Exam TMA4160 Cryptography
Suggested solutions

December 7, 2016

Problem 1

We use the obvious map between the English alphabet and the numbers $\{0, 1, 2, \ldots, 25\}$, which means that the plaintext letters `OR` correspond to 14 and 17, while the ciphertext letters `VK` correspond to 21 and 10, giving us the linear system of equations

$$21 \equiv 14a + b \pmod{26}$$
$$10 \equiv 17a + b \pmod{26}$$

Subtracting the first equation from the second, we get $-11 \equiv 3a$, or $a \equiv -99 \equiv 5$ and $b \equiv 21 - 14 \cdot 5 \equiv 3$. Using this key, we get the decryption

`orangutans by any other name` (spaces inserted for readability).

Problem 2

a) The usual fast exponentiation algorithm in this case simply reduces to four doublings. We use the doubling formula four times to get $2P = (11, 9)$, $4P = (17, 16)$, $8P = (10, 6)$ and $16P = (3, 18)$. Since $18 \equiv -1 \pmod{19}$, we see that $16P = -P$.

b) From a), we know that P has order 17, so the number of points must be divisible by 17. The only number divisible by 17 in the range allowed by Hasse's theorem is 17, so the number of points on the curve is 17. This is a prime number, so the group is cyclic.

Problem 3

We begin with $t = \lceil \sqrt{n} \rceil = 498$. The square root of $498^2 - n$ is not a square, but $499^2 - n = 42^2$, giving us that

$$n = (499 - 42)(499 + 42) = 457 \cdot 541.$$

Problem 4

The Schnorr signing algorithm computes $\gamma = r - \beta a \bmod n$, where a is the signing key, r is a random number, β is the hash of something known, and n is the known (prime) group order. When r is known, we can recover the signing key as

$$a = \beta^{-1}(r - \gamma) \bmod n.$$

Problem 5

a) This is obvious since $y_{k,i+1}$ is computed from y_{ki} by applying f, we get that $y_{k,i+1} = y_{l,j+1}$, $y_{k,i+2} = y_{l,j+2}$ and so on. Since only the last element in each sequence is in D, the sequences must therefore terminate at the same time, so $y_{k,n_k} = y_{l,n_l}$.

b) The obvious function is:

$$h(a,b) = \begin{cases} (a+1 \bmod p, b) & x^a g^b \in S_1 \\ (2a \bmod p, 2b \bmod p) & x^a g^b \in S_2 \\ (a, b+1 \bmod p) & x^a g^b \in S_3 \end{cases}$$

This is clearly easy to compute (one or two arithmetic operations).

As in the previous problem, we can compute a_{k,n_k} and b_{k,n_k} by repeatedly applying h (which we prove is correct using induction). We also want to keep track of $y_{k,i}$ so that we do not have to recompute $x^a g^b$ for every iteration (one group operation compared to many more to do the exponentiations).

c) The key here is that m is not too large, which means that we can sort the list of triples on the first coordinate using $O(m \log m)$ steps, which is an easy computation. Then a quick search will find two tuples with the same y, but distinct a's (modulo p), which gives us an equation of the form

$$x^a g^b = x^{a'} g^{b'} \qquad \Rightarrow \qquad x^{a-a'} = g^{b'-b}.$$

Computing $\log_g x$ is now easy.

d) We generate sequences for $k = 1, 2, \ldots, m$ by repeating the following steps m times:

- Sample a_{k1} and b_{k1} from the uniform distribution on $\{0, 1, 2, \ldots, p-1\}$ and compute $y_{k1} = x^{a_{k1}} g^{b_{k1}}$. (This ensures that y_{k1} is sampled from the uniform distribution on G.)
- Compute $y_{k,i+1} = f(y_{ki})$, $(a_{k,i+1}, b_{k,i+1}) = h(a_{ki}, b_{ki})$, for $i = 1, 2, \ldots$, until $y_{k,i+1} \in D$, when we set $n_k = i + 1$. (As long as f is "random-looking", this should happen after about p/r steps except with very small probability, so we can ignore this possibility.)

- Add $(y_{k,n_k}, a_{k,n_k}, b_{k,n_k})$ to the list.

Once this process stops, sort the list on the first coordinate. Then search for two tuples with identical first coordinates and distinct second coordinates. If no such tuples are found, we fail. Otherwise, we have tuples (y,a,b) and (y,a',b') with $a \not\equiv a' \pmod{p}$. We output the discrete logarithm

$$\log_g x = (b'-b)/(a-a') \bmod p.$$

Producing m sequences with a total of $\sum n_k$ elements requires roughly $\sum n_k$ group operations. We may ignore the initial exponentiation, since we expect that cost to be very small compared to $\sum n_k$.

In addition, we need to do $2\sum n_k$ arithmetic operations. Again, we may ignore the final arithmetic operations involved in computing $\log_g x$.

We may assume that $\sum n_k \approx mp/r$, so our estimate is that we need mp/r group operations and $2mp/r$ arithmetic operations.

e) Since we assume that f is "random-looking", we are now allowed to assume that all the y_{ki} have been sampled from the uniform distribution on G.

By the birthday paradox, an approximate lower bound on the probability that we find a collision among mp/r elements is

$$1-\exp(-m^2p/2r^2)$$

which begins to be significant when mp/r is about $\sqrt{p}$, which means that $m^2p/2r^2 \approx 1/2$.

For the purely random case, we should also consider the possibility that these collisions happen in the same sequence, which corresponds to one of the possibilities we ignored in d). But an approximate upper bound on this probability is $(p/r)^2/2p$, which over m sequences is upper bounded by $mp/2r^2$, which is small.