

Kerberos Authentication Service

- Kerberos is trusted authority with whom everyone shares keys
- When a client on a network wants to talk to a server, he issues a request for a *ticket* to Kerberos' Ticket Granting Server (TGS)
- Client uses this ticket always when he talks to the server, sometimes he also sends *authenticators*
- Clients and servers do not trust each other

1

Kerberos Authentication Service

- A ticket is used to pass securely to the server the identity of the client
 - It is good for a single client and single server for some period of time
 - It contains client's name and network address, server's name, timestamp and a session key, all encrypted with a key server shares with Kerberos
- An authenticator is generated whenever a client requests some service from the server
 - It is good only for one request
 - It contains client's name, a timestamp and an optional additional key, all encrypted with session key

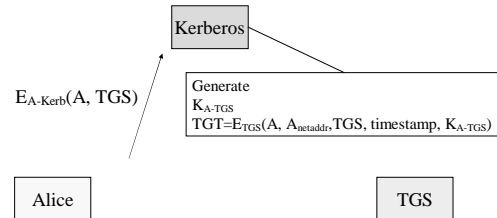
2

Kerberos Authentication Service

- To get initial ticket
 - Alice sends a message with her name and a name of a Ticket Granting Server (TGS) to Kerberos
 - Kerberos generates a session key K_{A-TGS} to be used between her and TGS and also generates Ticket Granting Ticket (TGT)
 - Kerberos encrypts K_{A-TGS} with Alice's secret key and sends that and TGT to Alice
 - Alice retrieves K_{A-TGS} and saves it and TGT

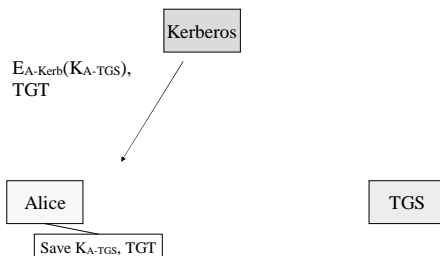
3

Getting Initial Ticket



4

Getting Initial Ticket



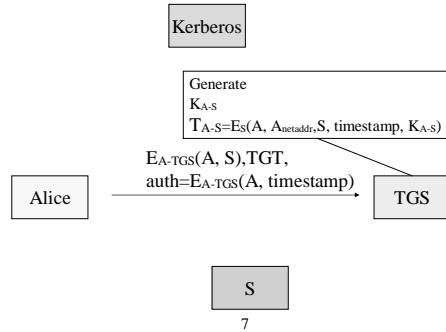
5

Kerberos Authentication Service

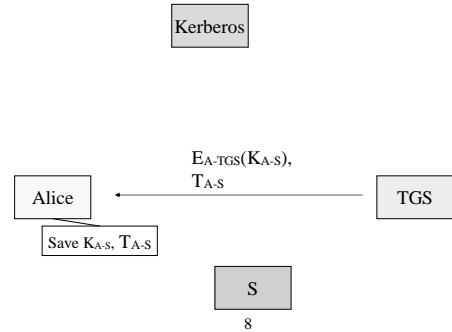
- To get ticket for specific server
 - Alice sends a request with her name and server's name to TGS, encrypted with K_{A-TGS} , accompanied with TGT and authenticator
 - TGS decrypts TGT with his secret key and retrieves K_{A-TGS}
 - TGS uses K_{A-TGS} to decrypt authenticator and compare Alice's information in authenticator with information in TGT, and compare timestamps
 - If everything matches he generates a session key K_{A-S} to be used between her and server and a valid ticket T_{A-S}
 - TGS encrypts K_{A-S} with K_{A-TGS} and sends this and T_{A-S} to Alice

