Computer and network security

Beau De Clercq

2020-2021

Contents

1 Introduction

2 Symmetric ciphers

3 Message authentication

3.1 Hash functions

A hash function H is a function that takes input data blocks of length M and returns a hash value of fixed size R.
A cryptographic hash function that also satisfies following conditions:

- One way property: it should be infeasible to find a data object that maps to a predefined hash value.
- Collision free property: it should be infeasible to find 2 data objects that map to the same hash value.
- Use padding to pad up input to fixed length and add the length l of the block in bits.

By satisfying the first two properties, hash functions can be used to determine if data has been altered.

3.1.1 Applications

Hash functions can be used in an number of applications:

- Message authentication: to ensure a message hasn't been altered.
- Digital signatures: ensure the authenticity of messages and identity of the sender.
- One-way password file: store hash value of password in plain text file.
- Intrusion/virus detection: store H(f) for each file to determine if files have been modified.
- Pseudorandom function: use H to generate pseudorandom private key.

3.1.2 Security requirements

Cryptographic hash functions must adhere to following security requirements:

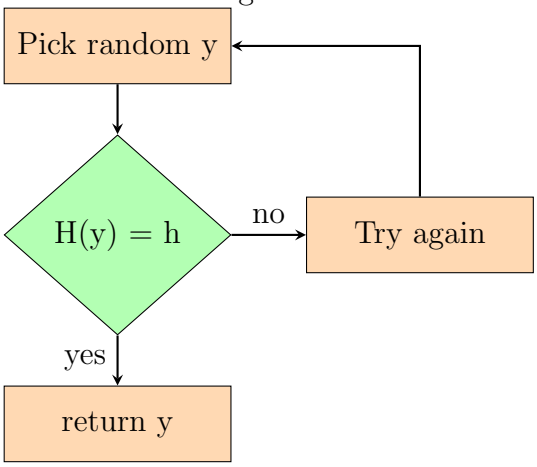
- Basic:
  - Input data can be of any size

  - Output is of fixed length
  - H(x) is easy to compute

- Advanced:
  - Given h, it is hard to find y: H(y) = h
  - It is hard to find y: y≠x & H(y)=H(x)
  - It is hard to find (x, y): H(x)=H(y)

3.1.3 Attacks

- Brute force preimage attack
  The goal of this attack is to find a y such that H(y) = h for a given hash value h.
  The attack itself goes as follows:

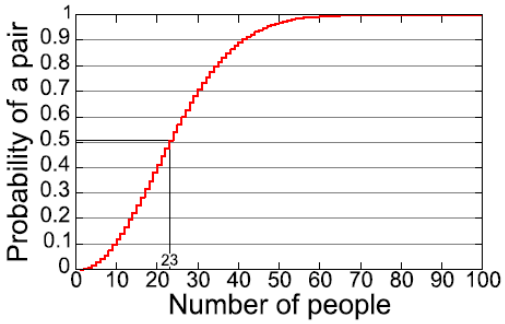


  On average this attack need 2^{m-1} attempts for an m-bit hash value.

- Brute force collision resistance attack
  The goal of this attack is to find two values x and y such that H(x) = H(y). This attack needs $2^{\frac{m}{2}}$ attempts for an m-bit hash value.

- Circumvent the collision resistance properties
- Goal: Find x and y such that $H(x) = H(y)$
- Using the birthday paradox, it can be shown that on average only $\sqrt{2^m} = 2^{m/2}$ attempts are needed (for an m-bit hash value)

Birthday paradox



Among 23 people, there is a 50% chance that two share a birthday

Generalization

Choosing random variables from a uniform distribution in the range of 0 to N – 1, the probability of a repeated element exceeds 0.5 after $\sqrt{N}$ choices have been made

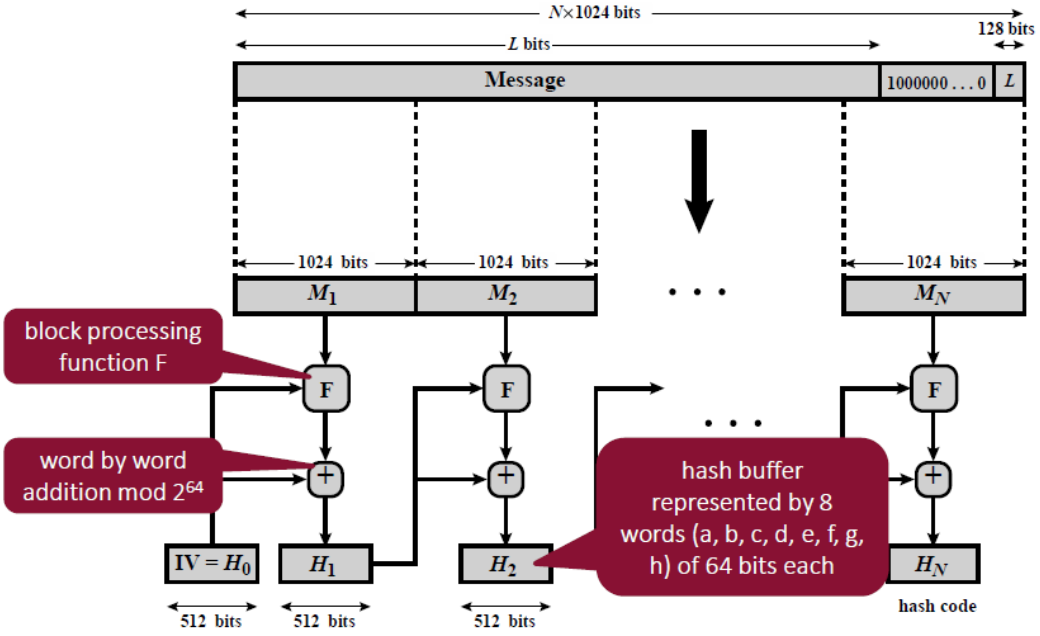
3.2 Secure Hash Algorithm (SHA)

