic, so we must be alert to occurrences of ‘bad cases’.
Pollard’s Rho Method

Pollard’s Rho method was introduced by John Pollard in the mid-70’s. This method quickly finds relatively small factors of composite numbers. It is a very simple factorization method which already runs several times faster than trial division for numbers whose smallest prime factor is about 1,000,000.

Further, it has the practical virtue that if a number has a small prime factor, then the method finds such a factor faster than it would find a large factor.

On the other hand, it is probabilistic, so we must be alert to occurrences of ‘bad cases’.

Finally, it has another disadvantage in that it is difficult to prove that it works as well as it does. From an experimental viewpoint, this may be of no consequence, but in applications where correctness really matters, this could be seen as a serious flaw.
Given integer $n$, initialize by setting $x = 2$ and $y = x^2 + 1$. 
Given integer $n$, initialize by setting $x = 2$ and $y = x^2 + 1$.

- Compute $g = (|x - y|, n)$
Pollard’s Rho Method

Given integer $n$, initialize by setting $x = 2$ and $y = x^2 + 1$.

- Compute $g = (|x - y|, n)$
- If $1 < g < n$, stop; $g$ is a proper factor of $n$
Given integer $n$, initialize by setting $x = 2$ and $y = x^2 + 1$.

- Compute $g = (|x - y|, n)$
- If $1 < g < n$, stop; $g$ is a proper factor of $n$
- If $g = 1$, replace $x$ by $x^2 + 1$ and $y$ by $(y^2 + 1)^2 + 1$ and repeat
Pollard’s Rho Method

Given integer $n$, initialize by setting $x = 2$ and $y = x^2 + 1$.

- Compute $g = (\lvert x - y \rvert, n)$
- If $1 < g < n$, stop; $g$ is a proper factor of $n$
- If $g = 1$, replace $x$ by $x^2 + 1$ and $y$ by $(y^2 + 1)^2 + 1$ and repeat
- If $g = n$, we have failure, and the algorithm needs to be reinitialized

Note: the need to reinitialize is rare.
Pollard’s Rho Method

Example
Given that $n = 31861$ is not prime, use Pollard’s Rho method to determine a prime factor of $n$. 

\[
\begin{align*}
(x, 31861) &= 1 \\
(5, 31861) &= 1 \\
(677, 31861) &= 1 \\
(29508, 31861) &= 1 \\
(27909, 31861) &= 151 \\
\end{align*}
\]

Therefore, $p = 151$ is a factor of 31861 and from there we can quickly get that $31861 = 151 \times 211$. 
Pollard’s Rho Method

Example
Given that \( n = 31861 \) is not prime, use Pollard’s Rho method to determine a prime factor of \( n \).

\[
\begin{array}{c|c|c}
  x & y & (x - y, 31861) \\
  2 & 5 & (3, 31861) = 1 \\
\end{array}
\]

Therefore, \( p = 151 \) is a factor of 31861 and from there we can quickly get that 31861 = 151 \times 211.\]
Example

Given that $n = 31861$ is not prime, use Pollard’s Rho method to determine a prime factor of $n$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$(x - y, 31861)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>$(3, 31861) = 1$</td>
</tr>
<tr>
<td>5</td>
<td>677</td>
<td>$(672, 31861) = 1$</td>
</tr>
</tbody>
</table>

Therefore, $p = 151$ is a factor of $31861$ and from there we can quickly get that $31861 = 151 \times 211$. 
Pollard’s Rho Method

Example

Given that $n = 31861$ is not prime, use Pollard’s Rho method to determine a prime factor of $n$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$(x - y, 31861)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>$(3, 31861) = 1$</td>
</tr>
<tr>
<td>5</td>
<td>677</td>
<td>$(672, 31861) = 1$</td>
</tr>
<tr>
<td>26</td>
<td>29508</td>
<td>$(26482, 31861) = 1$</td>
</tr>
</tbody>
</table>

Therefore, $p = 151$ is a factor of 31861 and from there we can quickly get that $31861 = 151 \times 211$. 
Pollard’s Rho Method

Example

Given that $n = 31861$ is not prime, use Pollard’s Rho method to determine a prime factor of $n$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$(x - y, 31861)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>$(3, 31861) = 1$</td>
</tr>
<tr>
<td>5</td>
<td>677</td>
<td>$(672, 31861) = 1$</td>
</tr>
<tr>
<td>26</td>
<td>29508</td>
<td>$(26482, 31861) = 1$</td>
</tr>
<tr>
<td>677</td>
<td>27909</td>
<td>$(27232, 31861) = 1$</td>
</tr>
</tbody>
</table>
Pollard’s Rho Method

Example

Given that $n = 31861$ is not prime, use Pollard’s Rho method to determine a prime factor of $n$.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$(x - y, 31861)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>$(3, 31861) = 1$</td>
</tr>
<tr>
<td>5</td>
<td>677</td>
<td>$(672, 31861) = 1$</td>
</tr>
<tr>
<td>26</td>
<td>29508</td>
<td>$(26482, 31861) = 1$</td>
</tr>
<tr>
<td>677</td>
<td>27909</td>
<td>$(27232, 31861) = 1$</td>
</tr>
<tr>
<td>12276</td>
<td>19675</td>
<td>$(7399, 31861) = 151$</td>
</tr>
</tbody>
</table>

Therefore, $p = 151$ is a factor of 31861 and from there we can quickly get that $31861 = 151 \times 211$. 
About Pollard’s Rho Method

- Playing on the birthday paradox, the number of cycles necessary to find a factor $p$ of $n$ should be roughly of the order $\sqrt{n}$. If $n$ is composite, then there is a prime factor $p \leq \sqrt{n}$. This method takes on the order of $\frac{4}{\sqrt{n}}$ cycles to find a proper factor.
About Pollard’s Rho Method

- Playing on the birthday paradox, the number of cycles necessary to find a factor $p$ of $n$ should be roughly be of the order $\sqrt{n}$. If $n$ is composite, then there is a prime factor $p \leq \sqrt{n}$. This method takes on the order of $\frac{4}{\sqrt{n}}$ cycles to find a proper factor.
- There is no limit on the number of cycles since, if $n$ is known to be composite, the algorithm will take the same time as trivial division.
About Pollard’s Rho Method

- Playing on the birthday paradox, the number of cycles necessary to find a factor $p$ of $n$ should be roughly be of the order $\sqrt{n}$. If $n$ is composite, then there is a prime factor $p \leq \sqrt{n}$. This method takes on the order of $\frac{4}{\sqrt{n}}$ cycles to find a proper factor.
- There is no limit on the number of cycles since, if $n$ is known to be composite, the algorithm will take the same time as trivial division.
- There should be a limit imposed on the number of cycles before making an adjustment. $\sqrt{n}$ would be too many, $\frac{4}{\sqrt{n}}$ would be too few. Since we are talking about large primes, $100\frac{4}{\sqrt{n}}$ should be enough to tell if an adjustment is needed.
About Pollard’s Rho Method

- Playing on the birthday paradox, the number of cycles necessary to find a factor $p$ of $n$ should be roughly be of the order $\sqrt{n}$. If $n$ is composite, then there is a prime factor $p \leq \sqrt{n}$. This method takes on the order of $\frac{1}{4}\sqrt{n}$ cycles to find a proper factor.
- There is no limit on the number of cycles since, if $n$ is known to be composite, the algorithm will take the same time as trivial division.
- There should be a limit imposed on the number of cycles before making an adjustment. $\sqrt{n}$ would be too many, $\frac{1}{4}\sqrt{n}$ would be too few. Since we are talking about large primes, $100\frac{1}{4}\sqrt{n}$ should be enough to tell if an adjustment is needed.
- It is acceptable to just let the algorithm run to success or failure if we know that $n$ is composite.
Playing on the birthday paradox, the number of cycles necessary to find a factor $p$ of $n$ should be roughly be of the order $\sqrt{n}$. If $n$ is composite, then there is a prime factor $p \leq \sqrt{n}$. This method takes on the order of $\sqrt[4]{n}$ cycles to find a proper factor.

There is no limit on the number of cycles since, if $n$ is known to be composite, the algorithm will take the same time as trivial division.

There should be a limit imposed on the number of cycles before making an adjustment. $\sqrt{n}$ would be too many, $\sqrt[4]{n}$ would be too few. Since we are talking about large primes, $100\sqrt[4]{n}$ should be enough to tell if an adjustment is needed.

It is acceptable to just let the algorithm run to success or failure if we know that $n$ is composite.

Since worst case is the same as trivial division, this is not a good method to test primality.
About Pollard’s Rho Method

- Playing on the birthday paradox, the number of cycles necessary to find a factor \( p \) of \( n \) should be roughly be of the order \( \sqrt{n} \). If \( n \) is composite, then there is a prime factor \( p \leq \sqrt{n} \). This method takes on the order of \( 4\sqrt{n} \) cycles to find a proper factor.
- There is no limit on the number of cycles since, if \( n \) is known to be composite, the algorithm will take the same time as trivial division.
- There should be a limit imposed on the number of cycles before making an adjustment. \( \sqrt{n} \) would be too many, \( 4\sqrt{n} \) would be too few. Since we are talking about large primes, \( 100\sqrt{n} \) should be enough to tell if an adjustment is needed.
- It is acceptable to just let the algorithm run to success or failure if we know that \( n \) is composite.
- Since worst case is the same as trivial division, this is not a good method to test primality.
- There is no guarantee that this method will produce a prime gcd \( g \), but this is generally the case.
Why The Method Works

There are two key ingredients: the probabilistic idea of the birthday paradox, and the Floyd circle detection method, which is used to make exploitation of the birthday paradox idea practical.
There are two key ingredients: the probabilistic idea of the birthday paradox, and the Floyd circle detection method, which is used to make exploitation of the birthday paradox idea practical.

So suppose \( n \) is a positive integer with proper divisor \( d \). It does not matter if \( d \) is prime or not, but only that \( d \) is smaller than \( n \).
Why The Method Works

There are two key ingredients: the probabilistic idea of the birthday paradox, and the Floyd circle detection method, which is used to make exploitation of the birthday paradox idea practical.

So suppose $n$ is a positive integer with proper divisor $d$. It does not matter if $d$ is prime or not, but only that $d$ is smaller than $n$.

