All exercise numbers from the textbook refer to the **second edition**.

1. Suppose that p, q, g are DSA public parameters (i.e. p, q are primes, and g has order q modulo p), and $A \equiv g^a \pmod{p}$ is Samantha's public (verification) key, while a is her private (signing) key. As we discussed in class, there are two main sorts of algorithms that Eve might use to extract a from A: collision algorithms (whose runtime depends on q), and the number field sieve (whose runtime depends on p). For simplicity, assume that Eve has a collision algorithm that can extract a in $\sqrt{q}$ steps, and an implementation of the number field sieve that can extract a in $e^{2(\ln p)^{1/3}(\ln \ln p)^{2/3}}$ steps (the true runtimes would involve a constant factor that would depend on implementation, and various other factors depending on the cost of arithmetic modulo p and of finding collisions).

   (a) Suppose that Samantha is confident that her private key will be safe as long as Eve does not have time to perform more than 2^{64} steps in either algorithm. How many bits long should she choose p to be? How many bits long should q be?

   (b) What if she instead wants to be safe as long as Eve doesn't have time for 2^{128} steps?

   (c) The NSA's recommendation for "Top Secret" government communications is to use 3072 bit values of p, and 384 bit values of q. How does this compare to your answers above? If the difference is significant, what might explain the discrepancy?

   For parts (a) and (b), it is sufficient to write a short script to find the minimum safe numbers of bits by trial and error (there are more efficient ways, of course).

2. Exercise 6.15.

3. Exercise 6.16.

4. Exercise 6.17.

5. Suppose that Eve intercepts a Menezes-Vanstone (see the previous problem and Table 6.13 in the text) ciphertext (R, c_1, c_2) send from Bob to Alice. Suppose that Eve knows, by other means, a list of 100 possible messages that Bob might send to Alice, and that the plaintext must be one of these (e.g. the message might be telling Alice which one of 100 locations they will meet in). Describe a method that Eve can use to determine which of these 100 candidates is the true plaintext, with relatively little computation (your method does not need to be completely fail-safe; it is ok if there are some extremely unlikely situations in which it will fail).

6. Samantha and Victor agree to the following digital signature scheme. The public parameters and key creation are identical to those of ECDSA. The verification procedure is different: to decide whether (s_1, s_2) is a valid signature for a document d, Victor computes

   $$\begin{aligned} w_1 &\equiv s_1^{-1} d \pmod{q} \\ w_2 &\equiv s_1^{-1} s_2 \pmod{q}, \end{aligned}$$

   then he check to see whether or not

   $$x(w_1 G \oplus w_2 V)\%q = s_1.$$

   If so, he regards (s_1, s_2) as a valid signature for d.

(a) Describe a signing procedure that Samantha can follow to produce a valid signature on a given document d. The procedure should be randomized in such a way that it will generate different signatures if executed repeatedly on the same document.

(b) Describe a "blind forgery" procedure that Eve can follow to create a signature (s_1, s_2) *and a document* d such that (s_1, s_2) is a valid signature for d under this scheme. Note that Eve does not need to be able to choose d in advance. The procedure should be randomized in such a way that it can generate many different forgeries (on many different documents).

Programming problems

Full formulation and submission: `https://www.hackerrank.com/m158-2016-pset-9`

Note. Both of these problems will make use of public parameters specified in this document.

`http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`

You certainly do not need to read and understand the whole document, just find the information you need for the algorithms. You'll need to look up how to convert hexadecimal strings to integers.

7. Write a program that determines whether a given ECDSA signature (s_1, s_2) for a document d is valid or not, given paramters and an ECDSA public key (notation as on page 322 of the textbook). The signatures will all use curve P-384 from the document above.

8. Write a program to decipher messages enciphered with the Menezes-Vanstone cryptosystem described in problems 6 and 7 (textbook exercises 6.17 and 6.18), given a private key and a ciphertext. The public parameters will be those of curve P-192 from the above document.