Network and Computer Security

Summary

Contents

1 Cryptography 3

1.1 Cryptographic Services 3

1.2 Cryptographic Building Blocks 3

1.2.1 Symmetric Cipher 3

1.2.2 Asymmetric Cipher 4

1.2.3 Cryptographic Hash 4

1.2.4 Composite Building Blocks 4

1.3 Symmetric Criptography 4

1.3.1 Symmetric Stream Ciphers 5

1.3.2 Symmetric Block Ciphers 5

1.4 Hash Functions 9

1.4.1 Message Integrity Codes 9

1.5 Asymmetric Ciphers 10

1.5.1 One-way Functions 10

1.5.2 Algorithms 10

1.5.3 Digital Signatures 11

1.6 Public Key Management 11

1.6.1 Digital Certificates 11

1.7 Secret Key Management 14

1.7.1 Key Generation 14

1.7.2 Key Distribution 14

2 Networks 17

2.1 Network Models 17

2.1.1 OSI and Internet 17

2.1.2 Address Resolution 17

2.2 Network Vulnerabilities 18

2.2.1 Physical Layer 18

2.2.2 Data Link Layer 18

2.2.3 Network Layer 19

2.2.4 Transport Layer 19

2.2.5 Application Layer 21

2.3 Network Security Models 22

2.4 Firewalls and Intrusion Detection Systems 23

2.4.1 Firewall Placement 23

2.4.2 Iptables 24

2.4.3 Types of Firewall 25

2.4.4 Intrusion Detection Systems 25

2.4.5 IDS classification 26
2.5 SSH and TLS 26
2.5.1 SSH Tunneling 27
2.5.2 TLS 27
2.5.3 TLS Protocols 28
2.5.4 Key Derivation 29
2.6 Authentication 29
2.6.1 Authentication With Passwords 29
2.6.2 Biometric Authentication 29
2.6.3 Authentication Protocols 30

Chapter 1

Cryptography

It is a widespread and dangerous belief that encrypting everything provides protection against anything. In reality, when the algorithm is known it is necessary to prevent several types of attacks, such as ciphertext-only, known-plaintext, chosen-plaintext or chosen-ciphertext. Cryptography must, then, protect the information against unauthorized insertion of information, modification of information in transit, replay of information and access to information.

1.1 Cryptographic Services

To this end, we need the following cryptographic services:

- Confidentiality: a service used to keep the content of the information from all, but those entities authorized to have it. Has the drawbacks of making debugging harder and may lead to information loss if the key is lost. Can be assured through symmetric of assymetric cipher.
- Integrity: a service that detects data manipulation by unauthorized entities (not the same thing as error detection codes). An intruder should not be able to substitute a false message for a legitimate one. Can be assured through MIC or a digital signature.
- Authenticity: a service used to ascertain the identity or the origin of a message. Can be assured through MIC or a digital signature, tough freshness requires adding a nonce to the message.
  - Entity authentication: verify the identity of an entity
  - Data origin authentication: confirm the creator of the message
  - Non-repudiation: a service which prevents an entity from denying. Can be assured by a digital signature. previous commitments or actions

1.2 Cryptographic Building Blocks

1.2.1 Symmetric Cipher

Uses the same key to cipher and decipher. In cryptographic function notation:

- $E(M, K)$: cipher message M with key K
- $D(C, K)$: decipher cryptogram C with key K

1.2.2 Asymmetric Cipher

A public/private pair of keys is used (KU/KR). In cryptographic function notation:

- $AE(M, KR)$: cipher message M with private key KR
- $AD(C, KU)$: decipher cryptogram C with public key KU

It is also possible to cipher with the public key and then decipher with the private key.

1.2.3 Cryptographic Hash

A cryptographic hash function does not use a key, instead receives an input message and returns a digest of the data. In cryptographic function notation:

- $H(M)$: hash of message M

The digest value is deterministic, has fixed size, a unique representation, is non-reversible and sensible to input changes.

1.2.4 Composite Building Blocks

- Hybrid cipher: a random symmetric key is generated and used to cipher the message. This key is then ciphered with the public key of the receiver and is sent along with the ciphered message. The receiver must decipher the symmetric key with its private key and use it do decipher the message
- Message integrity code: the MIC is created by creating the digest of the message and ciphering the result. It can then be used to check the integrity of the message by generating a new hash and comparing to the sent one.
- Digital signature: a digital signature is created by ciphering a digest with the private key. The receiver can then decipher with the public key and verify the identity of the sender.

1.3 Symmetric Criptography

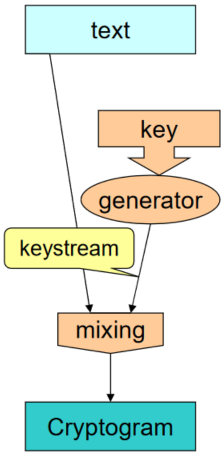
Symmetric ciphers use a single secret key that may be shared by 2 or more communicating parties. This allows confidentiality to all those who have the key and message authentication, with good performance. The main problem lies in key distribution

For N communicating parties, $N \times (N - 1)/2$ keys are needed for them to be able to communicate 1 - 1 secretly.

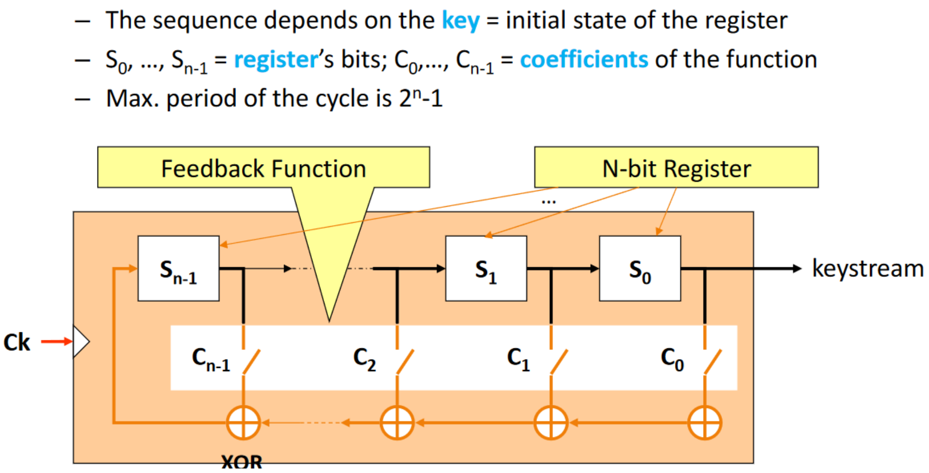
An example of a symmetric cipher is the one-time pad, where the message is XOR'd with the key. The security is based on the assumption that the key is never reused.

1.3.1 Symmetric Stream Ciphers

Symmetric stream ciphers are pratical approximations to the one-time pad. Keystreams are generated in a deterministic way from a fixed size key (approximation to real random sequence generators).



In this type of cipher, if the plain text is known, the keystream is exposed. The secure pseudo-random generators are sometimes based on LFSRs (Linear Feedback Shift Registers). It consists on a state machine that produces a cyclic sequence of bits:



- The sequence depends on the **key** = initial state of the register
- S_0, ..., S_{n-1} = **register**’s bits; C_0,..., C_{n-1} = **coefficients** of the function
- Max. period of the cycle is 2^n-1

1.3.2 Symmetric Block Ciphers

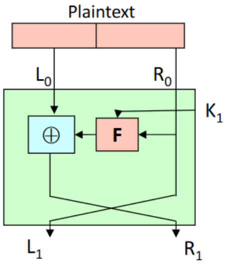
These are also based on approximations, using Shannon’s notions of confusion (repeated application of a complex function to a large block) and diffusion (permutation, substitution, expansion and compression).

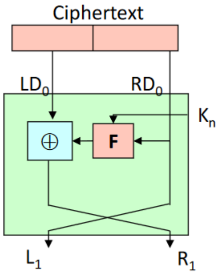
Feistel Network

The feistel network is a complex function most commonly used in block cipher algorithms. It applies a round function F over multiple rounds:

- Each round uses a different round key (K_i), obtained from the key (K)
- Text is split in left (L) and right (R) parts

The cipher and decipher processes are the same, with the keys used in the inverse order.





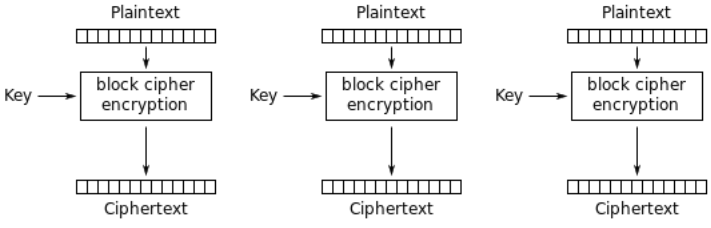
DES

The DES algorithm uses 64 bit blocks with 56 bit keys. It applies the feistel network algorithm on several rounds, after an initial permutation of the 64 bits, according to a fixed table.

