Lecturer: Prashant Shenoy *Scribe: Supriya Jagannath*

1. Introduction

1.1 Why is Security required?

An intruder can potentially perform:
• eavesdropping
• modification, insertion, or deletion of messages or message content

Four types of security threats to consider are:
1. Interception- situation that an unauthorized party has gained access to a service or data
2. Interruption- situation in which services or data become unavailable, unusable, destroyed, and so on
3. Modification- unauthorized changing of data or tampering with a service so that it no longer adheres to its original specifications
4. Fabrication- situation in which additional data or activity are generated that would normally not exist.

Interruption, modification, and fabrication can each be seen as a form of data falsification. Since users and resources are vast in number and widely spread across different administrative domains, security is essential in Distributed Systems.

Important security mechanisms are:
1. Encryption- transforms data into something an attacker cannot understand
2. Authentication- used to verify the claimed identity of a user, client, server, host, or other entity
3. Authorization- verify rights to perform requested action
4. Auditing- trace which clients accessed what, and which way

2. Cryptography

Encryption and decryption are accomplished by using cryptographic methods parameterized by keys. The purpose of encryption or encipherment is to transform a message or plaintext into an apparently random pattern, the cipher text, such that decryption of the cipher text is extremely difficult unless an appropriate algorithm is known. For a normal two-way cipher there will be encryption and decryption algorithms which are complementary; for a one-way cipher the encryption algorithm will have no known inverse, i.e. it is not possible to algorithmically convert cipher text to plaintext.

2.1 Encryption

DES Algorithm

The Data Encryption Standard (which is not an ISO standard) is a block cipher, dealing with 64-bit data blocks of data. A block is transformed into an encrypted (64 bit) block of output in 16 rounds, where each round uses a different 48-bit key for encryption.

Key Distribution

The previous section has dealt with some of the mechanisms for key distribution. Initial key distribution is often done by out-of-band means, e.g. using the postal service or face-to-face meetings. Once initial keys have been

distributed, it may be possible to use the data distribution mechanism (encrypted) to transmit new keys. However, weaknesses are usually human, and it is best to prevent users from ever being able to see their keys, or they may be tempted to send the clear text via electronic mail! Storage of keys must be similarly controlled, and it is wise to ensure that even inside a program keys are kept encrypted, so that if there is a program, machine, or file store crash, clear text keys cannot be easily discovered in the ruins.

Public Key Encryption

One difficulty with this approach is that the two parties must somehow agree on the shared key. In public key encryption, there will be separate keys for encryption and decryption. Since the encryption key is public, anyone can send encrypted message. Clearly, if public key cryptography is to work, key selection and encryption/ decryption must be done in such a way that it is for an intruder to either determine the private key or somehow otherwise decrypt or guess messages.

RSA Algorithm

The Rivest, Shamir and Adleman algorithm (RSA) [Rivest,78] belongs to a family of exponentiation ciphers. RSA makes extensive use of arithmetic operations using modulo-*n* arithmetic. They rely on the comparative ease of computing exponentials and the difficulty of computing logarithms. They also use modular arithmetic, where the modulus is usually a very large number.
Generating the private and public keys requires four steps:
1. Choose two very large prime numbers, *p* and *q*. How large should *p* and *q* be? The larger the values, the more difficult it is to break RSA, but the longer it takes to perform the encoding and decoding
2. Compute $n = p \times q$ and $z = (p - 1) \times (q - 1)$.
3. Choose a number *d* that is relatively prime to z.
4. Compute the number *e* such that $e \times d = 1 \ mod \ z$.

Another popular public-key encryption algorithm is the Diffie-Hellman algorithm. This algorithm is not as versatile as RSA in that it cannot be used to encrypt messages of arbitrary length; it can be used, however, to establish a symmetric session key, which is in turn used to encrypt messages.

Digital signatures using Public Keys

In a digital world, one often wants to indicate the owner or creator of a document, or to signify one's agreement with a document's content. It's a way of signing the message sent; they are supposed to be equivalent to physical signatures made to a document. A **digital signature** is a cryptographic technique for achieving these goals in a digital world. Sender cannot repudiate message never sent and same way the receiver can never fake a received message.
Example: suppose if A wants B to "Sign" a message M. B sends DB(M) to A , A computes if EB (DB(M)) == M , Then B has signed M.