The original SHA hash function family (sizes in bits)

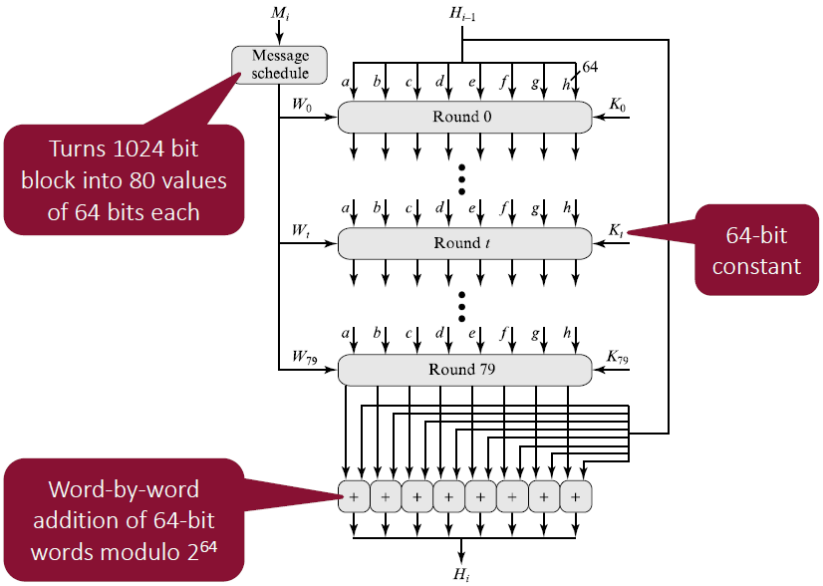
	SHA-1	SHA-2					
		SHA-224	SHA-256	SHA-384	SHA-512	SHA-512/224	SHA-512/256
Hash size	160	224	256	384	512	224	256
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	512	1024	1024	1024	1024
Word size	32	32	32	64	64	64	64
Rounds	80	64	64	80	80	80	80

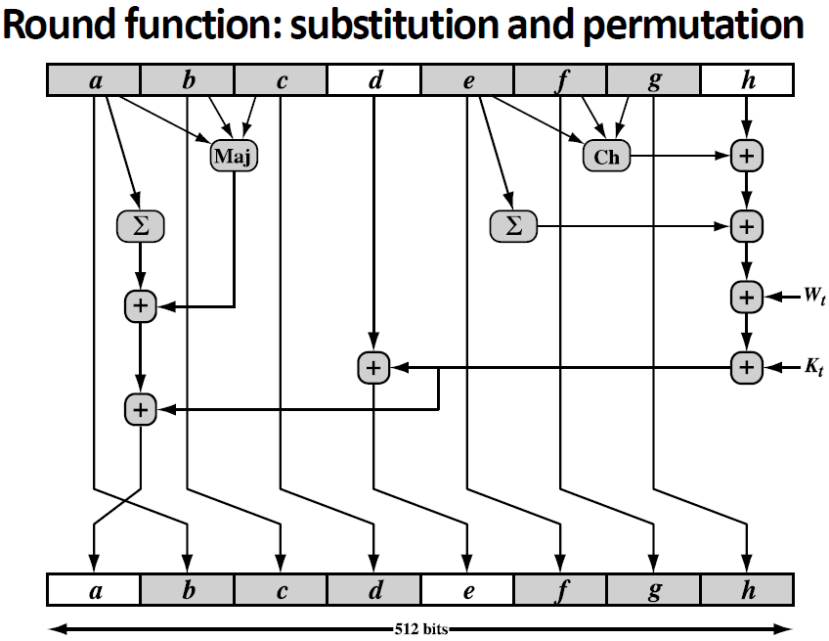
3.2.1 SHA512

SHA-512 overview



3.2.2 SHA512 block processing





In this 80 round process, a message of 1024 bits is transformed into 80 values of 64 bits each which are then individually processed in sequence by means of a round function.

After the final round a word-by-word addition 64-bit words mod2^{64} is per-

formed to obtain the final hashed value.

3.2.3 Message schedule

The first 16 words $W_0..W_{15}$ are derived directly from the input block M_i. All other words are derived as follows:

$$
\begin{aligned}
W_t &= W_{t-16} \boxplus \delta_0(W_{t-15}) \boxplus W_{t-7} \boxplus \delta_1(W_{t-2}) \\
\delta_0(x) &= ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x) \\
\delta_1(x) &= ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x) \\
ROTR^n(x) &= \text{circular right bit-shift by n bits} \\
SHR^n(x) &= \text{right bit-shift of x by n bits with} \\
&\quad \text{padding by n zeros on the left}
\end{aligned}
$$

3.3 Length extension attack and SHA3

3.3.1 Length extension attack

Type of attack where an attacker can use H(M_1) and the length of M_1 to calculate H($M_1|M_2$) for an attack controlled message M_2 without needing to know the content of M_1. This is done by taking the old message and using it as 'intermediary' input in the hashing algorithm.

