

## Fermat's "Little" Theorem

Fermat's little theorem is a statement about primes that nearly characterizes them.

Theorem: Let  $p$  be prime and  $a$  be an integer that is not a multiple of  $p$ . Then  $a^{p-1} \equiv 1 \pmod{p}$ .

Proof: Since  $\gcd(a, p) = 1$ , the set  $\{ai \pmod{p}; i = 1, \dots, p-1\}$  is a permutation of the set  $\{1, \dots, p-1\}$ . Therefore,

$$a^{p-1} \prod_{i=1}^{p-1} i = \prod_{i=1}^{p-1} (ai) \equiv \left( \prod_{i=1}^{p-1} i \right) \cdot 1 \pmod{p}.$$

Now  $\gcd\left(\prod_{i=1}^{p-1} i, p\right) = 1$  because  $p$ , being prime, has no divisor between 2 and  $p-1$ . Thus we can cancel the product  $(p-1)!$  and get  $a^{p-1} \equiv 1 \pmod{p}$ .

## Fast Exponentiation

Input: An integer  $n \geq 0$  and a number  $a$ .

Output: The value  $a^n$ .

```
e = n
y = 1
z = a
while (e>0) {
    if (e is odd) y = y * z
    z = z * z
    e = e/2
}
return y
```

Note that we did not require  $a$  to be an integer. We said  $a$  is a “number.” In fact, the algorithm works when  $a$  is anything that can be multiplied associatively, such as a real number or even a matrix. When  $a$  is a congruence class modulo  $m$ , we can use the algorithm to compute  $a^n \bmod m$  while keeping the numbers small (smaller than  $m$ , that is), by reducing modulo  $m$  after each multiplication. The modulus  $m$  need not be prime in this application.

Fermat's little theorem can almost be used to find large primes. The theorem says that if  $p$  is prime and  $p$  does not divide  $a$ , then  $a^{p-1} \equiv 1 \pmod{p}$ . Thus, this theorem gives a test for *compositeness*: If  $p$  is odd and  $p$  does not divide  $a$ , and  $a^{p-1} \not\equiv 1 \pmod{p}$ , then  $p$  is not prime.

If the converse of Fermat's theorem were true, it would give a fast test for *primality*. The converse would say, if  $p$  is odd and  $p$  does not divide  $a$ , and  $a^{p-1} \equiv 1 \pmod{p}$ , then  $p$  is prime. This converse is not a true statement, although it is true for most  $p$  and most  $a$ . If  $p$  is a large random odd integer and  $a$  is a random integer in  $2 \leq a \leq p - 2$ , then the congruence  $a^{p-1} \equiv 1 \pmod{p}$  almost certainly implies that  $p$  is prime. However, later we will see that there are more reliable tests for primality having the same complexity. Later we will prove a true statement quite similar to the false converse.

Definition: An odd integer  $p > 2$  is called a **probable prime to base  $a$**  if  $a^{p-1} \equiv 1 \pmod{p}$ . A composite probable prime to base  $a$  is called a **pseudoprime to base  $a$** .

If we knew all base  $a$  pseudoprimes  $< L$ , then the following would form a correct primality test for odd integers  $p < L$ :

1. Compute  $r = a^{p-1} \pmod{p}$ .
2. If  $r \neq 1$ , then  $p$  is composite.
3. If  $p$  appears on the list of pseudoprimes  $< L$ , then  $p$  is composite.
4. Otherwise,  $p$  is prime.

Although this algorithm has occasionally been used, there are much better tests, some having the same complexity.

There are only three pseudoprimes to base 2 below 1000. The first one is  $p = 341 = 11 \cdot 31$ . By fast exponentiation or otherwise, one finds  $2^{340} \equiv 1 \pmod{341}$ .

One difficulty with this test is that lists of pseudoprimes, to base 2, say, do not reach high enough to encompass the range of primes of cryptographic interest. A second problem is that there are too many pseudoprimes to any particular base; the list of all of them would be too long.