From the birthday paradox we know if we have more than $\sqrt{d}$ integers $x_1, x_2, \ldots, x_t$, then the probability if greater than $\frac{1}{2}$ that two of these are the same modulo $d$. 
Why The Method Works

There are two key ingredients: the probabilistic idea of the birthday paradox, and the Floyd circle detection method, which is used to make exploitation of the birthday paradox idea practical.

So suppose $n$ is a positive integer with proper divisor $d$. It does not matter if $d$ is prime or not, but only that $d$ is smaller than $n$.

From the birthday paradox we know if we have more than $\sqrt{d}$ integers $x_1, x_2, \ldots, x_t$, then the probability if greater than $\frac{1}{2}$ that two of these are the same modulo $d$.

The idea of Pollard’s Rho method is that $\sqrt{d}$ is much smaller than $\sqrt{n}$, we should expect that if we choose a random sequence of integers $x_1, x_2, \ldots$ there will be two the same modulo $d$ long before there are two the same modulo $n$. 
That is, supposing that \( x_i \equiv x_k \pmod{d} \) but not already knowing what \( d \) is, we can (often) find \( d \) by computing

\[
g = (x_i - x_k, n)
\]
That is, supposing that $x_i \equiv x_k \pmod{d}$ but not already knowing what $d$ is, we can (often) find $d$ by computing

$$g = (x_i - x_k, n)$$

Keep in mind that computation of gcd’s using the Euclidean Algorithm is relatively cheap.
That is, supposing that \( x_i \equiv x_k \pmod{d} \) but not already knowing what \( d \) is, we can (often) find \( d \) by computing

\[
g = (x_i - x_k, n)
\]

Keep in mind that computation of gcd’s using the Euclidean Algorithm is relatively cheap.

More precisely, if \( x_i \equiv x_k \pmod{d} \) but \( x_i \not\equiv x_k \pmod{n} \) then this gcd will be a multiple of \( d \).
A too-naive implementation of this as a means of hunting proper factors of \( n \) is to compute the gcd \((x_i - x_j, n)\) as we go along, for all \( i < j \). This is a bad idea: if we hope that we need about \( \sqrt{d} \) random integers in order to find the proper factor \( d \), then we will have had to make on the order of

\[
\frac{1}{2}(\sqrt{d})^2 = \frac{1}{2}d
\]

computations of gcd’s. We might as well have used trial division.
A too-naive implementation of this as a means of hunting proper factors of \( n \) is to compute the gcd \((x_i - x_j, n)\) as we go along, for all \( i < j \). This is a bad idea: if we hope that we need about \( \sqrt{d} \) random integers in order to find the proper factor \( d \), then we will have had to make on the order of

\[
\frac{1}{2}(\sqrt{d})^2 = \frac{1}{2}d
\]

computations of gcd’s. We might as well have used trial division.

We need a clever way of taking advantage of the birthday paradox. It is here that we make use of the specific manner in which Pollard’s Rho algorithm creates the supposedly random integers.
Implementation

First, we have to pretend that the function \( f(x) = x^2 + 1 \) (taken modulo \( n \)) used in the algorithm is a random map from \( \mathbb{Z}/n \) to itself. Since it appears that no one can prove much of anything in this direction, we won’t try to say precisely what this might mean.
First, we have to pretend that the function $f(x) = x^2 + 1$ (taken modulo $n$) used in the algorithm is a random map from $\mathbb{Z}/n$ to itself. Since it appears that no one can prove much of anything in this direction, we won’t try to say precisely what this might mean.

But in any case, if we make a sequence of supposedly random numbers by

\[
\begin{align*}
x_0 &= 2 \\
x_1 &= f(x_0) \\
x_2 &= f(x_1) \\
x_3 &= f(x_2) \\
&\vspace{1em} \vdots
\end{align*}
\]

Then, each element in the sequence determines the next one completely. That is, if ever $x_j \equiv x_i \pmod{d}$ with $i < j$ then also inevitably $x_{j+1} \equiv x_{i+1}$, $x_{j+2} \equiv x_{i+2}$, and so on.
Implementation

In particular, for all \( t \geq j \),

\[
x_t \equiv x_{t-(j-i)} \pmod{d}
\]
Implementation

In particular, for all $t \geq j$, 

$$x_t \equiv x_{t-(j-i)} \pmod{d}$$

Further, for the same reason,

$$x_t \equiv x_{t-(j-i)} \equiv x_{t-2(j-i)} \equiv \ldots \equiv x_{t-l(j-i)} \pmod{d}$$

as long as $t - l(j - i) \geq i$. That is, there is more structure here than if we merely had a growing set of random numbers.
Note

Keep in mind that we don’t necessarily care about the very first case that some $x_i \equiv x_j$, but only about a relatively early case.
Keep in mind that we don’t necessarily care about the very first case that some \( x_i \equiv x_j \), but only about a relatively early case.

**Floyd’s cycle-detection method** is the following efficient way of looking for matches. First, we do not keep in memory the whole list of \( x_i \)’s, as this would be needlessly inefficient. Rather, we just remember the last one computed. But at the same time we separately compute a sequence

\[
\begin{align*}
y_1 & \equiv x_2 \\
y_2 & \equiv x_4 \\
y_3 & \equiv x_6 \\
& \vdots \\
y_i & \equiv x_{2i}
\end{align*}
\]
The efficient way to compute the sequence of $y_i$’s is by noting that

$$y_{i+1} \equiv (y_i^2 + 1)^2 + 1 \pmod{n}$$

And, we only remember the last $y$ computed.
The efficient way to compute the sequence of $y_i$’s is by noting that

$$y_{i+1} \equiv (y_i^2 + 1)^2 + 1 \pmod{n}$$

And, we only remember the last $y$ computed.

At each step we only remember the last $x_i$ and $y_i$ computed and consider $(x_i - y_i, n)$. 
Why will this most likely find a proper factor? Let $j$ be the first index so that $x_j \equiv x_i \pmod{d}$ for some $i < j$. As noted above, this means that

$$x_t \equiv x_{t - l(j-i)}$$

whenever $t - l(j - i) \geq i$. 
Why will this most likely find a proper factor? Let $j$ be the first index so that $x_j \equiv x_i \pmod{d}$ for some $i < j$. As noted above, this means that

$$x_t \equiv x_{t-l(j-i)}$$

whenever $t - l(j - i) \geq i$.

So taking $t = 2s$,

$$y_s \equiv x_{2s} \equiv x_{2s-l(j-i)} \equiv x_{2s-s} \equiv x_s \pmod{d}$$

for all $s$ with $2s - l(j - i) \geq i$. This proves that the trick used above really does succeed in finding $x_i$ and $x_j$, which are the same modulo $d$, assuming that they exist.
Example

Use Pollard’s Rho method to find a proper factor of 1649.
Pollard’s $p - 1$ Method

This method is specialized, working well only to find a prime factor $p$ so that $p - 1$ is divisible by only ‘small’ factors, and not working particularly well outside those cases. Still, application of this test is relatively simple, so to prevent factorization attacks, it must be taken into account.
Pollard’s $p − 1$ Method

This method is specialized, working well only to find a prime factor $p$ so that $p − 1$ is divisible by only ‘small’ factors, and not working particularly well outside those cases. Still, application of this test is relatively simple, so to prevent factorization attacks, it must be taken into account.

Fix an integer $B$. An integer $n$ is $B$-smooth if all its prime factors are less than or equal to $B$. In this context $B$ is a smoothness bound.
Pollard’s $p − 1$ Method

This method is specialized, working well only to find a prime factor $p$ so that $p − 1$ is divisible by only ‘small’ factors, and not working particularly well outside those cases. Still, application of this test is relatively simple, so to prevent factorization attacks, it must be taken into account.

Fix an integer $B$. An integer $n$ is $B$-smooth if all its prime factors are less than or equal to $B$. In this context $B$ is a smoothness bound.

An integer $n$ is $B$-power smooth if all its prime power factors are less than or equal to $B$. 
Smoothness

Example

100 is 5-smooth because none of its prime factors are greater than 5.
Smoothness

Example

100 is 5-smooth because none of its prime factors are greater than 5.
100 is 25-power smooth because none of the powers of its prime factors exceed 25.
Smoothness

Example

100 is 5-smooth because none of its prime factors are greater than 5. 100 is 25-power smooth because none of the powers of its prime factors exceed 25.

Given an integer \( n \), Pollard’s \( p - 1 \) algorithm finds a prime factor \( p \) for \( n \) such that \( p - 1 \) is \( B \)-smooth.
Pollard’s $p − 1$ Method

- Keeping $B$ small is desirable. It should be much smaller than $\sqrt{n}$ or nothing is gained over trivial division.
Pollard’s \( p - 1 \) Method

- Keeping \( B \) small is desirable. It should be much smaller than \( \sqrt{n} \) or nothing is gained over trivial division.
- A too small value for \( B \) will cause the algorithm to not find factors.
Pollard’s $p – 1$ Method

- Keeping $B$ small is desirable. It should be much smaller than $\sqrt{n}$ or nothing is gained over trivial division.
- A too small value for $B$ will cause the algorithm to not find factors.
- The question of how large we should expect the prime factors of a randomly chosen number to be is not trivial and there is no simple answer.
Pollard’s $p - 1$ Method

Given an integer $n$ known to be composite, but not a prime power, and given a smoothness bound $B$, choose a random integer $b$ with $2 \leq b \leq n - 1$ and compute $g = (b, n)$. If $g \geq 2$ then $g$ is a proper factor and stop. Otherwise let $p_1, p_2, \ldots, p_t$ be the primes less than or equal to $B$. For $i = 1, 2, 3 \ldots, t$, let $q = p_i$ and

- Compute $l = \left\lfloor \frac{\ln n}{\ln p_i} \right\rfloor$
- Replace $b$ by $b^q$
- Compute $g = (b - 1, n)$
- If $1 < g < n$, stop; $g$ is a proper factor
- Else if $g = 1$, continue
- If $g = n$, stop (failure)
Pollard’s $p − 1$ Method

Given an integer $n$ known to be composite, but **not a prime power**, and given a smoothness bound $B$, choose a random integer $b$ with $2 \leq b \leq n − 1$ and compute $g = (b, n)$. If $g \geq 2$ then $g$ is a proper factor and stop. Otherwise let $p_1, p_2, \ldots, p_t$ be the primes less than or equal to $B$. For $i = 1, 2, 3 \ldots, t$, let $q = p_i$ and

- Compute $l = \left\lfloor \frac{\ln n}{\ln p_i} \right\rfloor$
- Replace $b$ by $b^q$
- Compute $g = (b − 1, n)$
- If $1 < g < n$, stop; $g$ is a proper factor
- Else if $g = 1$, continue
- If $g = n$, stop (failure)

Note: If $B$ is relatively large, we can save some time by not computing all the primes $p_1, \ldots, p_t$ at the beginning, but only as needed, hoping to find a factor before all are used.
Why It Works

If we can arrange that for some integer $b$ we have $1 < (b - 1, n) < n$, then certainly we have found a proper factor $g = (b - 1, n)$. 
Why It Works

If we can arrange that for some integer $b$ we have
$1 < (b - 1, n) < n$, then certainly we have found a proper factor
$g = (b - 1, n)$.