6

Getting Ticket for Server S



Getting Ticket for Server S

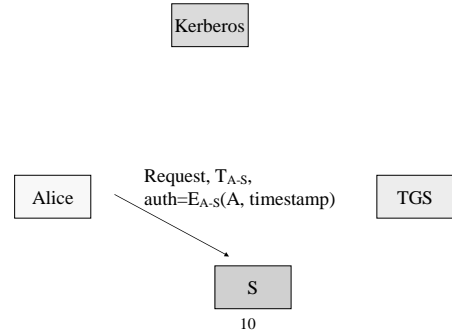


Kerberos Authentication Service

- > To request service
 - > Alice sends a valid ticket $T_{A,S}$ and authenticator
 - > Server decrypts $T_{A,S}$ with his secret key and retrieves $K_{A,S}$
 - > Server uses $K_{A,S}$ to decrypt authenticator and compare Alice's information in authenticator with information in $T_{A,S}$, and compare timestamps
 - > If everything matches he grants the request
 - > For applications that require mutual authentication server will send to Alice a timestamp encrypted with $K_{A,S}$

9

Getting Ticket for Server S



SSH

- > Protocol for secure remote login
- > Protocol architecture:
 - > Transport layer protocol provides
 - > Server authentication
 - > Confidentiality
 - > Integrity with perfect forward secrecy
 - > Compression (optional)
 - > User authentication protocol
 - > Authenticates the client to the server
 - > Runs over transport layer protocol
 - > Connection protocol
 - > Multiplexes the encrypted tunnel into several logical channels
 - > Runs over user authentication protocol

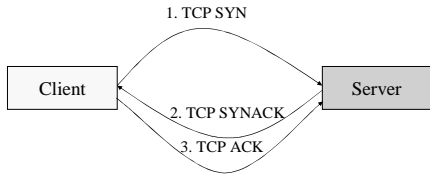
11

SSH Transport Protocol

1. Client contacts the server, performs TCP 3-way handshake
2. Both sides send the protocol and software version numbers
3. Key exchange

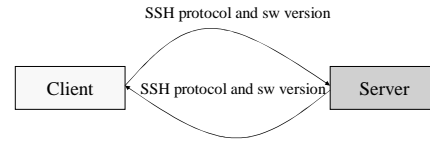
12

3-way handshake



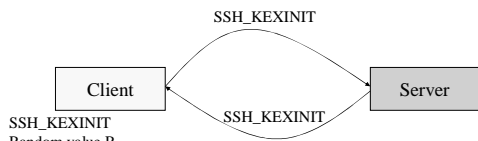
13

Negotiate protocol



14

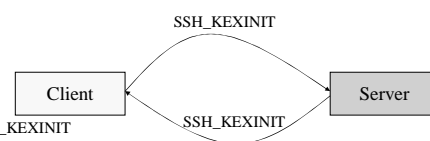
Negotiate Key Exchange



- SSH_KEXINIT
- Random value R
- Key exchange algorithms
- Server host key algorithm
- Encryption algorithm client-to-server
- Encryption algorithm server-to-client
- MAC algorithm client-to-server
- MAC algorithm server-to-client
- Compression algorithm client-to-server
- Compression algorithm server-to-client

15

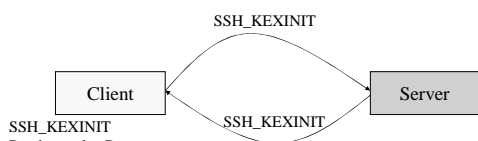
Negotiate Key Exchange



- SSH_KEXINIT
- Random value R
- Key exchange algorithms ← diffie-hellman-group1-sha1
- Server host key algorithm
- Encryption algorithm client-to-server
- Encryption algorithm server-to-client
- MAC algorithm client-to-server
- MAC algorithm server-to-client
- Compression algorithm client-to-server
- Compression algorithm server-to-client

