[Assignment 1]

1. With a 2-bit prefix, there would have been 18 bits left over to indicate the network. Consequently, the number of networks would have been 218 or 262144. However, all 0s and all 1s are special, so only 26142 are available.

2. For a class B network, the upper 16 bits form the network number and the lower 16 bits are the subnet and host fields. Of the lower 16 bits, the most significant 4 bits are: 1111. This leaves 12 bits for the host number, so 4096 host addresses exit.
Addresses 0 and -1 are special, so the maximum number of r hosts is 4096

3. You say, ARP does not provide a service to the network layer, it is part of the network layer and helps provide a service to the transport layer. The issue of IP addressing does not occur in the data link layer. Data link layer protocols are like protocols HDLC, PPP, etc. They move bits from one end of a line to the other.

4. Even though each datagram arrives intact, it is possible that datagrams arrive in the wrong order, so TCP has to be prepared to reassemble the parts of a message properly.

5. If we configure same static IP address on two cards that connected on the same Ethernet, both of the cards become unusable. I have tried to configure same static IP address of two windows machines on the same Ethernet, and both of these two machines pop out the message window, and ask you to change the IP address. It because that the windows operating system would not allow the same IP address that configured on the same network segment. But if we consider the OS that we used allows you to statically configure same IP address on the same network segment, what will happen? The ARP request will receive two answers with different hardware addresses. And the sender will be confused who is the correct receiver. Generally, we would not allow these to happen.
It is possible to configure two cards with the same IP address on different Ethernet, which depends on your network IP address allocation policy. We had discussed the IP address is uniquely to identify the location of your machine. Generally speaking, the ISP would not allocate the same public IP address to different customers. If the customer assign their machine with the same IP address that used by other customers, first of all, if the router "generously" propagate this routing information.
It may cut one of the machines off the Internet. Because the network routing will make the shortest path so the near machine, there might cause misdirect the packets to the wrong machine. But for totally separate two networks or the network implement with private IP address, it works.

6. Answer to question 6:
Answer (a): E4, the Ethernet interface address for source node H4.
Answer (b) : E6, the LAN 2 Ethernet interface address for the router R.
Answer (c): I9, destination host H9's IP (interface) address.
Answer (d): E7, the LAN 3 Ethernet interface address for the router R.
Answer (e): E9, host H9's Ethernet interface address.
Answer (f) : I4, source host H4's IP (interface) address.

Answer (g): Yes it will. B's filtering table is empty, so it does not know where the (destination) node with Ethernet address E6 is (see answer to (b) above).
Consequently, it will ``flood'' a copy of the frame on to LAN 1. Note that B now learns that (source) node E4 is off of its interface E3.
Answer (h): No it will not. The frame carrying the reply on LAN 2 will have destination Ethernet address E4, and B now knows that E4 is off of its interface E3 (see answer to part (g) above), which is the same interface it receives the frame on. Note that this frame will have source Ethernet address E6, so B now learns that node E6 is also off of its interface E3.
Answer (i): Yes it will. The frame will be carrying destination Ethernet address E5, and B still has not learned where E5 is. Consequently, it will ``flood'' a copy of the frame on to LAN 1.
Answer (j): No it will not. The packet will carry destination IP address I5, and source host H4's routing table will have shown that the node with address I5 is on the same network as H4 itself. Consequently, the Ethernet frame encapsulating the packet will have destination address E5, and will be ignored by node R.

7. Genuity (Level 3)

8. **

[Assignment 2]
Chapter 2.7 Homework
2.2 Anyone who knows which hash function is being used can forge a message.

2.3 Without the use of public key technology, they will still need to know each other's keys to do the verification, which means they can still forge each other's messages.
2.4 You can forge a signature on any message with the same hash as one that has been legitimately signed.

2.5 Alice doesn't need to know KAB; when Bob challenges with r_B, Alice just opens a second connection to bob and challenges him with r_B, then uses his response to respond to his firstconnection challenge. She can abort the second connection.

Chapter 3.7 Homework
3.1 You must distinguish between 264! different mappings. This takes $\log_2 2^{64}! \approx 2^{70}$ bits to represent. You could consecutively number all the permutations and just store the permutation number. While this would be most efficient in terms of storage, figuring out which permutation a given number specified might take a little while. So simplest is probably best-just store a 2^{64}-element table of 64-bit output values indexed by the 64-bit input value. This takes a full 2^{70} bits.
Note that a terabyte is 2^{43} bits.

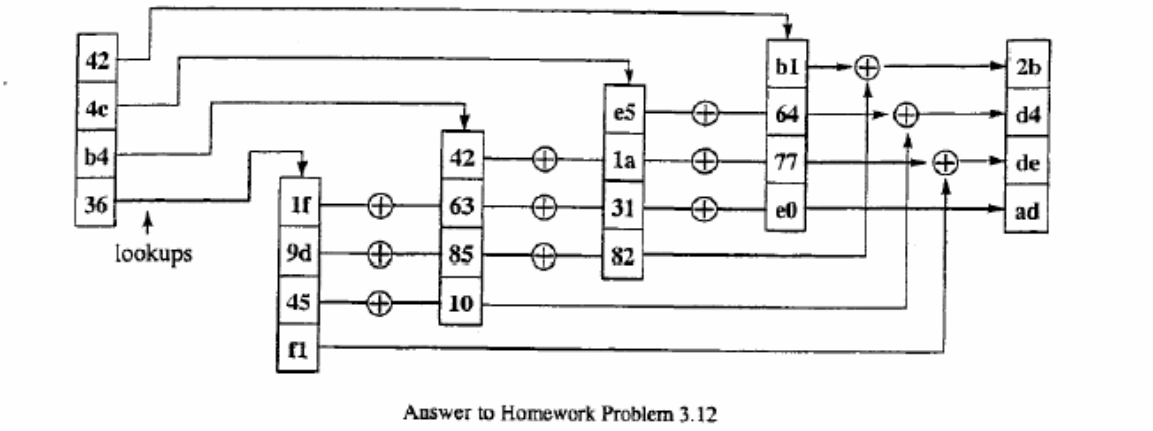
3.3 There are 2^{56} possible keys and 2^{64} possible ciphertext blocks for a particular plaintext block.
So only about 2^{56-64}=1/256 of the possible ciphertext blocks can be obtained by with a DES key.

3.4 Modify the DES scheme to not do an initial permutation of the key. Any message encrypted with regular DES using key K is also encrypted with the modified scheme using a key K' derived from K by applying DES's initial key permutation. If we can break the modified scheme, we can decrypt the message.

3.7 Number the bits of the 56-bit key from left to right with consecutive integers starting at 1, but skip multiples of 8. i.e. 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63. The rest of the description is unchanged.

3.8 Weak keys have each of C0 and D0 all ones or all zeroes. Since each C_i is a permutation of C_0 and each D_i is a permutation of D_0, each of the C_is is the same as C_0 and each of the D_is is the same as D0. Since K_i depends only on C_i and D_i all the K_is are the same. So the K_is are the same forwards and backwards, meaning that encryption and decryption are identical.

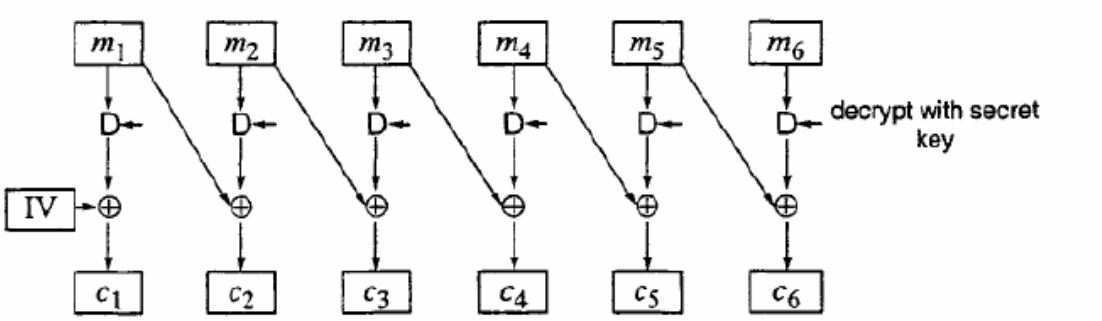
3.12



Answer to Homework Problem 3.12

Chapter 4.5 Homework
4.3 Essentially the same attack works. Of course, Table B is made by using its keys to encrypt rather than decrypt.

