**Nonce-** It is included in a key exchange algorithm to defend against replay attacks, It is often a random or a psuedo-random number, It must contain enough bits to be distinguishable from all other nonces recently used.

**S/Key Prot:** seed = 2; first pass = 3; H(2) = 5, h(5) = 1, h(1) = 4, h(4) = 0,h(0) = 3, h(3) = 6, h(6) = 2 => **hash chain** 2>5>1>4>0>3>6 **pass chain** 6>**\*3**>0>4>\***1**>5>2

**on receive; the server**: computes the hash of the returned password and admits the user if the hashed value is equal to the last correct password returned by that user. **Two Key Alice-Bob gen by alice no one break confidentiality** EK+B (EKA (m))

**ACLs vs Capabilities -** ACLs are a slice-by-column instantiation of an access control matrix, whereas Capabilities are a slice-by-row instantiation of an access control matrix

**Per-Subject-Rev** - is reviewing all the accesses a given subject has in a system. It is easy to undertake per-subject review with Capabilities as capabilities capture the rows or an access control matrix, that is, all the access that a given subject is allowed.

**SetUserID/SetGroupID -** one of the 3 special protection bits in UNIX file system. When set on a file it indicates that when the file is executed by any authorized user that executing process be run with the privileges of the owning user. **Advantage**: Programs needing privileged access can be run by users without having to give them all privileged access. **Disadvantage**: Possible vector for privilege escalation - if a program owned by root that has SetUID is compromised say through buffer overflow then the adversary gets root privileges.

**Stick bit-**When applied only the owner of any file can rename, move, or delete that file. **UNIX – Owner, group, other**

**Discretionary –** System Users ultimate source for defining access policy; need access at MAC and DAC(both) levels in order to view files.

**Bell-Lapadula –** Security levels, Top Secret> secret > confidential > restricted > unclassified; No read up; can only read obj of less or equal security. A subject can only “append” (write-only) into an object of greater or equal security level, and read equal or below.

**\*-Property in BLP** – no-write-down policy, user not allowed to write to lower class, only upper, used to prevent info leakage to lower levels.

**DAC under MAC BLP –** User doesn’t access if mac doesn’t allow even if dac allows, and vice versa.  
**Mutual Auth –** multiple entities authenticate each other **Multi-factor Auth –** multi factors of one individual(pass, phone)

**Dig Sig -** message digest created using public-key (or asymmetric key) cryptography that provides message integrity, origin authenticity, and non-repudiation. **Increase length? -** Yes, because the signature or MAC has to be transmitted along with the original message. Typically, signature or MAC alone cannot be used to recover the message. **RSA-Key pair (N,d,e)** - Signature = {h(m)}^d mod N

**RSA sig Keyed hash? -** No it doesn’t have to be as the hash is not available to the adversary in the open - enciphered with private-key. Using a keyed-hash will mean that the users will have to share a symmetric along with the private key which takes away from the advantages of public-key crypto. When using a cryptographic hash anyone who knows pub-key of sender can verify the signature. When using keyed-hash only folks who share the symmetric key with sender can verify it.

**Order Encryption/Signing Matter? -** Yes, it does. If one encrypts the message first and then signs its, then an interceptor of the message could strip the original signature and add their own signature, which could be a problem. Similarly, if one signs the message then encrypts it, then the receiver could decrypt and encrypt the signed message with another user’s public-key and forward it to that user and make him/her believe that the original sender intended the message to him/her.

**Digital cert/Cert Chain -** A digital certificate is a shared public data structure asserting something about the world. It helps assert that an entity knows a secret, in our case, a verification key. Certificate chain refers to a chain of certificates that verify the following certificate. The chain can be traversed all to the way to the root cert, which came from the Certificate Authority (CA).

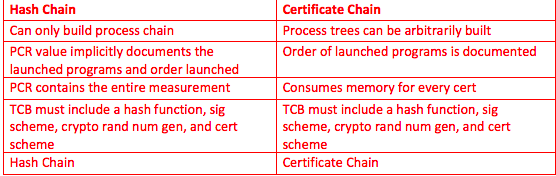
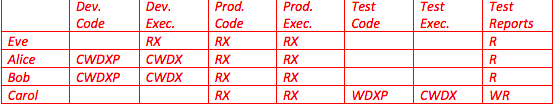
**Role of R1/R2 in Needham Schroeder -** R1 and R2 are random numbers used as Nonces to provide replay protection or freshness. R1 and R2 should be unpredictable and should be unique (not re-used) for that session or time-period of interest. **Alice send R2-1** prove she know skey

