Chapter 8 Review Questions

Question 1.
Confidentiality is the property that the original plaintext message can not be determined by an attacker who intercepts the ciphertext-encryption of the original plaintext message. Message integrity is the property that the receiver can detect whether the message sent (whether encrypted or not) was altered in transit. The two are thus different concepts, and one can have one without the other. An encrypted message that is altered in transmit may still be confidential (the attacker can not determine the original plaintext) but will not have message integrity if the error is undetected. Similarly, a message that is altered in transit (and detected) could have been sent in plaintext and thus would not be confidential.

Question 2.
(i) User's laptop and a web server; (ii) two routers; (iii) two DNS name servers.

Question 3.
One important difference between symmetric and public key systems is that in symmetric key systems both the sender and receiver must know the same (secret) key. In public key systems, the encryption and decryption keys are distinct. The encryption key is known by the entire world (including the sender), but the decryption key is known only by the receiver.

Question 4.
In this case, a known plaintext attack is performed. If, somehow, the message encrypted by the sender was chosen by the attacker, then this would be a chosen-plaintext attack.

Question 5.
An 8-block cipher has 2^8 possible input blocks. Each mapping is a permutation of the 2^8 input blocks; so there are $2^8!$ possible mappings; so there are $2^8!$ possible keys.

Question 6.
If each user wants to communicate with N other users, then each pair of users must have a shared symmetric key. There are N*(N-1)/2 such pairs and thus there are N*(N-1)/2 keys. With a public key system, each user has a public key which is known to all, and a private key (which is secret and only known by the user). There are thus 2N keys in the public key system.

Question 7.
a mod n = 23 , b mod n = 4. So (a*b) mod n = 23*4=92

Question 8.
175

Question 9.
One requirement of a message digest is that given a message M, it is very difficult to find another message M' that has the same message digest and, as a corollary, that given a message digest value it is difficult to find a message M'' that has that given message digest value. We have "message integrity" in the sense that we have reasonable confidence that given a message M and its signed message digest that the message was not altered since the message digest was computed and signed. This is not true of the Internet checksum, where we saw in Figure 7.18 that it easy to find two messages with the same Internet checksum.

Question 10.

No. This is because a hash function is a one-way function. That is, given any hash value, the original message cannot be recovered (given h such that h=H(m), one cannot recover m from h).

Question 11.
This is scheme is clearly flawed. Trudy, an attacker, can first sniff the communication and obtain the shared secret s by extracting the last portion of digits from H(m)+s. Trudy can then masquerade as the sender by creating her own message t and send (t, H(t)+s).

Question 12.
Suppose Bob sends an encrypted document to Alice. To be verifiable, Alice must be able to convince herself that Bob sent the encrypted document. To be non-forgeable, Alice must be able to convince herself that only Bob could have sent the encrypted document (e.g.,, non one else could have guess a key and encrypted/sent the document) To be non-reputiable, Alice must be able to convince someone else that only Bob could have sent the document. To illustrate the latter distinction, suppose Bob and Alice share a secret key, and they are the only ones in the world who know the key. If Alice receives a document that was encrypted with the key, and knows that she did not encrypt the document herself, then the document is known to be verifiable and non-forgeable (assuming a suitably strong encryption system was used). However, Alice can *not* convince someone else that Bob must have sent the document, since in fact Alice knew the key herself and could have encrypted/sent the document.

Question 13.
A public-key signed message digest is "better" in that one need only encrypt (using the private key) a short message digest, rather than the entire message. Since public key encryption with a technique like RSA is expensive, it's desirable to have to sign (encrypt) a smaller amount of data than a larger amount of data.

Question 14.

This is false. To create the certificate, certifier.com would include a digital signature, which is a hash of foo.com's information (including its public key), and signed with certifier.com's private key.

Question 15.

For a MAC-based scheme, Alice would have to establish a shared key with each potential recipient. With digital signatures, she uses the same digital signature for each recipient; the digital signature is created by signing the hash of the message with her private key. Digital signatures are clearly a better choice here.

Question 16.
The purpose of the nonce is to defend against the replay attack.

Question 17.

Once in a lifetimes means that the entity sending the nonce will never again use that value to check whether another entity is "live".

Question 18.

In a man-in-the-middle attack, the attacker puts himself between Alice and Bob, altering the data sent between them. If Bob and Alice share a secret authentication key, then any alterations will be detected.