4.6 It would certainly work, in the sense of allowing encryption and decryption of messages. In this encryption scheme, block n of the plaintext is ⊕'d with the output of the D step for block n+1 to get block n+1 of ciphertext.



One problem with this is that if Trudy knows the plaintext and ciphertext for a set of messages, she can mix and match the blocks of those messages almost as easily as with ECB. The reason is that block n of plaintext ⊕'d with block n+1 of ciphertext is D of block n+1 of plaintext, and once Trudy knows D of a desired block of plaintext, she can ⊕ it with the plaintext of the previous block to produce correct ciphertext.
More seriously, since block n+1 of ciphertext depends only on blocks n and n+1 of plaintext, patterns of ciphertext blocks indicate patterns in the plaintext, which provides a big clue for cryptanalysis. And if D of block n+1 of plaintext is known, it can be ⊕ 'd with block n+1 of ciphertext to get block n of plaintext.

[Assignment 3]

6.2 The attacker will not be able to decrypt the Diffie-Hellman values sent to him and so will not be able to compute the shared secrets.

6.3 d is the multiplicative inverse of e mod $(p\text{-}1)(q\text{-}1)$, and so is unique up to multiples of $(p\text{-}1)(q\text{-}1)$.

6.5 You don't need to transmit information on which p, q, and g are being used.

6.8 $(m_1^j)^d \bmod n = (m_1^j)^d \bmod n$, so to compute your signature on $m_1^j \bmod n$, Fred just raises your signature on m_1 to the jth power, mod n.

$(m_1^{-1})^d \bmod n = (m_1^d)^{-1} \bmod n$, so to compute your signature on $m_1^{-1} \bmod n$, Fred just computes the inverse mod n of your signature on m_1.

$(m_1 \cdot m_2)^d \bmod n = m_1^d \cdot m_2^d \bmod n$, so to compute your signature on $m_1 \cdot m_2 \bmod n$, Fred just multiples your signature on m_1 by your signature on m_2, mod n.

So for the general case of $m_1^j \cdot m_2^k \bmod n$, Fred gets your signature on $m_1^{\operatorname{sgn} j} \bmod n$ and raises it to the $|j|$th power, mod n, then gets your signature on $m_2^{\operatorname{sgn} k} \bmod n$ and raises it to the $|k|$th power, mod n, and finally multiplies the results together, mod n. [$\operatorname{sgn} x = x/|x|$]

6.10 In ElGamal, the signature X is $S_m + d_m S \bmod (p\text{-}1)$, where d_m is the message digest of $m|T_m$, S_m is the secret number, and S is the signer's private key. X, m, and T_m are transmitted, and the message digest function is public (as are p, q, and T), so we can compute d_m. If we also know S_m, we can compute $d_m S \bmod (p\text{-}1) = X - S_m$. Under the assumption that $q = (p\text{-}1)/2$ is prime, we know $d_m S \bmod q$ and can multiply by $d_m^{-1} \bmod q$ to get $S \bmod q$. Then $S \bmod (p\text{-}1)$ is either $S \bmod q$ or $S \bmod q + q$. We can verify which it is by seeing which gives T when used as the exponent of g.

Similarly, if we have two different signatures X_1 and X_2 using the same secret S_m, we can take their difference $X_1 - X_2 = (S_m + d_1 S) - (S_m + d_2 S) = d_1 S - d_2 S = (d_1 - d_2)S \bmod (p\text{-}1)$. Since we know d_1 and d_2, we can compute $(d_1 - d_2)^{-1} \bmod q$, and thus $(d_1 - d_2)^{-1}(X_1 - X_2) \bmod q$, which is $S \bmod q$. As before, this lets us find $S \bmod (p\text{-}1)$.

7.1 By Euclid's algorithm, we know that there are integers u and v such that $um + vn = \gcd(m,n)$. Dividing both sides by $\gcd(m,n)$ gives $u(m/\gcd(m,n)) + v(n/\gcd(m,n)) = 1$, so $m/\gcd(m,n)$ and $n/\gcd(m,n)$ are relatively prime.

7.2 Since bc is a multiple of a, $bc = ka$ for some integer k. Since a and b are relatively prime, there are integers u and v such that $ua + vb = 1$. Swapping sides, then multiplying both sides by c gives $c = uac + vbc = uac + vkac = (uc + vkc)a$.

7.3 $mq = r \bmod n$ iff there is an integer k such that $qm + kn = r$. Since the left side is divisible by $\gcd(m,n)$, this can only happen if r is a multiple of $\gcd(m,n)$. Dividing both sides by $\gcd(m,n)$ gives $qm' + kn' = r'$ where $m' = m/\gcd(m,n)$, $n' = n/\gcd(m,n)$, and $r' = r/\gcd(m,n)$. Euclid's algorithm gives us integers u and v such that $um' + vn' = 1$, so multiplying both sides by r' gives $ur'm' + vr'n' = r'$. So we can use $q = ur'$, and this is unique mod n'. For it to be unique mod n, we must have $n = n'$, i.e. $\gcd(m,n) = 1$; otherwise we can add any multiple of n' to q.

7.7 For $n = 1$, $\phi(n) = n$. For n prime, $\phi(n) = n\text{-}1$, so $\phi(n)/n \rightarrow 1$ as $n \rightarrow \infty$.

7.8 For n the product of consecutive primes, we have $\phi(n)/n \rightarrow 0$ as as $n \rightarrow \infty$. The first few values are $\phi(2)/2 = 1/2$, $\phi(6)/6 = 1/3$, $\phi(30)/30 = 4/15$, $\phi(210)/210 = 8/35$.

7.9 No. If n is a positive integer, there are only n elements in Z_n, so $\phi(n) \leq n$, with equality only for $n = 1$, since if $n > 1$, 0 is not relatively prime to n.

7.13 By the Chinese Reminder Theorem, we know that we can represent $a \bmod n$ number x uniquely as a $(k+1)$tuple where the ith component is the mod $p_i^{\alpha i}$ number x mod $p_i^{\alpha i}$, and we can multiply in this representation just by multiplying componentwise mod $p_i^{\alpha i}$. So a square root of 1 mod n must be represented as a tuple whose ith component is a square root of 1 mod $p_i^{\alpha i}$, and any such tuple will represent a square root of 1 mod n.

How many square roots of 1 are there mod $2^{\alpha 0}$? If $\alpha_0 \leq 1$, 1 is the only square root of 1. If $\alpha_0 = 2$, 1 and -1 are the two square roots of 1. Otherwise, 1, -1, $2^{\alpha 0-1} + 1$. and $2^{\alpha 0-1} - 1$ are the four square roots of 1. To see this, note that $1 = (x+1)^2 = x(x+2)+1$ iff $x(x+2) = 0$

which is true mod $2^{\alpha 0}$ iff $x(x+2) = 2^{\alpha 0}m$ for some integer m, and that happens exactly when x or $x+2$ is 0 or $2^{\alpha 0-1}$ mod $2^{\alpha 0}$.

How many square roots of 1 are there mod $p_i^{\alpha i}$, where p_i is an odd prime? In this case 1 and -1 are the two square roots of 1. To see this, note that $1= (x+1)^2 = x(x+2)+1$ iff $x(x+2) = 0$ which is true mod $p_i^{\alpha i}$ iff $x(x+2) = p_i^{\alpha i} m$ for some integer m, and that happens exactly when x or $x+2$ is 0 mod $p_i^{\alpha i\ 0}$. (Only for 2 do we get a free power of p and thus extra square roots of 1.)

