University College London
Department of Computer Science

Cryptanalysis Lab 5

J. P. Bootle

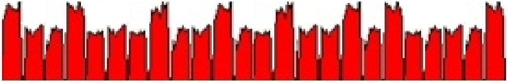
May 27, 2018

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

Side Channel Attacks

Bob's RSA implementation has public key $(N, e) = (183181, 5)$ where N is a product of two primes p and q. He receives a ciphertext c from Alice. Bob uses the following square-and-multiply algorithm to compute $m = c^d \mod N$.

```
def BobPower(a,k,n):
    K = bin(k)[2:]              # K is binary expansion of k,
    A = a % n                   # with the most significant bit
    c = 1                       #stored in K[0]
    if int(K[0])==1:
        c = (c*A) % n           #modular multiplication here
    for j in range(1,len(K)):
        c = (c^2) % n           #modular squaring is cheap
        if int(K[j])==1:
            c = (c*A) % n       #modular multiplication uses
    return c                    #more power
```



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) The power usage of Bob's CPU as he decrypts the ciphertext is given in the graph shown. What value for the decryption exponent d is suggested by the power usage graph?

(b) Using the values of d, e and N, can we compute p and q?

Continued Fractions and RSA

For any real number r, its **continued fraction representation** is a (possibly infinite) sequence of integers $[q_0; q_1, q_2, \ldots]$ such that

$$r = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{q_4 + \ldots}}}}$$

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) (Bonus Question) If $r = \frac{a}{b}$, show that the continued fraction representation of r can be computed with Euclid's Algorithm on (a, b).

(b) SAGE contains functions for computing continued fraction expansions. Try "$a = continued\_fraction(pi); a$".

(c) By truncating the continued fraction expansion of a number, we can obtain a rational approximation to that number. The rational number A_n/B_n representing the continued fraction expansion $[q_0; q_1, \ldots, q_n]$ is called the nth convergent. Try "$a.convergent(3)$", and compare the decimal expansion of this number to that of π. To how many decimal places do the two values agree?

(d) (Bonus Question) It is known that if $|r-m/n| < 1/2n^2$, then m/n is a convergent to r. For an RSA public/private key-pair, show that if $N = pq$ with $q < p < 2q$, and $d < N^{1/4}/3$, then k/d is a convergent to e/N, where $ed - 1 = k\phi(N)$.

(e) Let $N = 90581, e = 17993$ be an RSA public-key. Use continued fractions to find d.

Generating Discrete Logarithm Instances

Recall that a prime p is called a 'strong prime' if $p = 2q + 1$, where q is also prime.

The following function generates random discrete logarithm instances. On input n, the function first finds the smallest strong prime p that is greater than n. Thus, $\mathbb{Z}_p^*$ has a subgroup of order q. Finally, the function generates two random elements g, h of the subgroup, and outputs $[p, q, g, h]$.

```
def dlog_gen(n):
    p = next_prime(n)
    while not is_prime( floor((p-1)/2) ):
        p = next_prime(p)
    x = randint(1,p-1)
    y = randint(1,p-1)
    g = x*x % p
    h = y*y % p
    return [p,floor( (p-1)/2 ),g,h]
```

Copy and paste the code into SAGE. This function will be used to generate discrete logarithm instances for the following questions.

Index Calculus Algorithm

Exercise 3. In this exercise, you will use an index calculus algorithm to find discrete logarithms, using SAGE like a pocket-calculator.

(a) Use the code in the previous section to generate a discrete logarithm instance (p, q, g, h) with $p > 1000$.

(b) Compute $z = g^a h^b \mod p$ for random values of a and b. Write a function called "smooth_factors" that checks whether z is 13-smooth. The function should return "False" if z is not smooth, and should return a vector of exponents if z is smooth. For example, if $z = 2^3 \cdot 3^2 \cdot 13$, the the corresponding output would be $(3, 2, 0, 0, 0, 1)$.

(c) If z is a product of small primes, then store the exponents of the product in the rows of a matrix E. Store a and b in the rows of a matrix M. Repeat the process above until the number of rows of E is one greater than the number of columns. For example, for $z = 2^3 \cdot 3^2 \cdot 13$, then $g^{-a}h^{-b} \cdot 2^3 \cdot 3^2 \cdot 13$ the corresponding row in the matrix E would be $(3, 2, 0, 0, 0, 1)$, and in M, (a, b).