3.3.2 SHA3

Add image from slides

3.4 Message authentication

There are 8 types of attacks:

1. Disclosure of message content
2. Traffic analysis
3. Masquerade
4. Content modification
5. Sequence modification
6. Timing modification
7. Source repudiation
8. Destination repudiation

Items 1 and 2 can be countered using confidentiality mechanisms, eg symmetric encryption.
Items 3 to 6 can be countered by using message authentication to verify that the received message hasn't been altered.
Digital signatures can be seen as an extension to message authentication as it is an authentication technique that also includes measures to counter repudiation by the source.

A message authentication function is a function that produces an authenticator (a value to be used to authenticate a message). There are 3 classes of message authentication functions:

- Hash functions:
  - Concatenate message M with secret key S and hash the stream
  - Send message M together with hash value: if it is intercepted an attacker can't create a new hash value since the key is unknown
  - Check authentication: combine M with S, hash the stream and compare with received hash value
- Message encryption, eg using symmetric encryption, provides both confidentiality and authentication if key K is secret. If the decrypted message M is not meaningful then the message can be considered altered.
- Message authentication code (MAC): generates fixed-size cryptographic checksum based on message and secret key
  - Does not need to be reversible
  - send message with checksum: if the checksum calculated by the receiver doesn't match then the message has been altered
  - Cannot be used to provide digital signatures: a signature need to be verifiable by anyone and MAC requires the use of a secret key

3.4.1 Security requirements for MAC

- If an opponent observes M and $MAC(K, M)$, it should be infeasable to construct M' such that $MAC(K, M') == MAC(K, M)$.

- Given 2 randomly chosen messages M and M', the probability that $MAC(K, M) == MAC(K, M')$ should be 2^{-N} for an N-bit code. $-- >$ codes should be distributed uniformly and random over the entire output space
- If M' is a known transformation of M, then $\mathbb{P}(MAC(K, M) = MAC(K, M') = 2^{-N}$.

3.5 Message authentication codes

3.5.1 Cipher-based MAC (CMAC)

- M is split into fixed size blocks $M_1, ..., M_N$.
- M_1 is encrypted with key K of k bits.
- Resulting block is then XORed with next input block, the result is then encrypted with the same key K. This process is repeated until block M_{N-1}.
- At the final block M_N: XOR M_N with M_{N-1} and b-bit constant K_1 derived from the original key.
- Encrypt the result using key K, then the leftmost significant bits represent the tag.
- If M is not a multiple of b: last block is padded with 10..0 and K_2 is used in stead of K_1.
- Determining K_1 and K_2:

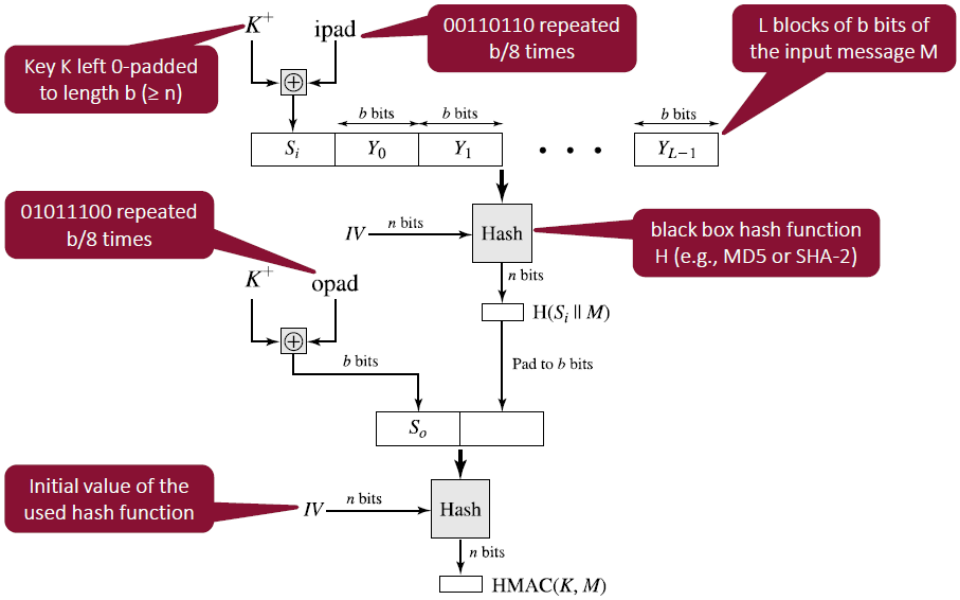
$$L = E(K, 0^b), K_1 = L * x, K_2 = L * x^2$$

where multiplication happens in $GF(2^b)$

3.5.2 Hash-based MAC (HMAC)

- Advantages: hash functions execute faster and library code is widely available.
- Main issue: a secret key needs to be used when using a hash function.

- Objectives:
  - To use, without modification, available hash functions;
  - To allow for easy replaceability of the hash function;
  - To preserve the hash function's original performance;
  - To use and handle keys in a simple way;
  - To have a well understood cryptographic analysis.
- Structure:



- Cryptographic strengths:
  - HMAC can be proven secure provided that the embedded hash function has some reasonable cryptographic strengths.
  - A successful attack on HMAC is equivalent to one of the following:
    * The attacker is able to compute an output of the compression function.
    * The attacker finds collisions in the hash function.

