Exercise 1: Proving knowledge of an RSA pre-image using a Σ protocol

For an RSA modulus $N = pq$, where $p, q$ are prime numbers, $\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N \mid \gcd(a, N) = 1\}$ forms a multiplicative group of order $|\mathbb{Z}_N^*| = \varphi(N) = (p - 1)(q - 1)$. Hence, for any prime integer $e$ such that $\gcd(e, \varphi(N)) = 1$, the function $f(x) : \mathbb{Z}_N^* \to \mathbb{Z}_N^* : x \to f(x) = x^e \mod N$ is a permutation over $\mathbb{Z}_N^*$. Each $y \in \mathbb{Z}_N^*$ thus has a unique pre-image $x = f^{-1}(y) \in \mathbb{Z}_N^*$. We consider the problem of proving the knowledge of an $e$-th root $y^{1/e} \mod N$.

1. For a public key $y = x^e \mod N$, describe a Schnorr-like Σ-protocol for proving the knowledge of $x \in \mathbb{Z}_N^*$.

   **Hint:** The protocol should prove knowledge of $y$’s pre-image under some homomorphism in the same way as Schnorr’s protocol does. Consider a Σ protocol where challenges are chosen in $\{0, \ldots, e - 1\}$.

2. Prove that this protocol has the special soundness property.

   **Hint:** Use Bézout’s Lemma which says that, for any integers $a, b \in \mathbb{Z}$, there exist $u, v \in \mathbb{Z}$ such that $u \cdot a + v \cdot b = \gcd(a, b)$.

3. Prove that, for $e = 3$, the protocol is perfectly zero-knowledge (even against a possibly malicious verifier) by describing a simulator. Is it also perfectly ZK for an exponentially large prime $e$?

4. Now, suppose that $e$ divides $\varphi(N) = (p - 1)(q - 1)$. What does the Σ protocol prove about $y \in \mathbb{Z}_N^*$?
Exercise 2: Proving discrete logarithm inequalities

In a cyclic group \( \mathbb{G} \) of prime order \( q \), we have seen how to prove the equality of two discrete logarithms \( x \in \mathbb{Z}_q \) given public group elements \( g, h \in \mathbb{G}, X = g^x \) and \( Y = h^x \). We now consider the problem of proving the inequality of discrete logarithms. Namely, given \( g, h \in \mathbb{G} \) as well as \( X = g^x \) and \( Y = h^y \), the prover \( P \) wants to convince a verifier \( V \) that \( x \neq y \) without revealing any further information. (Note that \( \log_g(h) \) is not available to anyone).

1. Suppose that the prover \( P \) (who knows witnesses \( x, y \in \mathbb{Z}_q \)) reveals \( Z = h^x \) and uses a \( \Sigma \) protocol (such as the Chaum-Pedersen protocol) to generate an interactive proof that \( \log_g(X) = \log_h(Z) \). The verifier can get convinced by checking that \( Z \neq Y \). Why is this protocol not honest-verifier zero-knowledge?

2. We now modify the above protocol to make it honest-verifier zero-knowledge

   a. Let us assume that, instead of revealing \( Z = h^x \), the prover \( P \) reveals \( Z = (h^x/Y)^r \) for some randomly chosen \( r \in R \mathbb{Z}_q \). Which condition should the verifier \( V \) check about \( Z \)?

   b. Given that \( 1_G = (g^r/X)^r \), describe a \( \Sigma \) protocol that can be used by \( P \) and \( V \). Prove its honest-verifier zero-knowledge and special soundness properties.

Exercise 3: Range proofs

Let \( (G_1, G_2, G_T) \) be cyclic groups of prime order \( p \) with a bilinear map \( e : G_1 \times G_2 \to G_T \). Let also

\[
PK := (u = h^{x_1}, v = h^{x_2}, h) \in G_1^3
\]

be a public key for the public-key encryption scheme based on the Decision Linear assumption, for which the underlying secret key is \( SK = (x_1, x_2) \in \mathbb{Z}_p^2 \). We consider the problem of proving that

\[
C = (C_1, C_2, C_3) = (u^a, v^b, g_1^a \cdot h^{a+b}) \in G_1^3,
\]

where \( g_1 \in G_1 \), is an encryption of a small message \( m \in \{0, \ldots, 2^L - 1\} \), where \( L \ll \log p \).

