University College London
Department of Computer Science

Cryptanalysis Lab 4

J. P. Bootle

March 3, 2016

jonathan.bootle.14@ucl.ac.uk
Version 2.0

The Fermat Factorisation Algorithm

Click on the green letter before each question to get a full solution. Click on the green square to go back to the questions.

Exercise 1.

(a) Given that $1309 = 47^2 - 30^2$, what is the prime factorisation of 1309?

(b) Let N, a, b be odd, positive integers such that $N = ab$. Show that N can be expressed as the difference between two square numbers.

(c) Write a function 'Fermat' which implements the following factorisation algorithm. The function should take input N, and output a, b such that $N = ab$. This method is called the Fermat Factorisation algorithm.

- Compute `n = ceil(sqrt(N))`.
- Beginning with n, compute $n^2 - N$.
- Check if the value you get is a square. If so, output a factorisation of N.

- Increment n and repeat until a factorisation is found.

(d) Use your code to find the factors of $N = 1488391, 1467181, 1456043$. Can you see a connection between the running time of your code and the prime factors of N?

Implementing the Pollard-Rho Algorithm

Click on the green letter before each question to get a full solution. Click on the green square to go back to the questions.

EXERCISE 2.

(a) Write a function `pollard_rho` which implements the low-memory version of the Pollard-Rho algorithm. The function should take input N, and output a, b such that $N = ab$. Run the algorithm for a fixed number of iterations. You may wish to structure your code as follows.

- Definition of a sub-function for iteration.
- Set the number of iterations to do.

- Main loop using the iterative function.
- At each step of the main loop, compute a greatest common divisor.
- Return a factorisation $[a, b]$ or output 'Fail'.

(b) According to the analysis of the running time of the Pollard-Rho algorithm, how many iterations should we expect to use before the algorithm succeeds in finding a factorisation?

(c) Test your algorithm by attempting to factorise the integers $M_n = 2^n - 1$, for $n = 80, 85, 90$. What is the largest value of n that your program can handle in 10 seconds?

Elliptic Curve Factorisation Algorithm

Click on the green letter in front of each sub-question (e.g. (a)) to see a solution. Click on the green square at the end of the solution to go back to the questions.

EXERCISE 3. In this exercise, you will use Sage to explore how integers are factored using elliptic curves.

(a) To create an elliptic curve Ep defined by $y^2 = x^3 + ax + b$ over $\mathbb{F}_p$, use `E = EllipticCurve(GF(p),[a,b])`. Create an elliptic curve Ep defined by $y^2 = x^3 + x + 4$, over the finite field of size 11.

(b) To create a curve EN defined by $y^2 = x^3 + ax + b$ over $\mathbb{Z}_N$, use `E = EllipticCurve(Integers(N),[a,b])`. Create a curve EN defined by $y^2 = x^3 + x + 4$, over the ring of integers modulo 438713.

(c) Type $PN = EN(100584, 115601)$ to create the corresponding point on EN. Similarly, type $Pp = Ep(100584, 115601)$ to create the same point, reduced modulo 11, on Ep. Type `Pp` to view the point modulo 11. The point should be expressed as $(x : y : 1)$. The point at infinity is $(0 : 1 : 0)$.

(d) Type `Ep.cardinality()` to find out the number of elliptic curve points modulo 11. What is the number of points? Type `a*Pp` to compute multiples of the point Pp. What is the order of Pp in the elliptic curve group Ep?

(e) Set $QN = 8 * PN$ and use SAGE to compute QN. Now, try to compute $9 * PN = QN + PN$. What happens? Compute the difference between the x coordinates of PN and QN, and compute

the greatest common divisor of this with N. Look at the formulae for adding elliptic curve points. Does this explain the error?

(f) Set $N = 20077$. Consider the curve E defined by the equation $y^2 = x^3 + x + 5$. Assume that N has a prime factor p with $| E(\mathbb{Z}_p) |$ being 5-powersmooth. Given that $P = (427, 466)$ is a point on $E(\mathbb{Z}_N)$, factor N.

Solutions to Exercises

Exercise 1(a) We have $1309 = (47 + 30)(47 - 30) = 77 \cdot 17$. □

Exercise 1(b) Write $N = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2$. Each bracketed expression is a whole number, because N is odd, so a, b are both odd, and therefore $a \pm b$ is even. □

Exercise 1(c) The following code implements the Fermat Factorisation algorithm.

```
def fermat(N):
    n = ceil(sqrt(N))
    while True:
        M = n*n-N
        m = floor(sqrt(M))
        if m == sqrt(M):
            return [n+m,n-m]
        n = n+1
```

□

Exercise 1(d) The Fermat factorisation method finds factors of N as $n+m$ and $n-m$, where $N = n^2 - m^2$. The value of $n+m$ is at least $\sqrt{N}$ and increases as n is incremented. Therefore, Fermat factorisation runs fastest on integers N which have factors close to $\sqrt{N}$. □

Exercise 2(a) The following code implements the Pollard-Rho algorithm.

```
def pollard_rho(N):
    n = floor(sqrt(sqrt(N))) # adjust this value
    ai = randint(1,N-1)
    a2i = ai
    for k in range(1,n):
        ai = (ai*ai + 1) % N
        a2i = (a2i*a2i + 1) % N
        a2i = (a2i*a2i + 1) % N
        d = gcd(abs(ai-a2i),N)
        if not (d in [1,N]):
            return [d,floor(N/d)]
    return 'fail'
```

□

Exercise 2(b) According to the heuristic analysis based on the Birthday paradox, we would expect to succeed after $O(\sqrt{p})$ iterations, where p is the smallest prime factor of N. □

Exercise 3(a) Use `Ep = EllipticCurve(GF(11),[1,4])` to produce the correct elliptic curve. □

Exercise 3(b) Use `EN = EllipticCurve(Integers(438713),[1,4])` to produce the correct curve. □

Exercise 3(d) There are 9 points on the elliptic curve Ep defined modulo 11. The order of a group element divides the order of the group, 9, and Pp is not the point at infinity, so we only have to check whether $3 * Pp$ is equal to the point at infinity or not. However, $3 * Pp = (3 : 1 : 1)$, so 3 is not the order of Pp. Therefore, the order of Pp is 9. □

Exercise 3(e) You can use `a = randint(0,1009)` to get a. □