1. (10 marks)
In this question we will present a reasonable key exchange protocol that turns out to be insecure.
We will assume the Setting for Discrete Log Conjectures as on Page 3 of Notes #8, and the DDHH assumption from Page 5.

GEN works as follows on security parameter $n$, which we will assume is even. Informally, GEN generates a Diffie-Hellman pair of keys, as well as a pair of keys for a secure signature scheme. More formally, on security parameter $n$:

Let $p = p_n$ and $g = g_n$. GEN chooses a random member of $\{1, 2, \ldots, p - 1\}$ represented as an $n$-bit string $\text{pri}1$ and computes the $n$-bit string $\text{pub}1 = g^{\text{pri}1} \mod p$. GEN also chooses a pair of keys for a secure signature scheme; let us say that these $n$-bit keys are $\text{pri}2$ and $\text{pub}2$, and signatures are of length $n$.

The private key then becomes $[\text{pri}1, \text{pri}2]$ and the public key becomes $[\text{pub}1, \text{pub}2]$.

The protocol, informally, is as follows:

(A, B, 0) encrypts a random $n/2$-bit string $r$ to (B, A, 1), together with an appropriate signature;

(B, A, 1) then encrypts a random $n/2$-bit string $s$ to (A, B, 0), together with an appropriate signature; the parties output $r \oplus s$ as the $n/2$-bit session key. More rigorously, we present the role-0 and the role-1 algorithm separately.

Process (A, B, 0) works as follows. We denote $A$’s private key as $[\text{pri}1A, \text{pri}2A]$ and $B$’s public key as $[\text{pub}1B, \text{pub}2B]$.

- Choose a random $x \in \{0, 1, 2, \ldots, p - 2\}$ and compute the $n$-bit string
  $\alpha \leftarrow g^x \mod p$, the $n/2$-bit string
  $r \leftarrow h(\text{pub}1B^x \mod p)$, and the $n$-bit string
  $\beta \leftarrow \text{SIGN}_{\text{pri}2A}(\alpha, A, B)$. (Recall that $A$ and $B$ are $n$-bit strings.)
  Send string $[\alpha, \beta]$ on the output channel.

- (To understand this part, see the second part of the role-1 protocol below.)
  Receive $n$-bit strings $\gamma$ and $\delta$ on the input channel.
  Use $\text{pub}2B$ to verify that $\delta$ is a signature by $B$ of $[\alpha, \gamma, A, B]$, and if not abort with output FAIL.
  Check that $\gamma \in \{1, 2, \ldots, p - 1\}$, and if not abort with output FAIL.
  Compute $s \leftarrow h(\gamma^{\text{pri}1A} \mod p)$. 
• Output \( r \oplus s \) as the \( n/2 \)-bit session key.

Process \( \langle B, A, 1 \rangle \) works as follows. We denote \( B \)'s private key as \([\text{pri1}_B, \text{pri2}_B]\) and \( A \)'s public key as \([\text{pub1}_A, \text{pub2}_A]\).

• Receive \( n \)-bit strings \( \alpha \) and \( \beta \) on the input channel.
  Use \( \text{pub2}_A \) to verify that \( \beta \) is a signature by \( A \) of \([\alpha, A, B]\), and if not abort with output \text{FAIL}.
  Check that \( \alpha \in \{1, 2, \ldots, p - 1\} \), and if not abort with output \text{FAIL}.
  Compute \( r \leftarrow h(\alpha^{\text{pri1}_B} \pmod{p}) \).

• Choose a random \( y \in \{0, 1, 2, \ldots, p - 2\} \) and compute the \( n \)-bit string
  \( \gamma \leftarrow g^y \pmod{p} \), the \( n/2 \)-bit string
  \( s \leftarrow h(\text{pub1}_A^y \pmod{p}) \), and the \( n \)-bit string
  \( \delta \leftarrow \text{SIGN}_{\text{pri2}_B}(\alpha, \gamma, A, B) \).
  Send string \([\gamma, \delta]\) on the output channel.

• Output \( r \oplus s \) as the \( n/2 \)-bit session key.