Multiplying the number of square roots of 1 in each component gives the desired result.

[Assignment 5]

4.8 exercises:

3.

a. My assumption here is that the system has no integrity controls.

This is true. If a system lacks integrity, then data can be changed without restraint. So, anone can change another user's authentication information, allowing them access to that user's account—and allowing them to see any data for that user or, bygeneralizing this in the obvious way, any user on the system.

If some integrity controls work, then the ability of the system to provide confidentiality depends on the effectiveness of the integrity controls and their use to protect critical information.

b. My assumption here is that the system has no confidentiality controls.

If there is no confidentiality, then all authentication information will be available. Unless authentication mechanisms do not use secret information (for example, biometrics or positions), any user can authenticate as another user. Hence there is no integrity.

Now suppose authentication information does not rely on confidentiality. Can the data in a file be kept confidential? To do so, either the user must be prevented from reading the file (for which there are no controls) or from reading the data in the file (for example, by cryptography). In the latter case, if the data is encrypted on the system, the key must be available, and as there is no confidentiality, the key can be read. If the data is not encrypted on the system, then the data cannot be used on the system but will remain confidential. So, the answer is that the system cannot provide integrity. But if data is protected when placed on the system, it will remain protected as long as confidentiality mechanisms were applied to the data itself (and not to the containing object) and the mechanisms to undo them are not on the system.

4. The problem with the cryptographer's claim is how to protect the keys. At some point, the cryptographic keys must be available to encipher, decipher, or validate the integrity of data. If the keys are kept in memory, they must be protected, either by other cryptographic keys (which require similar protection) or by non-cryptographic access control mechanisms. If the keys are kept off-line (for example, in a smart card or a dongle), access to the external unit must be protected either by cryptographic keys (which

require the protection discussed earlier) or non-cryptographic access control. By a simple process of induction, or *reductio ad absurdam*, non-cryptographic based access control mechanisms must be used at some point, refuting the cryptographer's claim.

5.
a. This is an example of a discretionary access control policy because an individual user can set permissions to allow or deny access to the object based upon a user's identity.
b. This is an example of an originator controlled access control policy because the author (who is the originator) controls the dissemination of the memorandum.
c. This is an example of a mandatory access control policy because the right to enter the room cannot be changed by an individual user. Also, it depends upon a property of the entity ("general") rather than the identity of the entity ("John Smith").
d. This is a combination of an originator controlled access control policy and a discretionary access control policy. The originator, which is the registrar, controls dissemination of the data, but the student also has some control, and allows access to the individual record based upon the identity of the faculty member.

5.5 exercises:
2. Let A's compartment be (L_A, C_A) and B's be (L_B, C_B). The simple security condition says that A can read B if and only if $L_A \geq L_B$ and $C_B \subseteq C_A$. The *-property says that A can write B if and only if $L_A \geq L_B$ and $C_B \subseteq C_A$. Remember that TOPSECRET ≥ SECRET ≥ CONFIDENTIAL ≥ UNCLASSIFIED.
a. L_{Paul} = TOPSECRET ≥ SECRET = L_{doc}, so Paul cannot write the document. Paul cannot read the document either, because $C_{doc} = \{ B, C \} \not\subset \{ A, C \} = C_{Paul}$.
b. L_{Anna} = CONFIDENTIAL ≥ CONFIDENTIAL = L_{doc}, but $C_{doc} = \{ B \} \not\subset \{ C \} = C_{Anna}$ so Anna cannot read the document, and $C_{Anna} = \{ C \} \not\subset \{ B \} = C_{doc}$, so Anna cannot write the document.
c. L_{Jesse} = SECRET ≥ CONFIDENTIAL = L_{doc}, and $C_{doc} = \{ C \} \subseteq \{ C \} = C_{Jesse}$, so Jesse can read the document. As $L_{Jesse} > L_{doc}$, however, Jesse cannot write the document.
d. As L_{Sammi} = TOPSECRET ≥ CONFIDENTIAL = L_{doc} and $C_{doc} = \{ A\} \subseteq \{ A, C \} = C_{Sammi}$, Sammi can read the document. But the first inequality means Sammi cannot write the document.
e. As $C_{Robin} = \emptyset \subseteq \{ B \} = C_{doc}$ and L_{doc} = CONFIDENTIAL ≥ UNCLASSIFIED = L_{Robin}, Robin can write the document. However, because $L_{doc} \geq L_{Robin}$, she cannot read the document.

6.6 exercises:
1. Let (S_L, S_C) be the security clearance and category set of the security level L. The Bell-LaPadula model allows a subject in L to read an entity in L′ if, and only if, L dominates L′. By definition 5–1, this means $S_L \leq S_{L'}$ and $S_{C'} \subseteq S_C$. As the security and integrity levels are the same, (S_L, S_C) also defines the integrity clearance and category set of the integrity level L. The Biba model allows a subject in L to read an entity in L′ if, and only if, L′ dominates L. Again, this means $S_{L'} \leq S_L$ and $S_C \subseteq S_{C'}$. Putting the two relations together, a subject in L can read an entity in L′ if, and only if, $S_L \leq S_{L'}$ and $S_{L'} \leq S_L$, and $S_{C'} \subseteq S_C$ and $S_C \subseteq S_{C'}$. By asymmetry, this means a subject in L can read an entity in L′

if, and only if, $S_L = S_{L'}$ and $S_{C'} = S_C$, or $L = L'$. Hence the subject and the entity being read must be at the same security (integrity) level. A similar argument shows that, if a subject can write to an entity, the subject and entity being written must be at the same security (integrity) level.

3. The TPs can be executed in parallel, provided the Bernstein conditions are met. The only problem with executing TPs in parallel occurs when two of them try to update or write the same CDI at the same time. The Bernstein conditions state that multiple processes may read an object at the same time, but at most one process may write to an object and, during that time, no process may read the object. This ensures that the values in the object are consistent.
Similarly, if two TPs try to alter a CDI simultaneously, there is a race condition rendering the result indeterminate; if one TP is writing the object and others are reading it at the same time, the values read may be inconsistent.
5. The new relation would contain tuples of the form (user, TP, { CDI_1, …, CDI_n }), meaning that the user user could execute the transaction procedure TP on the constrained data items CDI_1, …, CDI_n. One problem is how to handle the CDIs that the TP could operate on, but that no user is allowed to perform that TP on those CDIs. One way is to define a pseudo-user called "nobody" and put these CDIs into a tuple with the user as "nobody". The "allowed" and "certified" relations are fundamentally different. ER1 defines what (type of) entities the transaction procedure can operate on, and ER2 defines who can apply the transaction procedure to specific constrained data items. If they were collapsed into one, two different mechanisms would be needed to enforce the rule. Hence separating them keeps the model simple and clean.

[Assignment 6]
The following questions are listed in the online posted chapter file.
7.7
1. (Might be multiple solutions) Using Chinese Wall model (CW model) to emulate Bell-LaPadula model (BLP model), we need to define the security level first. Since CW model only has two levels: sanitized and unsanitized, we will be able to emulate two levels of BLP model.
Each CDI contains only on CD. A CD is defined as either sanitized, O} or {unsanitized, O}.
The security rule in BLP is S can read O if and only if S dom O. This can be simply achieve by CW model security condition, S can read O if there is an object O' such that S has accessed O' and CD(O')=CD(O). Here the security level is "equal". Similar as to "write", in BLP, S can write O if and only if O dom S, which can be achieved by CW-* property condition (1) S can read O, which is obvious. (2) S can read O' → CD(O')=CD(O). (2) tells us S can only write to a single
object O.

2. (Might be multiple solutions) Bonus points of this question.

