Massachusetts Institute of Technology
6.857: Network and Computer Security
Professor Ronald L. Rivest

Handout 14
October 15, 1998

Solutions to Problem Set 3

Problem 3-1. There were many approaches to this problem. Most solutions immediately noted that factoring is much harder than finding primes, and so elected to 'create' the filling and then test if the numbers around it were prime. Some also restricted the filling to special types of numbers which made it easier to test the numbers around it. Many used Miller-Rabin to test, and a few tested using elliptic curve primality provers on their results 'just in case'.

There were solutions in LISP, Java, C, C++, and Basic, and sizes varied from around 30 bits to our class record of 6166 bits. The following is one method of solving the problem:

William Chuang
Mark Silis
Serena Chan

Problem Description

A pair (p, r) of primes is known as *twin primes* if the equation $r = p + 2$ holds. Examples of twin primes are (17,19) and (41,43). An *Oreo cookie* is defined to be a triple (p, q, r) where (p, r) is a pair of twin primes, together with a factorization of the intermediate value $q = p + 1$.

Our objective is to find a "fresh" oreo cookie – one whose factorizable middle value (the *"filling"*) is divisible by 1998.

Addendums include:

- Use a computer program (do not look in the literature)
- Use a "big number" subroutine library
- Find the largest fresh Oreo cookie you can

Search algorithms

First attempt

Our first attempt to find "fresh" Oreo cookies involved writing a C++ program that used the *Integer* class (thanks for the tip, Yoav!).

The program took command-line arguments specifying the bit-size of the starting search number. It then chose $2^{\text{bit size}} + 1$ and tested it for primality using several iterations of the Fermat primality test (initially 40 iterations with the first 40 primes). If this number passed the test, we added 2 to it and tested that for primality as well. If this also passed the test, we printed out the number.

This was satisfactory for small twin primes (in the 35-45 bit range), but once we reached larger numbers (60+ bits) the factorization of the "filling" required too many hours to be feasible. We calculated that factoring some of the 200-bit "fillings" would require at least several weeks, unless we were lucky.

Second attempt

Rather than futilely search for CPU cycles to factor our "filling", we rewrote the Oreo cookie code.

We first generated a table of the first 12,000 primes. The search algorithm was changed to calculate the "filling" and then check if the "cookies" on either side of the "filling" were primes (once again using Fermat's primality test).

We needed to allow for greater variation in our testing space, so we threw in a triple for-loop around the test and multiplied our starting number by 1998*primes[i]*primes[j]*primes[k], where (i,j,k) were the loop variables. If the search exceeded the loop space then the starting number was multiplied by 2 and the search re-started. While this unfortunately limits the search space greatly, we felt there wasn't much choice in the matter.

Prime number generation

The prime-number table was generated using Erastothenes' sieve with a floating table window. Standard programs to do this include "primes" which is distributed with many BSD systems. We would have used a larger prime table, but the number of primes quickly became prohibitive to add to the table in terms of work.

Prime number validation

Within the search program we validated primes using Fermat's (Little) Theorem, which states:

> If p is a prime and if a is any integer, then $a^p = a(\text{mod } p)$. In particular, if p does not divide a, then $a^{p-1} = 1(\text{mod } p)$.

Unfortunately, this is not a definitive test – numbers which pass the test are termed "weak probable prime base" (or PRP). It is possible for pseudoprimes (composite PRP's) such as 25, 91, 217, and 341 to pass the test.

We run the test over the first 100 primes (ie, using multiple bases) in an attempt to screen out as many pseudoprimes as possible. Unfortunately, there are composite integers called **Carmichael numbers** for which Fermat's test fails until we run across one of its factors. Carmichael numbers are "rare" (there are only 2163 of them less than 25 billion), but there are infinitely many of them. Since our hoped-for search space is greater than 600-bits, we needed a better test or proof. **ECPP - Elliptical Curve Primality Proving**

We decided to take any Oreo cookies we found and run the twin primes through the Elliptic Curve Primality Proof. This is not a test like Fermat's little theorem, but can be used to certify primality of a number.