## A true converse of Fermat's little theorem

Theorem: Let  $m > 1$  and  $a$  be integers such that  $a^{m-1} \equiv 1 \pmod{m}$ , but  $a^{(m-1)/p} \not\equiv 1 \pmod{m}$  for every prime  $p$  dividing  $m - 1$ . Then  $m$  is prime.

Proof: The congruence  $a^{m-1} \equiv 1 \pmod{m}$  implies that  $\gcd(a, m) = 1$  and the order  $e$  of  $a$  modulo  $m$  divides  $m - 1$ . The second condition,  $a^{(m-1)/p} \not\equiv 1 \pmod{m}$  for every prime  $p$  dividing  $m - 1$ , shows that  $e$  is not a proper divisor of  $m - 1$ . Therefore,  $e$  must equal  $m - 1$ . But  $e$  divides  $\phi(m)$ . Hence,  $m - 1 \leq \phi(m)$ . But if  $m > 1$  is composite, then  $\phi(m) \leq m - 2$ . Thus,  $m$  cannot be composite.

This theorem can be used to prove primeness of almost any prime  $m$  for which we know the factorization of  $m - 1$ . If  $m$  is an odd prime, then usually a small prime  $a$  can be found quickly which will satisfy all the conditions. The principal difficulty in using the theorem to prove that a prime  $m$  is prime is not the search for  $a$ , but rather finding the factorization of  $m - 1$ . If  $m - 1$  has been factored, then one can use this simple algorithm to try to prove it is prime.

## Lucas-Lehmer $m - 1$ primality test

1. Choose  $a = 2$  or choose a random  $a$  in  $2 \leq a \leq m - 1$ .
2. Compute  $r = a^{m-1} \bmod m$ .
3. If  $r \neq 1$ , then  $m$  is composite.
4. Check that  $a^{(m-1)/p} \not\equiv 1 \pmod{m}$  for each prime  $p$  dividing  $m - 1$ .
5. If all these incongruences are true, then  $m$  has been proved prime.

6. If they are not satisfied, then either choose another  $a$  (either the next small prime or a new random  $2 \leq a \leq m - 1$ ) and go back to Step 2, or else give up if many  $a$  have already been tried.

If  $m$  is a large prime, then the expected number of  $a$  this algorithm must try before finding one that works is known to be  $< 2 \ln \ln m$ . If  $m$  is a large composite, then the algorithm will almost certainly stop in Step 3.

Many cryptographic algorithms require prime numbers of a certain size. If the prime need not be secret, then one can get one from a book or web site. Alternatively, one can form a random large prime by one of the methods for finding secret primes.

Every prime has a short, simple proof of its primality, but it is usually difficult to discover such a proof when the prime is large.

There are three ways to find large secret primes for cryptographic use.

1. Test random large numbers and choose the first probable prime. In other words, use “industrial-grade primes.”
2. Test random large numbers for being probably prime. When you find one, prove rigorously that it is prime.
3. Use random numbers to construct a large prime having special form which permits an easy rigorous proof of its primality.

## Stronger Probable Prime Tests

Definition: An odd positive integer  $n$ , with  $n - 1 = 2^s d$ , where  $d$  is odd, is a **strong probable prime to base  $a$**  if either  $a^d \equiv 1 \pmod{n}$  or  $a^{d \cdot 2^r} \equiv -1 \pmod{n}$  for some  $0 \leq r < s$ . A **strong pseudoprime to base  $a$**  is a composite strong probable prime to base  $a$ .

The left to right variation of fast exponentiation computes  $a^{n-1} \pmod{n}$  by first finding  $a^d \pmod{n}$ , and then squaring the result  $s$  times modulo  $n$ . Thus, fast exponentiation automatically produces the remainders, which are compared to  $+1$  or  $-1$  in the definition.

Every prime  $p$  is a strong probable prime to every base  $a$  it does not divide because  $a^{p-1} \equiv 1 \pmod{p}$  and the only solutions to  $x^2 \equiv 1 \pmod{p}$  are  $x \equiv \pm 1 \pmod{p}$ .

One can show that there are infinitely many strong pseudoprimes to every base  $a \geq 1$ .