3.
a. If organization i is to share one document with any other organization, there must be one compartment for each pair (i, j), where j= 1, …,k and j≠i. Thus there must be

$\binom{k-1}{2} = (k-1)(k-2)$ compartments.

b. Now there are m-tuples rather than pairs. This is equivalent to the number of ways to choose m out of k-1 items. Thus there must be

$\binom{k-1}{m} = \frac{(k-1)!}{m!(k-m-1)!}$ compartments.

4.
a. The primary advantage of roles in this situation is the enforcement of least privilege. Being in multiple groups simultaneously is similar to being in multiple roles simultaneously. That may allow one to perform an action in one group that affects functions of another group, much as the Windows Administrator user may mean to delete one user's files, but accidentally mistype a command and delete other users' files. Being in one role at a time prevents this.
b. The answer to this question depends on how the groups are organized. Roles are organized around job function. Groups may be organized around anything. Assuming the former, the groups in the system function as groups, because each user may be in at most one group, and the group selected depends on the functions that the user is to perform. This is exactly equivalent to a role. If the groups are not organized around job function, then the groups differ from roles because they do not restrict the particular job functionality of the user.

6. (There might be different presentations) Let r1, r2 ∈ R, authr1(s):{prescribe}, aurhr2(s):{purchase medicine}, trans(r):{dispense}, meauth(r){role cannot prescribe to him/herself}. Then we have: $(\forall r1,r2,\ r3 \in R)[r2 \in meauth(r1) \rightarrow [\forall s1,s2 \in S[r1 \in authr1(s1) \wedge r2 \in authr2(s2) \wedge t \in trans(r) \wedge canexec(s2,t) \rightarrow actr(s2)=\varphi]]]$.

14.8
2. In what follows, "r" represents the read right, "w" represents the write right, and "x" represents the execute right.
a. The access control list for file x is ((Alice, rw), (Bob, r)).
The access control list for file y is ((Alice, r), (Bob, rw)).
The access control list for file z is ((Alice, x)).
b. The capability list for user Alice is ((x, rw), (y, r), (z, x)).
The capability list for user Bob is ((x, r), (y, rw)).

4. If root can change the ACL (it happens in many Unix systems), the attacker can try to usurp the root privilege. Then the attacker can get access to what ever files she wants.

6. The user should give the editor process the minimum number of capabilities needed to perform the task. In this case, the capability to edit the file xyz is sufficient. As the editor has no capabilities for other files, it cannot access other files.
This is more difficult in an ACL-based system. The user can turn off access permission for herself on all files except xyzzy and then run the editor, but this requires that she own all other files (or be authorized to alter their ACL in some way). Further, as the editor process runs with her privileges, it can turn permissions back on unless she surrenders the ownership privilege.
So she cannot reduce the editor's protection domain without reducing hers. Hence she cannot be sure the editor cannot access any other file unless she ensures that she cannot.

8.
a. This occurs when p executes in a ring above, or equal to the top ring of, d's access bracket. As d's access bracket is (5, 6), this means p must execute in any of rings 0 to 5.
b. This occurs when p executes in a ring below the top ring in d's access bracket, and no greater than the ring at the bottom of d's access bracket. As d's access bracket is (5, 6), this means p must execute in ring 6.
c. This occurs when p executes in a ring above the access bracket of d. As d's access bracket is (5, 6), this means that p must execute in a ring with a number greater than 6.

The following questions are listed in our textbook
9.10
1. Requiring the desired account name makes it easy for system B to look just at that account name's .rhost file to see if access should be granted. Without the requirement, system B would have to search through all users' .rhost files to see if any allow access; alternatively it could compile the information in all the users' .rhost files to see if any allow access; alternatively it could compile the information in all the users' .rhost files into a table which lists, for each remote system an user, all local accounts that allow access by that remote system and user.

10.11
1. Take the password, group it into 64-bit blocks, $\oplus$ the blocks together, and take MD5 of the result.

[Assignment 7]
Chapter 9: Overview of Authentication Systems
9.2
No, Knowledge of the hash of Alice's password, which is stored in the server database, is sufficient to impersonate Alice's workstation to Bob.

Chapter 11: Security Handshake Pitfalls
11.1

File transfer normally requires the receiver to send various control information back to the sender. This information might include such things as acknowledgments, requests for retransmission of damaged or lost packets, and requests to suspend or resume transmission. Without this feedback, Trudy will have no idea if she was successful. She might send packets too fast so that some get dropped, or too slow, causing a protocol timeout. Some of her packets might be lost or damaged, and she won't know that they have to be transmitted.

11.3

$A \oplus R$ is not secure, because an eavesdropper who discovers the session key can instantly compute Alice and Bob's shared secret A.

$\{R + A\}_A$ and $\{R\}_{R+A}$ are secure.

$\{A\}_A$ is not secure, because it is the same for all sessions.

11.5

No. An eavesdropper can reply Alice's messages at any time. If Bob can't remember his current challenge, he won't know that the challenge response is to a previous challenge.

11.7

Any timestamp older that 10 minutes is considered to have expired, and will thus be deemed invalid.

11.9

In the 5th message, Bob received the ticket, which contains the session key K_{AB}, "Alice", and Bob's nonce N_B, all encrypted with K_{Bob}. The KDC sent the ticket to Alice in a message encrypted with K_{Alice}, so only Alice could have decrypted it to send to Bob.

11.11

Instead of transmitting a cleartext password, the phone should receive a challenge from the phone company and respond with the encrypted challenge. With public key technology, the phone encrypts the challenge with its private key and the phone company database contains only the public key, so a cloner gains nothing by stealing the database. With secret key technology, theft of the database enables the cloning of phones.

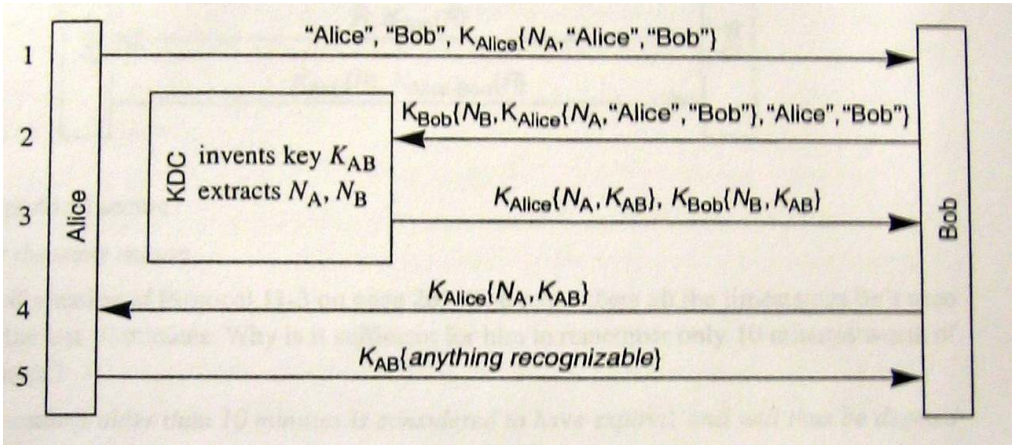
11.13

$MD5(K_{Alice\text{-}Bob} \vee R)$ is not secure, because Trudy, impersonating Bob, can send to Alice a sequence of challenges R that will reveal $K_{Alcie\text{-}Bob}$. First she sends the challenge 0 and notes Alice's response. Then she send challenges 1, 2, 4, 8, …, and she notes which responses are the same as that for 0. Those that are the same correspond to the one bits in $K_{Alice\text{-}Bob}$. The rest of the bits in $K_{Alice\text{-}Bob}$ are zero. (If R is shorter that $K_{Alice\text{-}Bob}$, not all the bits will be revealed by this technique, but the key will still be compromised.)

$MD5(K_{Alice\text{-}Bob} \oplus R)$ is secure, since MD5 conceals its argument and $\oplus$ merely flips bits.

[Assignment 8]
Chapter 11
11.2
Trudy can't impersonate Alice since Alice must prove knowledge of the secret before Bob will encrypt anything. In an unlikely scenario, Trudy might be able to impersonate Bob. If Alice is opening multiple connections to Bob in parallel, and tries a second before the first times out, Trudy can trick Alice into encrypting her own challenge.