Block Cipher Modes

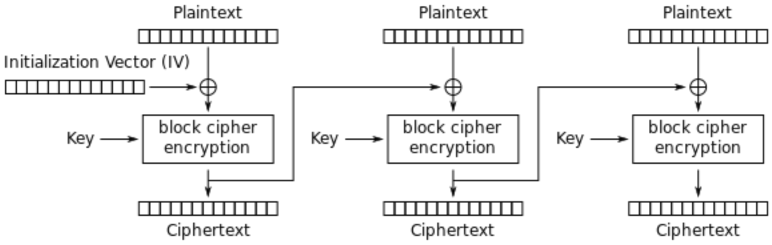
When using a plaintext of different size than the block, a cipher mode must be used.

- *Electronic Code Book* (ECB)
  - Encryption using independent blocks
  - Weakness in reproducing patterns from the original text; identical blocks produce the same ciphertext



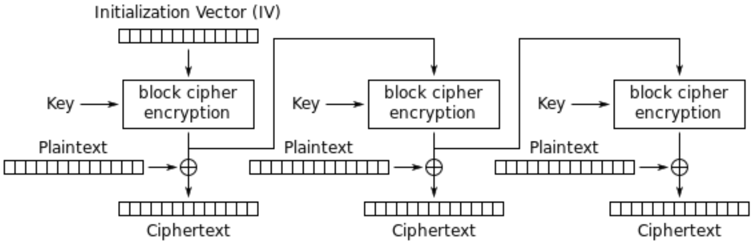
Electronic Codebook (ECB) mode encryption

- *Cipher Block Chaining* (CBC)
  - Plaintext is XORed with the ciphertext of the previous block before encryption
  - Reduces the risk of pattern replication
  - Uses an initialization vector in the first block (necessary for decryption) and requires padding: bits to compose entire blocks of the size required by the algorithm



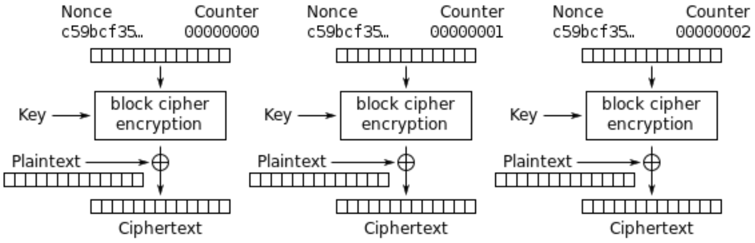
Cipher Block Chaining (CBC) mode encryption

- *Output Feedback* (OFB)
  - Retains the advantages of CFB, and the ciphertext is not used in the next block, allowing block cipher operations to be performed in advance, enabling parallel XOR as soon as the text (plaintext or ciphertext) is available



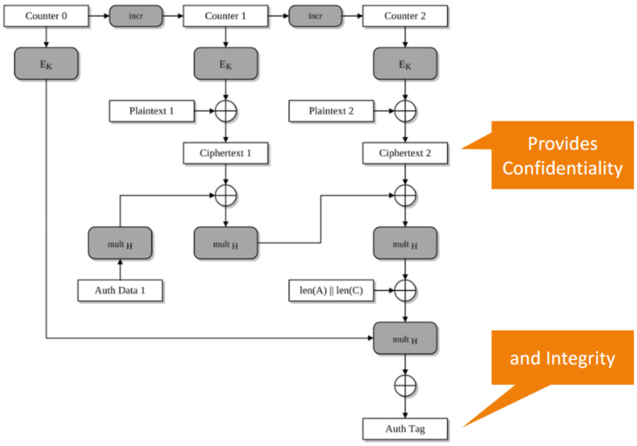
Output Feedback (OFB) mode encryption

- *Counter Mode* (CTR)
  - Standard cipher mode in AES
  - Uses a nonce and a counter, which must be different in each cipher operation



Counter (CTR) mode encryption

- *Galois Counter Mode* (GCM)



Padding

Sometimes it may be necessary to pad the last block, when the message size is not divisible by the block length. This may be done by:

- Padding with zero (null) bytes, spaces (0x20), all bytes of the same value
- Padding with random bits
- Padding with 0x80 (1000 0000) followed by zero (null) characters
- Padding with the PKCS#5 scheme: Sequence of bytes, each of which equal to the number of padding bytes (e.g. if 24 bits of padding need to be added, the padding string is "03 03 03": 3 bytes times 8 bits equals 24)

AES

AES is the current encryption standard. Supports keys of 128, 192 and 256 bits with 128 bit blocks. It runs in 10, 12 or 14 rounds, in which it substitutes bytes, shifts rows, mixes columns and adds the round key XOR state with key material)

1.4 Hash Functions

These are cryptagraphic functions, but not ciphers. They must be:

- Collision resistance: Computationally infeasible to find two inputs that give the same hash
- Preimage resistance: Given a hash, it's computationally infeasible to find an input that produces that hash
- Seconde preimage resistant: Given a hash value and the corresponding input, it's computationally infeasible to find a second input that generates that same hash

The most common algorithms are:

- MD5 (128 bits): very weak
- SHA-1 (160 bits): weak
- SHA-2 (256 to 512 bits)
- SHA-3 (256 to 512 bits)

Attacks on hash functions are sometimes done by brute force ($P(collision) = 2^{m-1}$) and most often through the birthday attack (pick M, M', M'', M'''... and obtain hashes until any 2 are identical) ($P(collision) = 2^{m/2}$).

1.4.1 Message Integrity Codes

The objective is to detect changes to a message. It allows the checking of a message's integrity and, with freshness can provide authenticity, thus, it's sometimes called MAC.

Under the assumption that the sender and recipient have a shared secret key K, the idea is to send the messgae and the MAC; if the message (or the MAC) is modified by an attacker, the recipient will be able to detect it. Attacker cannot create a valid MIC because he does not have K.

Implementations of MAC could be as follows:

- Hash the message and encrypt the digest
- Using a keyed-function (CMAC, usually with CBC)
- Using a keyed-hash (HMAC)

1.5 Asymmetric Ciphers

Asymmetric cryptography uses a pair of public/private keys. It allows for confidentiality, authentication and integrity (digital signatures). Has the advantage of only needing N key pairs, but the performance is much worse.

For confidentiality, the public key is used to cipher and the private to decipher. For authentication it's the opposite.

1.5.1 One-way Functions

Also known as trapdoor functions. The idea is for the function to be easy to compute in one direction, but hard to invert. These are usually used in asymmetric cryptography.

1.5.2 Algorithms

RSA

In this algorithm, the plaintext is divided into blocks, which are treated as a number. The keys are generated as follows:

- Choose two prime numbers p and q
- Define $n = pq$ and $z = \phi(n) = (p-1)(q-1)$
- Choose $e < n$ such that e is coprime with z
- Calculate d such that $ed \mod z = 1$
- The public key is $K_u = (e, n)$, and the private key is $K_r = (d, n)$

To encrypt, compute:

$$E(K_u, m) = m^e \mod n = c$$

And to decrypt:

$$D(K_r, c) = c^d \mod n = m$$

RSA is often used to produce signatures. To sign M we calculate $S = (hash(M))^d \mod n$ and to verify we check $hash(M) == S^e \mod n$. Only the owner of the private key can sign, but anyone with the public key can verify the signature.

ECC

Elliptic curve cryptography offers the same security as RSA with smaller bit sizes. In this case, the "hard" problem is the elliptic curve logarithm:

- $Q = k \cdot P$, where Q, P belong to an elliptic curve (e.g. $y^2 = x^3 - 3x + b \mod p$)
- Easy to compute Q, given k and P
- Hard to find k, given Q and P
- P is a base point (parameter of the curve)

1.5.3 Digital Signatures

Digital signatures use an asymmetric cipher and a hash function. The basic algorithm is as follows:

- Sign: $S(doc) = E(K_r, hash(doc))$
- Validate: $D(K_u, S(doc)) == hash(doc)$

Today often ECDSA is adopted (with elliptic curve encryption).

1.6 Public Key Management

In the past, public keys were distributed by means of public directories or announcements. Currently, the distribution is done through the use of digital certificates.