16

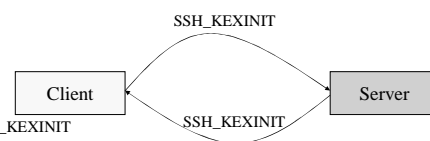
Negotiate Key Exchange



- SSH_KEXINIT
- Random value R
- Key exchange algorithms ← ssh-dss, ssh-rsa, x509v3-sign-rsa, x509v3-sign-dss, spki-sign-rsa, spki-sign-dss, pgp-sign-rsa, pgp-sign-dss
- Server host key algorithm
- Encryption algorithm client-to-server
- Encryption algorithm server-to-client
- MAC algorithm client-to-server
- MAC algorithm server-to-client
- Compression algorithm client-to-server
- Compression algorithm server-to-client

17

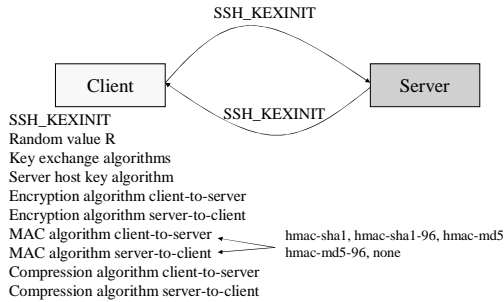
Negotiate Key Exchange



- SSH_KEXINIT
- Random value R
- Key exchange algorithms
- Server host key algorithm
- Encryption algorithm client-to-server ← 3des-cbc, blowfish-cbc, twofish256-cbc, twofish192-cbc, twofish128-cbc, aes256-cbc, aes192-cbc, aes128-cbc, serpent256-cbc, serpent192-cbc, serpent128-cbc, arcfour, idea-cbc, cast128-cbc
- Encryption algorithm server-to-client
- MAC algorithm client-to-server
- MAC algorithm server-to-client
- Compression algorithm client-to-server
- Compression algorithm server-to-client

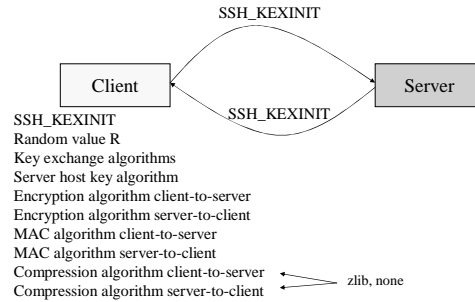
18

Negotiate Key Exchange



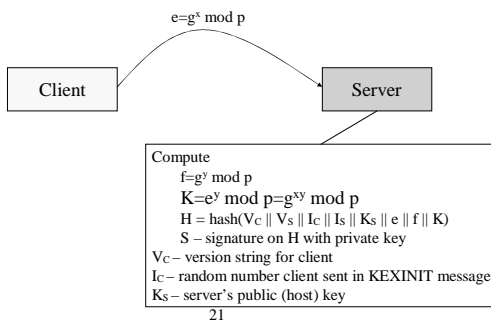
19

Negotiate Key Exchange



20

Key Exchange (Diffie-Hellman)



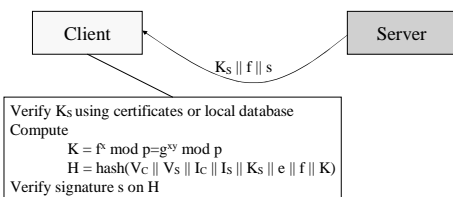
21

Host keys

- Each host has a host key – private/public key pair used for authentication
- Two different trust models exist:
 - First time client contacts a new host he stores the public key and associates it with the server name. Next time client checks if public key matches the stored value. If you have specified strict checking mismatches will be rejected, otherwise not.
 - Host's name-to-key mapping is certified by a certification authority, client only knows CA's public key

22

Key Exchange (Diffie-Hellman)



23

Example

```
ssh -v copland
debug1: Connecting to copland [128.175.13.92] port 22.
debug1: Connection established.
debug1: identity file /usr/sunshine/ssh/identity type -1
debug1: Remote protocol version 1.99, remote software version OpenSSH_3.7.1p2
debug1: match: OpenSSH_3.7.1p2 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_3.7.1p1
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client blowfish-cbc hmac-md5 none
debug1: kex: client->server blowfish-cbc hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
Warning: Permanently added 'copland,128.175.13.92' (RSA) to the list of known hosts.
debug1: ssh_rsa_verify: signature correct
```

