CS370/ECE499 November 22, 2016

Intro to Computer Security

Problem Set 3: Public-Key Cryptography, Key Exchange Protocols and Access Control

1) Digital Signatures
   a) What is a digital signature? What security properties does it provide?
      A digital signature is a mathematical scheme for demonstrating the authenticity of a digital message or documents.
      Properties:
      - Origin authenticity
      - Message Integrity
      - Non-Repudiation
   b) Do digital signatures and MACs increase the length of message to be transmitted? Explain Why?
      Yes, because along with the message the Digital Signature or MAC needs to be sent. Even in cases such as direct application of the RSA where the message is carried as part of the signature (message recovering) there is some expansion due to padding.

   c) Using the notation from the class, show how a message m is signed with an RSA key-pair (N, d, e).
      Sig(m) = (m)^d mod N ; Assuming d is private exponent and e is public exponent and that no hashing is used.

      Sig(m) = (h(m))^d mod N; when hashing is used

   d) Does the hash function used in an RSA signature need to be a keyed hash function? Why or why not?
      No, because the signature cannot be created without the private key so no extra security is added by using a keyed hash function.

   e) When encrypting and signing a message m, does the order of encryption and signature operations matter? Explain.
      Yes, the order of the encryption and signature operations matter.

      Sign & then Encrypt operation is vulnerable to surreptitious forwarding when Alice can decrypt the message sent to her by Bob, re-encrypt it to someone else making them believe that Bob sent that message to them.

      Similarly, Encrypt & the Sign operation cannot protect the ownership of the message as Alice can remove the signature from the message sent to her and forward it to someone else claiming the ownership of the encrypted ciphertext.

2) Time-Variant Parameters.
   a) What is the role of R1 and R2 (or N_A and N_B in Handbook of Applied Cryptography) in Needham Schroeder Protocol? What properties should R1 and R2 have?

R1 and R2 are Nonces used for detecting replay and in this case are random numbers. Thus they have to be unpredictable (otherwise adversary can guess them), fresh - not used in the recent past, should be large enough in size to minimize probability of collisions.

b) Why does Alice have to send r2-1 in the last message of Needham-Schroeder? Can she have not sent r3 instead?
Sending r2-1 proves that Alice was able to access the r2 sent in previous message which was enciphered with a key only Alices is supposed to have. This confirms that Bob is indeed talking to Alice and that she has the same session key as Bob.

c) Can a timestamp be used instead on R1? What is the advantage and disadvantage of using one over the other (i.e., R1 or N_A vs. timestamps) in security protocols? (Hint: see the discussion on this in Handbook of Applied Cryptography – 10.3.1).
Yes, timestamps can be used. Indeed Kerberos protocol uses timestamps. Use of Timestamps requires time synchronization among participating entities (server, Alice and Bob). Use of random numbers means time synchronization is not needed but requires some storage or state to track previously used nonces so a replay can be detected.

3) Long Lived and Session Keys
   a) What are pseudorandom numbers? Why are they used?
   Pseudorandom numbers are a sequence of numbers whose properties approximate the properties of random numbers and can be generated using pseudo-random number generation algorithms.
   They are used in cryptographic operations, simulations etc.
   b) What is the difference between session keys and interchange keys? Why are session keys needed? Do we need session keys when there is a shared symmetric interchange key between two-parties or are they only needed when using asymmetric cryptography?
   Session keys are used to actually encrypt or protect the message while the interchange keys are used for protecting the exchange of session keys.

   All keys have a “cryptoperiod” in practice - “duration” during which a key can be used. This duration is either expressed in time units or in the number of bytes or messages to be encrypted/hashed etc.

   Using session keys extends the life or cryptoperiod of interchange keys (hence long-lived keys) as they are only used to encrypt keys which are much smaller than messages in practice. This reduces the cost of establishing a new/fresh key between entities like Alice and Bob.

Session keys also provide other benefits like -- backward and forward security. That is compromise of old keys doesn't impact the security of current sessions and vice versa.

c) Why is a trusted-third party desirable/needed for key-exchange? Is such an entity only desirable/needed when using symmetric keys or is such an entity also desirable/needed when using asymmetric keys as well?