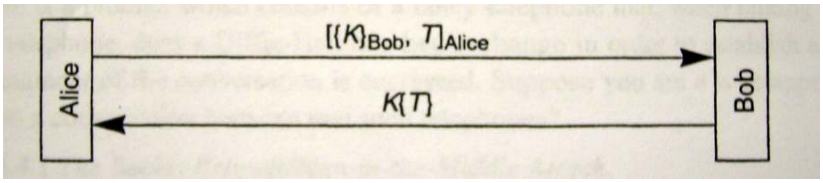
11.4
Essentially, we replace N_C by $K_{Alice}\{N_A, \text{"Alice"}, \text{"Bob"}\}$:



The KDC can check that Bob is making the request, that Alice made the request that Bob is forwarding inside his request, and that Alice and Bob want to talk to each other. The KDC extracts Alice's nonce to include in Alice's ticket and Bob's nonce to include in Bob's ticket. The respective nonces in the tickets assure Alice and Bob that their tickets are freshly created by the KDC.

11.6
No. An eavesdropper can replay Alice's messages at any time. If Bob can't remember his current challenge, he won't know that the challenge response is to a previous challenge.

11.8
Alice picks a session key K and sends along a timestamp. She encrypts K with Bob's public key and signs the entire message. Bob responds with the timestamp encrypted with K:



Bob knows it's Alice from the signature and timestamp. Alice knows it's Bob because only he can decrypt K.

11.10
In Protocol 11-18 (Needham-Schroeder), N_1 must be unpredictable so that Trudy can't impersonate the KDC by giving Alice and old ticket to Bob with a key that Trudy had stolen earlier.
In Protocol 11-19 (Expanded Needham- Schroeder) and Protocol 11-21 (Kerberos), N_1 still must be unpredictable.
In Protocol 11-20 (Otway-Rees), N_C must be unpredictable, as described in the text.

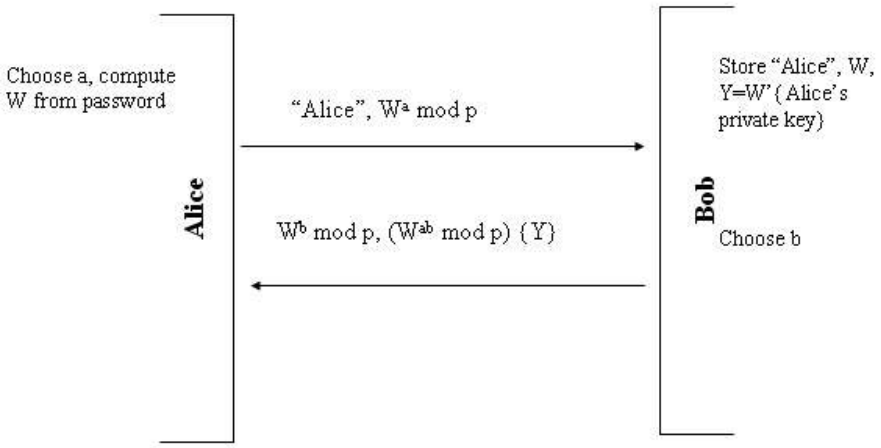
11.12
See §6.4.1 The Bucket Brigade/Man-in-the-Middle Attack.

Chapter 12
12.1
The Lamport Hash value changes for each login session, and it is computationally infeasible to compute future hash values from past hash values.

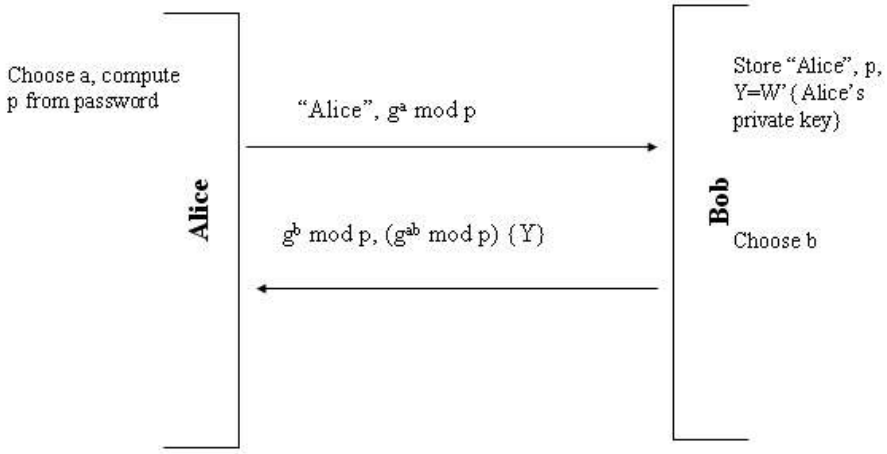
12.5
Bob can store another quantity n', which is the next n to send to Alice. Normally n and n' are the same. During authentication, Bob reads n', decrements the stored n', and then sends the value read to Alice. When Bob receives a response from Alice, he applies $hash^{n-n'}$ to it and compares the result to the stored hashn. If it matches, he atomically modifies the database, setting n to n' and replacing the stored $hash^n$ with Alice's response.

12.9
Only Bob knows the right modulus p associated with Alice, so only Bob can compute b and 2^b mod p correctly. An impostor could not thus compute the correct value of the first hash, and Alice would detect this when she tried to verify it. Likewise, since Alice's reply to Bob requires that she knows W, an attacker who stole Bob's database still would not be able to compute Alice's response hash correctly.

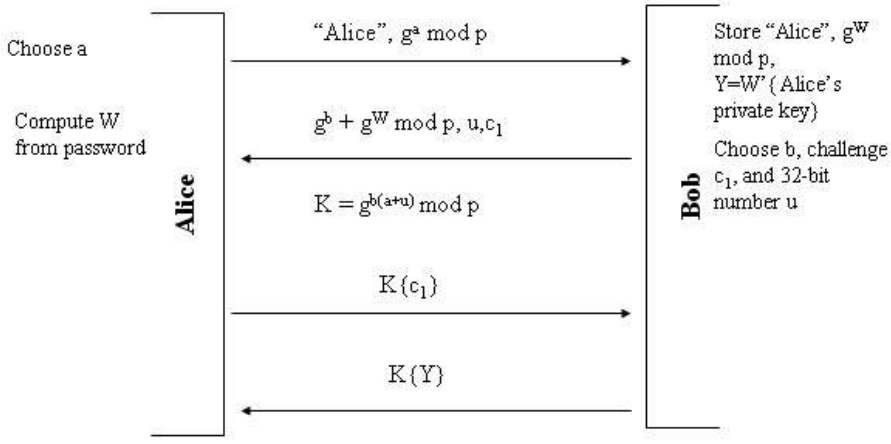
12.13



Credential download protocol based on SKEKE



Credential download protocol based on PDM



Credential download protocol based on SRP

[Assignment 9]

Chapter 13:
13.1
Let Alice be the user. In this variant, Alice's TGT is just K_{Alice}("Alice", SA}, where Alice (actually, her workstation) has invented the session key S_A. This causes a problem for the KDC when it wants to get the session key from the TGT. The KDC has to try user keys until it finds the correct key to decrypt the TGT. To avoid this problem, Alice's name/instance/realm (unencrypted) should always be paired with the TGT when transmitted (it is already included in the encrypted TGT.)

Since the only purpose of the TGT is to allow use of the short-term session key S_A instead of the long-term master key K_{Alice} when Alice and the KDC talk, and since knowledge of K_{Alice} is all that is required for mutual authentication of Alice and the KDC, there is no difference in security between the two schemes except when Alice changes her password (and thus her key K_{Alice}) during a session. If Alice believes someone has learned her password and therefore changes it, with normal Kerberos the TGT obtained by the impostor remains valid until expiration. With the modified scheme, the TGT would be immediately invalidated, though tickets could continue to be used until expiration.