Question 19.

Alice provides a digital signature, from which Bob can verify that message came from Alice. PGP uses digital signatures, not MACs, for message integrity.

Question 20.

False. SSL uses implicit sequence numbers.

Question 21.

The purpose of the random nonces in the handshake is to defend against the connection replay attack.

Question 22.

True. The IV is always sent in the clear. In SSL, it is sent during the SSL handshake.

Question 23.

After the client will generate a pre-master secret (PMS), it will encrypt it with Alice's public key, and then send the encrypted PMS to Trudy. Trudy will not be able to decrypt the PMS, since she does not have Alice's private key. Thus Trudy will not be able to determine the shared authentication key. She may instead guess one by choosing a random key. During the last step of the handshake, she sends to Bob a MAC of all the handshake messages, using the guessed authentication key. When Bob receives the MAC, the MAC test will fail, and Bob will end the TCP connection.

Question 24.
False. Typically an IPsec SA is first established between Host A and Host B. Then all packets in the stream use the SA.

Question 25.
False. IPsec will increment the sequence number for every packet it sends.

Question 26.
False. An IKE SA is used to establish one or more IPsec SAs.

Question 27.
01011100

Question 28.
True

Question 29.
Filter table and connection table. The connection table keeps track of connections, allowing for a finer degree of packet filtering.

Question 30.
True

Question 31.
True

Question 32.
If there isn't a packet filter, than users inside the institution's network will still be able to make direct connections to hosts outside the institution's network. The filter forces the users to first connect to the application gateway.

Question 33.
True

Chapter 8 Problems

Problem 1.
The encoding of "This is an easy problem" is "uasi si my cmiw lokngch".

The decoding of “rmij'u uamu xyj” is “wasn't that fun”.

Problem 2.
If Trudy knew that the words “bob” and “alice” appeared in the text, then she would know the ciphertext for b,o,a,l,i,c,e (since “bob” is the only palindrome in the message, and “alice” is the only 5-letter word. If Trudy knows the ciphertext for 7 of the letters, then she only needs to try 19!, rather than 26!, plaintext-ciphertext pairs. The difference between 19! and 26! is 26*25*24...*20, which is 3315312000, or approximately 10^9.

Problem 3.
Every letter in the alphabet appears in the phrase “The quick fox jumps over the lazy brown dog.” Given this phrase in a chosen plaintext attack (where the attacker has both the plain text, and the ciphertext), the Caesar cipher would be broken - the intruder would know the ciphertext character for every plaintext character. However, the Vigenere cipher does not alway translate a given plaintext character to the same ciphertext character each time, and hence a Vigenere cipher would not be immediately broken by this chosen plaintext attack.

Problem 4.
(a) The output is equal to 00000101 repeated eight times.
(b) The output is equal to 00000101 repeated seven times + 10000101.
(c) We have $(A^R B^R C^R)^R = CBA$, where A, B, C are strings, and R means inverse operation. Thus:

1. For (a), the output is 10100000 repeated eight times;
2. For (b), the output is 10100001 + 10100000 repeated seven times.

Problem 5.
(a) There are 8 tables. Each table has 2^8 entries. Each entry has 8 bits.
number of tables * size of each table * size of each entry = $8*2^{8*}$ 8= 2^{14} bits

(b) There are 2^{64} entries. Each entry has 64 bits. 2^{71} bits

Problem 6.
(a) 100100100 ==> 011011011
(b) Trudy will know the three block plaintexts are the same.
(c) c(i) = K_S(m(i) XOR c(i-1))
c(1) = K_S(100 XOR 111) = K_S (011) = 100
c(2) = K_S(100 XOR 100) = K_S (000) = 110
c(1) = K_S(100 XOR 110) = K_S (010) = 101

Problem 7.