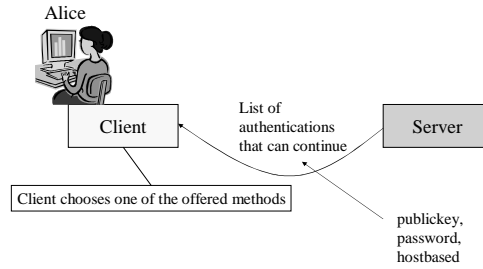
24

User Authentication Protocol

- › Runs on top of transport layer
- › Authenticates the client's user to the server
- › Uses value H from key exchange as session identifier

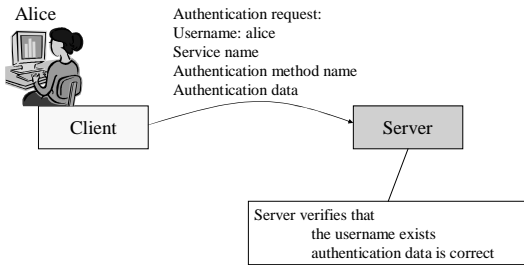
25

User Authentication



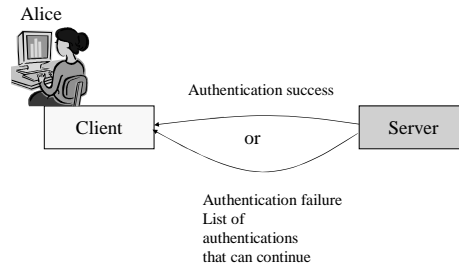
26

User Authentication



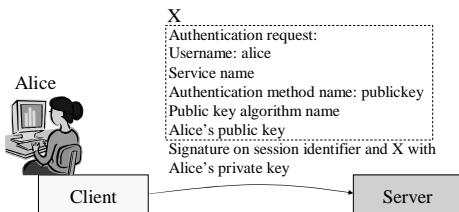
27

User Authentication



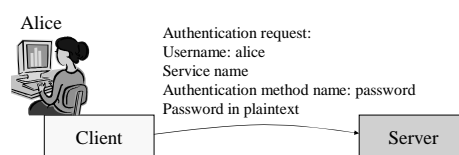
28

Public Key Authentication



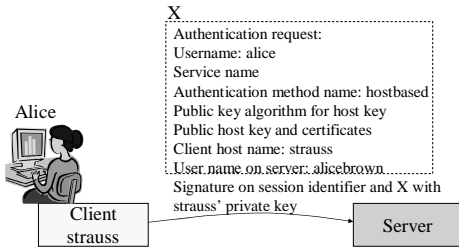
29

Password Authentication



30

Host Based Authentication



Server sometimes does DNS lookup to verify that source address in packets and client host name match

Connection Protocol

- A channel can be open for terminal sessions, forwarded connections, etc.
- All channels are multiplexed onto a single connection
- Channels can be open from either side
 - Send channel request describing the type of channel
 - Request is granted if such channel is available

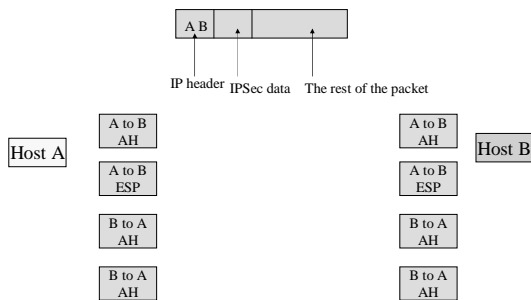
IPSec

- Protocol for providing security at IP level
 - Access control
 - Authentication
 - Integrity
 - Replay control
 - Data confidentiality
- Consists of two protocols:
 - Authentication Header (AH) protocol
 - Encapsulating Security Payload (ESP) protocol