The bases  $+1$  and  $-1$  are not interesting because every odd composite integer  $n$  is a pseudoprime and a strong pseudoprime to both of these bases.

It is easy to see that every strong probable prime is a probable prime to the same base, because the definition says that we will get  $\pm 1$  at some step before the last step in computing  $a^{n-1} \pmod{n}$  by fast exponentiation, and this number will be squared at least once.

## Lucas Probable Prime Tests

In this section, we develop the theory of binary linear recurrences and a probable prime test using them. The test is based on a generalization of the following theorem, which we will prove later.

Theorem: If  $n$  is prime,  $u_i$  is the  $i$ -th Fibonacci number and  $(n/5)$  is the Legendre symbol, then  $n$  divides  $u_{n-(n/5)}$ .

Examples:

Since  $(3/5) = -1$ , 3 divides  $u_4 = 3$ .

Since  $(11/5) = +1$ , 11 divides  $u_{10} = 55$ .

Since  $(5/5) = 0$ , 5 divides  $u_5 = 5$ .

Definition: The **Lucas sequences with parameters**  $P$  and  $Q$  are the two sequences  $\{u_n\}$  and  $\{v_n\}$  defined by  $u_0 = 0$ ,  $u_1 = 1$ ,  $v_0 = 2$ ,  $v_1 = P$ , and  $u_n = Pu_{n-1} - Qu_{n-2}$ ,  $v_n = Pv_{n-1} - Qv_{n-2}$ , for  $n \geq 2$ . We sometimes write  $u_n = u_n(P, Q)$  and  $v_n = v_n(P, Q)$  to show the dependence on the parameters  $P$  and  $Q$ . Let  $x^2 - Px + Q$  be the **recurrence polynomial** associated to the Lucas sequences, let  $D = P^2 - 4Q$  be the discriminant of this polynomial and let  $\alpha$  and  $\beta$  be the two zeros of the polynomial.

To get the Fibonacci numbers  $u_n$ , let  $P = 1$  and  $Q = -1$ . In that case,  $v_n = v_n(1, -1)$  are called the **Lucas numbers**. The recurrence polynomial is  $x^2 - x - 1$ , with discriminant  $D = 5$  and roots  $\alpha, \beta = (1 \pm \sqrt{5})/2$ .

The parameters  $P$  and  $Q$  will always be integers. In this case, all the  $u_n$  and  $v_n$  are integers. Usually, we will also assume that  $D = P^2 - 4Q$  is not a square. This implies that  $D \neq 0$ , so  $\alpha \neq \beta$ . From the equation  $(x - \alpha)(x - \beta) = x^2 - Px + Q$ , we see that  $\alpha + \beta = P$  and  $\alpha\beta = Q$ . If we let  $\alpha = (P + \sqrt{D})/2$  and  $\beta = (P - \sqrt{D})/2$ , then  $\alpha - \beta = \sqrt{D}$ . It is easy to show by induction on  $n$  that

$$u_n = \frac{\alpha^n - \beta^n}{\alpha - \beta} = \frac{\alpha^n - \beta^n}{\sqrt{D}} \quad \text{and} \quad v_n = \alpha^n + \beta^n$$

for  $n \geq 0$ .

There is a simple way to compute Lucas sequences using  $2 \times 2$  matrices. Define  $L = \begin{bmatrix} P & -Q \\ 1 & 0 \end{bmatrix}$  and, for  $n \geq 0$ ,  $A_n = \begin{bmatrix} u_{n+1} & v_{n+1} \\ u_n & v_n \end{bmatrix}$ . Then  $A_0 = \begin{bmatrix} 1 & P \\ 0 & 2 \end{bmatrix}$ . A simple induction shows that  $A_n = L^n A_0$  for  $n \geq 0$ , where  $L^0$  means the identity matrix.