In this algorithm, let

$$p = 1 + p_1^{e_1} \cdots p_t^{e_t}$$

be a prime factor of $n$ such that $p - 1$ is $B$-smooth, with some integer
exponents $e_i$. 
Why It Works

If we can arrange that for some integer \( b \) we have \( 1 < (b - 1, n) < n \), then certainly we have found a proper factor \( g = (b - 1, n) \).

In this algorithm, let

\[
p = 1 + p_1^{e_1} \cdots p_t^{e_t}
\]

be a prime factor of \( n \) such that \( p - 1 \) is \( B \)-smooth, with some integer exponents \( e_i \).

For any integer \( b \) relatively prime to \( p \) by Fermat’s Little Theorem

\[
b^{p-1} \equiv 1 \pmod{p}
\]

That is,

\[
b^{p_1^{e_1} \cdots p_t^{e_t}} \equiv 1 \pmod{p}
\]
Why It Works

If we can arrange that for some integer $b$ we have $1 < (b - 1, n) < n$, then certainly we have found a proper factor $g = (b - 1, n)$.

In this algorithm, let

$$p = 1 + p_1^{e_1} \cdots p_t^{e_t}$$

be a prime factor of $n$ such that $p - 1$ is $B$-smooth, with some integer exponents $e_i$.

For any integer $b$ relatively prime to $p$ by Fermat’s Little Theorem

$$b^{p-1} \equiv 1 \pmod{p}$$

That is,

$$b^{p_1^{e_1} \cdots p_t^{e_t}} \equiv 1 \pmod{p}$$

The quantity

$$l_i = \left\lfloor \frac{\ln n}{\ln p_i} \right\rfloor$$

is larger than or equal to $e_i$. 
Why It Works

Let

\[ T = p_1^{l_1} \cdots p_t^{l_t} \]
Let 

\[ T = p_1^{e_1} \cdots p_t^{e_t} \]

Then \( p_1^{e_1} \cdots p_t^{e_t} \) divides \( T \), so certainly

\[ b^T \equiv 1 \pmod{p} \]

for any integer \( b \ni (b, p) = 1 \), That is,

\[ p \mid (b^T - 1, n) \]

Note that the actual algorithm actually calculates the gcd's more often than indicated at the end of the proof. This provides some opportunities to avoid the cases that \( (b^T - 1, n) = n \).
Failure

Failure of this algorithm can result for two different sorts of reasons.

1. There are no prime factors $p$ of $n$ so that $p - 1$ is $B$-smooth. In this case, the gcd computed will always be 1.
Failure of this algorithm can result for two different sorts of reasons.

1. There are no prime factors $p$ of $n$ so that $p - 1$ is $B$-smooth. In this case, the gcd computed will always be 1.

2. All the prime factors $q$ of $n$ have $q - 1$ being $B$-smooth. In this case, $(b^T - 1, n) = n.$
Failure of this algorithm can result for two different sorts of reasons.

1. There are no prime factors $p$ of $n$ so that $p - 1$ is $B$-smooth. In this case, the gcd computed will always be 1.

2. All the prime factors $q$ of $n$ have $q - 1$ being $B$-smooth. In this case, $(b^T - 1, n) = n$.

This algorithm computes extra gcd’s so that even if all gcd’s are 1 or $n$, so long as there is at least one occurrence of $n$ among the gcd’s, there is hope that the algorithm can succeed; simply start with a different initial random $b$. 
Example

Find a prime factor of 9991.

Note that $9991 = 97 \cdot 103$ and $97 - 1 = 2^5 \cdot 3$, so $97 - 1$ is \{2, 3\}-smooth. (We don't know this yet in real life but could easily check and find a few small prime factors.)

Initialize $b = 3$.

$\lfloor \frac{\ln 9991}{\ln 2} \rfloor = 13$

So, compute $b^{2^{13}} \pmod{9991} \equiv 229$ and assign that value to $b$.

Using the Euclidean Algorithm, we can compute $(9991, 229 - 1) = 1$

So, we fail to find a factor in this round.
Example

Find a prime factor of 9991.

Note that $9991 = 97 \cdot 103$ and $97 - 1 = 2^5 \cdot 3$, so $97 - 1$ is \{2, 3\}-smooth. (We don’t know this yet in real life but could easily check and find a few small prime factors.)
Find a prime factor of 9991.

Note that $9991 = 97 \cdot 103$ and $97 - 1 = 2^5 \cdot 3$, so $97 - 1$ is \{2, 3\}-smooth. (We don’t know this yet in real life but could easily check and find a few small prime factors.)

Initialize $b = 3$.

$$\left\lfloor \frac{\ln 9991}{\ln 2} \right\rfloor = 13$$
Example

Find a prime factor of 9991.

Note that $9991 = 97 \cdot 103$ and $97 - 1 = 2^5 \cdot 3$, so $97 - 1$ is \{2, 3\}-smooth. (We don’t know this yet in real life but could easily check and find a few small prime factors.)

Initialize $b = 3$.

$$\left\lfloor \frac{\ln 9991}{\ln 2} \right\rfloor = 13$$

So, compute

$$b^{2^{13}} \pmod{9991} \equiv 229$$

and assign that value to $b$. 
Find a prime factor of 9991.

Note that $9991 = 97 \cdot 103$ and $97 - 1 = 2^5 \cdot 3$, so $97 - 1$ is \{2, 3\}-smooth. (We don’t know this yet in real life but could easily check and find a few small prime factors.)

Initialize $b = 3$.

$$\left\lfloor \frac{\ln 9991}{\ln 2} \right\rfloor = 13$$

So, compute

$$b^{2^{13}} \pmod{9991} \equiv 229$$

and assign that value to $b$.

Using the Euclidean Algorithm, we can compute

$$(9991, 229 - 1) = 1$$

So, we fail to find a factor in this round.
First Example

\[
\left\lfloor \frac{\ln 9991}{\ln 3} \right\rfloor = 8
\]

So, calculate \( b^{8} \pmod{9991} \equiv 3202 \) and assign that value to \( b \).

Again, using the Euclidean Algorithm, compute \((9991, 3202 - 1) = 97\). And we have found one of the proper factors to be 97.
First Example

\[ \left\lfloor \frac{\ln 9991}{\ln 3} \right\rfloor = 8 \]

So, calculate

\[ b^{3^8} \pmod{9991} \equiv 3202 \]

and assign that value to \( b \).
First Example

\[
\left\lfloor \frac{\ln 9991}{\ln 3} \right\rfloor = 8
\]

So, calculate

\[b^{3^8} \pmod{9991} \equiv 3202\]

and assign that value to \(b\).

Again, using the Euclidean Algorithm, compute

\[(9991, 3202 - 1) = 97\]

And we have found one of the proper factors to be 97.
Example

Factor 3801911 using the Pollard $p - 1$ method.
Factor 3801911 using the Pollard $p-1$ method.

With quick calculation, we see that 3801991 is $\{2, 3, 5, 7\}$-smooth. Initialize $b = 3$ (we don’t generally use 2 here is since $n$ is odd).

$$\left\lfloor \frac{\ln 3801911}{\ln 2} \right\rfloor = 21$$
Example

Factor 3801911 using the Pollard $p - 1$ method.

With quick calculation, we see that 3801991 is \{2, 3, 5, 7\}-smooth. Initialize $b = 3$ (we don’t generally use 2 here is since $n$ is odd).

$$\left\lfloor \frac{\ln 3801911}{\ln 2} \right\rfloor = 21$$

Now, compute

$$b^{21} \pmod{3801911} \equiv 3165492$$

and assign that value to $b$. 
Factor 3801911 using the Pollard $p - 1$ method.

With quick calculation, we see that 3801991 is $\{2, 3, 5, 7\}$-smooth. Initialize $b = 3$ (we don’t generally use 2 here is since $n$ is odd).

\[
\left\lfloor \frac{\ln 3801911}{\ln 2} \right\rfloor = 21
\]

Now, compute

\[
b^{2^{21}} \pmod{3801911} \equiv 3165492
\]

and assign that value to $b$.

Using the Euclidean Algorithm we see

\[
(3801911, 3165492 - 1) = 1
\]

so we fail on this round.
Another Example

\[ \left\lfloor \frac{\ln 3801911}{\ln 3} \right\rfloor = 13 \]

Compute \( b^{13} \pmod{3801911} \) and assign this value to \( b \).

Using the Euclidean Algorithm, we see \((3801911, 2431606 - 1) = 1\) and so we have failed this round as well.
Another Example

\[\left\lfloor \frac{\ln 3801911}{\ln 3} \right\rfloor = 13\]

Compute

\[b^{3^{13}} \pmod{3801911} \equiv 2431606\]

and assign this value to \(b\).
Another Example

\[ \left\lfloor \frac{\ln 3801911}{\ln 3} \right\rfloor = 13 \]

Compute

\[ b^{3^{13}} \pmod{3801911} \equiv 2431606 \]

and assign this value to \( b \).

Using the Euclidean Algorithm, we see

\[ (3801911, 2431606 - 1) = 1 \]

and so we have failed this round as well.
Another Example

\[
\left\lfloor \frac{\ln 3801911}{\ln 5} \right\rfloor = 9
\]

Compute \( b^{5^9} \pmod{3801911} \) and assign that value to \( b \).

