Protection and Security

B. Ramamurthy

Access Matrix

- A general model of access control as exercised by a file or database management system is that of an access matrix.
- Basic elements of the model are:
  - Subject: An entity capable of accessing objects. The concept of subject equates that of a process.
  - Object: Anything to which access is controlled. Ex: files, programs, segments of memory.
  - Access right: The way in which an object is accesses by the subject. Examples: read, write, and execute.

Access Matrix (contd.)

	File 1	File 2	File 3	File 4	Acct1	Acct2	Printer1
userA	Own R, W		Own R, W		Inquiry Credit		
userB	R	Own R, W	W	R	Inquiry Debit	Inquiry Credit	P
userC	R,W	R		Own R, W		Inquiry Debit	

Access Matrix Details

- Row index corresponds to subjects and column index the objects.
- Entries in the cell represent the access privileges/rights.
- In practice, access matrix is quite sparse and is implemented as either access control lists (ACLs) or capability tickets.

ACLs

- Access matrix can be decomposed by columns, yielding access control lists.
- For each object access control list lists the users and their permitted access rights.
- The access control list may also have a default or public entry to covers subjects that are not explicitly listed in the list.
- Elements of the list may include individual as well group of users.

Windows NT(W2K) Security

- Access Control Scheme
  - name/password
  - access token associated with each process object indicating privileges associated with a user
  - security descriptor
    - access control list
    - used to compare with access control list for object

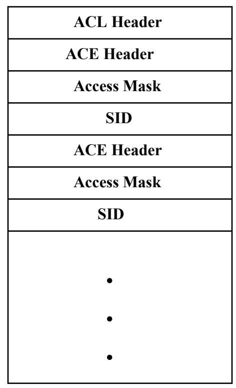
Access Token (per user/subject)

Security ID (SID)
Group SIDs
Privileges
Default Owner
Default ACL

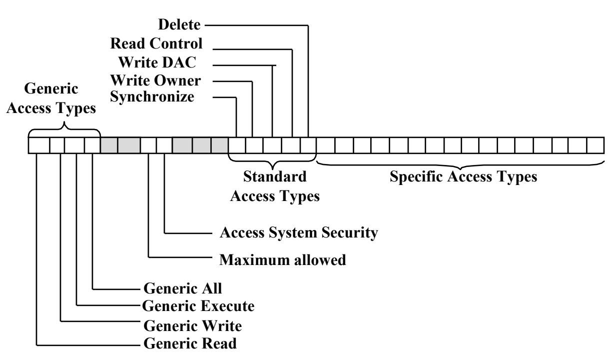
Security Descriptor (per Object)

Flags
Owner
System Access Control List (SACL)
Discretionary Access Control List (DACL)

Access Control List



Access Mask



Access Control Using ACLs

- When a process attempts to access an object, the object manager in W2K executive reads the SID and group SIDs from the access token and scans down the object's DACL.
- If a match is found in SID, then the corresponding ACE Access Mask provides the access rights available to the process.

RSA Encryption

To find a key pair *e*, *d*:

1. Choose two large prime numbers, *P* and *Q* (each greater than 10100), and form:
   N = *P* x *Q*
   Z = (*P*–1) x (*Q*–1)
2. For *d* choose any number that is relatively prime with *Z* (that is, such that *d* has no common factors with *Z*).

We illustrate the computations involved using small integer values for *P* and *Q*:

$P = 13$, $Q = 17$ –> $N = 221$, $Z = 192$

$d = 5$

3. **To find *e* solve the equation:**
   $e \times d = 1 \bmod Z$

That is, *e* x *d* is the smallest element divisible by *d* in the series *Z*+1, 2*Z*+1, 3*Z*+1, ...

$e \times d = 1 \bmod 192 = 1, 193, 385, \ldots$

385 is divisible by *d*

e = 385/5 = 77

RSA Encryption (contd.)

To encrypt text using the RSA method, the plaintext is divided into equal blocks of length k bits where $2^k < N$ (that is, such that the numerical value of a block is always less than N; in practical applications, k is usually in the range 512 to 1024).

k = 7, since $2^7 = 128$

The function for encrypting a single block of plaintext M is: (N = P X Q = 13X17 = 221), e = 77, d = 5:

$E'(e,N,M) = M^e \bmod N$

for a message M, the ciphertext is $M^{77} \bmod 221$

The function for decrypting a block of encrypted text c to produce the original plaintext block is:

$D'(d,N,c) = c^d \bmod N$

The two parameters e,N can be regarded as a key for the encryption function, and similarly d,N represent a key for the decryption function.

So we can write K_e = <e,N> and K_d = <d,N>, and we get the encryption function: $E(K_e, M) = \{M\}_K$ (the notation here indicating that the encrypted message can be decrypted only by the holder of the private key K_d) and $D(K_d, = \{M\}_K) = M$.

<e,N> - public key, d – private key for a station

Application of RSA

- Lets say a person in Atlanta wants to send a message M to a person in Buffalo:
- Atlanta encrypts message using Buffalo's public key B → E(M,B)
- Only Buffalo can read it using it private key b: E(b, E(M,B)) → M
- In other words for any public/private key pair determined as previously shown, the encrypting function holds two properties:
  - E(p, E(M,P)) → M
  - E(P, E(M,p)) → M

How can you authenticate "sender"?

- (In real life you will use signatures: the concept of signatures is introduced.)
- Instead of sending just a simple message, Atlanta will send a signed message signed by Atlanta's private key:
  - E(B,E(M,a))
- Buffalo will first decrypt using its private key and use Atlanta's public key to decrypt the signed message:
  - E(b, E(B,E(M,a)) → E(M,a)
  - E(A,E(M,a)) → M

Digital Signatures

- Strong digital signatures are essential requirements of a secure system. These are needed to verify that a document is:
- Authentic : source
- Not forged : not fake
- Non-repudiable : The signer cannot credibly deny that the document was signed by them.

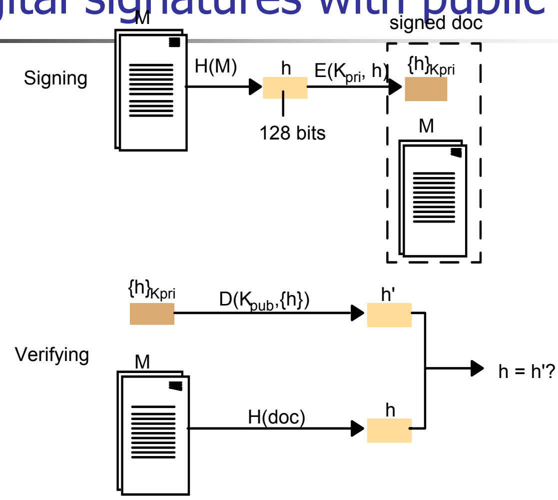
Digest Functions

- Are functions generated to serve a signatures. Also called secure hash functions.
- It is message dependent.
- Only the Digest is encrypted using the private key.

Alice's bank account certificate

1. *Certificate type*	Account number
2. *Name*	Alice
3. *Account*	6262626
4. *Certifying authority*	Bob's Bank
5. *Signature*	$\{Digest(field\ 2 + field\ 3)\}_{K_{Bpriv}}$

Digital signatures with public keys



Low-cost signatures with a shared secret key