This is not just a pretty formula. It provides a quick way to compute  $u_n$  and  $v_n$  when  $n$  is huge. The fast exponentiation algorithm given above applies to anything we can multiply associatively, including matrices. Thus we can compute  $L^n$  in our formula with only  $O(\log n)$  matrix multiplications. If we wish to compute  $u_n \bmod m$  or  $v_n \bmod m$ , we should reduce each matrix entry modulo  $m$  as it is computed. This will keep the numbers small ( $< m^2$ ) even if  $n$  has hundreds of digits.

We need the formulas in the next theorem to prove the generalization of the theorem stated above.

Theorem: For integers  $n \geq 0$  we have

$$2^{n-1}u_n = \sum_{\substack{i=0 \\ i \text{ odd}}}^n \binom{n}{i} P^{n-i} D^{(i-1)/2}$$

and

$$2^{n-1}v_n = \sum_{\substack{i=0 \\ i \text{ even}}}^n \binom{n}{i} P^{n-i} D^{i/2}.$$

Proof: Begin with the formula for  $u_n$  in terms of the two roots of the recurrence polynomial.

$$u_n = \frac{\alpha^n - \beta^n}{\alpha - \beta} = \frac{(P + \sqrt{D})^n - (P - \sqrt{D})^n}{2^n \sqrt{D}}.$$

Apply the binomial theorem to the two binomial powers and get

$$2^n u_n = \frac{1}{\sqrt{D}} \sum_{i=0}^n \binom{n}{i} P^{n-i} \left( (\sqrt{D})^i - (-\sqrt{D})^i \right).$$

When  $i$  is even, the terms  $(\sqrt{D})^i - (-\sqrt{D})^i$  cancel, but they add when  $i$  is odd. Hence

$$2^n u_n = \frac{2}{\sqrt{D}} \sum_{\substack{i=0 \\ i \text{ odd}}}^n \binom{n}{i} P^{n-i} (\sqrt{D})^i.$$

We obtain the first formula when we cancel one  $\sqrt{D}$  and divide by 2. The second formula is proved the same way, starting from  $v_n = \alpha^n + \beta^n$ .

The next theorem generalizes the theorem stated above and proves it.

Theorem: If  $p$  is an odd prime not dividing  $PQ$ , then

$$\begin{aligned}u_{p-(D/p)} &\equiv 0 \pmod{p}, \\u_p &\equiv (D/p) \pmod{p} \text{ and} \\v_p &\equiv v_1 = P \pmod{p}.\end{aligned}$$

If also  $\gcd(p, D) = 1$ , then

$$v_{p-(D/p)} \equiv 2Q^{(1-(D/p))/2} \pmod{p}.$$

Proof: First let  $n = p$  in the formula for  $u_n$  in the theorem just proved. Note that since  $p$  is prime, it divides every binomial coefficient  $\binom{p}{i}$  with  $1 \leq i \leq n - 1$ . The only remaining term with odd  $i$  in the sum is the one with  $i = p$ . Also,  $2^{p-1} \equiv 1 \pmod{p}$  by Fermat's little theorem. We find

$$u_p \equiv D^{(p-1)/2} \equiv (D/p) \pmod{p},$$

by Euler's criterion. This proves the second formula.

To prove the first one, let  $n = p + 1$  in the formula for  $u_n$  in the theorem just proved. Since  $p$  is prime, it divides the binomial coefficients  $\binom{p+1}{i}$  with  $2 \leq i \leq p - 1$ . The only odd  $i$  not in this interval are  $i = 1$  and  $i = p$ , so the sum reduces to two terms. We have  $2^p \equiv 2 \pmod{p}$  by Fermat's little theorem. We find

$$\begin{aligned} 2u_{p+1} &\equiv (p+1)P^p D^0 + (p+1)P^1 D^{(p-1)/2} \equiv \\ &\equiv P(1 + (D/p)) \pmod{p}, \end{aligned}$$