1.6.1 Digital Certificates

Certificates are documents signed by a certification entity (CA, public organization or company)

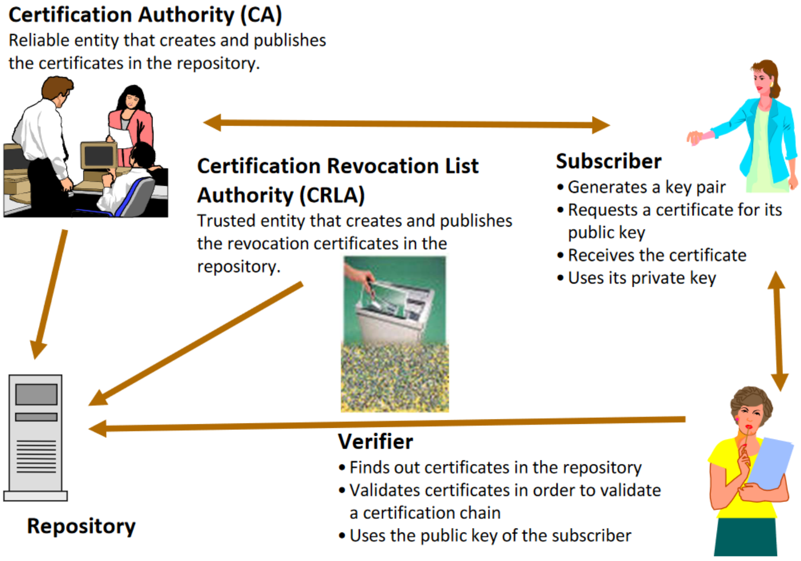
- Public documents
- Have a digital signature
- Used to distribute public keys through unsecure channels
- Contain version, serial number (of the CA), issuer, validity, etc.

Certificate Issuance

- Keys to assure confidentiality
  - The public key of X is used by the sender to assure confidentiality of the data sent to X
  - The private key of X is used to decipher the received information
  - These keys can be refreshed frequently
- Keys to assure authentication
  - The private key of X is used to sign the content
  - The corresponding public key to validate the signature
  - These keys should not be renewed frequently

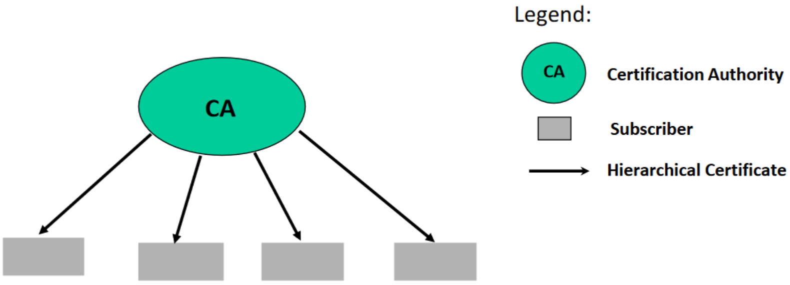
Certificate Distribution

There is a PKI (Public Key Infrastructure) to manage certificates in a certain context (e.g. the web). It encompasses a set of CAs (and similar entities), policies and mechanisms, and supports several operations, like the creation of key pairs or distribution of certificates.

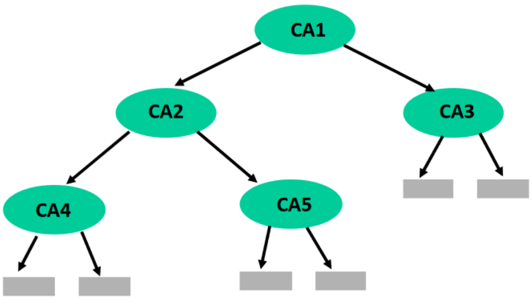

Certification Authority (CA)
Reliable entity that creates and publishes the certificates in the repository.
Certification Revocation List Authority (CRLA)
Trusted entity that creates and publishes the revocation certificates in the repository.
Subscriber
• Generates a key pair
• Requests a certificate for its public key
• Receives the certificate
• Uses its private key
Verifier
• Finds out certificates in the repository
• Validates certificates in order to validate a certification chain
• Uses the public key of the subscriber
Repository

A CA can establish trust relations by issuing public key certificates of other CAs or by requiring certification of its public key to other CAs. Some typical trust relations include:

- Flat: only the root CA is trusted. Verifying entities check the certificates validity with the public key of the CA.

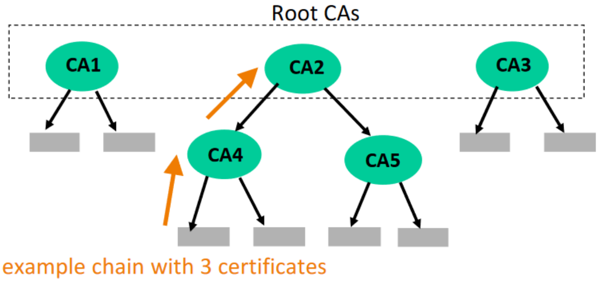


- Hierarchical: there is a tree of CAs. CAs issue certificates to subscribers and other CAs and verifying entities verify the certificates of the subscribers by sequentially checking the certificates up to the root certificate (verifying entitites trust CA1)



The use of intermediary certificates protects the PKI root certificate and delegates the signing authority to another organization (better for scalability)

- List of certificates: The verifying entities trust the keys of several root CAs. The verifying entities validate the chain of certificates that lead to any of the CAs



Browsers and operating systems have root stores with lists of trusted CAs.

Certificate Revocation

A certificate may be withdrawn for several reasons:

- Corresponding private key compromised
- Certificate owner does not operate service any longer
- Key ownership has changed
- ...

Two approaches may be used:

- Certificate revocation lists (CRL)
  - Lists of revoked certificates
  - Should be regularly checked by the certificate holders
  - Intermediate certificates should be checked too
  - CRLs can become large

- Online Certificate Status Protocol (OCSP)
  - Allows live revocation checks over the network
  - Uses a request-response model
  - Information must be fresh
  - Servers must have high availability

There is also a method called OCSP stapling. The idea is that the web server obtains a fresh OCSP response and "staples" it to the certificate given to the web browser. This is more efficient and secure, but not universally supported.

1.7 Secret Key Management

The management of secret keys deals with several aspects, such as key generation, distribution and lifetime.

1.7.1 Key Generation

Adequate random values generators should be used for appropriate key generation. Several sources of entropy may be used to create a random number:

- Ticks since boot
- Current time
- Low-level system info
- ...

Key size depends on the algorithm's strength, lifetime and usage of the key, as well as the attacker's power.

1.7.2 Key Distribution

Perfect forward secrecy (PFS) is a desirable characteristic of a key agreement protocol. It gives assurance that session keys will not be compromised even if the private key of the server is compromised.

PFS protects past sessions against future compromises of keys

Manual Distribution

Keys may be manually distributed through physical support. Usually done for personal or large sets of keys.

Distribution With Shared Values

Distribution with long-term shared secrets is useful, allowing the exchanging temporary secrets between entities that already share some secret information. The exchange has the form:

$$A \to B : \{K_s\}_{KEK}$$

Where the secret key is encrypted using a symmetric cipher, with the KEK key, only known by A and B. B verifies the message freshness and contents. This method does not assure PFS, as the disclosure of KEK reveals all session keys that have been exchanged between the communicating parties.

Instead of using a shared key, the receiver's public key may also be used.

Distribution Without Shared Values

The Diffie-Hellman algorithm is often used in scenarios with no shared values.

In this algorithm, the objective is to obtain a secret number K, shared between A and B, but without communicating it in clear.

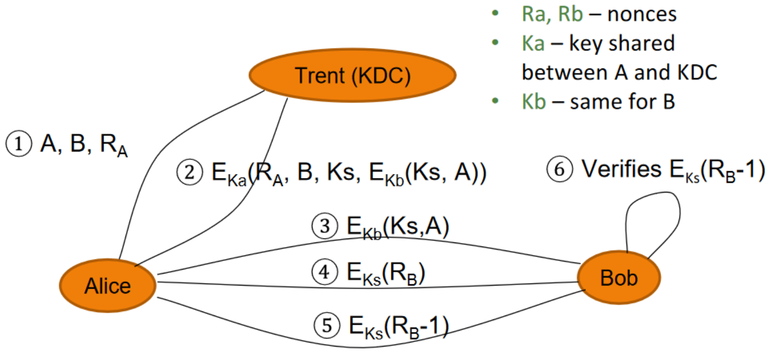
- Choose two public prime numbers m and n (n large).
- A generates a random number x_a and calculates $y_a = m^{x_a} \mod n$.
- B generates a random number x_b and calculates $y_b = m^{x_b} \mod n$.
- y_a and y_b are made public.
- Each party calculates K locally:

$$K = y_b^{x_a} \mod n = y_a^{x_b} \mod n$$

This type of distribution does not provide authentication and is vulnerable to man-in-the-middle attacks. However, if both secret values a and b are ephemeral (e.g. used only once), then there is PFS.