When we compute the gcd, we see \((3801911, 2604247 - 1) = 1801\) and so we have found a proper factor of 3801911.
Another Example

$$\left\lfloor \frac{\ln 3801911}{\ln 5} \right\rfloor = 9$$

Compute

$$b^{5^9} \pmod{3801911} \equiv 2604247$$

and assign that value to $b$. 
Another Example

$$\left\lfloor \frac{\ln 3801911}{\ln 5} \right\rfloor = 9$$

Compute

$$b^{5^9} \pmod{3801911} \equiv 2604247$$

and assign that value to $b$.

When we compute the gcd, we see

$$(3801911, 2604247 - 1) = 1801$$

and so we have found a proper factor of 3801911.
Note: the gcd’s don’t need to be calculated at each intermediate step for this algorithm to work, but it allows us to skip some coincidences. It also may be more effective to calculate all of the values of the same type at once (i.e. all of the floors, then all of the powers, then all of the gcd’s) for efficiency’s sake.
Another Pollard $p - 1$ Example

Example

Use Pollard’s $p - 1$ method with factor base $B = \{2, 3\}$ to find a factor $p$ of 11227 so that $p - 1$ is $B$-smooth.
Another Pollard $p - 1$ Example

Example

Use Pollard’s $p - 1$ method with factor base $B = \{2, 3\}$ to find a factor $p$ of 11227 so that $p - 1$ is $B$-smooth.

Initialize $b = 3$.

\[
\left\lfloor \frac{\ln 11227}{\ln 2} \right\rfloor = 13
\]

Compute

\[b^{2^{13}} \pmod{11227} \equiv 9396\]

and assign this value to $b$. Using the Euclidean Algorithm, we see

\[(11227, 9396 - 1) = 1\]

and so we have failed this round.
Set $b = 9396$.

$$\left\lfloor \frac{\ln 11227}{\ln 3} \right\rfloor = 8$$

Compute

$$b^{38} \pmod{11227} \equiv 7631$$

and assign this value to $b$. Using the Euclidean Algorithm, we see

$$(11227, 7631 - 1) = 109$$

So, we see that $11227 = 109 \cdot 103$, where 109 is \{2, 3\} smooth.
Example

Factor $n = 54541557732143$ using the Pollard $p - 1$ method. $n$ is \{2, 3, 5\}-smooth.
Initialize \( b = 3 \).

\[
\left\lfloor \frac{\ln 54541557732143}{\ln 2} \right\rfloor = 45
\]

Compute

\[
b^{2^{45}} \pmod{54541557732143} \equiv 1333741139152
\]

and assign this value to \( b \). Using the Euclidean Algorithm, we see

\[
(54541557732143, 1333741139152 - 1) = 1
\]

and so we have failed this round.
Solution

$$\left\lfloor \frac{\ln 54541557732143}{\ln 3} \right\rfloor = 28$$

Compute

$$b^{3^{28}} \pmod{54541557732143} = 22167690980770$$

and assign this value to $b$. Using the Euclidean Algorithm, we see

$$(54541557732143, 22167690980770 - 1) = 1$$

and so this round fails as well.

$$\left\lfloor \frac{\ln 54541557732143}{\ln 5} \right\rfloor = 19$$

Compute

$$b^{5^{19}} \pmod{54541557732143} = 2268486536233$$

and assign this value to $b$. Using the Euclidean Algorithm, we see

$$(54541557732143, 2268486536233 - 1) = 54001$$
Pocklington-Lehmer Criterion

This is a specialized technique which is especially useful to test primality of numbers $N$ so that the factorization of $N - 1$ is known.
This is a specialized technique which is especially useful to test primality of numbers $N$ so that the factorization of $N - 1$ is known.

This is most often applied to numbers of special forms, such as Fermat numbers $2^{2^n} + 1$. 
This is a specialized technique which is especially useful to test primality of numbers \( N \) so that the factorization of \( N - 1 \) is known.

This is most often applied to numbers of special forms, such as Fermat numbers \( 2^{2^n} + 1 \).

A more sophisticated variant of it gives the Lucas-Lehmer test for primality of Mersenne numbers \( 2^n - 1 \), among which are the largest explicitly known primes.
Pocklington-Lehmer Criterion

This is a specialized technique which is especially useful to test primality of numbers $N$ so that the factorization of $N - 1$ is known.

This is most often applied to numbers of special forms, such as Fermat numbers $2^{2^n} + 1$.

A more sophisticated variant of it gives the Lucas-Lehmer test for primality of Mersenne numbers $2^n - 1$, among which are the largest explicitly known primes.

The test is not probabilistic, so it gives proofs of primality when it is applicable. The idea originates in the word of Eduard Lucas in 1876 and 1891, and is a sort of true converse to Fermat’s Little Theorem.
This is a specialized technique which is especially useful to test primality of numbers $N$ so that the factorization of $N - 1$ is known.

This is most often applied to numbers of special forms, such as Fermat numbers $2^{2^n} + 1$.

A more sophisticated variant of it gives the Lucas-Lehmer test for primality of Mersenne numbers $2^n - 1$, among which are the largest explicitly known primes.

The test is not probabilistic, so it gives proofs of primality when it is applicable. The idea originates in the word of Eduard Lucas in 1876 and 1891, and is a sort of true converse to Fermat’s Little Theorem.

This technique is applied to prove primality of integers $N$, or to cut down the search space for proper factors, when the factorization of $N - 1$ is known.
Lemma

Let $N$ be a positive integer, $q$ a prime divisor of $N - 1$. Let $b$ be an integer so that $b^{N-1} \equiv 1 \pmod{N}$ but $(b^{\frac{N-1}{q}} - 1, N) = 1$. Let $q^e$ be the exact power of $q$ dividing $N - 1$. Then any positive divisor $d$ of $N$ satisfies

$$d \equiv 1 \pmod{q^e}$$
Lemma

Let $N$ be a positive integer, $q$ a prime divisor of $N - 1$. Let $b$ be an integer so that $b^{N-1} \equiv 1 \pmod{N}$ but $(b^{\frac{N-1}{q}} - 1, N) = 1$. Let $q^e$ be the exact power of $q$ dividing $N - 1$. Then any positive divisor $d$ of $N$ satisfies

$$d \equiv 1 \pmod{q^e}$$

To prove, first let $d$ be a positive divisor of $N$. Since we can write $d$ as a product of primes

$$d = \prod_i p_i^{e_i}$$
A Lemma

Lemma

Let N be a positive integer, q a prime divisor of N − 1. Let b be an integer so that $b^{N-1} \equiv 1 \pmod{N}$ but $(b^{\frac{N-1}{q}} - 1, N) = 1$. Let $q^e$ be the exact power of q dividing $N - 1$. Then any positive divisor d of N satisfies

$$d \equiv 1 \pmod{q^e}$$

To prove, first let d be a positive divisor of N. Since we can write d as a product of primes

$$d = \prod_i p_i^{e_i}$$

If we can prove that $p_1 \equiv 1 \pmod{q^e}$, then certainly $d \equiv 1 \pmod{q^e}$, since multiplication modulo $q^e$ behaves nicely. So, it suffices to consider the case of prime divisors $p = d$ of N.
The hypothesis of the lemma gives

\[ b \cdot b^{N-2} \equiv 1 \pmod{N} \]

so \( b \) is multiplicatively invertible modulo \( N \), so it is relatively prime to \( N \). So, \( b \) is relatively prime to \( p \) as well.
A Lemma

The hypothesis of the lemma gives

$$b \cdot b^{N-2} \equiv 1 \pmod{N}$$

so $b$ is multiplicatively invertible modulo $N$, so it is relatively prime to $N$. So, $b$ is relatively prime to $p$ as well.

Let $t$ be the order of $b$ in the multiplicative group $(\mathbb{Z}/p)^*$: that is, $b^t \equiv 1 \pmod{p}$ but no smaller exponent will do. Fermat’s Little Theorem gives $b^{p-1} \equiv 1 \pmod{p}$, so $t|p - 1$. 
On the other hand, since \((b^{N-1} - 1, N) = 1\), certainly \((b^{N-1} - 1, p) = 1\), so \(p\) does not divide \(b^{N-1} - 1\), and

\[
b^{N-1} q \not\equiv 1 \pmod{p}
\]

That is, \(t \nmid \sqrt[\frac{N-1}{q}]\). Also, since \(b^{N-1} \equiv 1 \pmod{p}\), so \(t|N - 1\).
On the other hand, since \((b^{\frac{N-1}{q}} - 1, N) = 1\), certainly \((b^{\frac{N-1}{q}} - 1, p) = 1\), so \(p\) does not divide \(b^{\frac{N-1}{q}} - 1\), and

\[ b^{\frac{N-1}{q}} \not\equiv 1 \pmod{p} \]

That is, \(t \not| \frac{N-1}{q}\). Also, since \(b^{N-1} \equiv 1 \pmod{p}\), so \(t|N - 1\).

From \(t|N - 1\) and \(t \not| \frac{N-1}{q}\), we conclude that \(q^e|t\). Since \(t|p - 1\) and \(q^e|t\), we conclude that \(q^e|p - 1\). That is, \(p \equiv 1 \pmod{q^e}\) as asserted.
Theorem

Let $N - 1 = K \cdot U$, where $K$ and $U$ are relatively prime, the factorization of $K$ is known, and $K > \sqrt{N}$.

- If for each prime $q$ dividing $K$ there is a $b$ so that $b^{N-1} \equiv 1 \pmod{N}$ but $(b^{\frac{N-1}{q}} - 1, N) = 1$ then $N$ is prime.

- If $N$ is prime, then for each prime $q$ dividing $K$ there is a $b$ so that $b^{N-1} \equiv q \pmod{N}$ but $(b^{\frac{N-1}{q}} - 1, N) = 1$. 
If $N$ is prime, then $\mathbb{Z}/N$ has a primitive root $b$, which fulfills the condition.
Proof Of Last Theorem

If $N$ is prime, then $\mathbb{Z}/N$ has a primitive root $b$, which fulfills the condition.

Now suppose the conditions are fulfilled and show that $N$ is prime. Suppose for each $q$ dividing $K$ there is a $b$ so that $b^{N-1} \equiv 1 \pmod{N}$ but $(b^{\frac{N-1}{q}} - 1, N) = 1$. Then from the lemma all divisors of $N$ are congruent to 1 modulo $K$. 
If $N$ is prime, then $\mathbb{Z}/N$ has a primitive root $b$, which fulfills the condition.

