Shweta Prasad May 12, 2021

CS 2362 Problem Set 5

Collaborators: *None*

Problem 5-1

(a)

Given: $p = 541, q = 1223$ and public exponent $e = 159853$. The public modulus can be calculated as $n = p * q = 541 * 1223 = 661643$. The private exponent can be calculated as $d \equiv e^{-1} \mod \phi(n)$ where $\phi(n) = (p-1) * (q-1)$. Finding the modular inverse using the EEA, we get $d = 561517$.

(b)

Given, message $D = 123456789$. The message can be signed as follows:

$$\begin{aligned} s &= D^d \mod n \\ &= 123456789^{561517} \mod 661643 \\ &= 436186 \end{aligned}$$

(c)

For this sub-part, we assume that we only have access to the public key. Assuming that we know that both the values that make up the public key tuple are relatively small, we do the following. (The associated script can be found in `commit_RSA_forgery.py`) To calculate d, we need to factor n into p and q. This will allow us to find the totient so that we can calculate $d \equiv e^{-1} \mod \phi(n)$.

We take a bruteforce approach to factorizing n. To reduce the search space, we find the square root of n and decrement to look through odd numbers that divide n. Once we find an appropriate factor, finding the other is just a matter of dividing n by the one just found.

Once d is found, we simply have to do what we do in part (b) of this question.

1 Problem 5-2

In the Elgamal digital signature scheme, the public parameters are (p, α, β) and Alice's private exponent d is used to compute β. If the message to be signed is m, k is the ephemeral round key and the signature is denoted by s, the Elgamal signature scheme computes $r = \alpha^k \mod p$, and $s = (m - dr)k^{-1} \mod p-1$, and sends across the signed document as $(m, (r, s))$.

(a)

If Alice uses the same ephemeral key to sign two documents D_1 and D_2, then the same r is computed both times, and Bob can easily detect that this has happened by looking at the tuple (r, s) in both signed documents. That is, if D_1 is signed with (S_1^1, S_2^1) and D_2 is signed with (S_1^2, S_2^2), we have $S_1^1 = S_1^2 = \alpha^k \mod p = r$.

(b)

Bob can use the above observation to uncover Alice's private exponent d in the following manner:

We know,

$$S_2^1 = (D_1 - dr)k^{-1} \mod p - 1$$
$$S_2^2 = (D_2 - dr)k^{-1} \mod p - 1$$

Multiply both equations by k and we get a system of linear equations :

$$k.S_2^1 = (D_1 - dr) \mod p - 1$$
$$k.S_2^2 = (D_2 - dr) \mod p - 1$$

Then, we subtract one from the other:

$$k.(S_2^1 - S_2^2) = (D_1 - D_2) \mod p - 1$$
$$k = \frac{D_1 - D_2}{S_2^1 - S_2^2} \mod p - 1$$

Bob can thus find solutions for k if $S_2^1 - S_2^2, p - 1$ are relatively prime. If there are multiple solutions, Bob simply has to try all of them and verify which is correct. Once the correct ephemeral key is found, Bob simply has to substitute it in the signing equation to retrieve d, as follows:

$$d = \frac{D_1 - S_2^1 k}{r} \mod p - 1$$

2 Problem 5-3

(a)

Namor gets y from Victor, and has to compute the square root modulo n for y, which can only be done efficiently if Namor has the prime factors for n. Assuming p, q that give n are distinct odd primes that satisfy the given condition, every a in QR_n has exactly four square roots in $\mathbf{Z}_n^*$ where 2 roots come from each of the 2 primes. That is, since $a \epsilon QR_n$, $a \epsilon QR_p$ and $a \epsilon QR_q$; there are numbers $\sqrt{a} \mod p = \{\pm s_p\}$ and $\sqrt{a} \mod q = \{\pm s_q\}$ which can be combined (as one from each set) using the Chinese Remainder Theorem to give a distinct element in $\sqrt{a}$ modulo n. The given assumption guarantees that any y that Victor sends will have quadratic residues. Once Namor obtains the square roots, they send $s \epsilon \{b_{x,y} | x \epsilon \{\pm s_p\} and y \epsilon \{\pm s_q\}\}$ to Victor and Victor verifies it.

(b)

The above back and forth is repeated 20 times. If Namor did not know the prime factors, they would not be able to compute the square root efficiently.

3 Problem 5-6

Class NP is the set of all decidable languages that run in polynomial time when implemented using a non-deterministic, single tape Turing Machine. Additionally, this class is also defined as the set of all languages that can be verified in polynomial time by some Turing Verifier. A Turing Verifier is a Turing Decider which takes as input two components (x, y), and the runtime of V is measured in the size of the first part - x - of the input. y is usually the"witness", upon which the verification is run. A Poly-time verifier V takes at most $O(|x|^k)$ steps before coming to a halt and giving the status of verification. These verifiers are good grounds for non-interactive, single step zero knowledge proofs.

This question asks us to show that if a language L is undecidable, no non-interactive proof exists for it in polynomial time. Assuming that only languages in the class NP are fit to give non-interactive proofs, it is trivial to show that this is the case: the computational class NP is a subset of Turing Decidable languages, and by definition, all languages in the class have deciders that can decide it. If a language is outside the set of decidable languages, it cannot belong to NP and thus cannot have non-interactive proofs.