(d) Look up the "MatrixSpace" command in the SAGE documentation. Create a space $S2$ of 7×6 matrices over $GF(2)$, and a space Sq of 7×6 matrices over $GF(q)$. Next, create new matrices, $E2$ over $GF(2)$ and Eq over $GF(q)$, such that $E2 = E \mod 2$, and $Eq = E \mod q$.

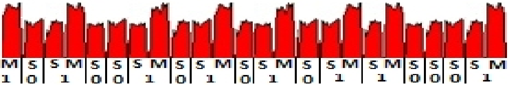
(e) Look up the commands "kernel" in the SAGE documentation. Use it to find vectors $v2$ and vq such that $v2 * M2 = \mathbf{0} \mod 2$ and $vq * Mq = \mathbf{0} \mod q$. If no such vectors exist, then repeat the process above in order to obtain new matrices.

(f) Look up the SAGE commands for the Chinese Remainder Theorem. Use these commands to produce a vector v which satisfies $v2 = v \mod 2$ and $vq = v \mod q$.

(g) The vector v satisfies $v * E = 0 \mod (p-1)$. Compute $(A, B) = v * M$, and hence find the discrete logarithm of h with respect to g.

Solutions to Exercises

Exercise 1(a) When computing $c^d \mod N$, the square-and-multiply algorithm will either do a squaring operation, or a squaring operation then a multiplication, depending on whether each bit in the binary representation of d is a 0 or a 1. The multiplication is usually more computationally intensive. This means that we can read off the binary representation of d straight from the graph.



This gives us $d = 72357$. □

Exercise 1(b) Since $N = pq$, we know that $\phi(N) = (p-1)(q-1) = pq - p - q + 1$. Thus $p + q = N - \phi(N) + 1$. Furthermore, in RSA, we know that $ed = 1 \mod \phi(N)$. Therefore, $ed - 1 = k\phi(N)$ for some positive integer k.

Now, consider the quadratic equation

$$X^2 - \left(N - \frac{ed-1}{k} + 1\right)X + N = X^2 - (p+q)X + pq = (X-p)(X-q) = 0$$

We already know N, e and d. If we guess values of k, we can try to use the quadratic formula to obtain p and q. Guessing $k = 2$ gives us $X^2 - 2290X + 183181$, and then we recover $p = 2207$ and $q = 83$ from the quadratic formula.

The disadvantage of this approach is that it seems to involve guessing k and we might have given up if k was large and prime.

Here is a second solution. We know that $ed - 1 = k\phi(n)$. For any a with $\gcd(a, N) > 1$, we have $a^{\phi(N)} \equiv 1 \mod N$. Substituting in the values of e and d, we know that $a^{361784} \equiv 1 \mod N$. Taking inspiration from the Miller-Rabin test, we can use this fact to try and find square roots of 1 not congruent to $\pm 1 \mod N$.

We divide 361784 by 2 as many times as possible, to get 45223. Now, we pick a random value of a between 1 and $N-1$. We check that $\gcd(a, N) = 1$ (if not, we have already factored N). Then, we raise to the power $45223 \mod N$, and then square repeatedly, hoping that we get a non-trivial square-root. For example, with $a = 2$, we get $A = 97109$, and find that $A^2 \equiv 1 \mod N$. Therefore, $(A+1)(A-1) \equiv 0 \mod N$, and $\gcd(A \pm 1, N)$ give factors of N. Finally, $\gcd(97110, 183181) = 83$ and $183181 = 83 \times 2207$.