1. Describe a first \( \Sigma \) protocol for proving that a ciphertext \( C = (C_1, C_2, C_3) = (u^a, v^b, g_1^a \cdot h^{a+b}) \) encrypts a bit \( b \in \{0, 1\} \).

2. Extend the first protocol into a \( \Sigma \) protocol for proving that \( 0 \leq m \leq 2^L - 1 \). The communication complexity of the resulting protocol should not exceed \( O((L \cdot \log p) \text{ bits}) \).

   Suggestion: Prove that the ciphertext \( C \) decrypts to the same message as a ciphertext obtained by homomorphically combining encryptions of bits.

3. Suppose that \( P \) and \( V \) both have access to public parameters \( pp \) containing \( w = g_1^\gamma \in G_1 \) and elements \( \sigma_i = g_2^{1/(\gamma + i)} \in G_2 \) for each \( i \in \{0, \ldots, 2^L - 1\} \) (note that \( pp \) is generated by a trusted entity that does not reveal \( \gamma \in \mathbb{Z}_p \) to \( P \) or \( V \)). Describe a \( \Sigma \) protocol for proving that \( m \in \{0, \ldots, 2^L - 1\} \) while communicating only \( O(\log p) \text{ bits} \).

   Suggestion: Consider a prover \( P \) that first reveals \( C_0 = (g_1^m \cdot w)^r \) (which can be seen as a Pedersen commitment to \( m \)), for some random \( r \in R \mathbb{Z}_p \), and proves that \( C_0 \) is consistent with \( (1) \).

Does the resulting protocol provide soundness against a computationally unbounded cheating prover? If not, which computational assumption does the soundness property rely on?
Exercise 4: Program Obfuscation

The aim of program obfuscation can be roughly stated as follows: find a way that we can give people programs they can run – without letting them figure out how the programs actually work. The primary application of program obfuscation is to protect an algorithm in such a way that it cannot be inferred from the software distributed.

Consider for instance the following program:

```
#define _ -F<00||--F-OO--;
int F=00,OO=00;main(){F_OO();printf("%1.3f\n",4.*-F/OO/OO);}F_OO()
{
  _-_-_-_
  _-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_-_-_-_
  _-_-_-_-_-_-_-_
  _-_-_-_
}
```

This program computes ... an approximation of $\pi$!

Another program, solving the $n$-queens problem, is the following:

```
int v,i,j,k,l,s,a[99];main(){for(scanf("%d",&s);*a-s;v=a[j*=v]-a[i],k=i<s,j+=(v=j<s&&(!k&!!printf(2+"\n\n%c",(l^v?2:-(!l<<!j))&&++
  l|a[i]<a&v&v-i+j&v+i-j))&!(l%=s),v||(i==j?a[i+=k]=0:++a[i])>=s+k&
  ++a[--i]));printf("\n\n");}
```

Let us call a program obfuscator, any program taking a program $P$ as input as producing $P'$ which is a “obfuscated” version of $P$.

Questions:

1. Precise the definition of a program obfuscator. What properties shall we require for a program obfuscator? Discuss the advantages/disadvantages of your definition.

2. What program obfuscation techniques do you know?

3. Suppose that you have an efficient program obfuscator (whatever the exact definition of “efficient” may be): what applications, in terms of confidentiality, can you think of for this obfuscator? (except of course copyright enforcement)

Suppose you have two programs $A, B$. Suppose that $A$ is:

```
SuperSecretPasswordProtectedStuff(string passwd)
{
```
if (password == "0xt438fh27266629zn28366492923aai3jnqobbyc4t") {
    print("Bravo voil passwd2 : 0xh798df003574jklksjklmgmv2431n");
} else {
    print("Wrong password...
");
}

and B is similar to program A, it contains the two password of A (passwd, passwd2) written somewhere. The difference is that B takes as input another program, say P. Then B(P) behaves as follows:

1. It executes P(passwd), which produces r.
2. If $r == passwd2$, it outputs a secret bit.

Questions:
1. What can we say about $B(A)$?
2. If A and B are obfuscated what happens?