In normal Kerberos, Alice can change her password (and thus her key K_{Alice}) during a session, because S_A is used for encryption, and S_A is in the TGT encrypted with K_{KDC} (so the TGT remains valid). In the variant, changing K_{Alice} invalidates the TGT, so to change her password, Alice would have to send a message containing the old TGT, and the new K_{Alice} encrypted with the unchanged SA. She would then compute a new TGT for herself.

Chapter 14:
14.2
This scheme requires the TGS to continually check tickets once they get sufficiently older than their start-time, rather than the one-time check and conditional reissue allowed with renewable tickets. On the other hand, if a ticket is issued with a future start-time in this scheme, it won't be checked if it is used shortly after its start-time; while in Kerberos V5 a postdated ticket starts off invalid and must be validated before it can be used at all.

So this scheme is inferior both in terms of efficiency and security.

Chapter 15:
15.4
Downloading Bob's key from whatever responds to what is believed to be his IP address. If someone can intercept traffic sent to Bob's IP address, they can give Alice (the requester) the wrong public key. However, the advantage of this scheme is that in order for Alice and Bob to communicate, only Alice and Bob need to be available-no directory,

no KDC, etc. This is computationally cheap since there is no authentication required to obtain Bob's public key.

Looking up Bob's key in a directory via an unauthenticated interaction. If someone breaks into the directory, they can install incorrect information. They can also impersonate the directory through various means, and five out faulty information, even if they can't actually break into the directory. This is computationally cheap since there is no authentication required to obtain Bob's public key.

Authenticated interaction with the directory. Now nobody can impersonate the directory, but the directory is an on-line trusted thing, so if someone were to compromise the directory, they could subvert security.

Having the directory sign the information you request. Similar security properties to above (authenticated interaction with the directory), but it means that Bob can obtain signed information from the directory and send it to Alice, rather than having everyone that wants to talk to Bob have to carry on an authenticated exchange with the directory.

Storing and retrieving certs from the directory. This is more secure because the trusted party can be off-line. Compromising the directory can be at worst a denial of service.

Having each principal keep its own certificate. This might make chain-building difficult, unless there were a globally agreed-upon root, in which case everyone could keep the complete chain from the root to themselves. It is less flexible, since Bob has to be available for Alice to obtain his cert before she can do something like compose an encrypted message for him.

[Assignment 10]
Chapter 16
16.1

- Protocol 16-2

PFS, escrow foilage against passive attacks, escrow foilage against active attacks (unless signature key is escrowed), no identity hiding.

- A modified form of Protocol 16-2 in which the first two messages are encrypted with the other end's public key rather than signed by the transmitter's private signature key. So in message 1 Alice sends {"Alice", g^a mod p} encrypted with Bob's public key, and Bob in message 2 sends {"Bob", g^b mod p} encrypted with Alice's public key.

PFS, escrow foilage against passive attacks, no escrow foilage for active attacks, identity hiding, no PFS for identity hiding.

- Protocol 16-4.

PFS, escrow foilage against passive and active attacks (assuming signature key not escrowed), identity hiding, PFS for identity hiding, active attacker can discover Alice's identity.

- Protocol 16-9, where Alice and Bob share a secret key S.

PFS, escrow foilage against passive attacks, no escrow foilage against active attacks, no identity hiding.

- Each side sends a nonce encrypted with other's public encryption key, resulting key is ⊕ of two nonces.

No PFS, no escrow foilage, no identity hiding.

- Assume Alice and Bob share a secret S. Design a protocol in which they can do mutual authentication and establish a shared secret with PFS. Can it be done without Diffie-Hellman or any other form of public key cryptography?

The protocol above ("protocol for Homework Problem 1") will work. And it cannot be done without Diffie-Hellman or some other means of PFS. A method of using RSA keys is to have one side create an ephemeral key pair, and send the public key to the other side, integrity protected with the shared secret key, and have the session secret (with which they'd mutually authenticate) be a function of a random number sent encrypted with the ephemeral public key and the shared secret key.

- Protocol 16-2, but with each side deterministically generating the Diffie-Hellman private numbers as described in §16.4 PFS-Foilage from a seed given to the client machine and escrowed at the server machine.

No PFS, no escrow foilage, no identity hiding.

16.5

Have Alice send her name, her public encryption key cert, and her Diffie-Hellman value encrypted with Bob's public key. Show such a protocol.

16.11

Bob knows it's the real Alice because she knows the key K, which is a function of the nonce as well as a Diffie-Hellman value signed by Alice. If someone had stolen a previous "a" and the signed g^a mod p, then they can impersonate Alice to Bob, however. Hopefully it is as unlikely someone can steal an a as stealing Alice's private key. If Alice uses a different a each time, then she knows it's the real Bob, and not someone replaying Bob's messages. But if she reuses a, then she can't tell.

To modify the protocol so she can tell, and tell, and allow both Alice and Bob to reuse their Diffie-Hellman values (to save computation), Alice sends a nonce in message 1, and have K be a function of both nonces as well as the Diffie-Hellman value.

16.15

If it is more convenient for one party to save state than the other, either party can encrypt and integrity protect the state it would have liked to keep and forward that information to

the other party. The encryption can be done with a secret key known only to a single party. For example, if it is more convenient for Alice to maintain state than Bob, then once Alice and Bob establish a shared secret key, Bob can encrypt that key with the name Alice and send the blob to Alice. When Alice wants to reauthenticate, if she remembers the shared secret key and the blob from Bob, she can the blob back to Bob and do a more efficient secret key based authentication. If Bob can remember state more conveniently, Alice could encrypt the shared secret and the name Bob under a key known only to Alice and have Bob store and return it.

Chapter 17

17.2

There is a lot of state that needs to be kept; the IP and perhaps layer 4 port mapping, and TCP sequence number modification (because of the ASCII transmission of the IP address inside the packet). So various solutions:

- Pick one of the NATs to be the exit point, and have the other NAT box only be a backup in case the first fails. If the backup NAT receives a packet, it tunnels it to the other one.
- As before, but have the main NAT box inform the backup NAT box of state, and allow the backup NAT box to forward packets for flows for which it has been informed of the state. For other packets, it forwards to the main NAT box.
- Hope that packets will tend to always go with the same NAT box, and have either NAT box create the state for that flow and share it with the other NAT box. In case box handle packets for the same flow before they get a chance to synchronize, that connection will break, but hopefully this will be a low-probability event.

17.5

Suppose one portion of your Intranet is connected to the Internet with firewall F1, and another portion of your Intranet is connected with firewalls F2 and F3. All addresses inside that portion are reachable equally well through F2 and F3. Since SAs are pairewise, F1 will have two SA's: one to F2, and one to F3. When F1 forwards a packet for destination D, it has to choose which SA to send it on, encrypting the packet with the key for the F1-F2 SA or with the key for the F1-F3 SA. Internet routing can route packets for D via either F2 or F3. If it chooses a different F than F1 assumed, then it will not work. So F1 has to specify which of the F's the Internet should deliver the packet to.

[Assignment 11]

Chapter 18:

2.