Now suppose the conditions are fulfilled and show that $N$ is prime. Suppose for each $q$ dividing $K$ there is a $b$ so that $b^{N-1} \equiv 1 \pmod{N}$ but $(b^{N-1/q} - 1, N) = 1$. Then from the lemma all divisors of $N$ are congruent to 1 modulo $K$.

If $N$ were not prime, then it would have a prime factor $d$ in the range $1 < d < \sqrt{N}$. But the condition $d \equiv 1 \pmod{K}$, with $K > \sqrt{N}$, contradicts this inequality. This, $N$ is prime.
A specialization of this gives a very efficient test for the primality of special types of numbers.
A specialization of this gives a very efficient test for the primality of special types of numbers.

**Corollary**

(Proth, 1878) Let $N = u2^n + 1$, where $u < 2^n$ and $u$ is odd. Suppose there is a $b$ so that $b^{\frac{N-1}{2}} \equiv -1 \pmod{N}$. Then $N$ is prime.
A specialization of this gives a very efficient test for the primality of special types of numbers.

**Corollary**

(Proth, 1878) Let $N = u2^n + 1$, where $u < 2^n$ and $u$ is odd. Suppose there is a $b$ so that $b^{\frac{N-1}{2}} \equiv -1 \pmod{N}$. Then $N$ is prime.

We have $N - 1 = 2^n \cdot u$. Since $u < 2^n$, $2^n > \sqrt{N}$, which is the inequality needed, with $K = 2^n$ and $U = u$. 
A specialization of this gives a very efficient test for the primality of special types of numbers.

**Corollary**

*(Proth, 1878)* Let $N = u2^n + 1$, where $u < 2^n$ and $u$ is odd. Suppose there is a $b$ so that $b^{\frac{N-1}{2}} \equiv -1 \pmod{N}$. Then $N$ is prime.

We have $N - 1 = 2^n \cdot u$. Since $u < 2^n$, $2^n > \sqrt{N}$, which is the inequality needed, with $K = 2^n$ and $U = u$.

And, $b^{\frac{N-1}{2}} \equiv -1 \pmod{N}$ implies $b^{\frac{N-1}{2}} - 1 \equiv -2 \pmod{N}$, so $(b^{\frac{N-1}{2}}, N)$ divides $(-2, N)$, which is 1, since $N$ is odd.
And there is further improvement possible, given by the following theorem. Note that the seemingly peculiar condition that the unfactored part have no prime divisors $\leq B$ for some given bound $B$ is a practical one, since in practice small prime divisors of $N - 1$ are readily removed by trial division and Pollard’s Rho method.
Another Theorem for Primality

**Theorem**

Suppose that \( N - 1 = K \cdot U \), where the factorization of \( K \) is known and \( (K, U) = 1 \), and all prime factors of the unfactored part \( U \) are greater than the bound \( B \). Suppose that \( B \cdot K \geq \sqrt{N} \). Suppose that for each prime \( q \) dividing \( K \) there is \( b \) (depending on \( q \)) so that

\[
b^{N-1} \equiv 1 \pmod{N} \text{ but } (b^{\frac{N-1}{q}} - 1, N) = 1.
\]

And suppose that there is \( b_0 \) so that \( b_0^{N-1} \equiv 1 \pmod{N} \) but \((b_0^K - 1, N) = 1\). Then \( N \) is prime. Conversely, if \( N \) is prime then these conditions are met.
Proof of Primality Theorem

Proof: Let $p$ be a prime divisor of $N$. From the lemma, $p \equiv 1 \pmod{K}$.
Proof of Primality Theorem

Proof: Let $p$ be a prime divisor of $N$. From the lemma, $p \equiv 1 \pmod{K}$.

Let $t$ be the exact order of $b_0$ in $(\mathbb{Z}/p)^*$, so $t|p - 1$ (by Fermat’s Little Theorem). Also, $t|N - 1$ and $t \nmid K$ by the last hypothesis. Keep in mind that $K = \frac{N - 1}{U}$. 

Thus, $N$ has no prime divisors $\leq \sqrt{N}$, so it is prime.

As a special application of this, we can test the primality of the Fermat numbers $N = F_n = 2^{2^n} + 1$. 
Proof of Primality Theorem

Proof: Let $p$ be a prime divisor of $N$. From the lemma, $p \equiv 1 \pmod{K}$.

Let $t$ be the exact order of $b_0$ in $(\mathbb{Z}/p)\*$, so $t|p - 1$ (by Fermat’s Little Theorem). Also, $t|N - 1$ and $t \nmid K$ by the last hypothesis. Keep in mind that $K = \frac{N - 1}{U}$.

It cannot be that $t$ is relatively prime to $U$, since $t \nmid K$ but $t|K \cdot U$. Since $U$ has all prime factors greater than $B$, $(t, U) > B$. Since $(K, U) = 1$, from $t|p - 1$ and $K|p - 1$, by unique factorization, $[t, K]|p - 1$. In particular, $(t, U) \cdot K$ divides $p - 1$. 

Thus, $N$ has no prime divisors $\leq \sqrt{N}$, so it is prime. As a special application of this, we can test the primality of the Fermat numbers $N = F_n = 2^{2^n} + 1$.
Proof of Primality Theorem

Proof: Let $p$ be a prime divisor of $N$. From the lemma, $p \equiv 1 \pmod{K}$.

Let $t$ be the exact order of $b_0$ in $(\mathbb{Z}/p)^*$, so $t|p - 1$ (by Fermat’s Little Theorem). Also, $t|N - 1$ and $t \not| K$ by the last hypothesis. Keep in mind that $K = \frac{N-1}{U}$.

It cannot be that $t$ is relatively prime to $U$, since $t \not| K$ but $t|K \cdot U$. Since $U$ has all prime factors greater than $B$, $(t, U) > B$. Since $(K, U) = 1$, from $t|p - 1$ and $K|p - 1$, by unique factorization, $[t, K]|p - 1$. In particular, $(t, U) \cdot K$ divides $p - 1$.

Since $(t, U) > B$ and $K \cdot B > \sqrt{N}$, this gives $p - 1 > B \cdot K > \sqrt{N}$. Thus, $N$ has no prime divisors $\leq \sqrt{N}$, so it is prime.
Proof of Primality Theorem

Proof: Let $p$ be a prime divisor of $N$. From the lemma, 
$p \equiv 1 \pmod{K}$.

Let $t$ be the exact order of $b_0$ in $(\mathbb{Z}/p)^*$, so $t|p - 1$ (by Fermat’s Little Theorem). Also, $t|N - 1$ and $t \not| K$ by the last hypothesis. Keep in mind that $K = \frac{N-1}{U}$.

It cannot be that $t$ is relatively prime to $U$, since $t \not| K$ but $t|K \cdot U$. Since $U$ has all prime factors greater than $B$, $(t, U) > B$. Since $(K, U) = 1$, from $t|p - 1$ and $K|p - 1$, by unique factorization, $[t, K]|p - 1$. In particular, $(t, U) \cdot K$ divides $p - 1$.

Since $(t, U) > B$ and $K \cdot B > \sqrt{N}$, this gives $p - 1 > B \cdot K > \sqrt{N}$. Thus, $N$ has no prime divisors $\leq \sqrt{N}$, so it is prime.

As a special application of this, we can test the primality of the Fermat numbers

$$N = F_n = 2^{2^n} + 1$$
The previous corollary gives only a sufficient condition for primality, but if we allow ourselves to invoke quadratic reciprocity, we can prove a necessary and sufficient condition in the case of Fermat numbers:

**Corollary**

(Pepin’s test) The $n^{th}$ Fermat number, $F_n = 2^{2^n} + 1$ is prime iff 

$$3^{\frac{F_n-1}{2}} \equiv -1 \pmod{F_n}$$
The Law Of Quadratic Reciprocity

The **law of quadratic reciprocity** is a theorem about modular arithmetic which gives conditions for the solvability of quadratic equations modulo prime numbers.
The law of quadratic reciprocity is a theorem about modular arithmetic which gives conditions for the solvability of quadratic equations modulo prime numbers.

Let $p > 2$ be a prime number. Then

- (Supplement 1)
  \[x^2 \equiv -1 \pmod{p}\] is solvable iff $p \equiv 1 \pmod{4}$

The law can be used to determine if any quadratic equation modulo a prime has a solution but offers no help in actually finding the solution.
The law of quadratic reciprocity is a theorem about modular arithmetic which gives conditions for the solvability of quadratic equations modulo prime numbers.

Let $p > 2$ be a prime number. Then

- (Supplement 1) $x^2 \equiv -1 \pmod{p}$ is solvable iff $p \equiv 1 \pmod{4}$
- (Supplement 2) $x^2 \equiv 2 \pmod{p}$ is solvable iff $p \equiv \pm 1 \pmod{8}$

The law can be used to determine if any quadratic equation modulo a prime has a solution but offers no help in actually finding the solution.
The Law Of Quadratic Reciprocity

The **law of quadratic reciprocity** is a theorem about modular arithmetic which gives conditions for the solvability of quadratic equations modulo prime numbers.

Let \( p > 2 \) be a prime number. Then

- (Supplement 1) \( x^2 \equiv -1 \pmod{p} \) is solvable iff \( p \equiv 1 \pmod{4} \)
- (Supplement 2) \( x^2 \equiv 2 \pmod{p} \) is solvable iff \( p \equiv \pm 1 \pmod{8} \)
- (Quadratic Reciprocity) Let \( q > 2 \) be another odd prime number, \( p \neq q \), and let \( q^* = \pm q \) where the sign is plus if \( q \equiv 1 \pmod{4} \) and minus if \( q \equiv -1 \pmod{4} \). Then \( x^2 \equiv p \pmod{q} \) is solvable if and only if \( x^2 \equiv q^* \pmod{p} \) is solvable.

The law can be used to determine if any quadratic equation modulo a prime has a solution but offers no help in actually finding the solution.
The law of quadratic reciprocity is a theorem about modular arithmetic which gives conditions for the solvability of quadratic equations modulo prime numbers.

Let \( p > 2 \) be a prime number. Then

1. (Supplement 1) \( x^2 \equiv -1 \pmod{p} \) is solvable iff \( p \equiv 1 \pmod{4} \)
2. (Supplement 2) \( x^2 \equiv 2 \pmod{p} \) is solvable iff \( p \equiv \pm 1 \pmod{8} \)
3. (Quadratic Reciprocity) Let \( q > 2 \) be another odd prime number, \( p \neq q \), and let \( q^* = \pm q \) where the sign is plus if \( q \equiv 1 \pmod{4} \) and minus if \( q \equiv -1 \pmod{4} \). Then \( x^2 \equiv p \pmod{q} \) is solvable if and only if \( x^2 \equiv q^* \pmod{p} \) is solvable.