4 Asymmetric encryption

4.1 Introduction

Asymmetric encryption uses both a public and private key, it's working is based on mathematical functions rather than substitution and permutations. AE addresses two concerns with symmetric encryption:

- Secret keys are distributed using a trusted key distribution center
- It does not enable digital signatures

4.1.1 Confidentiality using AE

Assume there is some source A that produces a message in plaintext X which is intended for destination B. B generates a related pair of keys: a public key, PU_b, and a private key, PR_b. PR_b is known only to B, whereas PU_b is publicly available and therefore accessible by A.
With the message X and the encryption key PU_b as input, A forms the ciphertext $Y = E(PU_b, X)$. The intended receiver, in possession of the matching private key, is able to invert the transformation: $X = D(PR_b, Y)$. An adversary, observing Y and having access to PU_b, but not having access to PR_b or X, must attempt to recover X and/or PR_b. It is assumed that the adversary does have knowledge of the encryption (E) and decryption (D) algorithms. If the adversary is interested only in this particular message, then the focus of effort is to recover X by generating a plaintext estimate $\widehat{X}$. Often, however, the adversary is interested in being able to read future messages as well, in which case an attempt is made to recover PR_b by generating an estimate $\widehat{PR_b}$.

4.1.2 Authentication using AE

In this case, A prepares a message to B and encrypts it using As private key before transmitting it. B can decrypt the message using As public key. Because the message was encrypted using As private key, only A could have prepared the message.Therefore,the entire encrypted message serves as a digital signature. In addition, it is impossible to alter the message without access to As private key, so the message is authenticated both in terms of source and in terms of data integrity.

In this scheme, the entire message is encrypted, which, although validating both author and contents, requires a great deal of storage. Each document must be kept in plaintext to be used for practical purposes. A copy also must be stored in ciphertext so that the origin and contents can be verified in case of a dispute. A more efficient way of achieving the same results is to encrypt a small block of bits that is a function of the document. Such a block, called an authenticator, must have the property that it is infeasible to change the document without changing the authenticator. If the authenticator is encrypted with the senders private key, it serves as a signature that verifies origin, content, and sequencing.

4.1.3 Combining confidentiality and authentication

It is possible to provide both the authentication function and confidentiality by a double use of the public-key scheme

- $Z = E(PU_b, E(PR_a, X))$
- $X = D(PU_a, D(PR_b, Z))$

where (PU_a, PR_a) is a key pair generated by the sender and is used for authentication and (PU_b, PR_b) is a key pair generated by the receiver and is used for confidentiality.
Encryption of a message M then goes as follows: $M \rightarrow E(M, PR_a) \rightarrow E(E(M, PR_a), PU_b) \rightarrow C$.
Decryption of a ciphertext C happens as follows: $C \rightarrow D(C, PR_b) \rightarrow D(D(C, PR_b), PU_a) \rightarrow M$.

4.1.4 Requirements for public-key cryptography

- It is computationally easy to generate the key pair (PU_b, PR_b).
- It is computationally easy, given the public key PU_b and a message M, to generate the corresponding ciphertext $C = E(PU_b, M)$.
- It is computationally easy, given the private key PR_b and a ciphertext C, to recover the plaintext $M = D(PR_b, C) = D(PR_b, E(PU_b, M))$.
- It is computationally infeasible, knowing PU_b or PR_b, to generate the other key.
- It is computationally infeasible, knowing PU_b and ciphertext C, to

recover the original message M.

- Optional: the two keys can be applied in any order.

4.2 Rivest-Shamir-Adleman (RSA)

4.2.1 RSA basics

In RSA, each plaintext block M is represented by a number smaller than n, resulting in a block size of at most ${}^2log(n) + 1$ bits. In practice a block size of i bits is used with $2^i \leq n \leq 2^{i+1}$.
Encryption and decryption happen according to following formulas:

- Encryption: $C = M^e \bmod n$
- Decryption: $M = C^d \bmod n = M^{ed} \bmod n$

Since both sender and receiver know n, only the sender knows e and d is known only to the receiver, the public key PU en private key PB become

- $PU = \{e, n\}$
- $PB = \{d, n\}$

Finding suitable values for e, d and n
Assuming M and n are relative prime, Euler's theorem states

$$M^{\phi(n)+1} \equiv M \bmod n = M^{ed} \bmod n \Leftrightarrow ed \equiv 1 \bmod \phi(n) \Leftrightarrow d \equiv e^{-1} \bmod \phi(n)$$

where $\phi(n)$ is the Euler totient function that returns the number of integers $k(1 \leq k \leq n)$ for which $gcd(n, k) = 1$.
How to guarantee M and n relative prime
If $n = pq$ with p and q prime, then M and n are relative prime if M is different from $1, p$ and $q \Rightarrow \phi(n) = (p-1)(q-1)(\Rightarrow ed \bmod \phi(n) = 1)$.

4.2.2 RSA ingredients

- Two prime numbers p and q, chosen privately
- Integer $n = pq$, calculated and publicly available
- Value e chosen by key generator with $gcd(\phi(n), e) = 1; 1 < e < \phi(n)$ publicly available
- Value $d \equiv e^{-1} \bmod \phi(n)$ calcu-

lated and private

Note that it should be infeasible to calculate p, q and that this is the case if n is large enough.