Distribution With Third Parties

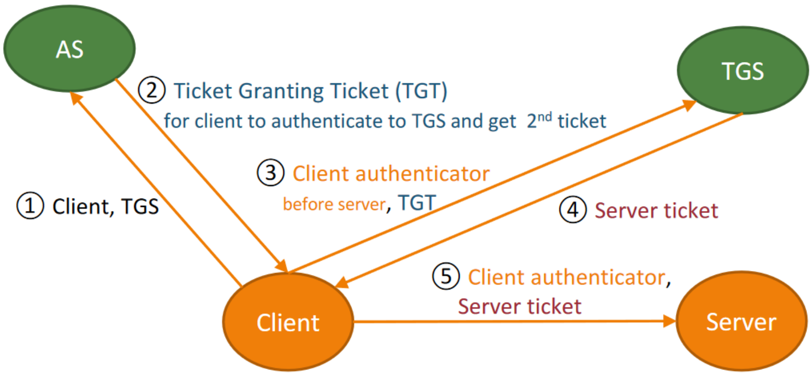
Distribution may be done through means of trusted third parties, such as a key distribution center. These centers act as mediators between the communicating parties, simplifying the management of long-term shared secrets.



This type of distribution assumes authentication and does not assure PFS.

Kerberos is another frequently used algorithm. It uses a ticket granting service

(TGT) that provides time-limited credentials (tickets) for several services/servers and an authentication service (AS), allowing clients to login.



Where tickets include client and server identifiers, timestamps with the beggining and end of validity and the session key value, all ciphered with the server key.

The authenticator has the client identifier and a timestamp of the request, ciphered with the session key.

Chapter 2

Networks

2.1 Network Models

2.1.1 OSI and Internet

There are seven layers in the OSI model:

1. Physical
2. Data Link
3. Network
4. Transport
5. Session
6. Presentation
7. Application

The network is not just computers and servers, there are also:

- Hubs: Send signals everywhere
- Switches: Send frames only where they need to go
- Routers: Look at the IP address from the incoming packet and forwards it. Have MAC address too.
- Gateways: Access point to other networks, with possible change of addressing and networking technology.

2.1.2 Address Resolution

In layers 2 and 3, MAC and IP addresses are used, respectively. IP addresses identify the network and the machine and MAC addresses may be converted to IPs with the ARP protocol. Some network ranges were reserved for private addressing.

2.2 Network Vulnerabilities

2.2.1 Physical Layer

Hubs

Hubs broadcast information on a shared medium, thus, are at threat of information leakage through sniffers.

Anyone can connect to a hub even if it is physically secure.

Sniffers

Usually, network adapters operate in a non-promiscuous mode (only listen to what is sent to their MAC). Sniffers, on the other hand, read all frames, regardless of MAC.

It is possible to indentify sniffers using tools (with several methods: latency, dns, os-specific, ...) or using the ARP method:

- Machines cache ARPs
- Send a non-broadcast ARP with our correct MAC address
- Then send a broadcast ping with the right IP but wrong MAC address
- Only a machine which has our correct MAC address from the sniffed ARP will respond

To prevent sniffing, switches may be used instead of hubs (does not fully solve). It is also possible to prevent the effectiveness of sniffing by using one-time passwords and encryption.

2.2.2 Data Link Layer

Switches typically send frames only to the destination MAC address, and thus, reduce the sniffing problem. There are, however, some ARP vulnerabilities present:

- MAC flooding: attacker sends several unsolicited ARP messages, overwhelming the switch with entries. When the table is filled, some switches stop accepting connections and others revert to Hub mode
- ARP Spoofing/Poisoning: An attacker sends a non-requested ARP message with a false IP-MAC address correspondence. ARP messages are in no way signed, so it is easy to falsify a message from any given MAC

Some preventive measures include the use of tools like *arpwatch*, to monitor the ARP to IP translation and the use of switches with fixed tables (with a cost in flexibility).

2.2.3 Network Layer

Routers support the indirect delivery of IP datagrams through the use of routing tables. Some threats in this layer include:

- Packet integrity: Data is not authenticated, so an attacker can change the source address of IP packets
- Information lead and DoS: Users have little to no guarantee concerning the routing path taken by the packets. An attacker might corrupt the routing tables by sending routing-update messages and effectively hijack the route

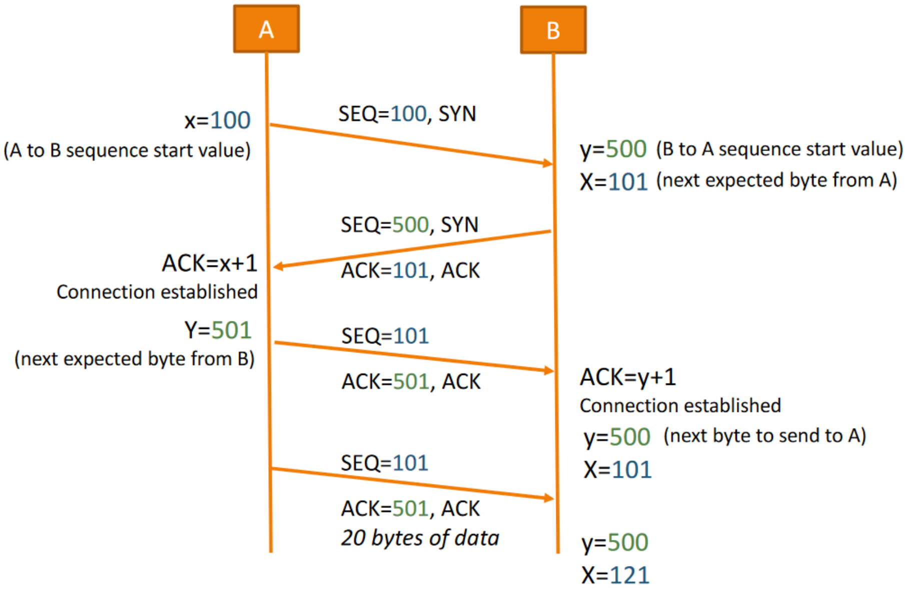
2.2.4 Transport Layer

UDP

UDP can be used to send and receive individual packets, without an established connection. It's just a thin addition to IP and is vulnerable to the same attacks.

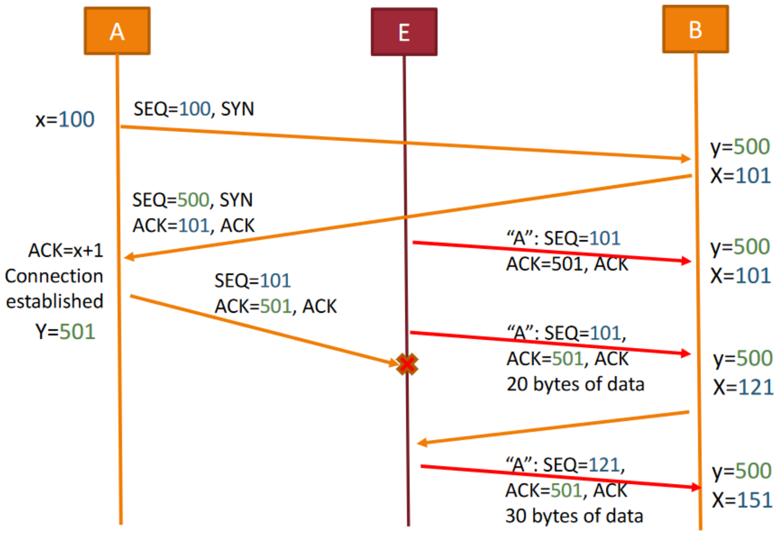
TCP

TCP can be used establish a connection to send and receive a data stream of bytes. An example of a TCP handshake:



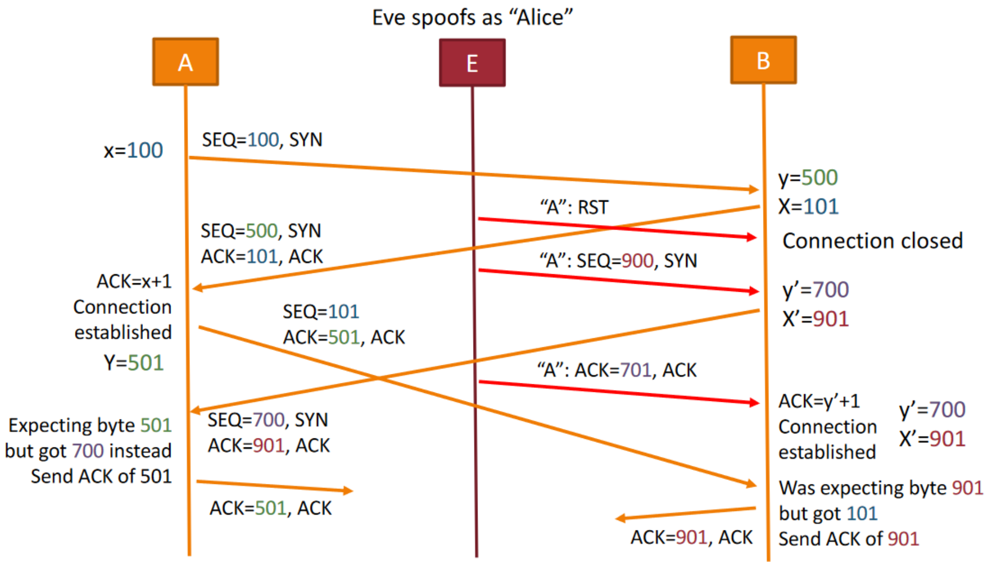
There are different techniques for TCP hijecking, depending on the attacker's capability to intercept communications:

- Full adversary-in-the-middle: the attacker is positioned to fully intercept the communication and can intercept the sequence numbers and take over the connection. The tool *shijack* is capable of doing this.



Eve can discard messages (some or all) and can send some messages as if she were Alice or Bob

- Weak adversary-in-the-middle TCP hijack: the attacker can only eavesdrop and spoof packets (cannot drop packets). Attacker must now exploit de-synchronization between hosts. Once the sender and the receiver are desynchronized, only the attacker can create data segments with correct numbers

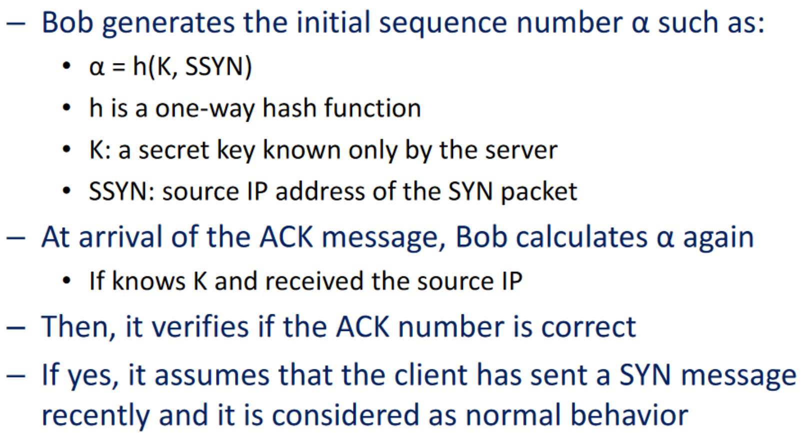


Eve knows correct sequence numbers and can send packets that will be accepted

The de-synchronization can be forged during the creation of a TCP/IP connection with a reset and with false acknowledgements. It can also be done for an already established connection by sending blank data to displace sliding windows.

- Blind TCP hijack: The attacker cannot capture return traffic from the host connection and only blindly sends malicious or manipulated packets. The attacker does not receive any confirmation of the desired effect through a packet capture and for the attack to be successful, the attacker must guess the sequence numbers of the TCP packets

Another common TCP attacks is syn flooding. It consists of overloading a host with incomplete TCP/IP connection requests. There is no definite solution for IPv4 SYN flooding; SYN cookies may also be used to mitigate flooding:



- Bob generates the initial sequence number α such as:
  - α = h(K, SSYN)
  - h is a one-way hash function
  - K: a secret key known only by the server
  - SSYN: source IP address of the SYN packet
- At arrival of the ACK message, Bob calculates α again
  - If knows K and received the source IP
- Then, it verifies if the ACK number is correct
- If yes, it assumes that the client has sent a SYN message recently and it is considered as normal behavior

Cooperation with firewalls and attack detectors is also helpful:

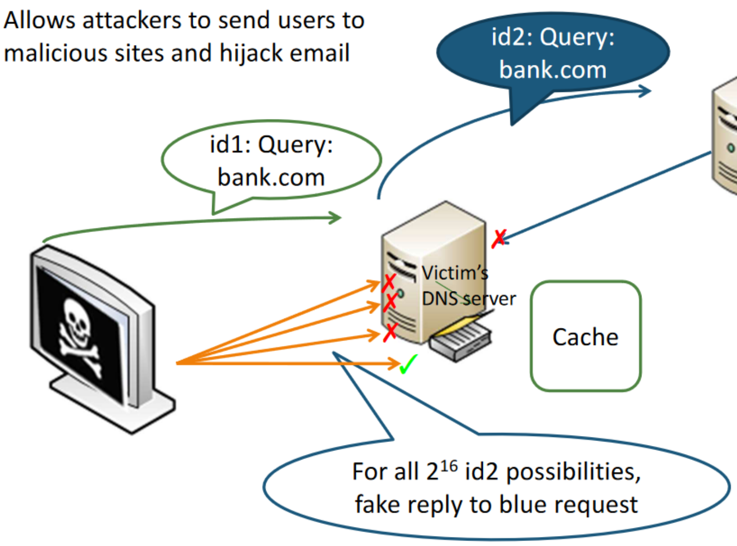
- Handshake relay: firewall stands in front of server and protects it until the handshake is complete

- Gateway: firewall keeps the connection alive on server and terminates it if the client leaves the connection open but without traffic

2.2.5 Application Layer

DNS

DNS translates domain names to IP addresses. In 2008 the Kaminsky attack was found:


Allows attackers to send users to malicious sites and hijack email
id2: Query: bank.com
id1: Query: bank.com
Victim's DNS server
Cache
For all 2^{16} id2 possibilities, fake reply to blue request

The attack is successful if it can guess the query ID value.

DNSSEC is DNS with digitally signed responses. Each zone has its own key-pair for signing and public Keys are published in the DNS itself. DNSSEC provides integrity and authenticity for RRs of the signed zones.

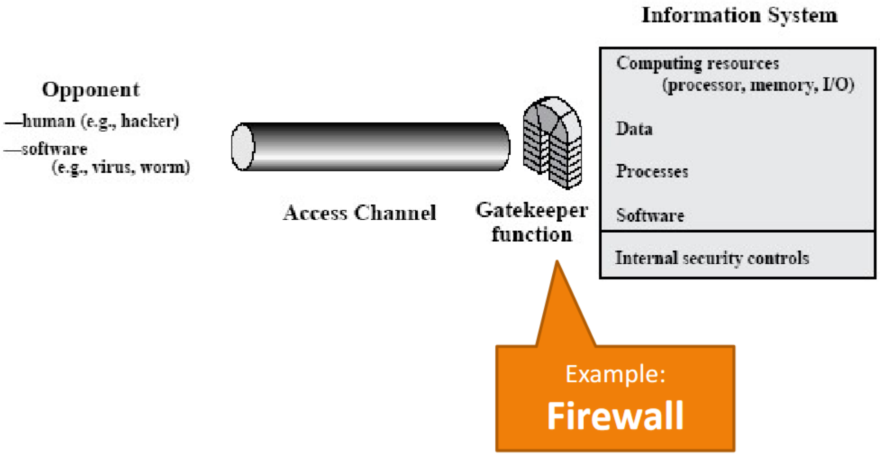
RCE

Remote code execution is a class of software security vulnerabilities. Most programming languages have some way to generate code in runtime and execute it, which has the potential of being abused by a malicious actor.

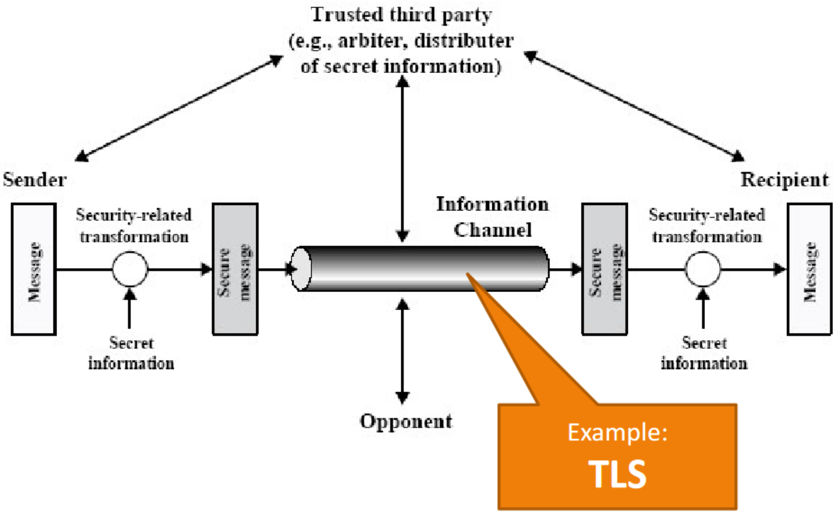
To prevent code injection, one must avoid using data as code as much as possible, as well as sanitize inputs.