Without a trusted third party, key exchange would become an O(N^2) operation for establishing keys among N parties. Having a trusted-third party to mediate key-exchange reduces this problem to O(N) as N parties need to set up a long-term key with only the third-party.

Such an entity is also needed in public-key settings. For example, to provide an authentic channel to distribute public-keys (authenticated directory service) or to bind public-keys with identities (Certificate Authorities). In the latter case the third-party doesn't have to be online to facilitate key-exchange.

4) Access Contol Concepts Concepts

a) The three most important components in access control, all starting with the letter 'A', are what?

Authentication: Verification that an entity is who it claims to be.

Authorization(from book): Granting of rights or permissions over an object to an entity.

(Authorization Check is to check that an entity is authorized to access an object before allowing them to access it; Authorization is often defined this way)

Audit/Accounting: Maintaining an audit trail (by logging all security relevant actions) so user can be held accountable for their actions

What is the primary difference between DAC and MAC access model?

- *Discretionary Access Control (DAC)*
  - *Access control is at the discretion of the user*
  - *Normal users can change access control state directly assuming they have appropriate permissions*
  - *E.g.: Access control implemented in standard OS's, e.g., Unix, Linux, Windows*
  - *typically identity based, but RBAC can also be used in DAC mode*
- *Mandatory Access Control (MAC)*
  - *Access decisions cannot be changed by normal users*
  - *Generally enforced by system wide set of rules*
  - *E.g: SELinux, Windows Vista Integrity Levels*
  - *typically uses security levels or labels.*

a) In access control, what does an "open policy" mean? What does a "closed policy" mean?

A system with an open policy defaults to allowing (open) access unless there is an explicit negative (deny) authorization.

A system with a closed policy defaults to denying (closed) access unless there is an explicit positive (allow) authorization.

b) What is the difference between a "role" in RBAC and a "group" commonly used in UNIX?
A role is a set of permissions needed for/associated with a job function; has relevance to or semantic meaning for the organization

Groups in Unix on the other hand are simply groupings of users. However, they can be created to have a semantic meaning and can be used to implement a basic form of RBAC

c) Explain the difference between Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC).

In RBAC access policies are defined in terms of roles – that is roles are assigned permissions, and users are given roles.

In ABAC, access policies are defined in terms of attributes of user, object and environment. A role can be used as an attribute in ABAC. Attributes can be used to define roles. Hybrids of RBAC-ABAC are a current research topic.

2. Access Control Matrix

Consider the following scenario. An organization employs product managers, programmers and testers. The organization operates with the following kinds of files: development code and executables, testing code and executables, test reports, and production code and executables.

Product Managers can view, and execute the development executables and production executables to verify correctness. Programmers can create, edit, delete, and execute development code and executables. Programmers can also promote development code to the test level. Testers can edit, delete, and execute test code and executables. The testers write test reports that can be read by everyone. The testers can promote test code to production level or demote it back to development. Everyone can view and execute production code and executables.

Eve is the product manager, Alice and Bob are programmers. Carol and Dave are testers

a) Define the rights the access control system would need to enforce the requirements for this scenario. Associate the abbreviation you will use in parts (b) and (c) with the definition.

One way to do this is as follows. Other variants are possible and are fine as long as they capture the scenario above accurately.

Read (R) – ability to view or read

Create (C) – ability to create object

Write (W) – ability to edit or write to an object

Delete (D) – ability to delete an object

Execute (X) – ability execute an object

Promote/Demote (P) – ability to change the classification of an object

b) Design an access control matrix for the scenario for the users.

	Dev. Code	*Dev. Exec.*	*Prod. Code*	*Prod. Exec.*	*Test Code*	*Test Exec.*	*Test Reports*
Eve		*RX*	*RX*	*RX*			*R*
Alice	*CWDXP*	*CWDX*	*RX*	*RX*			*R*
Bob	*CWDXP*	*CWDX*	*RX*	*RX*			*R*
Carol			*RX*	*RX*	*WDXP*	*CWDX*	*WR*
Dave			*RX*	*RX*	*WDXP*	*CWDX*	*WR*

c) Assume the Access Matrix is being implemented by a system using Access Control Lists. Write the Access Control List for the Development Executables.