4.3 Efficient RSA operations

4.3.1 Efficient exponentiation in modular arithmetic

To avoid integer overflow during calculations, we can use the property that $[(a \bmod n)(b \bmod n)] \bmod n = (ab) \bmod n$. Similarly we can make calculating large exponents more efficient by reusing intermediary results, eg $x^{13} = x * x^4 * x^8$. We can calculate $x \bmod n$, $x^2 \bmod n$, $x^4 \bmod n$ and $x^8 \bmod n$ where each reuses the previous result.
$\Rightarrow$ **Algorithm to find a^b mod n**
b can be expressed as a binary number $b_k b_{k-1} ... b_1 b_0$, or $b = \Sigma_{b_i \neq 0} 2^i$, therefore $a^b = a^{\Sigma_{b_i \neq 0} 2^i} = \Pi_{b_i \neq 0} a^{2^i}$.
$\Rightarrow a^b \bmod n = [\Pi_{b_i \neq 0} a^{2^i}] \bmod n = [\Pi_{b_i \neq 0} a^{2^i} \bmod n] \bmod n$
$\Rightarrow$Algorithm:

```
f = 1
for i = k down to 0 do
    f = (ff) mod n
    if b_i = 1 then
        f = (fa) mod n
return f
```

4.3.2 Efficient operation using public key

To make encryption as efficient as possible, chose an exponent with few 1 bits $\rightarrow$ the most common choice is $2^{16+1}, 3, 17$ since each of these numbers only has 2 1 bits. Note that small values are vulnerable to simple attacks but that this can be solved by adding a unique psuedorandom bit string as padding.

4.3.3 Efficient operation using private key

If e is chosen to be small, then d is large by definition. To speed up calculations for large d we can use the Chinese remainder theorem.

We need to compute $M = C^d \bmod n$. Now define $V_p = C^d \bmod p$ and $V_q = C^d \bmod q$. Applying the CRT gives $M = (V_p X_p + V_q X_q) \bmod n$ with $X_p = q(q^{-1} \bmod p)$ and $X_q = p(p^{-1} \bmod q)$.
Using Fermat's theorem we get that $V_p = C^{d \bmod (p-1)} \bmod p$ and $V_q = C^{d \bmod (q-1)} \bmod q$ where both $d \bmod (p-1)$ and $d \bmod (q-1)$ can be precalculated since p, q and d are known in advance.
By doing this precalculation, calculation of M is about 4 times more efficient.

4.3.4 Efficient key generation

1. Determine 2 prime numbers p and q.
   Any adversary will know $n = pq$, so in order to prevent brute force attacks p and q should be large enough and the method to find large p and q should be efficient.
   $\rightarrow$ Generate a random uneven number with the desired order of magnitude. If the number is determined to be prime, then return it. Else we generate a new number and repeat the process.

2. Select e or d and calculate the other.
   If p and q are found, than n is found and $\phi(n) = (p-1)(q-1)$ is found. Now lets select e such that $gcd(\phi(n), e) = 1$ and calculate d such that $d \equiv e^{-1}(\bmod \phi(n))$. Both values can be calculated simultaneously using the Extended Euclids Algorithm.
   One possible strategy is to select random numbers e until one is relative prime to $\phi(n)$.

4.4 Attacks against RSA

4.4.1 Brute force attacks

Try all possible private keys (d, n). RSA is safe if n is chosen large enough: minimum 3072 bits for n with RSA, in symmetric encryption 128 suffice.

4.4.2 Mathematical attacks

Factor the product of p and q given n. There are 3 approaches to this kind of attack.

1. Factor n into 2 primes p and q

2. Determine $\phi(n)$ directly without finding p and q

3. Determine d directly without finding $\phi(n)$

The first two options are mathematically equivalent and easier than the last option.
This gives rise to 2 threats to the use of large key sizes:

- Continuously improving computational power
- Refinement of factoring algorithms

There are a few safety precautions that can be taken:

- p and q should be nearly the same number of digits
- Both $(p-1)$ and $(q-1)$ should contain a large prime factor
- $gcd(p-1, q-1)$ should be small

4.4.3 Timing attacks

Exploit the running time of decryption algorithm. This is a ciphertext only attack where a snooper can determine the private key by keeping track of how long it takes to decipher the message. It is applicable to all public key encryption algorithms and for RSA, it uses the timings of the modular exponentiation algorithm. Counter measures include constant exponentiation time (which leads to degraded performance), adding a random delay or by using the blinding technique.
In the blinding technique the operation $M = C^d \bmod n$ is replaced by

1. Generate a random number $r \in]0, n-1]$
2. Compute $C' = Cr^e \bmod n$ where e is the public exponent
3. Compute $M' = (C')^d \bmod n$
4. Compute $M = M'r^{-1}$

This method results in a 2-10% performance loss.

4.4.4 Chosen ciphertext attacks

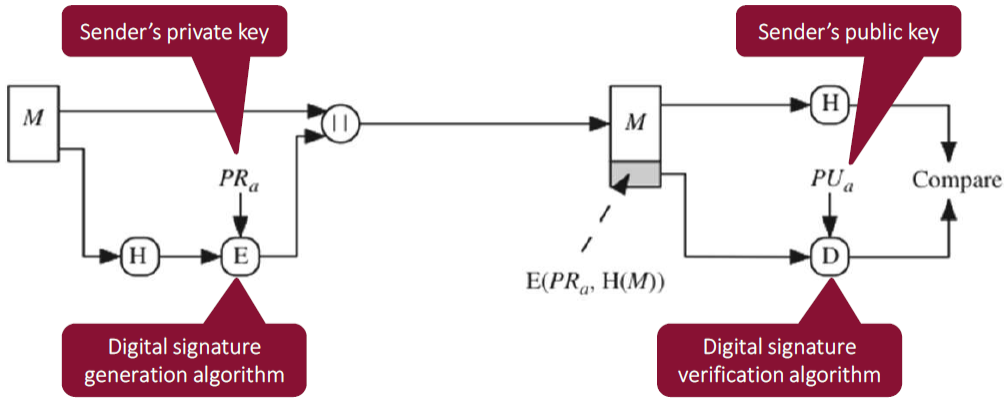
Exploit properties of RSA. In this attack the attacker chooses some ciphertext and is given the corresponding plaintext in return. Another possibility is that the attacker sends chosen blocks of data that yield additional information useful for cryptoanalysis.
Given the property that $E(PU, M_1)xE(PU, M_2) = E(PU, M_1xM_2)$ and $C = M^e \bmod n$, we can derive M using the CCA attack as follows

- Compute $x = (Cx2^e) \bmod n$
- Submit x as a chosen ciphertext and receive $y = x^d \bmod n$

$\Rightarrow x = (C \bmod n)x(2^e \bmod n) = (M^e \bmod n)x(2^e \bmod n) = (2M)^e \bmod n$
Therefore $y = (2M) \bmod n$, from which M can be easily deduced.