2.3 Network Security Models

Gatekeeper for access control:



Secure communication channel:



2.4 Firewalls and Intrusion Detection Systems

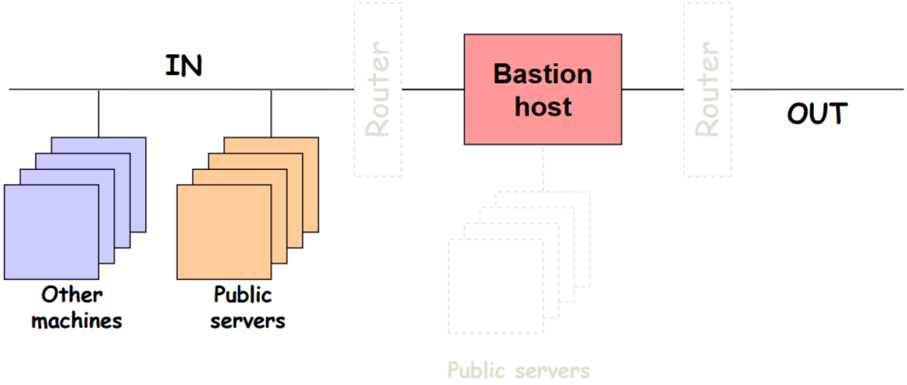
A Firewall is a means of protecting a local system or network of systems from network threats, creating a perimeter of defense.

Incoming TCP and UDP traffic should be denied by default and malformed packets should be blocked.

2.4.1 Firewall Placement

Dual-Homed Host Firewall

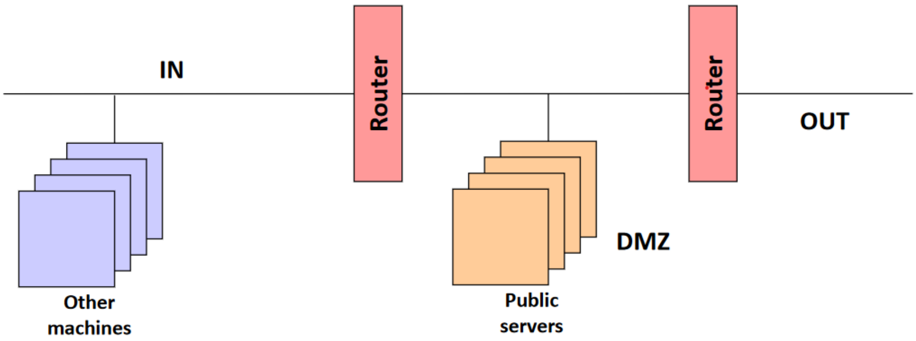
This architecture uses a bastion host with 2 home networks: IN and OUT



It's a simple an resource-efficient topology, but has some disadvantages: compromising the machine deactivates the firewall, all the processing load is in a single machine, and public servers are within the protected network.

Screened subnet

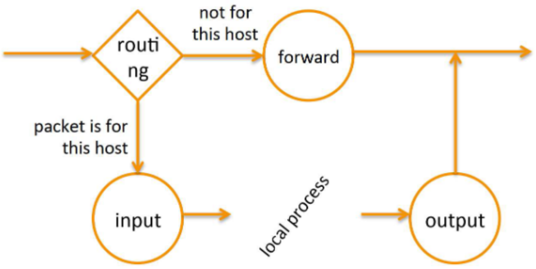
This approach also has two internal subnets: IN and DMZ.



There is a lower risk regarding the public services and of the firewall being compromised. However, there is also less control over the activities going on in the DMZ machines.

2.4.2 Iptables

Iptables has 3 chains (lists of rules): input, output forward. When a packet reaches a chain, the chain's rules decide the packet's fate: drop or accept.



2.4.3 Types of Firewall

There are several types of firewall:

- Packet filter:
  - Reject packets depending on the IP and transport layer headers
  - Can be stateful or stateless
- Circuit-level gateway:
  - Control iterations at the transport layer (typically TCP "circuit")
  - Similar to application-level gateways
- Application-level gateway:
  - Control iterations at the application layer
  - Protocol-specific proxy

Packet filters are faster but harder to configure and are unable to protect against "misbehaving" protocols. On the other hand, application-level gateways are slower but easier to configure, allow for more authentication mechanisms and more fine-grained control, with the disadvantage of being less adaptable to new protocols.

2.4.4 Intrusion Detection Systems

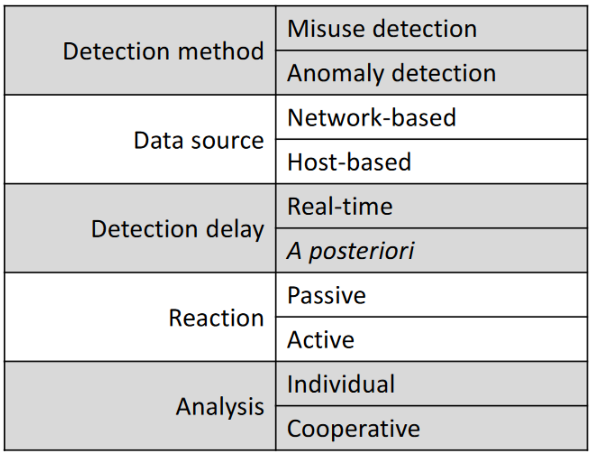
An IDS is a software that has the function to detect, identify, and respond to unauthorized or abnormal activities in the targeted system. IDS' complement firewalls and are capable of:

- Deep packet inspection (look at packet contents)
- Examine correlation among multiple packets

IDS' generate alarms and, as such, are prone to generate false positives/negatives.

2.4.5 IDS classification

There are several criteria to classify IDS:



Detection method	Misuse detection
	Anomaly detection
Data source	Network-based
	Host-based
Detection delay	Real-time
	A posteriori
Reaction	Passive
	Active
Analysis	Individual
	Cooperative

- Detection method:
  - Misuse detection: system activity analysis in search of known attack patterns (attack signatures)
  - Anomaly Detection: matches observed behavior with a model of normal behavior
- Data source:
  - Network-Based IDS: capture and do traffic analysis on network data (e.g., packets)
  - Host-Based IDS: capture and do analysis on host data
- Reaction:
  - Passive: only detect and report the detection results
  - Active: respond to attacks

2.5 SSH and TLS

SSH is a secure communication application and protocol over TCP. Allows secure remote sessions, file transfer and tunneling of traffic, while providing confidentiality and integrity.

It has server authentication, being a trust-on-first-use model (vullnerable to man-in-the-middle). SSH also uses Diffie-Hellman for key exchange. User authentication may be done through a password or a signature with the client's private key.

2.5.1 SSH Tunneling

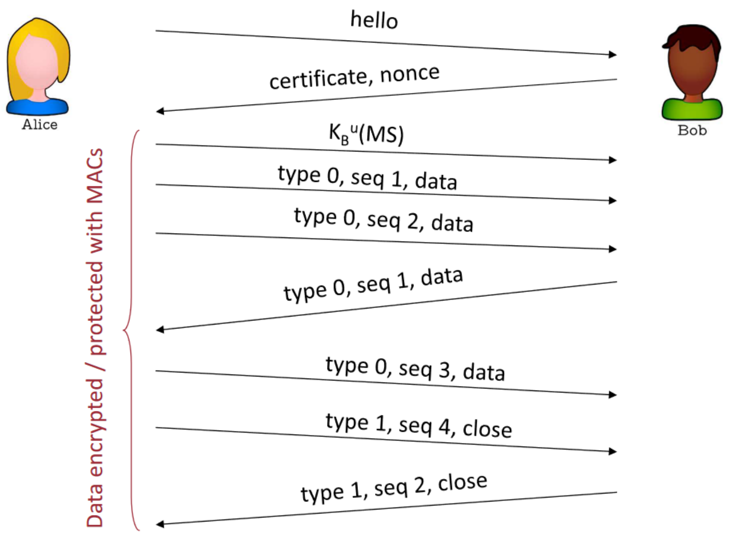
In the client machine, a mapping is created between a local TCP port and a port in the remote machine. The client application is configured to use the local port and, when using it, it will securely interact via SSH.