where we have again used Euler's criterion and also  $P^p \equiv P \pmod{p}$  by Fermat's little theorem. If  $(D/p) = -1$ , we see immediately that  $p$  divides  $u_{p+1} = u_{p-(D/p)}$ . If  $(D/p) = +1$ , then we have  $2u_{p+1} \equiv 2P \pmod{p}$ , so  $u_{p+1} \equiv P \pmod{p}$ . By the second formula, which we just proved,  $u_p \equiv (D/p) = +1 \pmod{p}$ . Substituting into the recurrence formula,  $u_{p+1} = Pu_p - Qu_{p-1}$ , we find  $P \equiv P(+1) - Qu_{p-1} \pmod{p}$ . This yields  $Qu_{p-1} \equiv 0 \pmod{p}$ . Since  $\gcd(p, Q) = 1$  we can divide by  $Q$  and find that  $p$  divides  $u_{p-1} = u_{p-(D/p)}$ . The other two congruences are proved the same way, using the formula for  $v_n$  in the theorem just proved.

Two matrices are **congruent** modulo  $n$  if their corresponding entries are congruent modulo  $n$ .

Let  $I$  denote the  $2 \times 2$  identity matrix.

Fermat's Little Theorem for Lucas Sequences

Theorem: Let  $L = \begin{bmatrix} P & -Q \\ 1 & 0 \end{bmatrix}$  be the matrix used to compute the Lucas sequences with parameters  $P$  and  $Q$ . Let  $D = P^2 - 4Q$ . Let  $p$  be a prime not dividing  $2PQD$ . If  $(D/p) = +1$ , then  $L^{p-1} \equiv I \pmod{p}$ . In any case,  $L^{p^2-1} \equiv I \pmod{p}$ .

Proof: Suppose  $(D/p) = +1$ . Then the previous theorem says that

$$A_{p-1} = \begin{bmatrix} u_p & v_p \\ u_{p-1} & v_{p-1} \end{bmatrix} \equiv \begin{bmatrix} 1 & P \\ 0 & 2 \end{bmatrix} = A_0 \pmod{p}.$$

But also  $A_{p-1} = L^{p-1}A_0$ . Since  $A_0$  has determinant 2, it is invertible modulo the odd prime  $p$ . Therefore,  $L^{p-1} \equiv I \pmod{p}$ . We have  $L^{p^2-1} = (L^{p-1})^{p+1} \equiv I^{p+1} = I \pmod{p}$ .

Now suppose  $(D/p) = -1$ . In this case, the previous theorem says that

$$A_p = L^p A_0 = \begin{bmatrix} u_{p+1} & v_{p+1} \\ u_p & v_p \end{bmatrix} \equiv \begin{bmatrix} 0 & 2Q \\ -1 & P \end{bmatrix} \pmod{p}.$$

Since  $A_0$  is invertible modulo  $p$ , with inverse

$$A_0^{-1} \equiv \frac{1}{2} \begin{bmatrix} 2 & -P \\ 0 & 1 \end{bmatrix} \pmod{p},$$

$$\text{we find that } L^p \equiv \begin{bmatrix} 0 & Q \\ -1 & P \end{bmatrix} \pmod{p} \text{ and } L^{p+1} = LL^p \equiv \begin{bmatrix} Q & 0 \\ 0 & Q \end{bmatrix} =$$

$QI \pmod{p}$ . Then  $L^{p^2-1} = (L^{p+1})^{p-1} \equiv Q^{p-1}I \equiv I \pmod{p}$  by Fermat's little theorem.

Why did we call this theorem, “Fermat’s little theorem for Lucas sequences?” Fermat’s little theorem says that if you raise  $a$ , relatively prime to a prime  $p$ , to the power  $p - 1$  modulo  $p$ , you will get the identity element 1 of  $R_p$ . The theorem says that if you raise  $L$ , which describes a Lucas sequence, to the power  $p^2 - 1$  modulo the prime  $p$ , relatively prime to  $2PQD$ , you will get the identity element  $I$  in the cyclic group of powers of  $L$  modulo  $p$ . Note that  $p^2 - 1$  is the order of the multiplicative group of the field  $\mathbb{F}_{p^2}$  with  $p^2$  elements, and this group is also cyclic.