4.5 Digital signatures

4.5.1 Introduction



Authentication vs signatures:

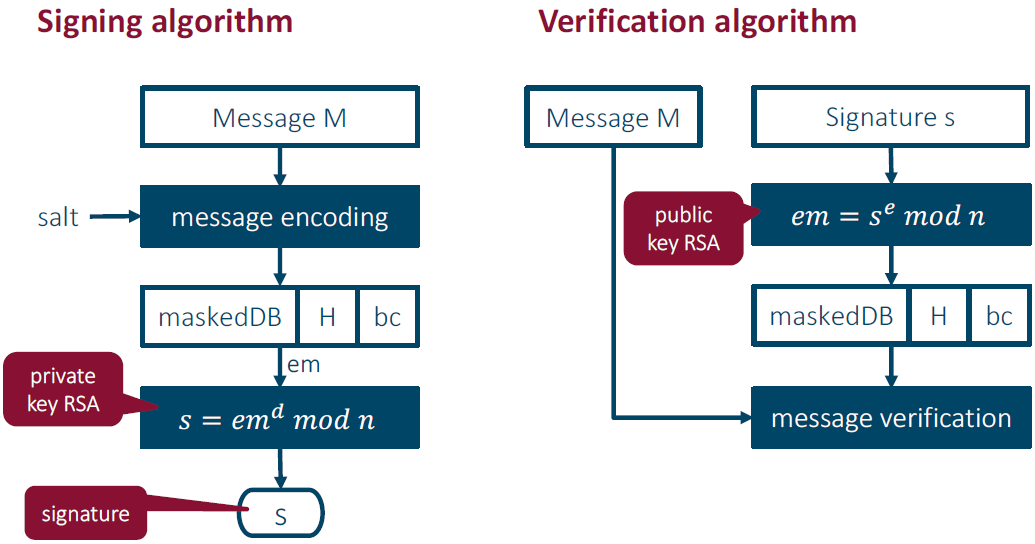
- Authentication: protects against third parties, only use single private key which is securely shared between sender and receiver → verify content of message was not altered.
- Signatures: provides additional securities, ensures that anyone can verify that the message was indeed sent by the sender.

4.5.2 Required properties of a digital signature

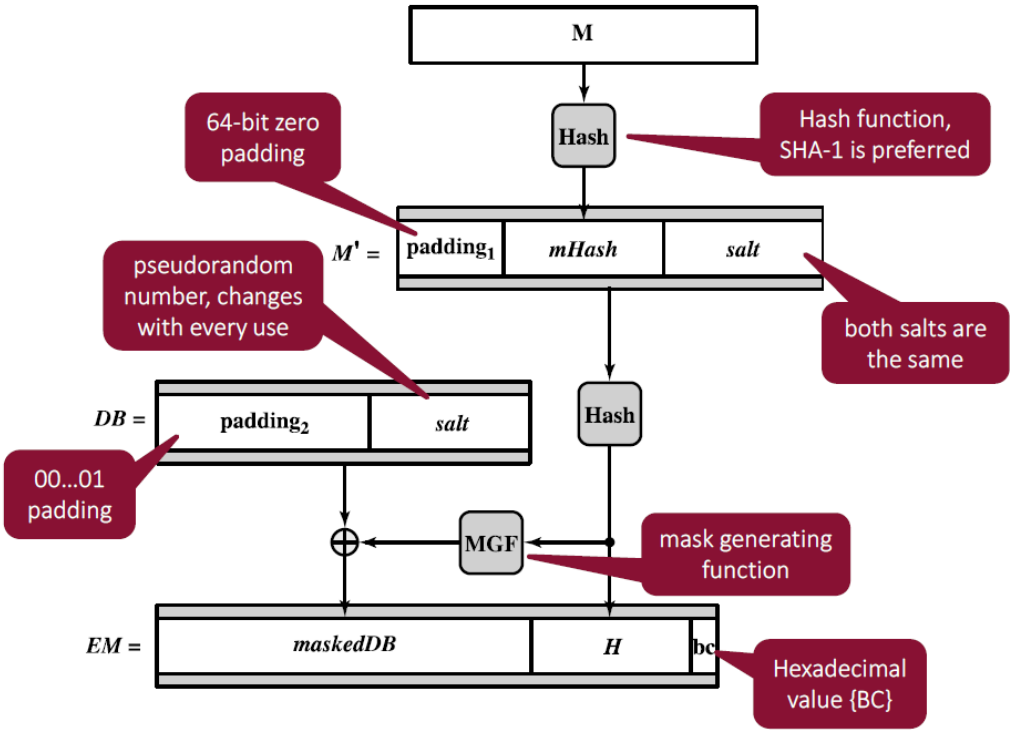
A digital signature must:

1. Verify the author and time of the signature → avoid misuse of private key
2. Authenticate the contents at the time of the signature
3. Be verifiable by third parties to resolve disputes → cannot be provided by MAC

4.5.3 RSA-PSS



We discuss the RSA Probabilistic Signature Scheme (RSA-PSS), which is the latest of the RSA schemes and the one that RSA Laboratories recommends as the most secure of the RSA schemes.
We show how the signature is formed by a signer with private key $\{d, n\}$ and public key $\{e, n\}$. Treat the octet string EM as an unsigned, nonnegative binary integer m. The signature s is formed by encrypting m as follows: $s = m^d \text{mod} n$.
Let k be the length in octets of the RSA modulus n. For example if the key size for RSA is 2048 bits, then $k = 2048/8 = 256$. Then convert the signature value s into the octet string S of length k octets.



