MATH 158
MIDTERM 2
11 NOVEMBER 2015

Name : Solutions

- The time limit is 50 minutes.
- No calculators or notes are permitted.
- For any problem asking you to write a program, you may write in a language of your choice or in pseudocode, as long as your answer is sufficiently specific to tell the runtime of the program.
- Each problem is worth 10 points.

1	/10	**2**	/10
3	/10	**4**	/10
5	/10	**6**	/10
		$\sum$	/60

(1) Suppose that Alice's RSA public key is the pair (N, e).

(a) Once Alice has decided on (N, e), how does she determine her decrypting exponent d? Why isn't Eve able to do the same thing, and decrypt messages intended for Alice?

She knows how to factor N as pq (p, q prime).
She computes $\varphi(N) = (p-1)(q-1)$, then

$$d \equiv e^{-1} \bmod \varphi(N).$$

Eve cannot perform this computation since she doesn't know the prime factors of N.

(b) Suppose that Alice wishes to use the same public key (N, e) to sign a document D. How does she compute the signature S? How does Victor (who only knows the public key) verify that the signature is correct?

$S \equiv D^d \bmod N$ (computed w/ a fast-powering mod N algorithm).

Victor verifies the signature by checking whether or not

$$S^e \equiv D \bmod N.$$

(2) (a) State the Prime Number Theorem.

If $\pi(N) = \#$ primes $p \leq N$,

then

$$\lim_{N\to\infty} \frac{\pi(N)}{N/\ln(N)} = 1.$$

(b) Estimate the number of primes between 1,000,000 and 1,001,000 (your answer may include logarithms, and will be marked correct if it is within 20% of the true value).

$$\pi(1001000) \approx \frac{1001000}{\ln(1001000)} \approx \frac{1001000}{\ln(10^6)}$$

& $\pi(1000000) \approx \frac{1000000}{\ln(10^6)}$

hence $\pi(1001000) - \pi(1000000) \approx \frac{1000}{\ln(10^6)} = \frac{1000}{6\ln 10} = \boxed{\frac{500}{3\ln 10}}$

is a good estimate.

Indeed, there are actually 75 such primes, and $\frac{500}{3\ln 10} \approx 72.38$.

(c) Estimate how many of these prime numbers are congruent to 1 (mod 6).

All such primes are coprime to 6, hence they are all 1 mod 6 or 5 mod 6.

Generally, primes spread evenly among the invertible congruence classes.

So about 50% are 1 mod 6

$\Rightarrow \boxed{\frac{250}{3\ln 10}}$ is a good estimate.

Indeed, there are 38 such primes, and $\frac{250}{3\ln 10} \approx 36.19$.

(3) Suppose that Samantha is using ElGamal parameters (p, g), and her public key is $A \in \mathbf{Z}/p$. You may assume that g is a primitive root modulo p. Samantha has just generated a valid ElGamal signature (S_1, S_2) for a document D.

(a) What congruence must be verified to check that this is a valid signature?

$$A^{S_1} \cdot S_1^{S_2} \equiv g^D \bmod p.$$

(b) Suppose that Eve examines this signature and discovers that $S_1 \equiv g^3 \pmod{p}$. Describe how Eve can use this information to compute Alice's private key a (such that $g^a \equiv A \pmod{p}$). You may assume that $\gcd(S_1, p-1) = 1$.

$$A^{S_1} \cdot (g^3)^{S_2} \equiv g^D \bmod p$$

$$\Leftrightarrow \quad g^{aS_1 + 3S_2} \equiv g^D \bmod p \qquad (\text{since } A \equiv g^a)$$

$$\Leftrightarrow \quad aS_1 + 3S_2 \equiv D \bmod (p-1) \qquad (\text{since } \mathrm{ord}_p(g) = p-1)$$

$$\Leftrightarrow \quad \underline{a \equiv S_1^{-1}(D - 3S_2) \bmod (p-1)}.$$

Therefore Eve may compute $S_1^{-1} \bmod (p-1)$ (which exists since $\gcd(S_1, p-1) = 1$), then $S_1^{-1}(D-3S_2) \,\%\, (p-1)$, which will be Alices private signing key, a.

(4) The number $p = 397$ is prime, and $g = 5$ is a primitive root modulo p. The prime factorization of $p - 1$ is $396 = 2^2 \cdot 3^2 \cdot 11$.

Eve has computed the following three (mod p) discrete logarithms.

$$\left.\begin{aligned} \log_{5^{99}\%p}(311^{99}\%p) &= 3 \\ \log_{5^{44}\%p}(311^{44}\%p) &= 6 \\ \log_{5^{36}\%p}(311^{36}\%p) &= 2 \end{aligned}\right\}$$

these are the first steps of the <u>Pohlig-Hellman algorithm</u>.

Using these three values, determine the value of $\log_5(311)$.