The law can be used to determine if any quadratic equation modulo a prime has a solution but offers no help in actually finding the solution.
Proof of Pepin’s Test

This is the case $k = 1$ of the previous corollary: on one hand, if the congruence holds then $F_n$ is prime. The converse is trickier. Suppose $F_n$ is prime. Then we can compute the quadratic symbol by using quadratic reciprocity.

\[
\left( \frac{3}{F_n} \right)_2 = (-1)^{\frac{(3-1)(F_n-1)}{4}} \cdot \left( \frac{F_n}{3} \right)_2 = \left( \frac{F_n}{3} \right)_2
\]
Proof of Pepin’s Test

This is the case $k = 1$ of the previous corollary: on one hand, if the congruence holds then $F_n$ is prime. The converse is trickier. Suppose $F_n$ is prime. Then we can compute the quadratic symbol by using quadratic reciprocity.

$$\left(\frac{3}{F_n}\right)_2 = (-1)^{(3-1)(F_n-1)/4} \cdot \left(\frac{F_n}{3}\right)_2 = \left(\frac{F_n}{3}\right)_2$$

Modulo 3 we compute

$$F_n = 2^{2n} + 1 = (2^{2n-1})^2 + 1$$
Proof of Pepin’s Test

This is the case \( k = 1 \) of the previous corollary: on one hand, if the congruence holds then \( F_n \) is prime. The converse is trickier. Suppose \( F_n \) is prime. Then we can compute the quadratic symbol by using quadratic reciprocity.

\[
\left( \frac{3}{F_n} \right)_2 = (-1)^{\frac{(3-1)(F_n-1)}{4}} \cdot \left( \frac{F_n}{3} \right)_2 = \left( \frac{F_n}{3} \right)_2
\]

Modulo 3 we compute

\[
F_n = 2^{2n} + 1 = (2^{2n-1})^2 + 1
\]

Since \( 2^{2n} + 1 \) is nonzero modulo 3, it is \( \pm 1 \) modulo 3, so it is the square of 1. Therefore, modulo 3

\[
F_n = 2^{2n} + 1 = (2^{2n-1})^2 + 1 \equiv 1 + 1 \equiv 2 \pmod{3}
\]
Proof of Pepin’s Test

This is the case $k = 1$ of the previous corollary: on one hand, if the congruence holds then $F_n$ is prime. The converse is trickier. Suppose $F_n$ is prime. Then we can compute the quadratic symbol by using quadratic reciprocity.

\[
\left( \frac{3}{F_n} \right)_2 = (-1)^{(3-1)(F_n-1)/4} \cdot \left( \frac{F_n}{3} \right)_2 = \left( \frac{F_n}{3} \right)_2
\]

Modulo 3 we compute

\[
F_n = 2^{2^n} + 1 = (2^{2^n-1})^2 + 1
\]

Since $2^{2^n} + 1$ is nonzero modulo 3, it is $\pm 1$ modulo 3, so it is the square of 1. Therefore, modulo 3

\[
F_n = 2^{2^n} + 1 = (2^{2^n-1})^2 + 1 \equiv 1 + 1 \equiv 2 \pmod{3}
\]

And 2 is not a square modulo 3, so

\[
\left( \frac{F_n}{3} \right)_2 = \left( \frac{2}{3} \right)_2 = -1
\]
Proof Of Pepin’s Test

And we can compute this as well by Euler’s Criterion:

**Euler’s Criterion**

For odd prime $p$ and integer $a$ such that $(a, p) = 1$ we have

$$
\left( \frac{a}{p} \right) \equiv a^{\frac{p-1}{2}} \pmod{p}
$$
And we can compute this as well by Euler’s Criterion:

**Euler’s Criterion**

For odd prime $p$ and integer $a$ such that $(a, p) = 1$ we have

$$\left( \frac{a}{p} \right) \equiv a^{\frac{p-1}{2}} \pmod{p}$$

Thus, if $F_n$ is prime, we can invoke quadratic reciprocity and Euler’s criterion to obtain the asserted congruence.
Fermat’s Numbers

It seems that only the first 5 Fermat numbers

\[ 2^0 + 1 = 3 \]
\[ 2^1 + 1 = 5 \]
\[ 2^2 + 1 = 17 \]
\[ 2^3 + 1 = 257 \]
\[ 2^4 + 1 = 65537 \]

are known to be prime. These were known by Fermat. It is known that the next 19 Fermat numbers are composite.
Fermat’s Numbers

It seems that only the first 5 Fermat numbers

\[ 2^0 + 1 = 3 \]
\[ 2^1 + 1 = 5 \]
\[ 2^2 + 1 = 17 \]
\[ 2^3 + 1 = 257 \]
\[ 2^4 + 1 = 65537 \]

are known to be prime. These were known by Fermat. It is known that

the next 19 Fermat numbers are composite.

Fermat claimed that the next Fermat number

\[ 2^5 + 1 = 4294967297 \]

is prime, but 100 years later, Euler found the proper factor 641:

\[ 4294967297 = 641 \cdot 6700417 \]
Euler did not use brute force, but instead sharply reduced the search space for the factors by proving and using the following lemma, whose proof resembles Fermat’s observation on possible prime factors of $b^n - 1$. This achieved a search speed-up by a factor of 128.
Euler did not use brute force, but instead sharply reduced the search space for the factors by proving and using the following lemma, whose proof resembles Fermat’s observation on possible prime factors of $b^n - 1$. This achieved a search speed-up by a factor of 128.

**Lemma**

*Every prime factor $p$ of $2^{2n} + 1$ satisfies*

$$p \equiv 1 \pmod{2^{n+2}}$$
Euler did not use brute force, but instead sharply reduced the search space for the factors by proving and using the following lemma, whose proof resembles Fermat’s observation on possible prime factors of \( b^n - 1 \). This achieved a search speed-up by a factor of 128.

**Lemma**

*Every prime factor \( p \) of \( 2^{2n} + 1 \) satisfies*

\[
p \equiv 1 \pmod{2^{n+2}}
\]

So, to look for factors of \( 2^{25} + 1 \), one only looks among the primes of the form \( k \cdot 2^7 + 1 \); these are 257, 641, etc. This is much easier than trial division by the 116 primes starting with 2,3, etc. and continuing to 641.
Euler

Proof: If $p$ divides $2^{2^n} + 1$, then $2^{2^n} \equiv -1 \pmod{p}$. Then $2^{2^n+1} \equiv 1 \pmod{p}$ and we conclude that the order of 2 in $(\mathbb{Z}/p)^*$ is exactly $2^{n+1}$. 
Proof: If $p$ divides $2^{2n} + 1$, then $2^{2n} \equiv -1 \pmod{p}$. Then $2^{2n+1} \equiv 1 \pmod{p}$ and we conclude that the order of 2 in $(\mathbb{Z}/p)^*$ is exactly $2^{n+1}$.

By Fermat’s Little Theorem, $2^{p-1} \equiv 1 \pmod{p}$, so the order of 2 divides $p - 1$. That is, $2^{n+1} | p - 1$. 
Proof: If $p$ divides $2^{2^n} + 1$, then $2^{2^n} \equiv -1 \pmod{p}$. Then $2^{2^n+1} \equiv 1 \pmod{p}$ and we conclude that the order of 2 in $(\mathbb{Z}/p)^*$ is exactly $2^{n+1}$.

By Fermat’s Little Theorem, $2^{p-1} \equiv 1 \pmod{p}$, so the order of 2 divides $p - 1$. That is, $2^{n+1} \mid p - 1$.

This nearly gives the result, but we can sharpen the conclusion further, as follows. For $n \geq 2$, this implies that $p \equiv 1 \pmod{8}$, so by quadratic reciprocity, $\left(\frac{2}{p}\right)_2 = 1$. That is, 2 is a square modulo $p$. 
Proof: If \( p \) divides \( 2^{2^n} + 1 \), then \( 2^{2^n} \equiv -1 \pmod{p} \). Then \( 2^{2^n+1} \equiv 1 \pmod{p} \) and we conclude that the order of 2 in \((\mathbb{Z}/p)^*\) is exactly \( 2^{n+1} \).

By Fermat’s Little Theorem, \( 2^{p-1} \equiv 1 \pmod{p} \), so the order of 2 divides \( p - 1 \). That is, \( 2^{n+1} | p - 1 \).

This nearly gives the result, but we can sharpen the conclusion further, as follows. For \( n \geq 2 \), this implies that \( p \equiv 1 \pmod{8} \), so by quadratic reciprocity, \( \left(\frac{2}{p}\right)_2 = 1 \). That is, 2 is a square modulo \( p \).

By Euler’s criterion, \( 2^{\frac{p-1}{2}} \equiv 1 \pmod{p} \). Therefore, in fact, the order of 2 in \((\mathbb{Z}/p)^*\) divides \( \frac{p-1}{2} \). From this we reach the conclusion of the lemma.
How Good Is This Speed-Up Trick?

The speed up trick is less and less effective in finding such divisors as \( n \) grows. For example, the 7\(^{th} \) Fermat number

\[
F_6 = 2^{26} + 1 = 18446744073709551617
\]

This can quickly be shown to be composite using Pepin’s test, but this is too large for Euler’s trick to be effective.

Example

Use the Pocklington-Lehmer Criteria to show that 11351 is prime.
The speed up trick is less and less effective in finding such divisors as \( n \) grows. For example, the 7\(^{th} \) Fermat number

\[
F_6 = 2^{26} + 1 = 18446744073709551617
\]

This can quickly be shown to be composite using Pepin’s test, but this is too large for Euler’s trick to be effective.

**Example**

Use the Pocklington-Lehmer Criteria to show that 11351 is prime.

If \( N = 11351 \) then \( N - 1 = 11350 = 2 \cdot 5^2 \cdot 227 \). This satisfies the form we need with \( K = 5^2 \cdot 227 \) and \( U = 2 \). Certainly, \( (K, U) = (5^2 \cdot 227, 2) = 1 \) as required, and \( K > \sqrt{11350} \).
How Good Is This Speed-Up Trick?