The first stage in generating an RSA-PSS signature of a message M is to generate from M a fixed-length message digest, called an encoded message. We define the following parameters and functions:

- Options
  - Hash: hash function with output hLen octets. The current preferred alternative is SHA-1, which produces a 20-octet hash value.
  - MGF: mask generation function. The current specification calls for MGF1.
  - $sLen$: length in octets of the salt. Typically sLen = hLen, which for the current version is 20 octets.
- Input
  - M: message to be encoded for signing.
  - $emBits$: This value is one less than the length in bits of the RSA modulus n.

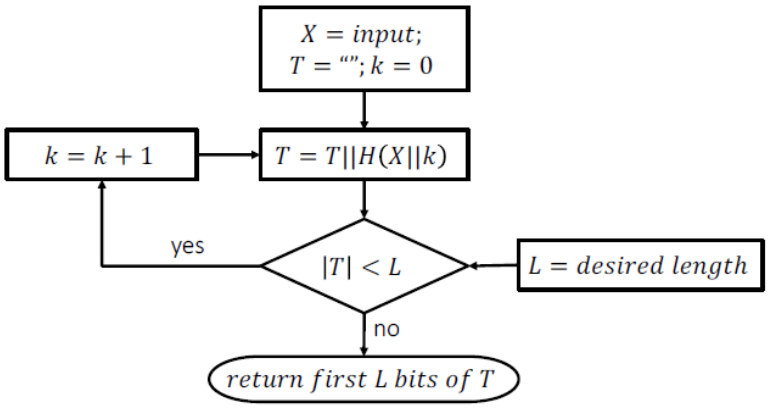
- Output
  - EM: encoded message. This is the message digest that will be encrypted to form the digital signature.
- Parameters
  - $emLen$: length of EM in octets = $emBits/8$.
  - padding1: hexadecimal string 00 00 00 00 00 00 00 00; that is, a string of 64 zero bits.
  - padding2 : hexadecimal string of 00 octets with a length (emLen - sLen - hLen - 2) octets, followed by the hexadecimal octet with 63 (emLen - sLen - hLen - 2) octets, followed by the hexadecimal octet with value 01.
  - salt: a pseudorandom number.
  - bc: the hexadecimal value BC.

The encoding process consists of the following steps.

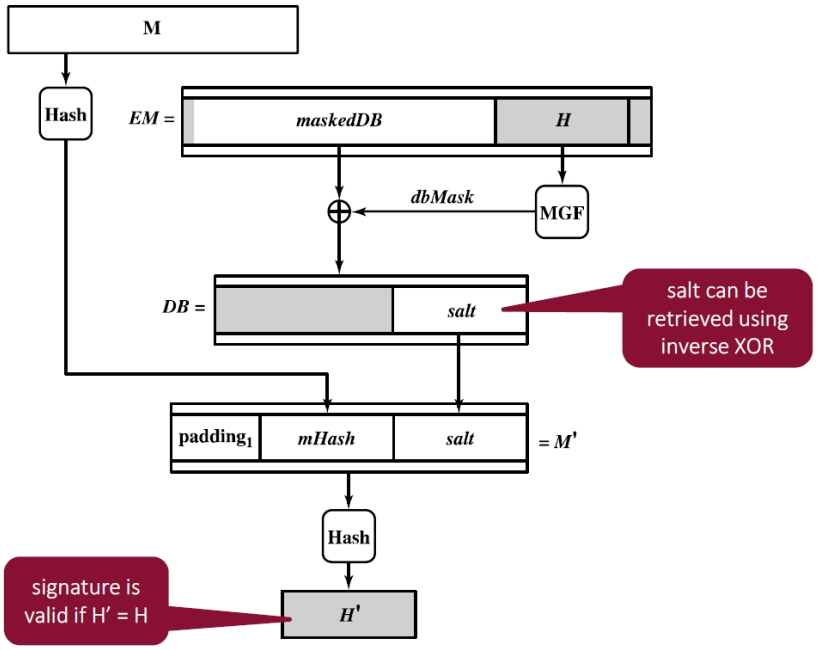
- Generate the hash value of M: $mHash = Hash(M)$
- Generate a pseudorandom octet string salt and form block $M' = padding1||mHash||salt$
- Generate the hash value of M': $H = Hash(M')$
- Form data block $DB = padding2||salt$
- Calculate the MGF value of H: $dbMask = MGF(H, emLen-hLen-1)$
- Calculate $maskedDB = DB \bigoplus dbMsk$
- Set the leftmost 8 $emLen-emBits$ bits of the leftmost octet in $maskedDB$ to 0
- $EM = maskedDB||H||0xbc$

4.5.4 RSA-PSS MGF (Mask Generating Function)

The goal is to generate a cryptographically secure variable length L hash code using a fixed length output cryptographic hash function H, eg SHA.



