Homework 9

Due **Friday**, November 12

(note that November 11 is a university holiday)

Implementation Part

1. Implement the Goldwasser-Micali probabilistic encryption scheme to securely send 1 bit of data.

   (a) Create a function `GenerateGMKey(b)` which creates a Goldwasser-Micali key from b-bit primes. In particular, it should output a *public key* $[N, a]$ where $N = pq$ is a product of (secret) b bit primes and $a \in \mathbb{Z}/N\mathbb{Z}$ is a quadratic *non*residue modulo p and q, as well as a *private key* which should just consist of one of the secret primes. (You can use part 1 to compute Legendre symbols.)

   (b) Write functions `GMEncrypt(publicKey,m)` and `GMDecrypt(privateKey,c)`. The first takes a Goldwasser-Micali public key, and a bit $m = \{0, 1\}$, and returns a ciphertext $c \in \mathbb{Z}/N\mathbb{Z}$ which is a square modulo N if and only if $m = 0$. The second recovers m from c and the private key and Legendre symbols. (*Note*: When you compute the random integer in the encryption function make sure it is larger than the square root of N. For extremely large N, you should make use of the sage integer square root function `isqrt()` since floats may round up to infinity and have trouble reconverting to ints).

   (c) Generate and print a Goldwasser-Micali key from 16 bit primes. Use it to encrypt and then decrypt both both 0 and 1. Confirm that you recover the bit correctly.

   (d) Implement a Goldwasser-Micali key from primes $p = 151$ and $q = 233$. Recover the bit from the cipher $c = 33482$.

2. Implement the RSA Digital Signature algorithm. It will look very similar to the first project, and you are welcome to reuse code from that assignment (especially for key generation).

   (a) Write a function `generateRSAKey(b)` which generates an RSA private signing key and public verification key from primes `b` bits long. The verification key will be a pair `[N,e]` where $N = pq$ is a product of secret primes b bits long and e is an integer prime to $(p-1)(q-1)$, and the signing key will be a pair `[N,d]` for the same N and d the inverse of e modulo $(p-1)(q-1)$ (sound familiar?).

   (b) Write functions

   `RSASign(signingKey,document)` and
   `RSAVerify(verificationKey,document,signedDocument)`.

   The former will sign a document with the signing key (by taking an appropriate root), and the second will verify that a document is correctly signed (by exponentiating).

   (c) Generate an RSA digital signature key from 16 bit primes. Use it to sign the document $D = 314159$. Run your verification algorithm for twice, once with the signed document and once with the unsigned document, and confirm you get the expected results.

   (d) I generated a RSA Digital Signature key, below is my public verification key.

$N =$
10522131111414083920142713395050769617442433657695255146537311896434311144361
52105697372008556663566615508174430418471897265040190399740387723795071615822
78783519296999687194872224184723574952162166440945091505292215584920700394034
42226365661647537411508608651137113856704007797414387581567049968246853446643
9.
$e =$
21016836287029986747723759800774101835281275854873180166245543058525887395193
96307766860254990165396054303259228348297307889093719835313656626522445205557
65407005232827058670837160067093166648952976725071881739083691398929472902959
38964443139784969754551470562127704086246141892044152375206351855610254115515.
You recieve two documents:
$D =$ 445915856905197344451931056052999335315688923420907486019700081 37
$D' =$
33737713092926002711997965938686781423985198340985423980678740668540837920165
714308759593604261668849045652971875876231 3
Run `intToText` to read this documents (note: they are not encrypted). Each claims to be signed by me, and comes with a digital signature:
$D^{sig} =$
84266656881633759645931434414589646373474140299696318568846558871354683992664
91115775348447213721512083388365998130055074496366371336684453834515335491064
69940898673531353559586336758802075651425026668697230132550765551653945548321
68424776641559903698464211162982834359695243021419912674971036001530626269241
$D'^{sig} =$
90592809313509991477767898543561252818730285233946708276918316631997361944737
64728663367105923665686312882726901254765910591149065525748838442609274074694
58299675381621160449150556922441295777712973489019131240798265816551541474157
14324848399155672059930064485203220452985824881282662030076919344957027591627
Which message is truthful?

