
Problem Set 2

This problem set is due *via email*, to `6857-hw@mit.edu` on *Monday, March 3* by the beginning of class.

You are to work on this problem set in groups of three or four people. Problems turned in by individuals, pairs, pentuples, etc. will not be accepted. Be sure that all group members can explain the solutions. See Handout 1 (*Course Information*) for our policy on collaboration. If you do not have a group, let us know.

Homework must be submitted electronically! Each problem answer must appear on a separate page. Mark the top of each page with your group member names, the course number (6.857), the problem set number and question, and the date. We have provided templates for L^AT_EX and Microsoft Word on the course website (see the *Resources* page).

Grading and Late Policy: Each problem is worth 10 points. Late homework will not be accepted without prior approval.

With the authors' permission, we will distribute our favorite solution to each problem as the "official" solution—this is your chance to become famous! If you do not wish for your homework to be used as an official solution, or if you wish that it only be used anonymously, please note this on your homework.

Problem 2-1. Finding primes

Let x be the product of the ten-digit phone numbers of the members of your group.

- (a) Find the least prime number p that is greater than your group's x .
- (b) A *safe prime* is a prime p such that $p - 1 = 2q$ for some prime q . Find the least safe prime p that is greater than your x . Find a generator of the subgroup QR_p of squares mod p .

Problem 2-2. RSA pitfalls

This problem has four subproblems, in three independent parts.

Alice challenges Bob to a friendly wager. The wager requires that she and Bob be able to flip a fair coin cryptographically. Bob agrees to the wager, and proposes the following coin-flipping protocol.

First, Alice selects an RSA public key (n, e) with corresponding secret key (p, q) . Alice then selects an $r \in Z_n^*$. She computes the RSA encryption $a = r^e \pmod n$, and sends a , along with the public key, (n, e) , to Bob. Bob selects $s \in Z_n^*$, and computes the RSA encryption $b = s^e \pmod n$. He sends b to Alice, who ensures that $a \neq b$ (and hence $r \neq s$). If so, Alice sends the secret key (p, q) to Bob. (In the event that $a = b$, they abort the protocol and start over from the beginning.)

The result of the coin flip is obtained by decrypting r and s using the secret key, taking the least significant bit of each, and computing the XOR of those two bits.

- (a) Alice refuses to use Bob's protocol, on the grounds that Bob could cheat by biasing the result of the coin flip. Show how.

Ben Bitdiddle wants to increase the security of RSA without paying a cost in efficiency. He theorizes that it may be possible to do so by increasing the size of *one* of the prime factors, rather than both of them.

Ben proposes a variant on RSA that he calls *unbalanced RSA*. Choose the RSA modulus n to be 5,000 bits long, the product of a 500-bit prime p and a 4,500-bit prime q . Since RSA is usually used just to exchange short symmetric keys we will assume that all messages being encrypted are smaller than p . Once the public exponent e is chosen, compute $d = e^{-1} \pmod{\phi(n)}$ and keep it secret, as usual.

Typically, increasing the size of the modulus n increases the amount of time required to decrypt a ciphertext $c = m^e \pmod n$ (since one has to compute $c^d \pmod n$). But since we know that $m < p$ we can use the Chinese

Remainder Theorem and compute $m = c^d \bmod p$. Ben claims that this allows us to increase the size of the modulus, thus achieving better security against the advances of factoring, while not losing efficiency. This, naturally, raises red flags in your head.

- (b) Ben figures that smaller public exponents mean less work during encryption, so he chooses $e = 3$. Argue that $e = 3$ is a poor choice of exponent.
- (c) Show how, with a *single* chosen ciphertext attack (i.e., obtaining the decryption of a ciphertext of your choice), you can learn the factorization of n , thus breaking Ben's scheme.

Ben, undaunted by the failure of unbalanced RSA, continues searching for shortcuts. He wants to use RSA to protect the privacy of both his business emails and personal emails. That is, he wants two RSA public keys: one which will be used to encrypt emails sent to his work email address, and one that will be used to encrypt emails sent to his personal email address. However, Ben feels it is too much trouble to generate and remember the factorization of two different RSA moduli. Therefore, he decides to use the same n for both public keys. Ben chooses $e_w = 7$ as his encrypting exponent for work emails and $e_p = 11$ as his encrypting exponent for personal emails.