One concern with signing data by encryption is that encryption and decryption are computationally expensive. Given the overheads of encryption and decryption, signing data via complete encryption/decryption can be overkill. A more efficient approach is to introduce hash functions into the digital signature. An important application of digital signatures is **public key certification** that is, certifying that a public key belongs to a specific entity. Public key certification is used in many popular secure networking protocols, including IPsec and SSL.

3. Authentication

This is a process where the user can prove the system its identity and obtain access to the system. Various authentication protocols are as follows.

AP 1.0:
Here Alice sends a message to bob identifying itself, but the problem with such a fundamental approach is any third party can also claim to be Alice.

AP2.0:
The problems faced in first step can be negated by authenticating source IP address is from Alice's machine, but another problem arises here where the third part can just do IP spoofing by using a simple python code.

AP3.0:
To overcome the problems faced in 2nd step we can use a secret password, but hacker can intercept this message "sniffing". Earlier protocols like telnet the username and password is in plaintext, hence easy to sniff and crack the system.

AP3.1: Use Encryption
To overcome the problems faced in AP 3.0 we can use Symmetric key encryption where Alice & bob both know a secure key. Both Alice & Bob communicate by encrypting the message with the shared common key. This proves the authentication of Alice or Bob, since only those two have the key. But this is going fail if the hacker captures the encrypted packet and does reply attack, where the same message is sent to bob at a later time. Hence, in this scenario the encryption fails.

Prof questions what is the solution to this problem? the answers to this is "using random number", "use time stamps" , "use time out mechanisms".

Authentication Using Nonce
AP 4.0:
To overcome the problems faced in AP3.1 we can make use of nonce, when Alice sends message to Bob (unencrypted) Bob sends a nonce to Alice, then Alice encrypts the nonce and proves to Bob that it's the true Alice.
A to B: msg = "I am A" /* note: unencrypted message! */
B to A: once-in-a-lifetime value, n
A to B: msg2 = encrypt(n) /* use symmetric keys */
B computes: if decrypt(msg2)==n
then A is verified ,else A is fradulent

Authentication using Public keys
AP 5.0:
Each entity (Machine) generates two key, public and private, private is supposed to be kept secret. The public key can be distributed so that others can contact.
A to B: msg = "I am A"
B to A: once-in-a-lifetime value, n
A to B: msg2 = DA(n)
B computes: if EA (DA(n))== n
then A is verified, else A is fradulent

Problems with AP 5.0
An intruder can do the man in the middle attack, i.e. Trudy can impersonate as Alice to Bob. There needs to be a secure key distribution. The protocol is only as secure as key distribution.

4. Message digests

They are essentially hash functions; it takes an arbitrary length of strings but outputs a fixed length hash and this hash function is signed using methods specified earlier. Since the hash is significantly smaller in size and hence not computationally intensive to make Digital Signatures on this, this seems to be more efficient and better approach than hashing the entire message to authenticate. When the receiver, receives the (message + signed hash), the received messaged can be hashed and verify the signed hash.

Properties of hash function
(a) Given a digest x, it is infeasible to find a message y such that H(y) = x,

(b) It is infeasible to find any two messages x and y such that H(x) = H(y)

5. Hash functions

MD5:

Takes arbitrary message and converts into 128 bits constant length key. But MD5 is not secure anymore.
SHA hash functions (secure hash algorithm):
SHA-1 : 160-bit function that resembles MD5, SHA-2: family of two hash functions (SHA-256 and SHA-512), Developed by NIST and NSA

6. Symmetric Key exchange: trusted server

Alice sends an initiative message to the KDC that it wants to contact Bob, the KDC generates a secret key that only KDC & Alice knows (KA,KDC) and using this key encrypts the Key shared between A & B (KA,B). The KDC also sends the same key in the same method to B, Hence now both Alice and Bob can exchange data securely using their respective keys.

7. Access Control:

This gives us the capacity to restrict access to the system.
Access control list - where for every resource there is list of which user can read, write, and exit.
Capabilities- Every user is basically given a list of capability, when it does an operation the user presents the list of capabilities and gains access.