It can be shown, using the Chinese Remainder Theorem, that this approach has a success probability of roughly $\frac{1}{2}$, in the case that N is a product of two distinct primes. □

Exercise 2(a) Using Euclid's Algorithm, we find integers $r_0, r_1, r_2, \ldots$ such that:

$$\begin{aligned} a &= q_0 b + r_0 \\ b &= q_1 r_0 + r_1 \\ r_0 &= q_2 r_1 + r_2 \\ &\vdots \\ r_{n-1} &= q_{n+1} r_n \end{aligned}$$

We substitute the expression for a into $\frac{a}{b}$ and rearrange to get

$$\frac{a}{b} = \frac{q_0 b + r_0}{b} = q_0 + \frac{r_0}{b} = q_0 + \frac{1}{\frac{b}{r_0}}$$

We can then substitute the expression for b and rearrange in a similar way to get

$$\frac{a}{b} = q_0 + \frac{1}{q_1 + \frac{1}{\frac{r_0}{r_1}}}$$

Repeating the same idea, we eventually arrive at

$$r = q_0 + \cfrac{1}{q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{q_4 + \ldots + \cfrac{1}{q_{n+1}}}}}}$$

□

Exercise 2(b) SAGE should display "[3; 7, 15, 1, 292, 1, 1, 1, 2, 1, 3, 1, 14, 2, 1, 1, 2, 2, 2, 2, ...]". □

Exercise 2(c) The third convergent to π is 355/113, which approximates π to 6 decimal places. □

Exercise 2(e) The first convergent is $1/5$, which shows that $d = 5$.

□

Exercise 3(a) An example of a discrete logarithm instance is $[p, q, g, h] = [1019, 509, 277, 487]$. □

Exercise 3(b) Using the following code, after a few tries, you can discover, for example, that $g^{308}h^{809} = 2^4 \cdot 3^2 \mod p$.

```
def factor_base(B):
    factor_base = [2]
    while factor_base[-1] < B:
        prime = next_prime(factor_base[-1])
        factor_base = factor_base + [prime]
    return factor_base
```

(continued on next page)

```
def smooth_factors(n):
    z=n
    exponent_list = []
    for prime in factor_base(13):
        exponent = 0
        while mod(z,prime)==0:
            z=z/prime
            exponent = exponent+1
        exponent_list = exponent_list + [exponent]
    if z>1:
        return False
    return exponent_list
```

□

Exercise 3(c) Using the example from the previous part, we have a row $(4, 2, 0, 0, 0, 0)$ in the matrix E and a row $(308, 809)$. Combining the following code snippets with "while" loops should enable you to produce a matrix.

```
a = randint(1,p-1)
b = randint(1,p-1)
z=(g**a)*(h**b)%p;z
exponent_list = smooth_factors((g**a)*(h**b)%p)
exponent_list
```

```
    M = M + [[a,b]];M
    E = E + [exponent_list];E
```

Finally, type "M = matrix(M)" and "E = matrix(E)". Example matrices may be found on the next page.

$$M = \begin{pmatrix} 308 & 809 \\ 575 & 611 \\ 576 & 447 \\ 531 & 280 \\ 676 & 132 \\ 603 & 940 \\ 854 & 140 \end{pmatrix}, \qquad E = \begin{pmatrix} 4 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \\ 3 & 1 & 0 & 0 & 0 & 1 \\ 3 & 0 & 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 2 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

□

Exercise 3(d) The following code produces the required matrices.

```
S2 = MatrixSpace(GF(2),7,6)
Sq = MatrixSpace(GF(q),7,6)
E2 = S2(E)
Eq = Sq(E)
```

Example matrices:

$$E2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}, \qquad Eq = \begin{pmatrix} 4 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \\ 3 & 1 & 0 & 0 & 0 & 1 \\ 3 & 0 & 1 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 2 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

□

Exercise 3(e) The "kernel" command produces several vectors which span the left kernel of a matrix. Taking the first vector each time, the following code produces the required vectors.

```
V2 = E2.kernel();V2;v2=V2[1];v2
Vq = Eq.kernel();Vq;vq=Vq[1];vq
```

□

Exercise 3(f) The following code produces the vector v.

```
v = vector([crt(ZZ(v2[i]),ZZ(vq[i]),2,q) for i in range(0,len(v2))])
```

Continuing with the running example, we have

$$v = (1, 552, 932, 424, 42, 806, 0)$$

□

Exercise 3(g) Since $g^a h^b = 2^{e_2} 3^{e_3} \dots 13^{e_{13}}$ for each pair of rows in M and E, by finding a vector v in the kernel of E, we are able to construct A and B such that $g^A h^B = 1 \mod p$. Now, the discrete logarithm can be computed as

$$k = -A * inverse\_mod(B, p-1)\%(p-1)$$

In our example, we find $k = 1012$.

□