ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

FS 2019

Prof. R. Wattenhofer

Computer Engineering II

Solution to Exercise Sheet Chapter 12

1 Quiz Questions

a) No: The (supposed) security depends on not knowing the shift x. If CAESAR is applied twice, you just chose another shift (and in the worst case, cancel out the encryption).

b) No. E.g., $2 * 3 * 5 - 1 = 29$, which is prime. But $30 * 29 - 1 = 869 = 11 * 79$. Even the first part is not correct, e.g.: $2 * 3 * 5 * 7 = 210$ and $210 - 1 = 11 * 19$.

c) Yes. An attacker could just flip the bit of the message.

d) No. The attacker could just hash the modified message as well.

2 Secret Sharing

a) example execution: Let $a_1 = 3$ and $s = 2$, with 2 neighbors. Thus, $f(x) = 2 + 3x$. We distribute, e.g., $(2, 8)$ and $(3, 11)$. With both pairs, $s = 2$ can be recovered.

b) Without obtaining t pairs, k can take any value, i.e., $t - 1$ pairs reveal no information on k.

3 The One Time Pad

a) If you apply the same one time pad twice, it cancels out, leaving you with the original message.

b) Essentially, you created a new one time pad. If both are truly random, then this method is not more secure, but also not less, it is the same.

c) The beauty of the one time pad is that it transforms the message into a random message. As thus, any string of length k could be the original message - you still know nothing except for the length of the message.

d) Let k be the OTP. $c_1 \oplus c_2 = m_1 \oplus k \oplus m_2 \oplus k = m_1 \oplus m_2$, i.e., the one time pad cancels out. You don't have it decrypted yet, but it is a lot more information than just a random string.

e) You can get, e.g., $m_3 \oplus m_4$, using similar techniques as above. I.e., $c_3 \oplus c_2 = m_3 \oplus k$ and $c_4 \oplus c_3 = m_4 \oplus k$, leading to $c_4 \oplus c_2 = m_4 \oplus m_3$.

4 Diffie-Hellman Key Exchange

a) The primitive roots are 3 and 5.

b) Alice sends $3^4 = 81 = 4 \mod 7$ and Bob sends $3^2 = 9 = 2 \mod 7$. As thus, they agree on $(3^4)^2 = 4^2 = 16 = 2 \mod 7$ (or $(3^2)^4 = 2^4 = 16 = 2 \mod 7$).

c) (individual solutions)

d) Alice picked $k_A = 3$, Bob picks $k_B = 2$. Alice sends $3^3 = 27 = 2 \mod 5$ to Bob and Bob sends $3^2 = 9 = 4 \mod 5$ to Alice. As thus, they agree on $(3^3)^2 = 2^2 = 4 \mod 5$ (or $(3^2)^3 = 4^3 = 64 = 4 \mod 5$).

5 Message Authentification

a) E.g., use sequence numbers.

b) The answer is no to both: Take any m and $m' = m + p$, then $h(m) = h(m')$. Similarly, given any $1 \leq m \leq p - 1$, $h(m) = m$.

c) Use a large prime p with a primitive root g. With m being the message, let the hash be $h(m) = g^m \mod p$. Now, finding an x s.t. $h(x) = h(m)$ is the desired hash is equivalent to solving the discrete logarithm problem.

6 Protocol Design

a) If Alice is supposed to send the same message every time, the signature is the same as well. If Eve intercepts this message once, she can resend with the same signature any time she wants in the future and Bob will respond.

Anybody can use Alice's public key to encrypt any message and send it to her, so she has no way of telling whether the responses she receives are from Bob or Eve.

Essentially, Alice and Bob's plan doesn't achieve anything, except after Bob receives the first "read", he can be sure that Alice at some point in the past has actually sent it.

b) When Alice wants Bob to respond, she will send a message with a timestamp, signed with her private key. The timestamp will prevent replay attacks, and the signature will authenticate Alice to Bob. For secrecy, the message can be encrypted with Bob's public key.

To reply to Alice, Bob will timestamp and sign his message with his private key before encrypting with her public key, to similarly authenticate himself and prevent replay attacks on Alice.

c) The first time Alice sends a message to Bob, she will send a key k_0 for a symmetric encryption scheme, signed by her secret key and encrypted with Bob's public key. From now on, they can stop using asymmetric encryption and signatures. Bob can accompany his response by an HMAC using k_0, and encrypt the whole message with k_0.

Messages from Alice in subsequent communication rounds i will include a key k_i, authenticated by an HMAC using k_{i-1} and encrypted with k_{i-1}. Again, Bob can respond with a message, an HMAC using k_i, and encrypt the whole message with k_i.

Changing the key every round ensures forward secrecy. Messages are authenticated by an HMAC using the key corresponding to the round, so replay attacks are prevented.