(a) We are given $p = 3$ and $q = 11$. We thus have $n = 33$ and $q = 11$. Choose $e = 9$ (it might be a good idea to give students a hint that 9 is a good value to choose, since the resulting calculations are less likely to run into numerical stability problems than other

choices for e.) since 3 and $(p-1)*(q-1)=20$ have no common factors. Choose $d=9$ also so that $e*d=81$ and thus $e*d-1=80$ is exactly divisible by 20. We can now perform the RSA encryption and decryption using $n=33$, $e=9$ and $d=9$.

```
letter  m      m**e          ciphertext = m**e mod 33
  d     4      262144              25
  o     15     38443359375         3
  g     7      40353607            19

ciphertext    c**d             m = c**d mod n   letter
     25        38146972265625      4               d
     3         19683               15              o
     19        322687697779        7               g
```

(b) We first consider each letter as a 5-bit number: 00100, 01111, 00111. Now we concatenate each letter to get 001000111100111 and encrypt the resulting decimal number m=4583. The concatenated decimal number m (= 4583) is larger than current n (= 33). We need m < n. So we use p = 43, q = 107, n = p*q = 4601, z = (p-1)(q-1) = 4452. e = 61, d = 73

ciphertext = m**e mod 4601

m**e= 213865776018280578040896021565305671886114998690297887338084388043028645956206139567258407209497648456409561187848752467850332361977771297302589617569184002920486328061975277854477915672551018944928209725081857698028817189 83

ciphertext = m**e mod 4601 = 402

c**d
= 1283813313619771634195712132539793287643533147482536209328405262793027158861012392053287249633570967493122280221453815012934241370540204581459871497938723214101470322779458649981794563339059 2

ciphertext = m**e mod 4601 = 4583

Problem 8.

p = 5, q = 11
(a) n = p*q = 55, z = (p-1)(q-1) = 40
(b) e = 3 is less than n and has no common factors with z.
(c) d = 27
(d) m = 8, m^e = 512, Ciphertext c= m^e mod n = 17

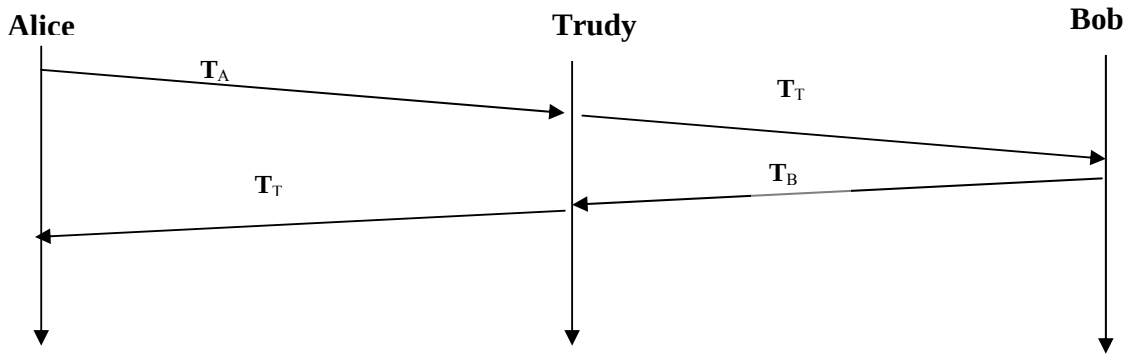
Problem 9.

	Alice	Bob
secrect key:	S_A	S_B
public key:	$T_A = (g\text{^}S_A) \bmod p$	$T_B = (g\text{^}S_B) \bmod p$
shared key:	$S = (T_B\text{^}S_A) \bmod p$	$S' = (T_A\text{^}S_B) \bmod p$

(a) $S = (T_B\text{^}S_A) \bmod p = ((g\text{^}S_B \bmod p)\text{^}S_A) \bmod p = (g\text{^}(S_BS_A)) \bmod p$
$= ((g\text{^}S_A \bmod p)\text{^}S_B) \bmod p = (T_A\text{^}S_B) \bmod p = S'$
(b and c) $p = 11$, $g = 2$

	Alice	Bob
secrect key:	$S_A = 5$	$S_B = 12$
public key:	$T_A = (g\text{^}S_A) \bmod p = 10$	$T_B = (g\text{^}S_B) \bmod p = 4$
shared key:	$S = (T_B\text{^}S_A) \bmod p = 1$	$S' = (T_A\text{^}S_B) \bmod p = 1$