The speed up trick is less and less effective in finding such divisors as $n$ grows. For example, the $7^{th}$ Fermat number

$$F_6 = 2^{2^6} + 1 = 18446744073709551617$$

This can quickly be shown to be composite using Pepin’s test, but this is too large for Euler’s trick to be effective.

**Example**

Use the Pocklington-Lehmer Criteria to show that 11351 is prime.

If $N = 11351$ then $N - 1 = 11350 = 2 \cdot 5^2 \cdot 227$. This satisfies the form we need with $K = 5^2 \cdot 227$ and $U = 2$. Certainly, $(K, U) = (5^2 \cdot 227, 2) = 1$ as required, and $K > \sqrt{11350}$.

What we need is, for each prime divisor of $K$, a $b \ni b^{N-1} \equiv 1 \pmod{N}$ but so that $(b^{\frac{N-1}{q}}, N) = 1$. 
For $p_1 = 5$, choose $b = 2$.

$$2^{11350} \equiv 1 \pmod{11351}$$

and

$$(2^{\frac{11350}{5}}, 11351) = 1$$

Therefore, 11351 is prime.
For \( p_1 = 5 \), choose \( b = 2 \).

\[ 2^{11350} \equiv 1 \pmod{11351} \]

and

\[ (2^{\frac{11350}{5}}, 11351) = 1 \]

For \( p_2 = 227 \), choose \( b = 7 \).

\[ 7^{11350} \equiv 1 \pmod{11351} \]

and

\[ (7^{\frac{11350}{227}}, 11351) = 1 \]

Therefore, \( 11351 \) is prime.
Example

Use the Pocklington-Lehmer Criteria to show that 121021 is prime.
Example

Use the Pocklington-Lehmer Criteria to show that 121021 is prime.

For $N = 121021$, $N - 1 = 121020 = 2^2 \cdot 3 \cdot 5 \cdot 2017$. This satisfies the form we need with $K = 3 \cdot 5 \cdot 2017$ and $U = 2^2$ and the other criteria are met.
Example

Use the Pocklington-Lehmer Criteria to show that 121021 is prime.

For $N = 121021$, $N - 1 = 121020 = 2^2 \cdot 3 \cdot 5 \cdot 2017$. This satisfies the form we need with $K = 3 \cdot 5 \cdot 2017$ and $U = 2^2$ and the other criteria are met.

or $p_1 = 3$, choose $b = 2$.

$$2^{121020} \equiv 1 \pmod{120121}$$

and

$$(2^{\frac{121020}{3}}, 121021) = 1$$
For $p_2 = 5$, choose $b = 2$.

$$2^{121020} \equiv 1 \pmod{120121}$$

and

$$(2^{\frac{121020}{5}}, \, 121021) = 1$$

Therefore, 121021 is prime.
Another Pocklington-Lehmer Criterion Example

For $p_2 = 5$, choose $b = 2$.

$$2^{121020} \equiv 1 \pmod{120121}$$

and

$$\left(2^{\frac{121020}{5}}, 121021\right) = 1$$

For $p_3 = 2017$, choose $b = 2$.

$$2^{121020} \equiv 1 \pmod{120121}$$

and

$$\left(2^{\frac{121020}{2017}}, 121021\right) = 1$$

Therefore, $121021$ is prime.
Use the Pockington-Lehmer criterion in the special case of Proth’s corollary to write 193 in the form $u2^k + 1$ with odd $u$, etc.
Use the Pocklington-Lehmer criterion in the special case of Proth’s corollary to write 193 in the form $u2^k + 1$ with odd $u$, etc. For $N = 193$, then $N - 1 = 192 = 2^6 \cdot 3$. So, 193 has the correct form to apply Proth’s corollary.
Proth’s Corollary Example

Example

Use the Pocklington-Lehmer criterion in the special case of Proth’s corollary to write 193 in the form $u2^k + 1$ with odd $u$, etc.

For $N = 193$, then $N - 1 = 192 = 2^6 \cdot 3$. So, 193 has the correct form to apply Proth’s corollary.

We need an integer $b$ that satisfies the criteria. We can see that $b = 10$ gives

$$10^{96} \equiv 192 \equiv -1 \pmod{193}$$

which proves that 193 is prime.
Another Proth’s Corollary Example

Example

Use the Pockington-Lehmer criterion in the special case of Proth’s corollary to write 353 in the form $u2^k + 1$ with odd $u$, etc.
Example

Use the Pocklington-Lehmer criterion in the special case of Proth’s corollary to write 353 in the form \( u2^k + 1 \) with odd \( u \), etc.

For \( N = 353 \), \( N - 1 = 352 = 2^5 \cdot 11 \).
Example

Use the Pocklington-Lehmer criterion in the special case of Proth’s corollary to write 353 in the form $u2^k + 1$ with odd $u$, etc.

For $N = 353$, $N − 1 = 352 = 2^5 \cdot 11$.

We need to find an integer $b$ satisfying the criteria of Proth’s corollary.

$$3^{176} \equiv 352 \equiv −1 \pmod{353}$$

which proves 353 is prime.
Even though the proof of primality can be laborous for numbers not of special forms, strangely enough it is sometimes possible to provide easily verifiable *certificates of primality*, using the Pocklington-Lehmer criterion. This amounts to giving information which puts the recipient in a position to verify the primality with only a modest amount of work.
For example, to prove $N$ is prime, it suffices to factor

$$N - 1 = K \cdot U$$

where the factorization of $K$ is completely known,
For example, to prove $N$ is prime, it suffices to factor

$$N - 1 = K \cdot U$$

- where the factorization of $K$ is completely known,
- $K > \sqrt{N}$,
What We Need To Show

For example, to prove $N$ is prime, it suffices to factor

$$N - 1 = K \cdot U$$

- where the factorization of $K$ is completely known,
- $K > \sqrt{N}$,
- for each prime $q$ dividing $K$ find $b_q$ so that $b_q^{N-1} \equiv 1 \pmod{N}$

but $(b_q^q - 1, N) = 1$
An Alternate Form

It suffices to factor

\[ N - 1 = K \cdot U \]

- where the factorization of \( K \) is completely known,
An Alternate Form

It suffices to factor

\[ N - 1 = K \cdot U \]

- where the factorization of \( K \) is completely known,
- where no prime factor of the unfactored part \( U \) is \( \leq B \) for some bound \( B \),

In either version, the factorization of \( K \), the list of the \( b \)'s and the primes \( q \) to which they are attached together comprise a certificate of primality for \( N \).

Of course, thus certificate must include certificates of primality for the smaller primes \( q \) which occur in the factorization of \( K \), which requires certificates for the primes occurring in these certificates, and so forth.
An Alternate Form

It suffices to factor

\[ N - 1 = K \cdot U \]

- where the factorization of \( K \) is completely known,
- where no prime factor of the unfactored part \( U \) is \( \leq B \) for some bound \( B \),
- \( B \cdot K > \sqrt{N} \),
An Alternate Form

It suffices to factor

\[ N - 1 = K \cdot U \]

- where the factorization of \( K \) is completely known,
- where no prime factor of the unfactored part \( U \) is \( \leq B \) for some bound \( B \),
- \( B \cdot K \geq \sqrt{N} \),
- for each prime \( q \) dividing \( K \) find \( b_q \) so that \( b_q^{N-1} \equiv 1 \pmod{N} \)
  but \( (b_q^{N-1} - 1, N) = 1 \),
- find \( b_0 \) so that \( b_0^{N-1} \equiv 1 \pmod{N} \)
  but \((b_0^{N-1} - 1, N) = 1 \),
- the factorization of \( K \), the list of the \( b \)'s and the primes \( q \) to which they are attached together comprise a certificate of primality for \( N \).
- Of course, thus certificate must include certificates of primality for the smaller primes \( q \) which occur in the factorization of \( K \), which requires certificates for the primes occurring in these certificates, and so forth.
An Alternate Form

It suffices to factor

\[ N - 1 = K \cdot U \]

- where the factorization of \( K \) is completely known,
- where no prime factor of the unfactored part \( U \) is \( \leq B \) for some bound \( B \),
- \( B \cdot K > \sqrt{N} \),
- for each prime \( q \) dividing \( K \) find \( b_q \) so that \( b_q^{N-1} \equiv 1 \pmod{N} \)
  but \( (b_1^q - 1, N) = 1 \),
- find \( b_0 \) so that \( b_0^{N-1} \equiv 1 \pmod{N} \) but \( (b_0^K - 1, N) = 1 \).
An Alternate Form

It suffices to factor

$$N - 1 = K \cdot U$$

- where the factorization of $K$ is completely known,
- where no prime factor of the unfactored part $U$ is $\leq B$ for some bound $B$,
- $B \cdot K > \sqrt{N}$,
- for each prime $q$ dividing $K$ find $b_q$ so that $b_q^{N-1} \equiv 1 \pmod{N}$
  but $(b_q^{N-1} - 1, N) = 1$,
- find $b_0$ so that $b_0^{N-1} \equiv 1 \pmod{N}$ but $(b_0^K - 1, N) = 1$

In either version, the factorization of $K$, the list of the $b$'s and the primes $q$ to which they are attached together comprise a certificate of primality for $N$. Of course, this certificate must include certificates of primality for the smaller primes $q$ which occur in the factorization of $K$, which requires certificates for the primes occurring in these certificates, and so forth.
An Alternate Form

It suffices to factor

\[ N - 1 = K \cdot U \]

- where the factorization of \( K \) is completely known,
- where no prime factor of the unfactored part \( U \) is \( \leq B \) for some bound \( B \),
- \( B \cdot K > \sqrt{N} \),
- for each prime \( q \) dividing \( K \) find \( b_q \) so that \( b_q^{N-1} \equiv 1 \pmod{N} \)
  but \( (b_1^q - 1, N) = 1 \),
- find \( b_0 \) so that \( b_0^{N-1} \equiv 1 \pmod{N} \) but \( (b_0^K - 1, N) = 1 \)

In either version, the factorization of \( K \), the list of the \( b \)'s and the primes \( q \) to which they are attached together comprise a certificate of primality for \( N \).