**TimeStamp instead of R1 - Random number nonce: Pro**: doesn’t require time synchronization. **Con**: require the ability to generate good random or pseudorandom numbers; need to be used in a challenge-response mode so may increase number of messages; need to keep short-term state in the protocol **Timestamps: Pro**: don’t need the generation of random numbers; don’t require short-term state in the protocol; can reduce the number of messages **Con**: Need secure, synchronized clocks across a distributed system

**4 sub-problems Trusted Computing problem? –** Measurement of program, attestation reading measurement, policy how to get only auth programs to execute, ownership who gets to authorize programs

**Manifest -** Manifest is a special type of certificate. The manifest signature is the manufacturer’s assertion that the manifest subject identifies the software it intended to release. R-read, c-create, w-write, d-delete, x-execute, p-promote

ACL(Dev. Exec.) = [(Alice, {CWDX}), (Bob, {CWDX}), (Eve, {RX})] CAPABILITY(Alice) = [(Dev. Code, {CWDXP}), (Dev. Exec., {CWDX}), (Prod. Code, {RX}), (Prod. Exec., {RX}), (Test Reports, {R})] **Hash Chain/Cert Chain:**

****

**TCB adversary physical access -** No, because the TCB signing key in the hardware, thus making it possible to an adversary with physical access to steal it.

**TCB continuous attestation – Attest currently running -** No. Attestation only tells us what programs were loaded, not the software that is currently running.

**Secured Boot vs Measured**: Measured boot loads a program, measured into PCR, and allowed to execute, whether there is a manifest or not. Secure boot allows a program to be loaded and measured into PCR, but if it is not white listed it’s not allowed to execute. Secure boot requires a manifest.

**Sealed Memory/Storage -** Sealed memory or storage refers to the TCB encrypting a program’s launch parameters. Encryption is with a symmetric key.

**two ownership models for trusted computing/TCB? –** Taking ownership, ownership transfer; taking ownership with sudo, ownership transfer req>machinery

**PseudoRandom -** Pseudorandom numbers are sequences of numbers that are not random (generated by a deterministic algorithm) but seem like cryptographically random numbers. They are used because true sources of randomness are not easily/cheaply available.

**Session/InterchangeKeys -** Session keys are short-term data encrypting keys meant to be used only for one session (e.g., TCP session). Interchange keys are long-term keys only meant for securing the exchange of session keys and not meant to be used for encrypting actual bulk data/messages. Session keys are needed to enhance/protect the life of Interchange keys. Breaking a cipher typically get easier the longer it is used (i.e., the more messages it is used encrypt). Having session keys is a way to prolong the life of the Interchange keys. In the case of public-key encryption, session keys help as symmetric ciphers are much more computationally efficient than public-key encryption. For the above reasons, session keys are desirable in both symmetric and asymmetric settings.

**Why is a trusted-third party desirable/needed for key-exchange?­ -** a trusted-third-party (TTP) N2 long-term interchange keys need to be set up and maintained between N parties to allow any-to-any (anyone among the N parties to be able tos ecurely communicate with any other) communication. This could be very challenging in large/open systems where N is large. With a trusted third-party only N long-term interchange keys need to set-up and maintained between the N parties and the TTP and this enables secure any-to-any communication through the use of protocols like Needham-Schroeder (actually Kerberos that we didn’t cover is used in practice; Needham-Schroeder has some issues). With public-keys the issue of setting up long-term interchange keys is solved to an extent but authentic versions of public-keys need to be made available. So while the secrecy requirement for key-exchange is removed, integrity requirement still exists. So trusted directories of public-keys (or Certificate authorities) or are still needed in asymmetric settings. So while the nature and level of trust is different some form of a TTP is needed.

**Access Control –** Authentiation: verification of entity, Authorization: grant permissions over object to entity, Audit/accounting: Maintain logging so user is accountable. **DAC –** Allow user to grant access to object he owns. **RBAC** user can’t pass rights to other because roles.

**MAC/DAC -** 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 Access control implemented in standard OS’s, e.g., Unix, Linux, Windows • 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

**Open/Closed Policy** – open = access unless deny, closed = no access unless allow

**RBAC vs Group -** 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 from of **RBAC –** roles are given permissions, and user are given roles **ABAC –** Access policies are defined in terms of attribute of user, object, environment. Role can be used as Attribute, attribute can define role making hybrid RBAC and ABAC.