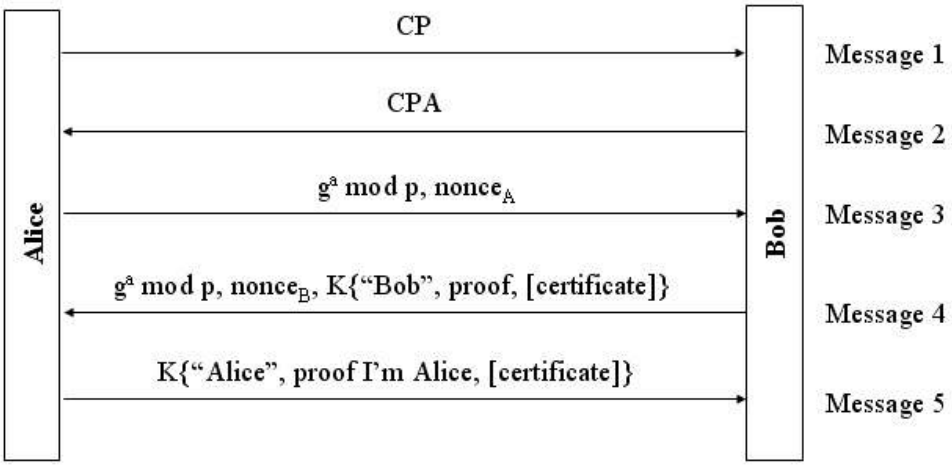
Preshared keys: better performance, easier to type in if configuration is done by typing the value into the two end nodes, but harder to configure than public keys because each pair of communication entities results in a key to be configured.

Public Signature keys: less likely to be escrowed, more likely to exist in an export situation than public encryption keys. Doesn't require knowing the other side's public key in advance.
Public encryption keys: Can hide both identities from active attackers. Another reason stated in the IKE RFC is that even if the Diffie-Hellman group is broken, it will be secure provided that the RSA is not broken. We are unenthusiastic about that reasoning because there's no reason to pick a Diffie-Hellman group with inadequate strength.

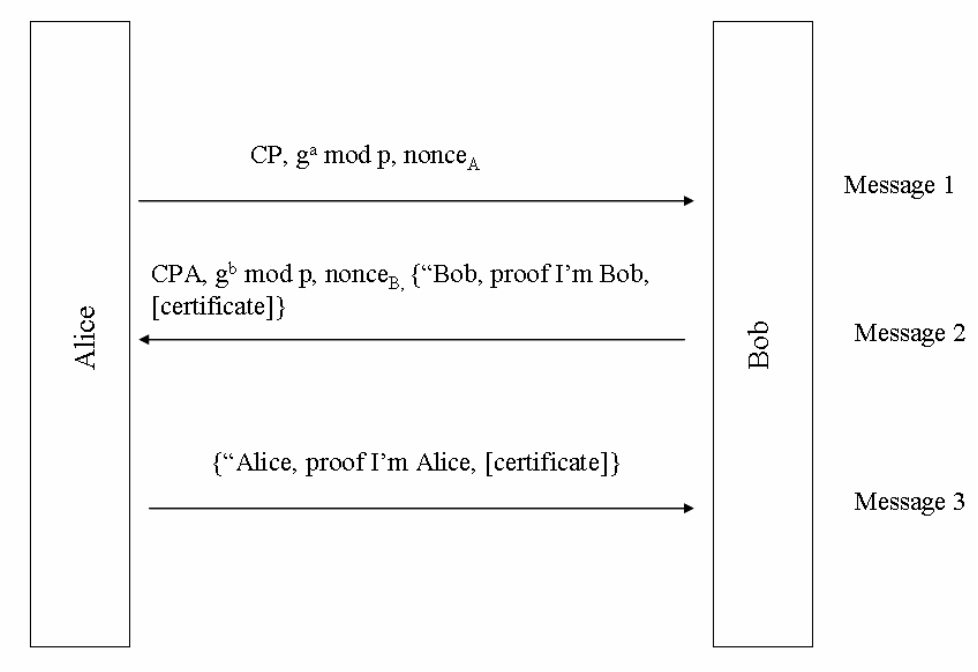
3.
Let Trudy be the person attempting to construct an IKE exchange that looks like it's between Alice and Bob. For all IKE public encryption variants, the following applies. Trudy chooses Diffie-Hellman numbers a and b for each side, and nonces for each side. The proof of identity is a function of the other side's nonce (which ordinarily would require knowledge of one's private key since it is transmitted encrypted with the public key, but in this case Trudy has chosen the nonce and therefore knows it), the Diffie-Hellman values, and the cookies, all of which Trudy knows. Trudy can also compute the session keys, since she knows all the inputs, including g^{ab} mod p, since she knows both a and b.

4.

This variant requires Bob to compute K, which involves computing g^{ab} mod p, before he can send message 4. And Alice can't start computing g^{ab} mod p until she receives message 4. So Bob has to compute between 3 and 4, and Alice has to compute between messages 4 and 5, whereas in the original 6-message version, Bob can send message 4 and then compute g^{ab} mod p. If the computation takes longer than the transmission, then Bob's computation of g^{ab} mod p won't slow anything down, since he'll be computing it while Alice is computing it.

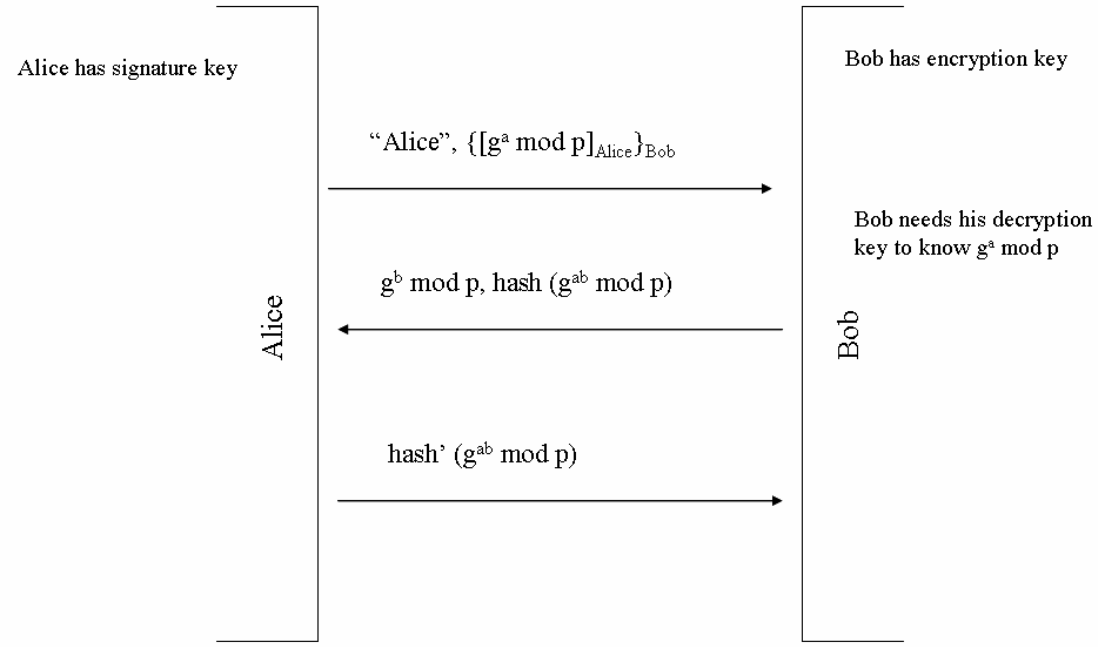


Shortened form of public signature keys, main mode



7.

In the following, Alice sends her Diffie-Hellman value signed with her signature key and encrypted with Bob's public key.



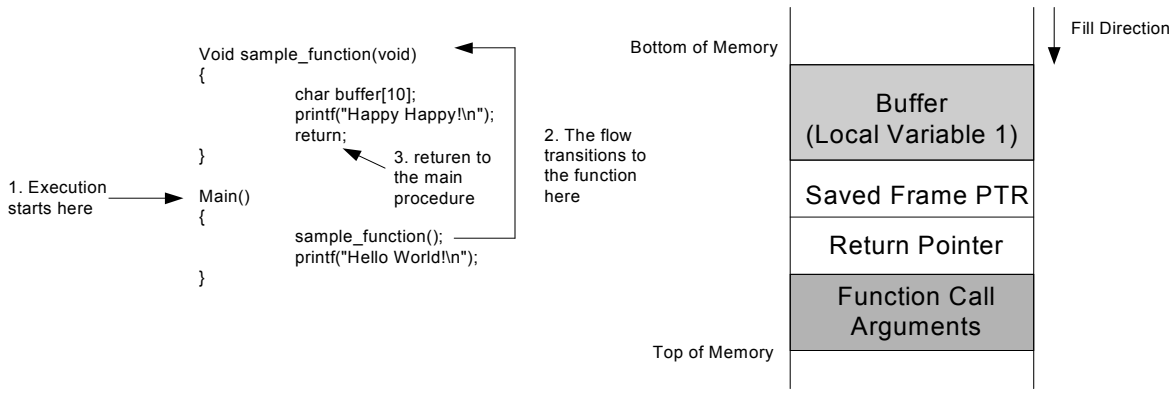
Chapter 19:

2.