- (d) Alice needs to send an urgent message to Ben. Normally, she would send it to his work e-mail, but unfortunately, all Blackberry service is down for the day. She decides to send the same message to Ben's work and personal e-mail addresses, in the hopes that he'll get it quickly, regardless of which inbox he happens to check first. Show how an eavesdropper, Eve, can use the two encrypted messages to read Alice's urgent message.

Problem 2-3. MAC variants

CBC-MAC is not a secure authentication code when an adversary can obtain authentication tags of different length messages. One way we might try to craft a secure scheme for messages of variable length is to append the message length m to the message M , and compute the CBC-MAC of the resulting string, $M||m$.

- (a) This proposed CBC-MAC scheme is not secure. Give an attack against it.

Now recall the XOR-MAC scheme from lecture. It can be instantiated for any block cipher F ; assume that F is a secure block cipher with a 64-bit block size.

Given a message M , the sender breaks M into blocks M_1, \dots, M_ℓ , each of length 32 bits (for simplicity we assume that the length of M is always a multiple of 32). Let $\langle i \rangle$ denote the binary representation of i as a string of exactly 31 bits, and let $\alpha.\beta$ denote the concatenation of two binary strings α and β . To authenticate M using a secret key K , the sender does the following:

- Pick a random 63-bit binary string r .
- Set $t = F_K(0.r) \oplus F_K(1.\langle 1 \rangle.M_1) \oplus \dots \oplus F_K(1.\langle \ell \rangle.M_\ell)$.
- Set the MAC of M to be the pair (r, t) .

The verification algorithm is similar to the above (we leave it to you to reconstruct).

- (b) Observe the initial 0 and 1 bits prepended to each block, e.g. in $F_K(0.r)$ and $F_K(1.\langle 1 \rangle.M_1)$. Suppose we were to remove them, and instead use a 64-bit r . Suggest an attack on this modified scheme.

Problem 2-4. Digital Signature Standard

Consider the Digital Signature Standard, or DSS (specified by NIST in FIPS 186):

- **Key generation:** the signer chooses a random large prime p , a random large prime q that divides $p - 1$, random $g \in \mathbb{Z}_p^*$ whose order modulo p is q , and random $x \in \mathbb{Z}_q^*$, and computes $y = g^x \bmod p$. The public key is (p, q, g, y) . The secret key is x .
- **Signing:** to sign a value $m \in \{0, \dots, p - 2\}$, the signer chooses a random $k \in \mathbb{Z}_q^*$; that is, a $k \in \{1, \dots, q - 1\}$ such that $\gcd(k, q) = 1$. The signer computes $r = (g^k \bmod p) \bmod q$, and

$$s = (H(M) + xr)k^{-1} \bmod q.$$

The signer repeats these steps until $r \neq 0$ and $s \neq 0$. The signature for m is the pair (r, s) .

- **Verification:** For message M with signature (r, s) , the verifier checks that $0 < r < q$ and $0 < s < q$, and also checks that $g^{H(M)} = y^{-r} r^s \bmod q$. If all conditions are true, the verifier accepts, otherwise he rejects.

Here, H is any hash function. The DSS recommends the use of SHA-1, but soon will be updated to recommend hash functions with larger output, e.g. SHA-256.

This scheme is believed to be secure, but there are many precautions that must be taken in its implementation.

- Show that the signer *must* keep k secret, in the sense that if k is known, an attacker can learn the signer's secret key.
- Show that if the same value of k is known to have been used to sign two different messages, the secret key can also be found.
- In contexts where it is difficult or impossible to access true random numbers, people sometime rely on so-called *pseudo-random number generators*. These are *deterministic* algorithms that start with a *random seed*. Their output is a sequence of numbers computed deterministically from the seed.

One simple and popular generator is called the *linear congruential generator*. Used in this context, there are two public values $a, b \in \mathbb{Z}_q^*$. The initial seed is a random, secret $k_0 \bmod q$. When an i th value is to be signed, the value $k_i = a \cdot k_{i-1} + b \bmod q$ is computed and used for k .

Is it safe for the signer to use such a generator for DSS signatures?