The four congruences of the Theorem above are valid at least for all primes  $p$  not dividing  $2PQD$ . In fact, when  $p$  is allowed to be composite, but  $\gcd(p, 2PQD) = 1$ , any two of the congruences imply the other two. Baillie and Wagstaff found that they seldom hold when  $p$  is an odd composite number. They focused on the first congruence when they made this definition.

**Definition:** A **Lucas probable prime with parameters  $P$  and  $Q$**  is an integer  $n > 1$  with  $\gcd(n, 2PQD) = 1$  and  $u_{n-(D/n)} \equiv 0 \pmod{n}$ , where  $D = P^2 - 4Q$ . A **Lucas pseudoprime with parameters  $P$  and  $Q$**  is a composite Lucas probable prime with the same parameters.

Baillie and Wagstaff showed that Lucas pseudoprimes are rare and defined Lucas analogues of Euler and strong pseudoprimes.

The bases  $a = \pm 1$  are avoided in probable prime tests because every odd number is a probable prime to these bases. Likewise, the parameters  $(P, Q) = (1, 1)$  and  $(-1, 1)$  must be avoided in Lucas probable prime tests because every odd  $n$  satisfies  $u_{n-(D/n)} \equiv 0 \pmod{n}$  with either of these choices.

We mentioned that  $D$  should not be a square. In fact,  $D$  should not even be a quadratic residue modulo  $n$  in a Lucas probable prime test on  $n$ . For if  $D \equiv b^2 \pmod{n}$ , then  $(D/n) = +1$ ,  $P \equiv b + 2 \pmod{n}$ ,  $Q \equiv b + 1 \pmod{n}$ ,  $\alpha \equiv Q \pmod{p}$ ,  $\beta \equiv 1 \pmod{p}$ , and  $u_{n-1} \equiv (Q^{n-1} - 1)/b \pmod{n}$ ; so, the Lucas test is an ordinary probable prime test in disguise. The complexity of a Lucas probable prime test is several times that of a probable prime test; so, one might as well perform a probable prime test with base  $a = b + 1$  rather than a Lucas probable prime test with  $D \equiv b^2 \pmod{n}$ .

Selfridge proposed the following method of choosing the parameters for a Lucas probable prime test that avoids the problem of  $D$  being a quadratic residue modulo  $n$ . Let  $D$  be the first member of the sequence  $5, -7, 9, -11, 13, -15, \dots$  for which the Jacobi symbol  $(D/n) = -1$ . Let  $P = 1$  and  $Q = (1 - D)/4$ . It is known that the expected number of  $D$ 's which must be tried, before a suitable one is found, is about 1.8. When  $n \equiv 2$  or  $3 \pmod{5}$ , the first discriminant,  $D = 5$ , is chosen and the Lucas sequence is the Fibonacci numbers.

Pinch has computed the pseudoprimes to base 2 up to  $10^{13}$ . With Selfridge's parameter choices for the Lucas sequence, not a single known strong pseudoprime to base 2 is also a Lucas pseudoprime. In fact, Pomerance, Selfridge and Wagstaff made this conjecture.

## CONJECTURE

*No odd composite positive integer is both a strong pseudoprime to base 2 and a Lucas pseudoprime with Selfridge's choice of parameters  $P$  and  $Q$ .*

In 1980, they offered \$30 for a proof or disproof of the conjecture, and have since raised this reward to \$620. The conjecture is certainly true for all integers  $< 10^{13}$ .

A simplified version of the conjecture asserts that there is no composite number  $n$  whose last decimal digit is 3 or 7, which is a strong pseudoprime to base 2 and which divides the Fibonacci number  $u_{n+1}$ .

Those cryptographers satisfied with “industrial-grade primes” should select strong probable primes to base 2 which are also Lucas probable primes, as in the Conjecture. The tests are simple, elegant and provide the added benefit that if you are the first to detect a failure of the conjecture, then you will collect \$620.

In 2002, the American National Standards Institute selected this algorithm for choosing industrial-grade primes for cryptography as ANSI Standard X9-80.

## Rigorous Proof of Primality

The Lucas-Lehmer primality test may be used iteratively to construct large, random primes.