Exercise 5: Information Declassification and PER

A way to describe information flows in programs can be done via the use of relations (instead of high-low bisimulations as seen in course). The general idea is as follows: Suppose that the values of a particular secret range over int, and that the value of the secret is not fixed - it is a parameter of the system. Without fixing a particular value for the secret, one way to describe how much an attacker knows (or can learn) about the secret is in terms of an equivalence relation. In this approach an attackers knowledge about the secret is modelled in terms of the attackers ability to distinguish elements of int. If the attacker knows nothing about the secret then this corresponds to saying that from the attackers viewpoint, any value in int looks the same as any other value. This is captured by the equivalence relation $\forall m,n \in \text{int}. m = n$.

Suppose that we model by $s : f \rightarrow f$, how the system maps the value of the secret to the observable output, then we write

$$s : \text{All} \Rightarrow \text{Id}$$

meaning that $\forall m,n \in \text{int}. m = n \Rightarrow s(m) = s(n)$, where Id is the identity relation. In other words it is the standard noninterference (zero information flow) property.

Questions:
1. Why do we use partial equivalence relations instead of equivalence relations in this model?
2. How do you model the fact that only the parity of a secret is declassified (and nothing more)?
3. In this model, is it possible to represent the downgrading of information from high to low without losing information for example representing the secure encryption of high level information?

Exercise 6:

The aim of this exercise is to study a protocol allowing Alice to sell secrets in such a way that, in one hand, she cannot know which secret she has sold, on the other hand, that he buyer cannot learn more than one secret. For this we give a protocol for two buyers (Bob and Carole) looking for two secrets. We add the constraint that neither Bob nor Carole can learn the secret chosen by the other buyer.

The fixed bit index FBI of x and y are the bits where the ith bit of x equals the ith bit of y.
\[
\begin{align*}
x &= 110101001011 \\
y &= 101010000110 \\
FBI(x, y) &= \{1, 4, 5, 11\}
\end{align*}
\]

We are reading the bits from right to left, with the right-most bit as zero.

In the following protocol, Alice is the seller, Bob and Carole are the buyers. Alice owns \(k\) secrets of \(n\)-bits \(S_1, S_2, \ldots, S_k\). Bob is buying \(S_b\) and Carole the secret \(S_c\).

1. Alice generates a public-key/private-key key pair and tells Bob (but not Carol) the public key. She generates another public-key/private-key key pair and tells Carol (but not Bob) the public key.

2. Bob generates \(k n\)-bit random numbers, \(B_1, B_2, \ldots, B_k\), and tells them to Carol. Carol generates \(k n\)-bit random numbers, \(C_1, C_2, \ldots, C_k\), and tells them to Bob.

3. Bob encrypts \(C_b\) (remember, \(S_b\) is the secret he wants to buy) with the public key from Alice. He computes the FBI of \(C_b\) and the result he just encrypted. He sends this FBI to Carol. Carol encrypts \(B_c\) (remember, \(S_c\) is the secret she wants to buy) with the public key from Alice. She computes the FBI of \(B_c\) and the result she just encrypted. She sends this FBI to Bob.

4. Bob takes each of the \(n\)-bit numbers \(B_1, B_2, \ldots, B_k\), and replaces every bit whose index is not in the FBI he received from Carol with its complement. He sends this new list of \(n\)-bit numbers, \(B'_1, B'_2, \ldots, B'_k\), to Alice. Carol takes each of the \(n\)-bit numbers \(C_1, C_2, \ldots, C_k\), and replaces every bit whose index is not in the FBI she received from Bob with its complement. She sends this new list of \(n\)-bit numbers, \(C'_1, C'_2, \ldots, C'_k\), to Alice.

5. Alice decrypts all \(C'_i\) with Bob’s private key, giving her \(k n\)-bit numbers: \(C''_1, C''_2, \ldots, C''_k\). She computes \(S_i \oplus C''_i\), for \(i = 1\) to \(k\), and sends the results to Bob. Alice decrypts all \(B'_i\) with Carol’s private key, giving her \(k n\)-bit numbers: \(B''_1, B''_2, \ldots, B''_k\). She computes \(S_i \oplus B''_i\), for \(i = 1\) to \(k\), and sends the results to Carol.

6. Bob computes \(S_b\) by XORing \(C_b\) and the \(b\)th number he received from Alice. Carol computes \(S_c\) by XORing \(B_c\) and the \(c\)th number she received from Alice.

Questions:

1. Give examples for the use of such a protocol. Why can it be useful to be able to distribute secrets anonymously?

2. Explain why the protocol is correct (ie meets all the requirements written in the introduction).

3. If Alice and Carole are working together how can they break the protocol and learn the secret buyed by Bob?