Tunneling is transparent for apps, but complex for admins and provides low flexibility for devs and users.

2.5.2 TLS

TLS provides secure channels over TCP/IP, authentication, integrity, confidentiality, and key distribution.

- Handshake:
  - Alice and Bob use their certificates, private keys to authenticate each other and exchange a shared secret
- Key derivation:
  - Alice and Bob use shared secret to derive set of keys
- Data transfer:
  - Data to be transferred is broken up into series of records
  - Data records: the stream is broken into a series of records, each with its length and MAC
  - Sequence numbers are used to prevent replay attacks and the order of records
  - Record types are used to prevent truncation (early closure) attacks. Type 0 is for data and type 1 is for closure

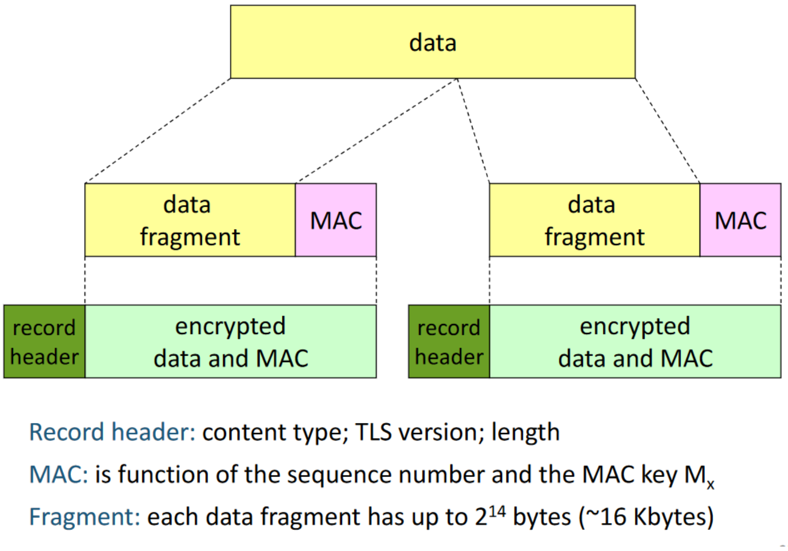


2.5.3 TLS Protocols

There are two TLS protocols:

- Record protocol: creation and verification of secure messages
- Handshake Protocol: exchange of identity and supported cipher suites, authentication of the communicating parties and key distribution

The record protocol can be described as follows:

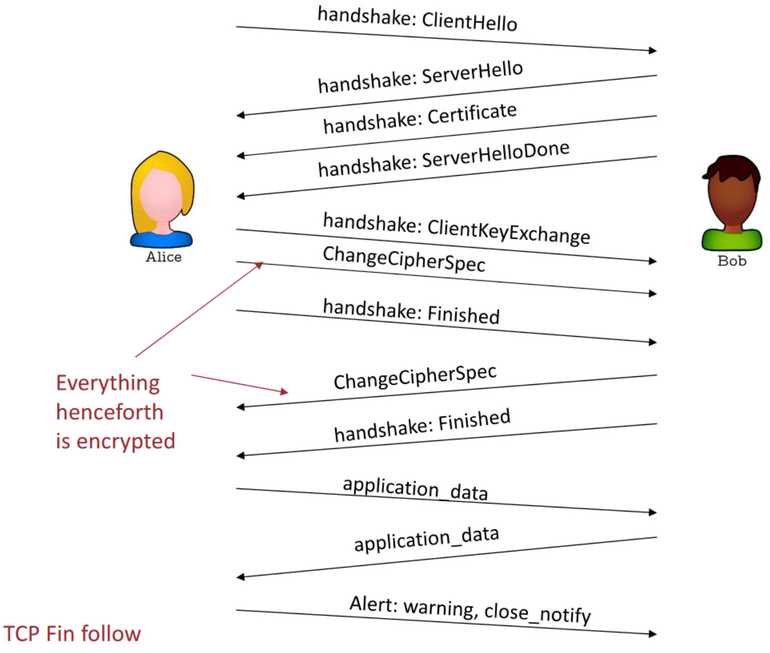


Record header: content type; TLS version; length

MAC: is function of the sequence number and the MAC key M_x

Fragment: each data fragment has up to 2^{14} bytes (~16 Kbytes)

And the handshake is as follows:



2.5.4 Key Derivation

The client nonce, server nonce, and pre-master secret input into pseudo random-number generator, produce a master secret.

Master secret and nonces input into another pseudo random-number generator, produce key block, cut into several other keys, such as the server and client MAC and encryption keys and initialization vectors.

2.6 Authentication

Authentication services must handle problems of security and trustworthyness. Their end goal is one of entity (user, machine, service) authentication, facilitating protocol usage and preventing impersonation from attackers.

2.6.1 Authentication With Passwords

This type of authentication requires the user to memorize a password and provide it when prompted. It is simple but passwords may be poorly selected by users, making them susceptible to password attacks.

A key derivation function (KDF), such as hashing, is often used when storing passwords in order to prevent their hijacking by attackers. A modern password-based KDF is PBKDF2, which uses a cryptographic hash function, salt and a high iteration count.

One-Time Passwords (OTP)

This variation requires the validation of a one-time password regarding a secret stored for the user. Such a password is only valid once and thus, pointless for an attacker to capture.

Has better security in remote authentications over insecure networks or in insecure terminals, however, some implementations may require the users to use some device/application to generate the one-time passwords.

Challenge-Response

This method requires the authenticator to provide a challenge and the user being authenticated transforms the challenge using the secret that he shares with the authenticator.

Its secure for authentication over unsecure networks, but the authenticator needs to have shared secrets with all users.

2.6.2 Biometric Authentication

The user is validated through biometrical data: fingerprint, physiognomy,... It solves the problem of choosing a good password and does not rely on the user's memory, but can be deceived sometimes and does not allow the transference of authentication between subjects.

The principle of biometric authentication has two phases:

- Registration: acquisition of information about the person
- Authentication: checking the obtained biometric data with the stored template

It should fulfill the following desirable properties: universality, unicity, stability, correctness, convenience and acceptance.

2.6.3 Authentication Protocols

Password Protocols

There is PAP and CHAP, both protocols are used by PPP (Point-to-Point Protocol). Authentication is unidirectional (A authenticates U)

- PAP: simple exchange of pair UID/password. Insecure, as the password is sent in plain text:

$$U \rightarrow A : username,\ password \tag{2.1}$$
$$A \rightarrow U : OK/not\ OK \tag{2.2}$$

- Challenge Handshake Authentication Protocol (CHAP):

$$U \rightarrow A : username \tag{2.3}$$
$$A \rightarrow U : authID,\ challenge \tag{2.4}$$
$$U \rightarrow A : Hash(authID,\ passoword,\ challenge) \tag{2.5}$$
$$A \rightarrow U : OK/not\ OK \tag{2.6}$$

  In this version, A stores the passwords, however, that problem may be solved by using only a hash of the password.

One-Time Password Protocols

- S/Key: Authenticator gives user a sequence of one-time passwords

$$OPT_1 = Hash(seed, password)\ ...\ OPT_n = Hash(OTP_{n-1})$$

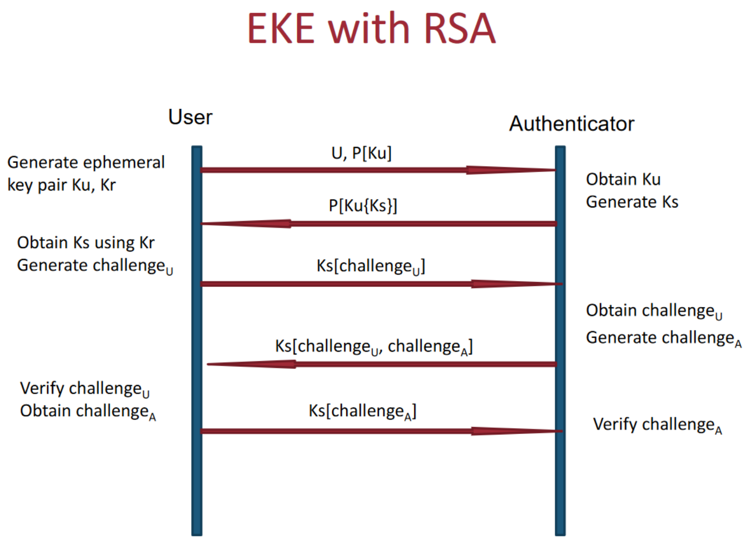
  For each user, the authenticator stores only the seed of the one-time password sequence and the current index. The authentication process is as follows:

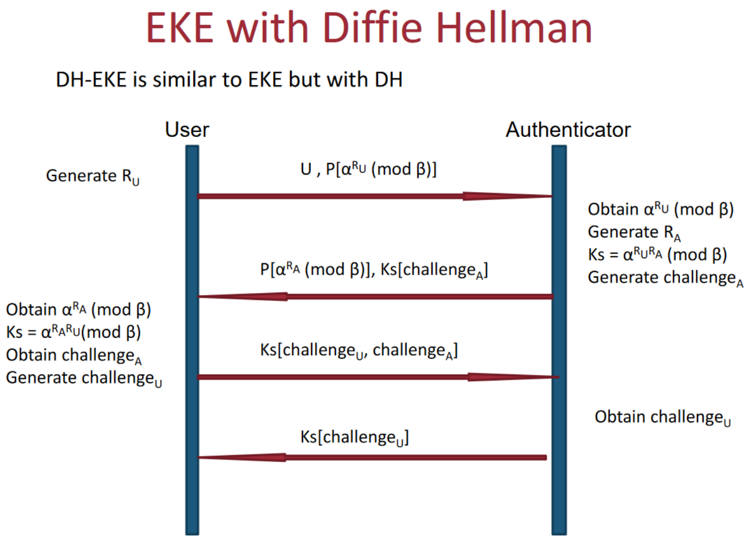
  - Authenticator sends index to the user
  - User sends $P = OPT_{index-1}$ to the authenticator (the one-time password)
  - Authenticator computes $Hash(P)$ and compares with stored OPT_{index}
  - If values are equal, then success, server stores $index-1$ and $OPT_{index-1}$
  - On the first time, the authenticator sends $index = n$, next $index = n - 1$, etc.

- RSA SecurID: generates a unique number every minute, essentially OPTs. The authentication process is as follows:
  - The user generates a one-time password, combining a PIN with the card number
  - The RSA server performs the same operation and verifies if the values are equal
  - Server is synchronized with real time (RSA Security Time Synchronization)

Asymmetric Keys Protocols

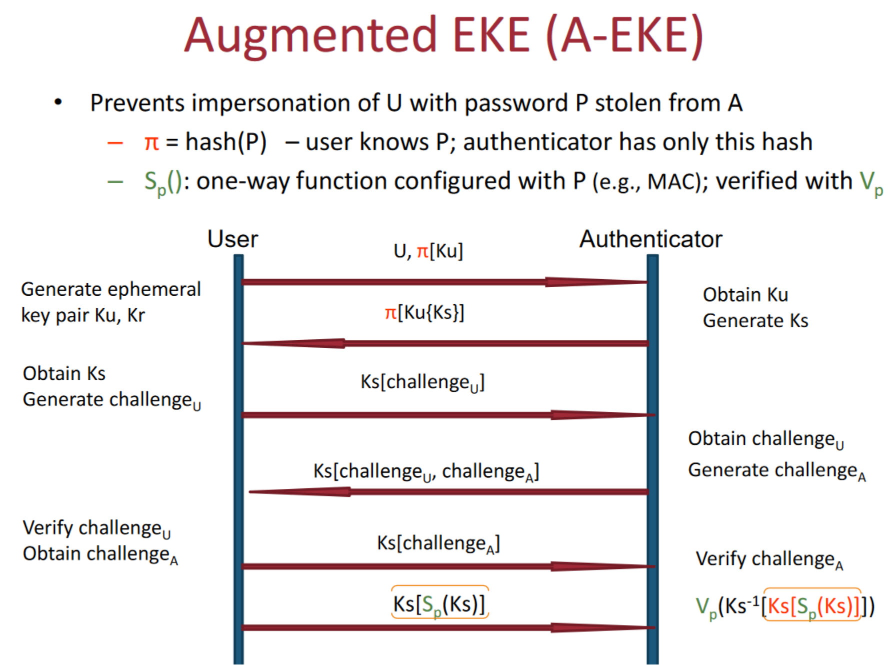
The main protocol used with asymmetric keys is EKE. It does a password-authenticated key agreement and is resistant to dictionary attacks.




EKE with Diffie Hellman
DH-EKE is similar to EKE but with DH
User
Authenticator
Generate R$_U$
U , P[α^{R_U} (mod β)]
Obtain α^{R_U} (mod β)
Generate R$_A$
Ks = $\alpha^{R_U R_A}$ (mod β)
Generate challenge$_A$
P[α^{R_A} (mod β)], Ks[challenge$_A$]
Obtain α^{R_A} (mod β)
Ks = $\alpha^{R_A R_U}$(mod β)
Obtain challenge$_A$
Generate challenge$_U$
Ks[challenge$_U$, challenge$_A$]
Obtain challenge$_U$
Ks[challenge$_U$]

It's safe from dictionary attacks since, even if an attacker was successful decrypting the public key, it would still have to compute the private key to obtain the the session key (i.e. decrypt $K_u\{K_s\}$).

It is also protected against man-in-the-middle and replay attacks. However, an attack to the password database in the authenticator would give access to P. The solution consists in only storing $Hash(P) = \pi$:



Augmented EKE (A-EKE)

- Prevents impersonation of U with password P stolen from A
  - π = hash(P) – user knows P; authenticator has only this hash
  - $S_p()$: one-way function configured with P (e.g., MAC); verified with V_p

User

Authenticator

Generate ephemeral key pair Ku, Kr

U, π[Ku]

Obtain Ku
Generate Ks

π[Ku{Ks}]

Obtain Ks
Generate challenge_U

Ks[challenge_U]

Obtain challenge_U
Generate challenge_A

Ks[challenge_U, challenge_A]

Verify challenge_U
Obtain challenge_A

Ks[challenge_A]

Verify challenge_A

Ks[S_p(Ks)]

$V_p(Ks^{-1}[Ks[S_p(Ks)]])$

The *Sp* function could be, for example, a cryptographic hash function combined with a unique salt for each user and the *Vp* function could be a time-constant comparison to mitigate timing attacks.

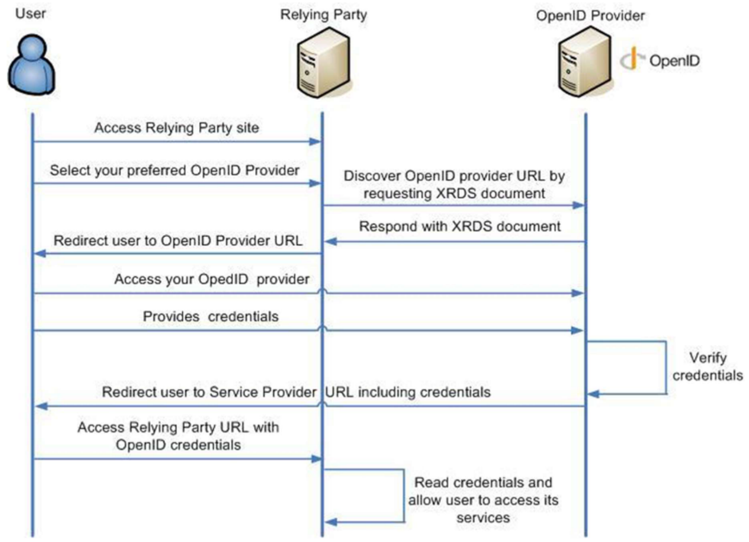
Web Authentication Protocols

- HTTP Basic authentication: when user asks for protected page, browser asks user for username and password for that page and sends them encoded in Base64
- HTTP Digest authentication: server sends a similar digest or nonce, the browser sends a hash instead of credentials
- In HTTPS, SSL and TLS are used
- FIDO2 user authentication: based on user-controlled cryptographic authenticators, such as a smartphone or hardware security key
- Session-based authentication: server sends cookie with session identified to the client. Whenever browser sends request to the server, cookie is inserted automatically

Web Single-Sign On Protocols

Single-sign on was developed as a solution for users who need to authenticate themselves in many devices, managing several passwords.

OpenID is the main single-sign on protocol. When a user wants to login in a relying party (OpenID supporter), the following exchange occours:


User
Relying Party
OpenID Provider
OpenID
Access Relying Party site
Select your preferred OpenID Provider
Discover OpenID provider URL by requesting XRDS document
Respond with XRDS document
Redirect user to OpenID Provider URL
Access your OpedID provider
Provides credentials
Verify credentials
Redirect user to Service Provider URL including credentials
Access Relying Party URL with OpenID credentials
Read credentials and allow user to access its services