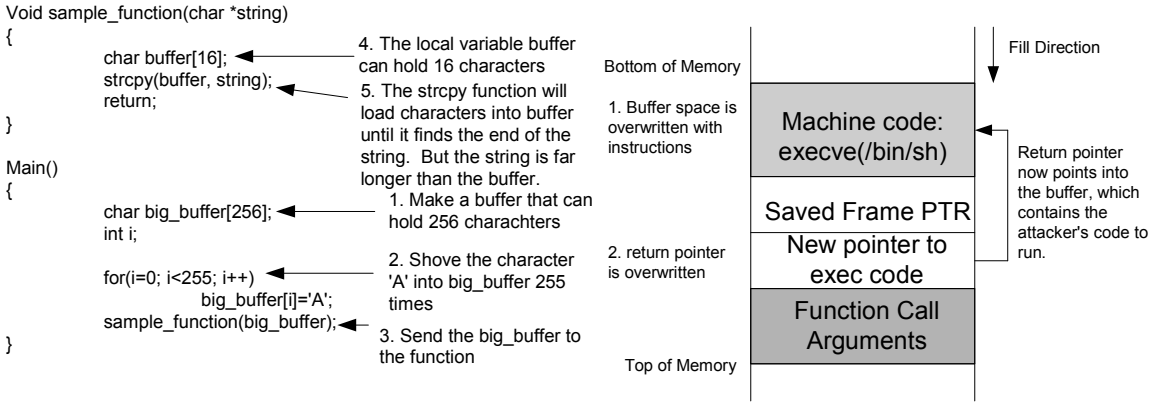
PFS, SSLv3, as implemented: client does a cheap RSA encrypt, and a cheap RSA verify signature (cheap because both use the public key). The server has an expensive decrypt and an expensive sign on every authentication.
With the recommended modification: the server only has to do the expensive sign once an hour or so. The disadvantage of the modification is that it requires putting in an expiration time, and therefore requires relatively synchronized clocks.
With Diffie-Hellman, each side has to do an expensive Diffie-Hellman exponentiation, and in addition the server would have to sign its Diffie-Hellman number on each interaction. If PFS is less than perfect, the server can reuse its Diffie-Hellman b value, and not have to keep re-signing that.

Solutions for additional questions:

1. Typically, transport mode is used for end-to-end communication between two hosts, for example, between a workstation and a server, or between two servers. Tunnel mode encapsulates an entire IP packet within an IP packet to ensure that no part of the original packet is changed as it is moved through a network. The entire original, or inner, packet travels through a tunnel from one point of an IP network to another. No routers along the way need to examine the inner IP header. Tunnel mode is useful in a configuration that includes a firewall or other sort of security gateway that protects a trusted network from external networks. Encryption occurs only between an external host and the security gateway or between two security gateways. This setup relieves hosts on the internal network of the processing burden of encryption and simplifies the key distribution task by reducing the number of needed keys
2. In transport mode, NAT may need to modify port numbers in TCP/UDP and TCP/UDP packets may be encrypted. Thus, ESP authentication would break if NAT changes ports. However, the ESP tunnel mode can still work. Two methods are used to enable ESP in both transport mode and tunnel mode. (a) NAT before ESP is applied at the sender side and ESP before NAT at the receiver side; (b) encapsulate ESP within UDP packet (why UDP?).
3. The examples of stack buffer overflow may vary. We only show one example as follows. A stack is a data structure that stores important information for processes running on a computer. The system writes down important little notes for it to remember and places these notes on the stack, a special reserved area in memory. Stacks are similar to stacks of dishes, in that they behave in a Last-in, First-Out manner (LIFO). Stacks are used to store information associated with function calls on the computer. Function calls are used by programmers to break code down into smaller pieces. A simple function, and the context switching operation in stack are shown as follows:



The stack buffer overflow example is show as follows:



[Assignment 12]
Chapter 20:
2. Because then the MD on the message only needs to be computed once. This would also apply if the integrity check is a keyed MD, provided that a new per-message secret S is chosen, the keyed MD computed using S, and then S distributed to each recipient by encrypting a copy of S with the secret key of each recipient.

4. Once Bob knows $[\{S\}_{Bob}]_{Alice}$, he can find S, and even prove to Charlie that Alice sent him S. But knowing S, he can forge any message he likes, because he can compute the integrity code based on S. The signature my Alice is needed to prove that S came from Alice. The encryption with Bob's public key is needed to hide S from anyone other than Bob.

6. A message from Alice to someone else, say Carol, has the secret key encrypted with Carol's public key and then signed by Alice. Bob can't forge Alice's signature, nor can he decrypt with Carol's key. He can pick an S and encrypt it with Carol's key, but then he can't forge Alice's signature on the result. Alternatively, he could pick something arbitrary as if it were signed by Alice and then encrypt it with Alice's public key, but then he couldn't decrypt with Carol's key to get the secret key that Carol would see.

8. Privacy can be provided by encrypting with public or secret key technology. Authentication can be provided with public key signatures or shared secret keys. Integrity can be provided using message digests suitably authenticated. Non-repudiation can be provided with public key signatures or notarization. Repudiability can be provided by shared secret key encryption or by the scheme of 12.8.2. Plausible Deniability Based on

Public key Technology. Roof of submission or delivery requires help from the mail delivery infrastructure, as does containment. Message flow confidentiality and anonymity can be provided with the help of cooperating third parties.

10. Let x be the expiration date of the certificate, r the date on the CRL that doesn't contain the certificate, and n the date of notarization of the purchase order. To prove the purchase order is legitimate, $n<\min(r,x)$, that is, the purchase order was notarized while the buyer's certificate was valid. The judge may also want to see that the certificate and CRL have notarization dates that are not long after r. (The certificate can be notarized with the purchase order, but the CRL must be notarized separately, since $r>n$.)

Chapter 26:

1. We know that $c_2=c_5$. Therefore, $c_1 \oplus m_2$ equals $c_4 \oplus m_5$, since each result, encrypted with the key yields the same value (c_2 and c_5, respectively, which are equal). Therefore, if m_2 were known, then since c is also known (the ciphertext is assumed to be visible to the attacker), then $c_1 \oplus m_2$ is now known, which equals $c_4 \oplus m_5$, so since c_4 is known and $c_4 \oplus m_5$ is known, m_5 can be computed.

An alternative explanation that takes advantage of the hint is that since $c_1 \oplus m_2$ equals $c_4 \oplus m_5$, then $c_1 \oplus c_4$ equals $m_2 \oplus m_5$, so since c_1, c_4, and m_2 are known, then m_5 can be computed.

2. Assuming the attacker sees a signed, encrypted message with recognizable plaintext so that brute force attack is possible, it takes 2^{40} operations to break the encryption key, and then 40 of the 64 bits of the integrity protection key are known. Then it only takes 2^{24} operations to search all possible integrity protection keys. So the workfactor is only 2^{40}.

3. It is OK for the challenge R to be predictable, as long as R is not reused. The attack we'd like to avoid is for an attacker to impersonate Bob and send Alice a value that Bob will send in the future, and trick Alice into giving an answer to that challenge. But since the challenge is sent encrypted with the shared key, even if the attacker can guess R, he cannot predict a value of $K_{\text{Alice-Bob}}\{R\}$ that Bob will transmit in the future.