3. Implement Elgamal digital signatures.

   (a) Write a function `generateElgamalKey(p,g)` which takes as input a prime p and a primitive root $g \in \mathbb{F}_p^*$, chooses a secret exponent a (at random) and returns a private signing key $[a, p, g]$ and a public verification key $[A, p, g]$.

   (b) Write functions

   `elgamalSign(signingKey,document)` and
   `elgamalVerify(verificationKey,document,signedDocument)`.

   The former will sign a document with the given signing key, returning the signed document as a pair $[S_1, S_2]$ where $0 \leq S_1 < p$ and $0 \leq S_2 < p-1$. The second will verify that the document was correctly signed. Both should follow the elgamal digital signature protocol defined in the October 22 lecture and described in Table 4.2 of [HPS].

   (c) If you sign 2 different documents with the same random element k, your Elgamal signing key becomes insecure. In problem 6 (below) you will describe an algorithm to steal that signing key. Implement that algorithm here. In particular, implement an algorithm

   `stealElgamalSignature(verificationKey,D,Dsig,D',D'sig)`.

which takes as input verification key associated to an Elgamal signature (which is public information), as well as 2 distinct documents signed with that key. It will first check if those 2 documents were signed with the same random value k, and if they were it will return the signers secret exponent a. (This attack was used to steal Sony's digital signature in 2013!).

(d) Let $p = 3700273081$, and $g = 7$. Create an Elgamal key, and use it to sign the document $D = 314159$. Then verify that the signature was valid.

(e) Suppose Samantha has a public verification key $[A, p, g] = [185149, 348149, 113459]$. Suppose Samantha signed the following 2 documents:

$$\begin{aligned} D &= 153405 \qquad D^{sig} = (S_1, S_2) = (208913, 209176) \\ D' &= 127561 \qquad D'^{sig} = (S'_1, S'_2) = (208913, 217800). \end{aligned}$$

Use `stealElgamalSignature` to steal Samantha's signing exponent.

4. Implement DSA.

(a) Write a function `generateDSA(p,q,g)` which takes as input primes p, q with $p \equiv 1 \mod q$, and an element $g \in \mathbb{F}_p^*$ of order q. It then chooses chooses a secret exponent a (at random) and returns a private signing key $[a, p, q, g]$ and a public verification key $[A, p, q, g]$.

(b) Write functions

`DSASign(signingKey,document)` and
`DSAVerify(verificationKey,document,signedDocument)`.

The former will sign a document with the given signing key, returning the signed document as a pair $[S_1, S_2]$ where $0 \leq S_i < q$ for $i = 1, 2$. The second will verify that the document was correctly signed. Both should follow the DSA protocol defined in the November 4 lecture and described in Table 4.3 of [HPS].

(c) Let $p = 48731$ and $q = 443$. Assume that 7 is a primitive root for $\mathbb{F}_p^*$. Use this to find an element of order 443 in $\mathbb{F}_p^*$. Call this element g.

(d) Generate a DSA key for p, q, g in part (c). Use it to sign the document $D = 314$ and verify that this signature is valid.

Written Part

In the written part we explore some attacks on the Elgamal Digital Signature algorithm.

5. First let's describe a way Eve can produce documents that appear to be signed by Sam. Let p be a prime number and $g \in \mathbb{F}_p^*$ a primitive root. Let i and j be integers such that $\gcd(j, p-1) = 1$. Let A be arbitrary. Set:

$$\begin{aligned} S_1 &\equiv g^i A^j \mod p \\ S_2 &\equiv -S_1 j^{-1} \mod p-1 \\ D &\equiv -S_1 i j^{-1} \mod p-1 \end{aligned}$$

(a) Show that the pair (S_1, S_2) is a valid Elgamal signature for the document D. In particular, this means Eve can produce valid Elgamal signatures.

(b) Explain why this doesn't mean that Eve can forge Sam's signature on a given document.

6. In this exercise we describe a security flaw in the Elgamal digital signature algorithm, caused by a careless signer. Suppose that Sam signed two distinct documents D and D' using the same random value k.

   (a) Explain how Eve can immediately recognize that Samantha has made this blunder.

   (b) Let the signature for D be $D^{sig} = (S_1, S_2)$ and the signature for D' be $D'^{sig} = (S_1', S_2')$. Explain how Eve can recover Samantha's secret signing key a.