Instructions

- Answer the following 5 questions, worth 20 points each.
- Please write the following text on the front page of your work (in English or Hebrew), and sign: “I declare that the answers submitted here are purely my own work; your-name, your-signature”.
- Put your answers in my mailbox, in the Wolfson Software Engineering Building, 3rd floor.

Questions (20 points each)

1. Let \( \text{DES}_K(P) \) denote the DES encryption of the plaintext \( P \) using key \( K \). Assume that \( P \) is a single block (64 bits). Let \( \overline{Z} \) denote the bitwise complement of \( Z \) (e.g., \( 00110001 = 11001110 \)).
   
   (a) If \( C = \text{DES}_K(P) \), what is \( \text{DES}_K(\overline{P}) = ? \) Explain your answer step-by-step through the DES tables.
   
   (b) A DES key \( K \) is called weak if \( E_K(E_K(x)) = x \) for all \( x \); in other words, encryption and decryption are the same for weak keys. DES has 4 weak keys, one of which is (hex) \( 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00 \). What are the other 3 weak keys? (ignore keys that are equal except for the 8 LSBs that DES does not use)
   
   (c) What pseudo-random stream is produced by using \( \text{DES}_K \) in OFB mode when \( K \) is a weak key?

2. The AES key schedule has an input of 16 bytes, which we can denote by an array of 4 words \((a, b, c, d)\), 4 bytes per word. The output of the key schedule is an array of 44 words, which we denote by \((a_0, b_0, c_0, d_0, \ldots, a_{10}, b_{10}, c_{10}, d_{10})\). Using this notation, the tuple \((a_i, b_i, c_i, d_i)\) is the round key for round \( i \).
   
   (a) How many AES keys \((a, b, c, d)\) exist such that round key 0 and round key 1 are identical: \((a_0, b_0, c_0, d_0) = (a_1, b_1, c_1, d_1)\)? Write an explicit expression for what such keys look like and explain your answer.
   
   (b) How many AES keys \((a, b, c, d)\) exist such that the round keys for rounds 0, 1, and 2 are all identical: \((a_0, b_0, c_0, d_0) = (a_1, b_1, c_1, d_1) = (a_2, b_2, c_2, d_2)\)? Explain your answer.

3. (a) Find an RC4 key \( k \) that brings the \( S \) array back to the state \( S = 0, 1, 2, \ldots, 255 \) after the Key Setup Algorithm (KSA). Note that \( k \) may be up to 256 bytes long.
   
   (b) Estimate how many such keys \( k \) exist.

4. RSA decryption requires the computation of \( m = c^d \mod n \) for \( n = pq \), \( p \) and \( q \) are primes, and \( d \) is the secret exponent.
   
   (a) Let \( c_p = c^d \mod p \) and \( c_q = c^d \mod q \). Write an explicit expression for \( m \) in terms of \( c_p \), \( c_q \), and the parameters of the system.
   
   (b) Which parts of the calculation in step (4a) can be pre-computed when the RSA system is generated?
(c) Typically \( d \gg p \) and \( d \gg q \). Using this observation, suggest a faster way to compute \( c_p \) and \( c_q \) than the definition in (4a) and prove that your computation is correct.

(d) Assume that additions take 0 time, multiplications mod \( p \) or mod \( q \) take \( t \) time, and multiplications mod \( n \) take \( 2t \) time. Assume also that \( p \) and \( q \) have approximately the same number of bits. Compare the decryption time using the basic definitions versus (4a) + (4c) [ignore any pre-computation costs].

(e) Can the method of (4a) be used for RSA encryption as well?

5. Consider an El-Gamal public-key crypto-system with public parameters \((p, g, g^a)\) where \( p \) is a prime modulus, \( g \) is a generator of \( \mathbb{Z}_p^* \).

(a) Denote the El-Gamal encryption of a message \( m \), using a specific random value \( k \), by

\[ \text{ElGamal}(m, k) = (\gamma, \delta). \]

Let \( r \in \mathbb{Z}_p^* \) be given, and let \( m' = mr \mod p \). Write an expression for \( \text{ElGamal}(m', k) \) in terms of \( \gamma, \delta \).

(b) Assume you have access to a deterministic black-box called \( \text{EG-Maybe}(\gamma, \delta) \). \( \text{EG-Maybe} \) takes an El-Gamal encryption, and attempts to crack (decrypt) the message. If it fails to decrypt, then \( \text{EG-Maybe} \) returns 0. \( \text{EG-Maybe} \) succeeds in decrypting only 1\% of the possible messages: for every choice of the random value \( k \), there exists a fixed (but unknown) set of \( p/100 \) messages \( m \) whose encryption \( \text{EG-Maybe} \) can break; the set of decryptable messages is different for each \( k \).

Design a randomized algorithm \( Z \), which uses \( \text{EG-Maybe} \) as a subroutine, and can decrypt every encrypted message \( (\gamma, \delta) \) with probability 1.

(c) Analyze the expected (average) number of \( \text{EG-Maybe} \) activations that \( Z \) uses until a decryption is found.