University College London
Department of Computer Science

Cryptanalysis Exercises Lab 03

P. Spacek, N. Courtois,
J. Bootle

February 7, 2019

n.courtois@ucl.ac.uk
Version 1.2

1. Safe Primes

Exercise 1.

(a) Let p be a prime such that $p = 2q + 1$, where q is also prime. We call p with this property a 'strong' prime (ambiguous term to avoid) or rather a 'safe' prime. Let g be a generator of $(\mathbb{Z}/p\mathbb{Z})^*$. How can we generate a group of order q?

2. Modular Inverses

A student proposed to compute the modular inverse of a mod n as follows:

$$a^{-1} = a^{\phi(n)-1}$$

Which theorem is this based on? When this is actually true? Explain what are 3 serious problems with this method.

2.1. Bézout Theorem

Bézout's Theorem: Let a and b be integers with greatest common divisor

$$d = GCD(a, b).$$

Then, there exist integers x and y such that

$$ax + by = d.$$

More generally, the integers of the form $ax + by$ are exactly the multiples of d.

Remarks. For integers it was known 150 years earlier. Bézout shows that it holds also for polynomials, "Théorie générale des équations algébriques", Paris, France, 1779.

2.2. Computing a modular inverse with [Extended] Euclid

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.

Click here for a reminder of the Extended Euclidean Algorithm.

EXERCISE 2. Let p and q be two distinct primes.

(a) Show how to use the extended Euclidean algorithm to simultaneously compute $p^{-1} \mod q$ and $q^{-1} \mod p$.

(b) What is the complexity of this approach in terms of bit operations?

(c) Compute $11^{-1} \bmod 17$ using this method.

(d) Implement the Extended Euclidean Algorithm in SAGE, and use it to compute $7^{-1} \mod 159$.

3. 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 3.

(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) The incomplete function 'Fermat' implements a factorisation algorithm. The function takes input N, and should output a, b such that $N = ab$. Please fill in the question marks to obtain a complete implementation of the Fermat factorisation algorithm.

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

(d) Use your completed 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?

Solutions to Exercises

Exercise 1(a) The order of g is $\phi(p) = p - 1 = 2q$. We can compute $g^2 \mod p$, and this element will have order q, generating a subgroup of size q. □

Exercise 2(a) If necessary, swap p and q so that $p > q$. Since p and q are distinct primes, $gcd(p, q) = 1$, and there exist integers A and B such that $Ap + Bq = 1$. Then $A = p^{-1} \mod q$ and $B = q^{-1} \mod p$. We compute these using the Extended Euclidean Algorithm.

One way to implement the extended Euclidean Algorithm is to use the back-tracking approach: work backwards in a GCD computation. Otherwise, the following method allows the answer to be calculated without working backwards.

Set $r_{-1} = p$ and $r_0 = q$. We also set $A_{-1} = 1$, $A_0 = 0$, and $B_{-1} = 0$, $B_0 = 1$. For each i, find a_{i+1}, r_{i+1} such that $r_{i-1} = a_{i+1}r_i + r_{i+1}$ as in the Euclidean Algorithm.

At each stage, compute $A_{i+1} = a_iA_i + A_{i-1}$ and $B_{i+1} = a_iB_i + B_{i-1}$. These values satisfy $A_ip - B_iq = (-1)^{i+1}r_i$. When the algorithm terminates after n steps, $r_n = gcd(p, q) = 1$. We take $A = (-1)^{n+1}A_n$ and $B = (-1)^nB_n$. □

Exercise 2(b) The Extended Euclidean Algorithm requires $O(\log(p)^2)$ bit operations. □

	a_i	A_i	B_i
-	-	1	0
-	-	0	1
$17 = 1 \cdot 11 + 6$	1	1	1
$11 = 1 \cdot 6 + 5$	1	1	2
$6 = 1 \cdot 5 + 1$	1	2	3

Figure 1: Gcd of 17 and 11

Exercise 2(c) Again, we can easily find the answer using the backtracking method. The alternative solution from an earlier part of the question is shown below.

Set $r_{-1} = 17, r_0 = 11$. Figure 1 shows working for the Extended Euclidean Algorithm. We find that $2 \cdot 17 - 3 \cdot 11 = 1$. Therefore $11^{-1} \bmod 17 \equiv -3 \equiv 14$.

□

Exercise 2(d) The SAGE code shown implements the Extended Euclidean Algorithm:

```
def gcd1(a,b):
    if mod(a,b) == 0:
        return [b,0,1]
    else:
        q = (a- (a%b) )/ b
        [d,r,s]=gcd1(b,a-q*b)
        return [d,s,r-q*s]
```

When run on 159 and 7, the output is $[1, 3, -68]$, so the answer is -68.

□

Exercise 3(a) We have $1309 = (47 + 30)(47 - 30) = 77 \cdot 17.$ □

Exercise 3(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 3(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 3(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 $\sqrt{N}$. □