**File 644 inside DIR 730 -** 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

**UNIX FS - The owner ID, group ID, and protection bits are part of the file’s inode; ACL is triple in inode, resource monitor is open system call.**

**20 job, 200 emp, 1500 permission - i**) DAC Model: 20 \* 200 \* 1500 = 6Million ii) RBAC: 20\*1500 + 200 \*20 = 20\*(1500+200) = 34000

**Secure Boot –** ACL is manifest sealed in the boot loader, boot loader is resource monitor, does PCR have right value after program loaded? Yes: program is unlocked and executed; No, then the program is erased from memory and denied access. Owner configures manifest sealed to the boot loader.

**Access Controls**

**Discretionary Access Control** – Decision based on identity of requestor and access rules, regular users can adjust policy

**Mandatory Access Control** – Decision made by testing labels assoc with processes, resources against system policy rules, user can’t change

**Role Based Access Control** – Policy defined in terms of roles rather than individual requestors

**Attribute based access** – Access policies defined in terms of attributes of user, resource, environment, context

**Requirements**: Reliable input (user auth and inputs are correct), Fine and coarse grain specifications, Open/closed policies, conflict resolution between policies, administrative policies (how to change access control)

**Principles**:Least privilege, Separation of duty (more than one entity to complete task), Dual control (two entities to implement policy change)

**Elements**: Subject (process), object (file), access right (read, write, exec, etc.)

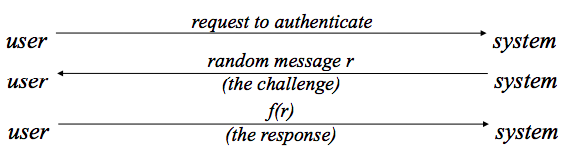
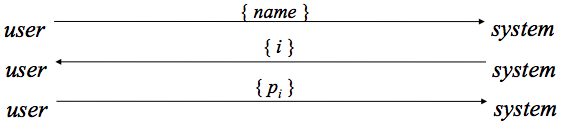
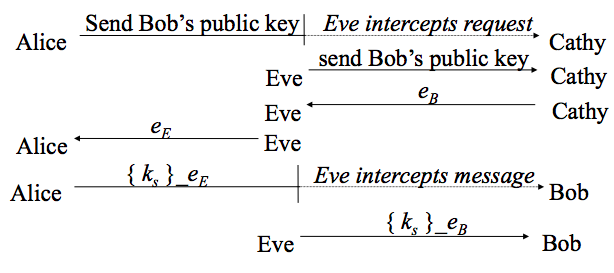
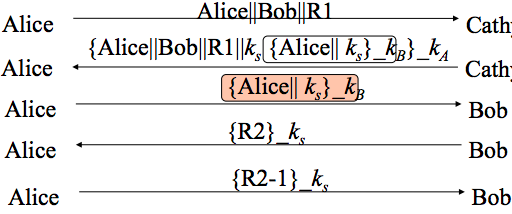
**ACL Scaling:** Group of user, RBAC, directory inheritance (rights transfer to obj in dir), negative rights(infer what is permitted by negation of what is stated to be denied)

**Revocation:** Revoking right for a subject from ACL. **Per Object review,** access to a given obj in ACL (easy) **Per-subject review** What object can a subject access (hard) search ACL of all object in system

**Capabilities** – Presented to access object, not completely contained by the OS, integrity is concern (tag mem, segmented mem, crypto hash)

**Auth Table –** Each row describes one right to one object by one user

**R1** – S0 has α\*, can transfer with or without \* to S (\* given means S can transfer α) **R2** – If S0 owns X, then S may grant any access to X to any other S **R3** – If S0 has ‘control’ or ‘ownership’ of S, S0 may remove any α from S’s access **R4** – S0 may copy porNon of ACM it controls or owns **R5** – Any subject can create an object X and may grant and delete access to X (because it is ‘owner’) **R6** – If S0 owns an object X it may destroy **R7** – Any subject S0 may create a new subject S. S0 owns S. S has control access to self. **R8** – If S0 owns S, it may remove S from system… Needham Schroeder, MITM Attack, S/Key, Challenge One Time Pass



kab = (PKbob)^ka mod p = (g^kb mod p) ka mod p = (PKalice)^kb mod p

m^e mod n= h(m) -> h(m)^d mod n = m, private key = d, public key = e,n. (n,d,e)