Let $x = \log_5(311)$, ie. $5^x \equiv 311 \mod p$.

Then

$$5^{99x} \equiv 311^{99} \equiv 5^{99\cdot 3} \mod p$$

$\Rightarrow$ $99x \equiv 99\cdot 3 \mod (p-1)$ (since $\mathrm{ord}_p(5) = p-1$)

$\Rightarrow$ $x \equiv 3 \mod \left(\frac{p-1}{99}\right)$ ie. $\underline{x \equiv 3 \mod 4}$.

Similarly, $\underline{x \equiv 6 \mod 9}$ and $\underline{x \equiv 2 \mod 11}$.

We must merge these with the Chinese Remainder Theorem.

$x = 3 + 4k$

$\Rightarrow 3 + 4k \equiv 6 \mod 9 \Rightarrow 4k \equiv 3 \mod 9$

$\Rightarrow 7\cdot 4k \equiv 7\cdot 3 \mod 9 \Rightarrow k \equiv 3 \mod 9.$

$\Rightarrow x = 3 + 4(3 + 9h) = 3 + 12 + 36h = 15 + 36h.$

$\Rightarrow 15 + 36h \equiv 2 \mod 11 \Rightarrow 36h \equiv -13 \mod 11 \Rightarrow 3h \equiv 9 \mod 11$

$\Rightarrow h \equiv 3 \mod 11.$

$\Rightarrow x = 15 + 36(3 + 11\ell) = 15 + 108 + (p-1)\cdot\ell$

$= 123 + (p-1)\ell$

$\Rightarrow$ $\underline{x \equiv 123 \mod (p-1)}$

So $\boxed{\log_5(311) = 123}$.

(5) Suppose that G is a finite group. Assume that you have access the following:

- A function `Gmult(a,b)`, which takes $a, b \in G$ and returns their product in G.
- A function `Ginv(a)`, which takes an element $a \in G$ and returns its inverse in G.
- A constant `Gid`, which is the identity element of G.
- A constant `Gord`, which is the integer $|G|$.

(a) Write a function `Gpow(a,k)`, which receives an element $a \in G$ and an integer $k \in \mathbf{Z}$, and returns the group element g^k. For full credit, your function should call the function `Gmult` at most $\mathcal{O}(\log |k|)$ times.

```
def Gpow(a,k):
    if k<0:
        a = Ginv(a)
        k =-k
    res = Gid
    while k>0:
        if k%2==1:
            res = Gmult(res, a)
        a = Gmult(a,a)
        k //= 2
    return res
```

(b) Assume that you also have access to a function `mod_inv(c,M)`, which takes integers $c, M \in \mathbf{Z}$ such that $\gcd(c, M) = 1$ and returns the inverse of c modulo M. Write a function `Groot(a,k)`, which receives an element $a \in G$ and an integer $k \in \mathbf{Z}$ such that $\gcd(k, |G|) = 1$, and returns an element $x \in G$ such that $x^k = a$. You may assume that the function `Gpow` from part (a) has been implemented correctly, and use it in your solution. For full credit, your function should call `Gmult` at most $\mathcal{O}(\log |k|)$ times (including the times it is called by `Gpow`).

```
def Groot(a,k):
    kinv = mod_inv(k, Gord)
    return Gpow(a, kinv)
```

(6) Suppose that Samantha and Victor agree to use a digital signature system that differs slightly from DSA. In this system, the parameters (p, q, g), public key A, and private key a are as in DSA. However, the equations describing a signature of a document D are now the following.

$$S_1 = g^k \% p \% q$$
$$S_2 = a^{-1}(kD - S_1)\% q \qquad \text{(where } a^{-1} \text{ denotes the inverse modulo } q\text{)}$$

Describe a verification procedure for this signature scheme. Your answer should be similar to the verification procedure of DSA.

For a correctly produced signature:

$$aS_2 \equiv kD - S_1 \mod q$$

$$\Rightarrow \quad aS_2 + S_1 \equiv kD \mod q$$

$$\Rightarrow \quad A^{S_2} \cdot g^{S_1} \equiv (g^k)^D \mod p$$

$$\Rightarrow \quad A^{D^{-1}S_2} g^{D^{-1}S_1} \equiv g^k \mod p$$

$$\Rightarrow \quad \boxed{A^{D^{-1}S_2} \cdot g^{D^{-1}S_1} \% p \% q = S_1}$$

where D^{-1} denotes the inverse mod q.

This equation is the analog of the DSA verification equation; it identifies S_1 uniquely as the reduction mod q of a product of powers of A and g.

<u>Note</u>. I've implicitly assumed that $D \not\equiv 0 \mod q$. If $D \equiv 0 \mod q$, we could instead check whether $A^{S_1} \cdot g^{S_2} \equiv 1 \mod p$, since $(g^k)^D \equiv (g^k)^0 \equiv 1$.

(additional space for work)