Of course, thus certificate must include certificates of primality for the smaller primes \( q \) which occur in the factorization of \( K \), which requires certificates for the primes occurring in these certificates, and so forth.
An Example

For example, consider \( N = 1000000033 \). By applying Miller-Rabin with bases 2, 3, 5, 7, 11, we see that this is a probable prime. It can be easy to remove several small factors from \( N - 1 \):

\[
N - 1 = 2^5 \cdot 3 \cdot 127 \cdot 82021
\]

where we can easily verify the primality of 2, 3 and 127 directly.
For example, consider \( N = 1000000033 \). By applying Miller-Rabin with bases 2, 3, 5, 7, 11, we see that this is a probable prime. It can be easy to remove several small factors from \( N - 1 \):

\[
N - 1 = 2^5 \cdot 3 \cdot 127 \cdot 82021
\]

where we can easily verify the primality of 2, 3 and 127 directly.

Also, presuming that we do this by trial division, we would know that the remaining \( U = 82021 \) has no prime factor \( \leq 127 \). That is, in the notation at hand, 82021 is 127-smooth. Taking

\[
K = 2^5 \cdot 2 \cdot 127 = 12192
\]

we see that the condition

\[
K \cdot B \approx 1548384 > \sqrt{N} \approx 31623
\]
Now we hunt for $b$’s for the primes dividing $N - 1$, and also for the leftover unfactored part $U$. Anticipating that $N$ is prime, it should be easy to satisfy the condition $b^{N-1} \equiv 1 \pmod{N}$. 
Now we hunt for $b$’s for the primes dividing $N - 1$, and also for the leftover unfactored part $U$. Anticipating that $N$ is prime, it should be easy to satisfy the condition $b^{N-1} \equiv 1 \pmod{N}$.

The more difficult part will be the requirements that $(b^{\frac{N-1}{q}} - 1, N) = 1$. First, with the divisor $q = 2$ of $N - 1$,

$$2^{\frac{N-1}{2}} \equiv 1 \pmod{N}$$

$$3^{\frac{N-1}{2}} \equiv 1 \pmod{N}$$

so neither 2 or 3 will work here, but

$$5^{\frac{N-1}{2}} \equiv -1 \pmod{N}$$

so $b_2 = 5$ will work.
Now we hunt for $b$’s for the primes dividing $N - 1$, and also for the leftover unfactored part $U$. Anticipating that $N$ is prime, it should be easy to satisfy the condition $b^{N-1} \equiv 1 \pmod{N}$.

The more difficult part will be the requirements that $(b^{\frac{N-1}{q}} - 1, N) = 1$. First, with the divisor $q = 2$ of $N - 1$,

\[
2^{\frac{N-1}{2}} \equiv 1 \pmod{N}
\]

\[
3^{\frac{N-1}{2}} \equiv 1 \pmod{N}
\]

so neither 2 or 3 will work here, but

\[
5^{\frac{N-1}{2}} \equiv -1 \pmod{N}
\]

so $b_2 = 5$ will work.

Notice that since $N$ is odd, if $A \equiv -1 \pmod{N}$, then $A - 1 \equiv -2 \pmod{B}$, and $A$ is necessarily relatively prime to $B$. 
For the prime $q = 3$ dividing $N - 1$, we may as well try 5 again:

$$5^{\frac{N-1}{3}} \equiv 566663896 \pmod{N}$$

and the Euclidean Algorithm shows that

$$(5^{\frac{N-1}{3}} - 1, N) = 1$$

So, $b_3 = 5$ also works.
For the prime $q = 3$ dividing $N - 1$, we may as well try 5 again:

$$5 \frac{N-1}{3} \equiv 566663896 \pmod{N}$$

and the Euclidean Algorithm shows that

$$(5 \frac{N-1}{3} - 1, N) = 1$$

So, $b_3 = 5$ also works.

For the prime $q = 127$ dividing $N - 1$,

$$5 \frac{N-1}{127} \equiv 915796555 \pmod{N}$$

and the required gcd checks to be 1. So, $b_{127} = 5$ works.
Finally, since we are using the slightly improved version, we need to find $b_0$ so that $b_0^{N-1} \equiv 1 \pmod{N}$ but $(b_0^K - 1, N) = 1$. Let’s try 5 again, since we already know the first condition is fulfilled.
Finally, since we are using the slightly improved version, we need to find $b_0$ so that $b_0^{N-1} \equiv 1 \pmod{N}$ but $(b_0^K - 1, N) = 1$. Let’s try 5 again, since we already know the first condition is fulfilled.

The Euclidean Algorithm shows that

$$(5^K - 1, N) = 1$$

Thus, the data

$$K = 2^5 \cdot 3 \cdot 127 = 12192$$

$$b_2 = b_3 = b_{127} = b_0 = 5$$

is a primality certificate for $N = 1000000033$. 
What To Do With This

Once given this information, all the checking that needs to be done can be accomplished efficiently. That is, a primality certificate may leave some work to the viewer, but that remaining work must be polynomial time work rather than something burdensome like trial division or a large number, etc.
What To Do With This

Once given this information, all the checking that needs to be done can be accomplished efficiently. That is, a primality certificate may leave some work to the viewer, but that remaining work must be polynomial time work rather than something burdensome like trial division or a large number, etc.

Notice that in the last example, we did not care that 82021 was prime but only that there were no prime factors \( \leq 127 \). Here, this saved 75 further trial divisions.
Example

Show $N$ is prime if $N$ is as follows:

$$N = 3^{53} - 2^{53} = 19383245658672820642055731$$

The Miller-Rabin test indicates that this is a probable prime. We will use the Pocklington theorem to give a primality certificate for it. It is most likely unreasonable to attempt to factor $N - 1$, so we only search for factors among primes under 10000:

$$N - 1 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 53 \cdot 263 \cdot 6621792797417598667$$
Example

Show $N$ is prime if $N$ is as follows:

$$N = 3^{53} - 2^{53} = 19383245658672820642055731$$
Example

Show $N$ is prime if $N$ is as follows:

$$N = 3^{53} - 2^{53} = 19383245658672820642055731$$

The Miller-Rabin test indicates that this is a probable prime. We will use the Pocklington theorem to give a primality certificate for it. It is most likely unreasonable to attempt to factor $N - 1$, so we only search for factors among primes under 10000:

$$N - 1 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 53 \cdot 263 \cdot 6621792797417598667$$
Using Miller-Rabin base 2, we find that the latter number

\[ t = 6621792797417598667 \]

is definitely composite. It is small enough that Pollard’s Rho method should be effective: we find (after 502 cycles)

\[ t = 906043 \cdot 7308475201969 \]
A More Sizeable Example

Using Miller-Rabin base 2, we find that the latter number

\[ t = 6621792797417598667 \]

is definitely composite. It is small enough that Pollard’s Rho method should be effective: we find (after 502 cycles)

\[ t = 906043 \cdot 7308475201969 \]

It is easy to check with technology that 906043 is prime. Miller-Rabin indicates that the larger factor is a probably prime. We want to use Pocklington to verify this.
Using Miller-Rabin base 2, we find that the latter number
\[ t = 6621792797417598667 \]
is definitely composite. It is small enough that Pollard’s Rho method should be effective: we find (after 502 cycles)
\[ t = 906043 \cdot 7308475201969 \]
It is easy to check with technology that 906043 is prime. Miller-Rabin indicates that the larger factor is a probably prime. We want to use Pocklington to verify this.

Factor \( u - 1 \) as
\[ u - 1 = 2^4 \cdot 3 \cdot 83 \cdot 17539 \cdot 104593 \]
and all of these factors are readily verified with technology to be prime.
A More Sizeable Example

To prove that the factor $u$ of $N - 1$ is prime, we try to meet the conditions of Pocklington-Lehmer, but $2, 3, 5, 7$ fail as $b_2$ and the first success comes with

$$11^{u-1} \equiv 1 \pmod{u}$$

and

$$(11^{u-1}/2 - 1, u) = 1$$

which means that $b_2 = 11$ will work.
A More Sizeable Example

To prove that the factor \( u \) of \( N - 1 \) is prime, we try to meet the conditions of Pocklington-Lehmer, but \( 2, 3, 5, 7 \) fail as \( b_2 \) and the first success comes with

\[
11^{u-1} \equiv 1 \pmod{u}
\]

and

\[
(11^{\frac{u-1}{2}} - 1, u) = 1
\]

which means that \( b_2 = 11 \) will work.

But,

\[
(3^{\frac{u-1}{3}} - 1, u) = 1
\]

and \( 3^{u-1} \equiv 1 \pmod{u} \) so we can use \( b_3 = 3 \). Likewise,

\[
(3^{\frac{u-1}{5}} - 1, u) = 1
\]

\[
(3^{\frac{u-1}{7}} - 1, u) = 1
\]

\[
(3^{\frac{u-1}{53}} - 1, u) = 1
\]

\[
(3^{\frac{u-1}{263}} - 1, u) = 1
\]
A More Sizeable Example

Thus, we’ve verified that $u$ is prime.
Thus, we’ve verified that $u$ is prime.

Going back, we have

$$N - 1 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 53 \cdot 263 \cdot 906043 \cdot 7308475201969$$

and now we know that all these factors are prime.
Thus, we’ve verified that $u$ is prime.

Going back, we have

$$N - 1 = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 53 \cdot 263 \cdot 906043 \cdot 7308475201969$$

and now we know that all these factors are prime.

Now using the Pocklington-Lehmer criterion on $N$:

$$2^N - 1 \equiv 1 \pmod{N}$$

and

$$(2^{N-1/2} - 1, N) = 1$$

so $b_2 = 2$ works for $N$. 

A More Sizeable Example

But 2 fails as a candidate for $b_3$, as does 3. But then 5 works:

$$5^{N-1} \equiv 1 \pmod{N}$$

and

$$(5 \frac{N-1}{3} - 1, N) = 1$$
$$(5 \frac{N-1}{5} - 1, N) = 1$$
$$(5 \frac{N-1}{7} - 1, N) = 1$$
$$(5 \frac{N-1}{53} - 1, N) = 1$$
$$(5 \frac{N-1}{263} - 1, N) = 1$$
$$(5 \frac{N-1}{906043} - 1, N) = 1$$
$$(5 \frac{N-1}{7308475201969} - 1, N) = 1$$

Thus 5 succeeds as $b_3$, $b_5$, etc. and so our $N$ is certified prime.
A final note about the complexity: for a 54 digit number, if we used the Pocklington-Lehmer approach but without the knowledge of the large primes, we would need to factor $M \cdot N \approx 10^{52}$ which by trial division would take 10 million years at a rate of a trillion trial divisions per second.