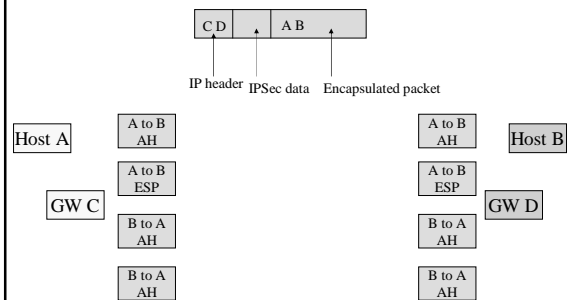
IPSec

- IPSec can be implemented “point-to-point” or between gateways
- The notion of Security Association (SA) is crucial to IPSec
 - A simplex connection that affords security services to traffic carried by it
 - Associated with security parameter index, destination address and protocol (AH or ESP)
- Two types of SAs are defined
 - Transport mode
 - Tunnel mode

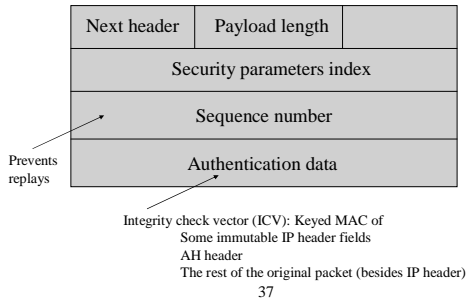
IPSec Transport Mode



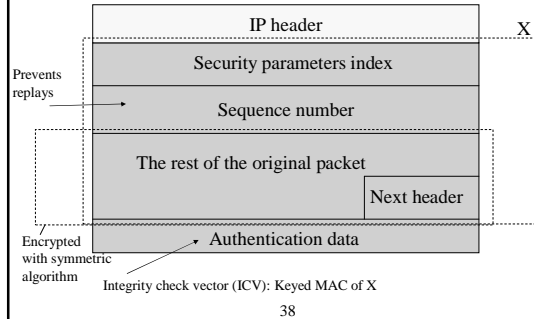
IPSec Tunnel Mode



Authentication Header



Encapsulating Security Payload



Pretty Good Privacy (PGP)

- Cryptographic program that provides
 - Privacy (encrypts messages)
 - Authentication
 - Message integrity
- Also enables file encryption
- Used in E-mail service

39

Pretty Good Privacy (PGP)

- Uses public cryptography for authentication and key exchange, symmetric cryptography for messages
 - Generates session key using randomness from keyboard and mouse movements
 - Constructs message digest and signs it with sender's private key
 - Compresses messages to reduce size and redundancy
 - Encrypts messages with session key
 - Encrypts session key with recipient's public key

40

Pretty Good Privacy (PGP)

- First time sender learns recipient's public key through certificate, stores it in a file
- Certificate contains
 - Public key (along with algorithm name RSA, DSA or Diffie-Hellman)
 - Some information about the owner
 - Preferred symmetric algorithm: Cast, IDEA or Triple-DES
 - Several signatures attesting the validity
- Signatures may be from CA or from other users
- Certificate has expiration date, it can also be revoked

41

PGP Web of Trust

- Every user can sign other user's certificate, it is upon the recipient to follow certificate chain and decide whether to trust the certificate
- Signature bestows reputation – I signed your key because you are trustworthy
 - If you sign Alice's key, this will mean that I co-sign it too
 - Once you sign someone's key (and add it to your list) you indicate level of trust: complete, marginal or none
 - Now when you receive key signed by Alice it will be regarded as valid, marginally valid or invalid based on trust
 - One valid or two marginally valid signatures are needed to establish a key as valid

42

S/MIME

- › Similar to PGP but uses different encryption algorithms and certificate formats
- › Also used for E-mail messages
- › Does not have “web of trust” concept