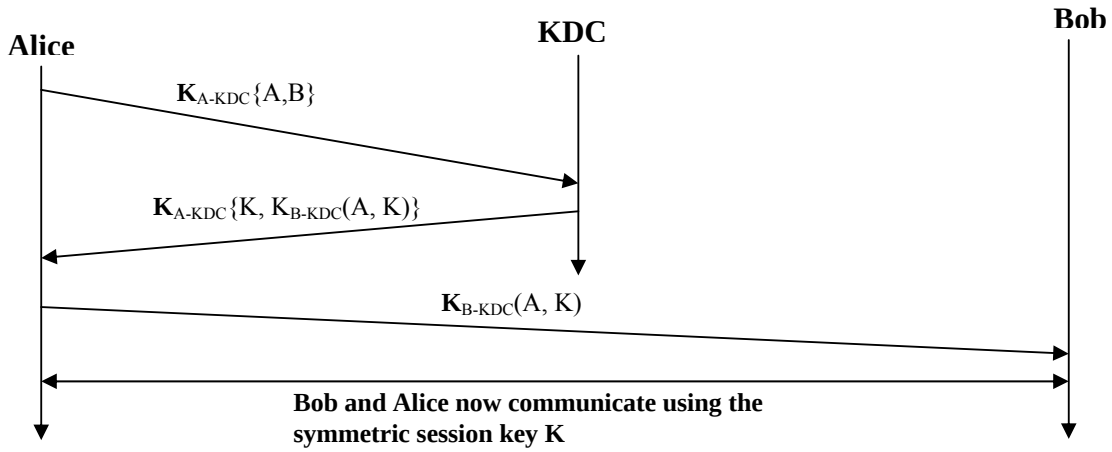
(d)



The Diffie-Hellman public key encryption algorithm is possible to be attacked by man-in-the-middle.

1. In this attack, Trudy receives Alice's public value (T_A) and sends her own public value (T_T) to Bob.
2. When Bob transmits his public value (T_B), Trudy sends her public key to Alice (T_T).
3. Trudy and Alice thus agree on one shared key (S_{AT}) and Trudy and Bob agree on another shared key (S_{BT}).
4. After this exchange, Trudy simply decrypts any messages sent out by Alice or Bob by the public keys S_{AT} and S_{BT}.

Problem 10.



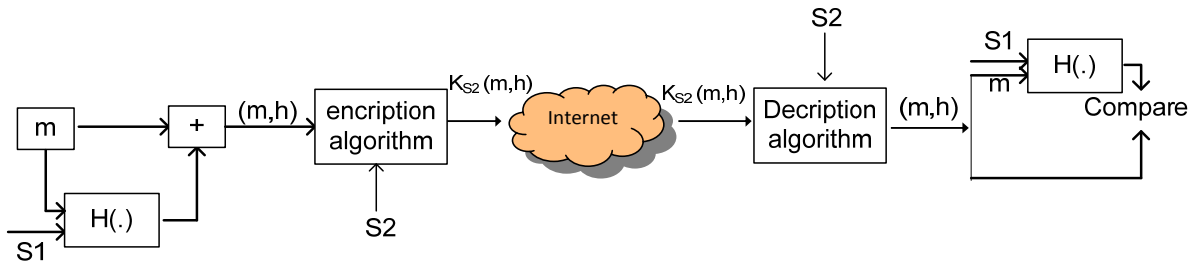
Problem 11.

The message

I	O	U	1
9	0	.	9
0	B	O	B

has the same checksum

Problem 12.



Problem 13.

The file is broken into blocks of equal size. For each block, calculate the hash (for example with MD5 or SHA-1). The hashes for all of the blocks are saved in the .torrent

file. Whenever a peer downloads a block, it calculates the hash of this block and compares it to the hash in the .torrent file. If the two hashes are equal, the block is valid. Otherwise, the block is bogus, and should be discarded.

Problem 14.

Digital signatures require an underlying Public Key Infrastructure (PKI) with certification authorities. For OSPF, all routers are in a same domain, so the administrator can easily deploy the symmetric key on each router, without the need of a PKI.

Problem 15.