Prove that this protocol is not secure. You do not have to do any “number theory” here. In fact, any protocol in which \( A \) encrypts a random string \( r \) to \( B \) using \( B \)'s infrastructure public key, and \( B \) encrypts a random string \( s \) to \( A \) using \( A \)'s infrastructure public key, and they use \( r \oplus s \) as the exchanged key, will be insecure, even if everything is properly signed.

2. (10 marks)
Prove that the following key exchange protocol is insecure. This protocol is similar to Protocol 2 of Notes #9, except instead of signing the role bits, we have Party 0 going first and Party 1 going second.

We use the same \text{GEN} as in Protocol 2, that is, \text{GEN} generates keys for a secure signature scheme, and we use the same notation for the signing algorithm. We describe the role 0 and the role 1 protocol separately.

\textbf{Protocol 2'}:

Process \( \langle A, B, 0 \rangle \) works as follows:

• Choose a random \( x \in \{0, 1, \ldots, p - 2\} \), compute \( \alpha = g^x \pmod{p} \), and send out the \( 2n \)-bit message: \([\alpha, \text{SIGN}_A(\alpha B)]\). (Note that \( A \)'s private key is used to sign a \( 2n \) bit string.)

• Receive a \( 2n \)-bit message \([\beta, \delta]\) where \(|\beta| = |\delta| = n \).
  Check that \( \beta \in \mathbb{Z}_p^* \), and use \( B \)'s public key to check that \( \delta \) is a valid signature of \([\beta A]\); if not, halt and output \text{FAIL}.

• Output \( h(\beta^x \pmod{p}) \) as the session key.
Process \( \langle B, A, 1 \rangle \) works as follows:

- Receive a 2\( n \)-bit message \([\alpha, \sigma] \) where \(|\alpha| = |\sigma| = n\).
  Check that \( \alpha \in \mathbb{Z}_p^* \), and use A’s public key to check that \( \sigma \) is a valid signature of \([\alpha B]\); if not, halt and output \text{FAIL}.
- Choose a random \( y \in \{0, 1, \ldots, p-2\} \), compute \( \beta = g^y \mod p \), and send out the 2\( n \)-bit message: \([\beta, \text{SIGN}_B(\beta A)]\). (Note that B’s private key is used to sign a 2\( n \) bit string.)
- Output \( h(\alpha^y \mod p) \) as the session key.

3. (15 marks – extra credit) It turns out that if we are careful, we can get a secure key exchange protocol by using an arbitrary semantically secure (as defined in Notes #10) public key encryption primitive instead of DDH in Protocol 2 from Notes #9. Say that \( G, E, D \) are the generating, encrypting, and decrypting functions for a \textbf{semantically secure} public key encryption primitive for encrypting strings of length \( n \).

As in Protocol 2, the public key infrastructure will consist of keys for a secure signature scheme.

Vaguely, the new protocol will work as follows. The role-1 party will use \( G \) to generate new keys \( pub \) and \( pri \); he will then send \( pub \), together with an appropriate signature, to the role-0 party. The role-0 party will then encrypt (using \( pub \)) a random session key \( k \) to the role-1 party, together with an appropriate signature.

More formally:

\(< B, A, 1 > \) works as follows:

- Use \( G \) to generate encryption keys \( pub \) and \( pri \).
  Compute \( \sigma = \text{SIGN}_B(1 A pub) \).
  SEND \( pub, \sigma \).
- RECEIVE strings \( e, \sigma' \) of the proper length.
  Verify that \( V E R_A([0 B pub] e, \sigma') = 1 \). (If not, \text{FAIL}.)
  Compute \( k = D_{pri}(e) \); if this decryption \text{FAILs}, then \text{FAIL}, else output \( k \) as the session key.

\(< A, B, 0 > \) works as follows:

- RECEIVE strings \( pub, \sigma \) of the proper length.
  Verify that \( V E R_B([1 A pub], \sigma) = 1 \). (If not, send out a string of 0’s and \text{FAIL}.)
- Choose a random \( k \in \{0, 1\}^n \) and compute
  \( e = E_{pub}(k, \text{RANDOMBits}) \) and \( \sigma' = \text{SIGN}_A(0 B pub e) \).
  SEND \( e, \sigma' \).
- Output \( k \) as the session key.

Prove that this new protocol is secure, with an outline similar to that of Protocol 2 in the notes. Give sufficiently many details that it is clear, for example, why we needed \( A \) to sign \( pub \).