ACL(Dev. Exec.) = [(Alice, {CWDX}), (Bob, {CWDX}), (Eve, {RX})]; Pictorial representation is fine as well.

d) Assume the Access Matrix is being implemented by a Capability system. Write the Capability list for Alice.

CAPABILITY(Alice) = [(Dev. Code, {CWDXP}), (Dev. Exec., {CWDX}), (Prod. Code, {RX}), (Prod. Exec., {RX}), (Test Reports, {R})]

3. UNIX Permissions

When a file in Unix is protected with mode “644” and is inside a directory with mode “730” describe ways in which the file can be compromised?

A file with 644 means rw for owner, and only r for group and world; so technically a member of the group associated with the file should be able to read but not write/modify

Directory with 730 means rwx for owner, wx for group and none for world; this means a member of the group associated with the directory can delete, rename or create (has wx on directory) files in the directory.

So if the groups associated with the file and directory are the same, then a member of the group can delete the file in question and create a new file with the same name but different content, which is effectively modifying the contents of the file in question (albeit technically it is a different file, with a different owner etc.). Having a sticky bit set on this directory will prevent this file being deleted by anyone other than the owner

4. RBAC

a) Would the access control for the scenario in Q2 above benefit from being implemented in a RBAC system? If yes, explain why and create access matrices that define an RBAC that would enforce this scenario? If not, describe why not and present another scenario that would be better defined as an RBAC system rather than a straight DAC.

Yes, permissions are all based on job functions and so this is a good fit for RBAC. Looking at the ACM also makes it clear that it is a good scenario for RBAC (rows with same permissions)

	Dev. Code	*Dev. Exec.*	*Prod. Code*	*Prod. Exec.*	*Test Code*	*Test Exec.*	*Test Reports*
Product Manager		*RX*	*RX*	*RX*			*R*
Programmer	*CWDXP*	*CWDX*	*RX*	*RX*			*R*
Tester			*RX*	*RX*	*WDXP*	*CWDX*	*WR*

	Product Manager	*Programmer*	*Tester*
Eve			
Alice			
Bob			
Carol			
Dave			

b) A company has 20 job functions. On average there are 200 employees in each job function. Similarly, on average an employee in each job function needs 1500 permissions to properly execute their task. Compare the number of assignments that need to be managed i) when using a DAC model vs. ii) when using RBAC model. Generalize the comparison to when the number of job functions is N, number of employees per job function is $U_{i,}$ where i indexes the job-function, and the number of permissions required per job function is $P_{i.}$

*i) DAC Model: 20 * 200 * 1500 = 6Million*

*ii) RBAC: 20*1500 + 200 *20 = 20*(1500+200) = 34000*

$\sum_{i=1}^{N} U_i * P_i$ *for DAC*

vs

$\sum_{i=1}^{N} U_i + P_i$ *for RBAC*

c) Consider a hierarchical Role-Based Access Control (RBAC) system where a role **Manager** inherits from a role **Clerk**. A **Manager** is permitted to perform operations *Review* and *Approve* on resource ***Report***, and a **Clerk** is permitted to perform operation *Edit* on resource ***Report***. User Alice is assigned to role **Manager**, and user Bob is assigned to role **Clerk**. For each statement below circle T if the statement is always true, and F if it can ever be false.

a. (T / F) Bob is necessarily also assigned to the role Manager.
b. (T / F) Alice necessarily has privileges to Edit.
c. (T / F) An RBAC session can have both Alice and Bob associated with it

4. Changing Access Control Matrix

	File 1	File 2	File 3	File 4	Subject A	Subject B	Subject C
Subject A	Own R W		Own R W		Control		Own
Subject B	R	Own R W	W	R*		Control	
Subject C	R W	R		Own R W			Control

Keeping in mind the rules governing access control matrix change covered in Section 4.3 (and discussed in class), and the access matrix shown above, answer whether or not the following changes to access matrix are allowed. **Explain in one sentence why or why not.**

a. (allowed / not allowed) Subject C wants to Transfer R on File 2 to Subject A

C is not owner of File 2 and it doesn't have the grant option on R for File 2 (no R*)

b. (allowed / not allowed) Subject A wants to Delete R on File 2 from Subject C

A owns C so he can delete R on File 2 from C even though he is not the owner of File 2.