Bob does not know if he is talking to Trudy or Alice initially. Bob and Alice share a secret key K_{A-B} that is unknown to Trudy. Trudy wants Bob to authenticate her (Trudy) as Alice. Trudy is going to have Bob authenticate himself, and waits for Bob to start:

1. **Bob-to-Trudy: "I am Bob"** Commentary: Bob starts to authenticate himself. Bob's authentication of himself to the other side then stops for a few steps.
2. **Trudy-to-Bob: "I am Alice"** Commentary: Trudy starts to authenticate herself as Alice
3. **Bob-to-Trudy: "R"** Commentary: Bob responds to step 2 by sending a nonce in reply. Trudy does not yet know $K_{A-B}(R)$ so she can not yet reply.
4. **Trudy-to-Bob: "R"** Commentary: Trudy responds to step 1 now continuing Bob's authentication, picking as the nonce for Bob to encrypt, *the exact same value that Bob sent her to encrypt in Step 3.*
5. **Bob-to-Trudy: "**$K_{A-B}(R)$" Bob completes his own authentication of himself to the other side by encrypting the nonce he was sent in step 4. Trudy now has $K_{A-B}(R)$. (Note: she does not have, nor need, K_{A-B}
6. **Trudy-to-Bob: "**$K_{A-B}(R)$" Trudy completes her authentication, responding to the R that Bob sent in step 3 above with $K_{A-B}(R)$. Since Trudy has returned the properly encrypted nonce that Bob send in step 3, Bob thinks Trudy is Alice!

Problem 16.

This wouldn't really solve the problem. Just as Bob thinks (incorrectly) that he is authenticating Alice in the first half of Figure 7.14, so too can Trudy fool Alice into thinking (incorrectly) that she is authenticating Bob. The root of the problem that neither Bob nor Alice can tell is the public key they are getting is indeed the public key of Alice of Bob.

Problem 17.

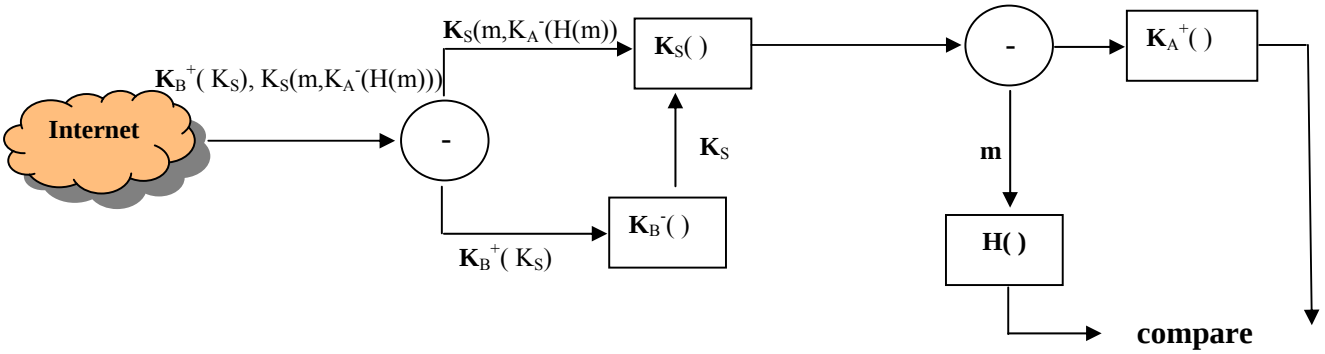


Figure: Operations performed by Bob for confidentiality, integrity, and authentication

Problem 18

(a) No, without a public-private key pair or a pre-shared secret, Bob cannot verify that Alice created the message.
(b) Yes, Alice simply encrypts the message with Bob's public key and sends the encrypted message to Bob.

Problem 19

a) Client
b) IP: 216.75.194.220, port: 443
c) 283
d) 3 SSL records
e) Yes, it contains an encrypted master secret
f) First byte: bc; Last byte: 29
g) 6

Problem 20.

Again we suppose that SSL does not provide sequence numbers. Suppose that Trudy, a woman-in-the-middle, deletes a TCP segment. So that Bob doesn't anything, Trudy needs to also adjust the sequence numbers in the subsequent packets sent from Alice to Bob, and the acknowledgment numbers sent from Bob to Alice. The result will be that Bob will, unknowingly, be missing a packet's worth of bytes in the byte stream.

Problem 21.

No, the bogus packet will fail the integrity check (which uses a shared MAC key).

Problem 22.
(a) F
(b) T
(c) T
(d) F

Problem 23.

If Trudy does not bother to change the sequence number, R2 will detect the duplicate when checking the sequence number in the ESP header. If Trudy increments the sequence number, the packet will fail the integrity check at R2.

Problem 24.

a) Since IV = 11, the key stream is 111110100000 …….