[Really simple large prime generation algorithm]

Begin with a prime  $p_1$ . Let  $i = 1$ . Repeat Steps 1 through 5 until  $p_i$  is large enough.

1. Let  $k$  be a random small integer and let  $n = 2kp_i + 1$ .
2. If  $2^{n-1} \not\equiv 1 \pmod{n}$ , then  $n$  is composite by Fermat's little theorem, so return to Step 1.

3. Otherwise,  $n$  is probably prime, so try to prove  $n$  is prime using the Lucas-Lehmer theorem just stated. Note that  $n - 1 = 2kp_i$  is easy to factor completely because it has the known prime factor  $p_i$ , which should be removed first, and because  $k$  is small. Try the primes  $< 30$  for possible values of  $a$ .

4. If you succeed in finding an  $a$  which satisfies the conditions of the theorem, then  $n$  is proved prime. Let  $p_{i+1} = n$  and  $i = i + 1$  and go to Step 1.

5. Otherwise, try a new random  $k$ . (Go to Step 1.)

During the construction of the last  $p_i$  one may have to restrict the size of  $k$  to produce a prime of the required size. Typical sizes for  $k$  before the last  $p_i$  might be 10 or 15 decimal digits—small enough to factor easily by trial division.

There are several enhancements to the Lucas-Lehmer Theorem that accelerate this algorithm. The first is that one can use different  $a$ 's for different prime factors  $p$  of  $n - 1$ , provided one checks  $a^{n-1} \equiv 1 \pmod{n}$  once for each  $a$  used.

Suppose the prime  $p_i$  must be secret, and is a factor of a public key. The algorithm has the advantage that  $p_i$  will be immune to discovery by the Pollard  $p - 1$  method, because  $p_i - 1$  has the large prime factor  $p_{i-1}$ .

However, the algorithm is slow because it builds up primes little by little. The next theorem, which may be viewed as another enhancement of the Lucas-Lehmer Theorem, allows one to jump ahead to larger primes much faster because it requires only a partial factorization of  $n - 1$ .

Pocklington-Lehmer Theorem: Let  $n$  be odd and  $n - 1 = FR$ , where the complete factorization of  $F$  is known. Suppose that for every prime  $p$  dividing  $F$  there is an integer  $a$  such that  $a^{n-1} \equiv 1 \pmod{n}$  and  $\gcd(a^{(n-1)/p} - 1, n) = 1$ . Then every prime factor of  $n$  is  $\equiv 1 \pmod{F}$ .

If also  $F \geq \sqrt{n}$ , then  $n$  is prime.

This theorem allows us to construct a new prime with about twice as many digits as the previous one.

[Doubling the size of a random prime]

Input: A prime  $p$ .

Output: A prime  $n$  near  $p^2$ .

```
repeat {
  let  $k$  be a random integer between  $p/2$  and  $p$ .
   $n = 2kp + 1$ 
  if  $2^{n-1} \bmod n$  is not 1, restart this loop.
  try to prove  $n$  is prime via
      the Pocklington-Lehmer Theorem.
  if you succeed, end the loop.
} until  $n$  is prime.
```

By the prime number theorem, the expected number of iterations of the loop needed to find a prime  $n$  is about  $\ln p$ .

In applying the Pocklington-Lehmer Theorem in the algorithm above, let  $F = p$  and  $R = 2k$ . It may seem strange to put the known factor 2 into  $R$ , but it would take longer to check the hypotheses of the Pocklington-Lehmer Theorem if we put the 2 in  $F$ . For the integer  $a$  of the theorem, try the ten primes  $< 30$ .

To construct a large prime near  $X$ , begin with a known prime near the  $2^i$ -th root of  $X$ , for some convenient  $i$ , and apply the algorithm  $i$  times with the known prime as the first input, and each subsequent input equal to the previous output. Adjust  $k$  in the final iteration of the loop to make the last  $n$  just the right size. The large prime  $p$  will have a rigorous proof of its primality and  $p-1$  will have a large prime factor to make  $p$  immune to discovery by the Pollard  $p-1$  method.