Theorem

> Let N be an integer prime to 6, E an elliptic curve over $\mathbf{Z}/N\mathbf{Z}$, together with a point P on E and m and s two integers with $s \mid m$. For each prime divisor q of s, we put $(m/q)P = (x_q : y_q : z_q)$. We assume that $mP = O_E$ and $gcd(z_q, N) = 1$ for all q. Then, if p is a prime divisor of N, one has $\#E(\mathbf{Z}/p\mathbf{Z}) \equiv 0 \text{ mod s}$.

We can combine the above theorem with Schoof's algorithm that computes $\#E(\mathbf{Z}/p\mathbf{Z})$ in time $O((\log p)^{8+\epsilon})$ to get the Goldwasser-Killian algorithm.

The ECPP relies on the concept of elliptical curves; the premise is to do the opposite of the Goldwasser-Killian algorithm. A pseudolisting of the procedure would be:

1.set $i = 0$, $N_0 = N$ (where N is a probable prime)

2.**while** $N_i > N_{small}$

- (a)find a fundamental discriminant $-D_i$ which is good for N_i. In other words, $N_i = \pi_i \pi_i^{'}$ in $\mathbf{K} = \mathbf{Q}(\sqrt{-D_i})$
- (b)if one of the $w(-D_i)$ numbers $m_1, \ldots, m_w$ ($m_r = N_K(v_r\pi_i - 1)$ where v_r is a unit in $\mathbf{K}$) is probably factored, go to step 2.3, else go to step 2.1
- (c)store $\{i, N_i, D_i, v_r\pi_i, m_r, F_i\}$ where $m_r = F_i N_{i+1}$ Here F_i is a completely factored integer and N_{i+1} is a probable prime. Set $i = i + 1$ and go to step 2.1
- (d)compute a root j of $H_{D_i}(X) \equiv 0 \text{ mod } N_i$
- (e)compute an equation of the curve E_i of invariant j and whose cardinality modulo N_i is m_i
- (f)find a point P_i on the curve E_i
- (g)check the conditions of the theorem with $s = N_{i+1}$ and $m = m_i$. In other words, check that $Q_i = F_i P_i \neq O_{E_i}$ but $sQ_i = O_{E_i}$

3.**end**

This algorithm is best explained in the paper **"Elliptic Curves and Primality Proving"** by A.O.L. Atkin and F. Morain [1]

Rather than implement this by ourselves, we used F. Morain's ECPP package which includes self-verification routines and provides "certificates" containing a long list of numbers that constitute the proof of primality for that number. This "certificate" can be used to more quickly verify the primality of the number.

The following is the cream of the largest cookie found:

Joshua Fisher
David Ratajczak
Scott Schneider
Eric Sit

$2^{409} \cdot 3^{233} \cdot 522 \cdot 7^{81} \cdot 11^{47} \cdot 13^{35} \cdot 17^{33} \cdot 19^{27} \cdot 238 \cdot 294 \cdot 311 \cdot 37^{8} \cdot 41^{5} \cdot 43^{9} \cdot 476 \cdot 53^{7} \cdot 59^{6} \cdot 61^{3} \cdot 67^{9}$
$71^{6} \cdot 73^{6} \cdot 79^{6} \cdot 83^{7} \cdot 89^{4} \cdot 97^{7} \cdot 101^{7} \cdot 103^{3} \cdot 107^{7} \cdot 109^{3} \cdot 113^{4} \cdot 127^{6} \cdot 131^{2} \cdot 137^{8} \cdot 139^{3} \cdot 149^{6} \cdot 151^{5} \cdot 157^{3}$
$163^{2} \cdot 167^{2} \cdot 173^{4} \cdot 179^{2} \cdot 181^{2} \cdot 191^{2} \cdot 193^{5} \cdot 197^{2} \cdot 199 \cdot 211 \cdot 223^{2} \cdot 227^{2} \cdot 229 \cdot 233 \cdot 239^{4} \cdot 241 \cdot 257^{2}$
$263^{3} \cdot 269^{3} \cdot 271^{3} \cdot 277 \cdot 281 \cdot 283^{3} \cdot 293^{2} \cdot 307 \cdot 311^{3} \cdot 313 \cdot 317^{2} \cdot 331^{3} \cdot 337 \cdot 347^{2} \cdot 353^{2} \cdot 359 \cdot 367$
$379 \cdot 389^{2} \cdot 397 \cdot 401^{2} \cdot 409^{2} \cdot 419 \cdot 439^{2} \cdot 443 \cdot 479^{3} \cdot 503 \cdot 509 \cdot 521 \cdot 523 \cdot 541 \cdot 547 \cdot 557 \cdot 569$
$571^{2} \cdot 599 \cdot 601^{2} \cdot 607 \cdot 631 \cdot 641 \cdot 643 \cdot 653 \cdot 659 \cdot 677 \cdot 683^{2} \cdot 691 \cdot 709 \cdot 719 \cdot 743 \cdot 751^{2} \cdot 757^{2}$
$773 \cdot 787 \cdot 797^{2} \cdot 823 \cdot 829 \cdot 863 \cdot 883 \cdot 919^{3} \cdot 929 \cdot 937 \cdot 941 \cdot 967 \cdot 977 \cdot 991 \cdot 1021 \cdot 1031 \cdot 1051$
$1063 \cdot 1069 \cdot 1103 \cdot 1117 \cdot 1151 \cdot 1153 \cdot 1163 \cdot 1171 \cdot 1181 \cdot 1187 \cdot 1193^{3} \cdot 1223 \cdot 1231 \cdot 1291 \cdot 1307$
$1361 \cdot 1399 \cdot 1427 \cdot 1439 \cdot 1447 \cdot 1459 \cdot 1567^{2} \cdot 1597 \cdot 1609 \cdot 1627^{2} \cdot 1637^{2} \cdot 1697 \cdot 1831 \cdot 1873 \cdot 1877$
$1931 \cdot 1979^{2} \cdot 1993^{2} \cdot 1999^{3} \cdot 2053^{2} \cdot 2087 \cdot 2129 \cdot 2141 \cdot 2161 \cdot 2239 \cdot 2287 \cdot 2311 \cdot 2341 \cdot 2393 \cdot 2437$
$2539 \cdot 2579 \cdot 2609 \cdot 2647 \cdot 2657 \cdot 2833 \cdot 2897 \cdot 3041 \cdot 3079 \cdot 3167 \cdot 3181 \cdot 3259 \cdot 3329 \cdot 3547 \cdot 3557$
$3613 \cdot 3623 \cdot 3637 \cdot 3733 \cdot 3877 \cdot 3917 \cdot 3919 \cdot 4001 \cdot 4049 \cdot 4157 \cdot 4253 \cdot 4283 \cdot 4297 \cdot 4513 \cdot 4651$
$4721 \cdot 4909 \cdot 4957 \cdot 5021 \cdot 5279 \cdot 5419 \cdot 5449 \cdot 5639 \cdot 5653 \cdot 5683^{2} \cdot 5701 \cdot 5801 \cdot 5843 \cdot 5869 \cdot 6257$
$6311 \cdot 6491 \cdot 6719 \cdot 7121 \cdot 7607 \cdot 8209 \cdot 8741 \cdot 8969 \cdot 9043 \cdot 9419 \cdot 9829 \cdot 9851 \cdot 9931 \cdot 10663 \cdot 10937$
$11243 \cdot 11317 \cdot 11519 \cdot 11549 \cdot 11579 \cdot 11677 \cdot 11813 \cdot 11827 \cdot 12511 \cdot 12763 \cdot 13063 \cdot 14389 \cdot 14479$
$14717 \cdot 15077 \cdot 15731 \cdot 15823 \cdot 15937 \cdot 17291 \cdot 19403 \cdot 19843 \cdot 19867 \cdot 20333 \cdot 22307 \cdot 22739 \cdot 23021$
$24113 \cdot 24793 \cdot 25169 \cdot 26891 \cdot 26927 \cdot 27407 \cdot 27437 \cdot 29387 \cdot 40169 \cdot 40709 \cdot 43801 \cdot 45481 \cdot 46279$
$49169 \cdot 52691 \cdot 54443 \cdot 57073 \cdot 62143 \cdot 64451 \cdot 64849$

[1] http://ultralix.polytechnique.fr/~morain/Articles/articles.english.html