Given, m = 10100000
Hence, ICV = 1010 XOR 0000 = 1010
The three fields will be:

IV: 11
Encrypted message: 10100000 XOR 11111010 = 01011010
Encrypted ICV: 1010 XOR 0000 = 1010

b) The receiver extracts the IV (11) and generates the key stream 111110100000 …….

XORs the encrypted message with the key stream to recover the original message:
01011010 XOR 11111010 = 10100000

XORs the encrypted ICV with the keystream to recover the original ICV:

1010 XOR 0000 = 1010

The receiver then XORs the first 4 bits of recovered message with its last 4 bits:
1010 XOR 0000 = 1010 (which equals the recovered ICV)

c) Since the ICV is calculated as the XOR of first 4 bits of message with last 4 bits of message, either the 1^{st} bit or the 5^{th} bit of the message has to be flipped for the received packet to pass the ICV check.

d) For part (a), the encrypted message was 01011010
Flipping the 1^{st} bit gives, 11011010
Trudy XORs this message with the keystream:
11011010 XOR 11111010 = 00100000

If Trudy flipped the first bit of the encrypted ICV, the ICV value received by the receiver is 0010
The receiver XORs this value with the keystream to get the ICV:
0010 XOR 0000 = 0010
The receiver now calculates the ICV from the recovered message:
0010 XOR 0000 = 0010 (which equals the recovered ICV and so the received packet passes the ICV check)

Problem 25.

Filter Table:

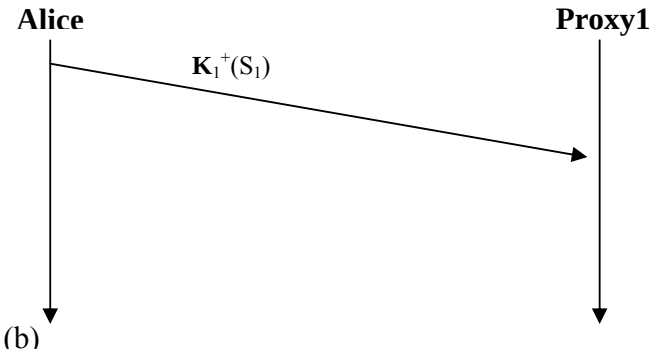
Action	Source address	Dest address	Protocol	Source port	Dest port	Flag bit	Check connection
allow	222.22/16	outside of 222.22/16	TCP	> 1023	22	any	
allow	outside of 222.22/16	222.22/16	TCP	22	> 1023	ACK	x
Allow	outside of 222.22/16	222.22.0.12	TCP	>1023	80	any	
Allow	222.22.0.12	outside of 222.22/16	TCP	80	>1023	any	
deny	All	all	all	all	all	all	

Connection Table:

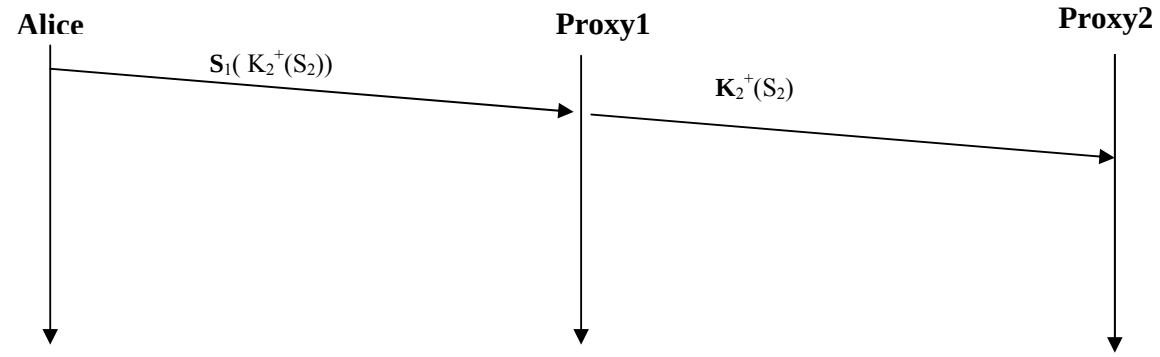
Source address	Dest address	Source port	Dest port
222.22.1.7	37.96.87.123	12699	22
222.22.93.2	199.1.205.23	37654	22
222.22.65.143	203.77.240.43	48712	22

Problem 26.

(a)



(b)



(c)