4.5.5 RSA-PSS signature verification



For signature verification, treat the signature S as an unsigned, nonnegative binary integer s. The message digest m is recovered by decrypting s as

follows: $m = se \bmod n$.
Then, convert the message representative m to an encoded message EM of length $emLen = modBits - \frac{1}{8}$ octets, where $modBits$ is the length in bits of the RSA modulus n.

5 Key distribution

5.1 Symmetric private key distribution

How to share a private key between 2 parties without others having access to it?

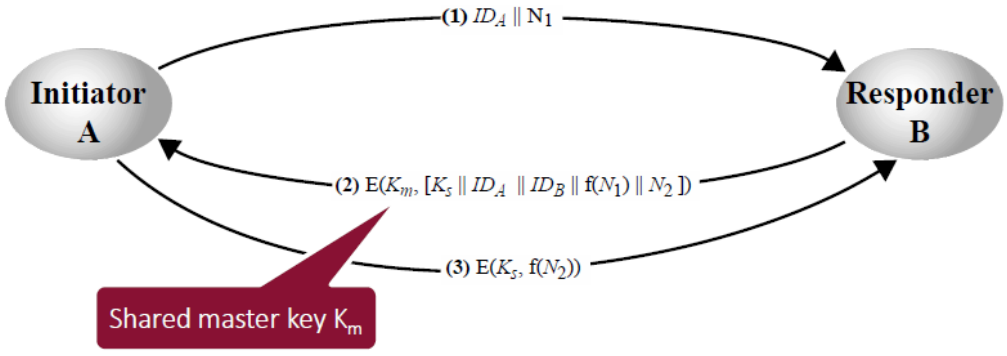
There are a few different distribution methods for parties A and B:

1. A selects the key and physically delivers it to B
2. Third party selects a key and physically delivers it to A and B
3. If A and B already share a key, a new key can be encrypted using the old key and transmitted over the network
4. If A and B have anencrypted connection to a third party C, C can deliver a key on the encrypted link to A and B

Solutions 1 and 2 are infeasible in large scale networks. If in solution 3 an attacker determines a single key then all future keys are compromised, due to the amount of keys that need to be maintained this solution too can only be used on small scale.
It thus follows that solution 4 is the preferred one.

5.1.1 Decentralized key control without third party

This way of working avoids the need for a central trusted authority, but it can only be used in small scale scenarios like local networks.



1. A send $ID_A||N_1$ to B where N_1 is a random nonce

2. B uses the previously shared master key K_m to encrypt a newly generated key K_s

3. After decryption A will send $E(K_s, f(N_2))$ to B

This approach uses $\frac{N(N-1)}{2}$ keys in a network with N hosts.

5.2 Key Distribution Centers (KDCs)

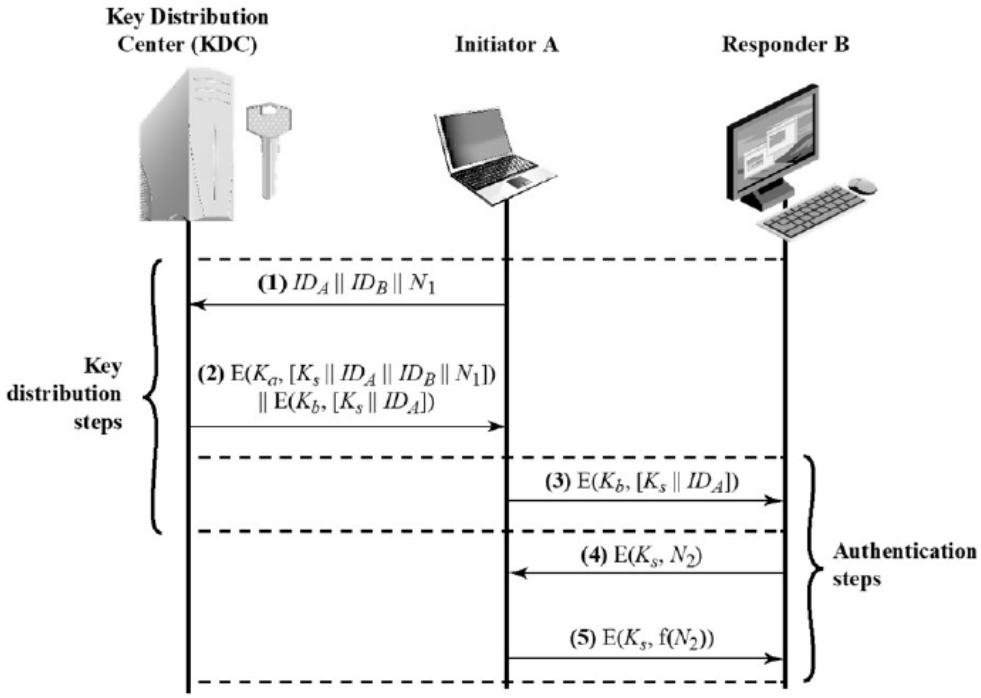
Centralized authority that distributes keys using symmetric encryption.

5.2.1 Hierarchical structure

Keys are stored on 2 levels:

- Session keys: temporary keys used between two end systems, discarded after logical session ends.
- Master keys: used to safely encrypt and transmit session keys $\rightarrow$ maintained between KDC and hosts, only used when host requests new session key.

5.2.2 KDC scenario



5.2.3 Hierarchical key control

A hierarchy of KDCs manages different parts of an internetwork where each local KDC distributes keys in the local network.
Higher layer KDCs can coordinate key distribution across networks.

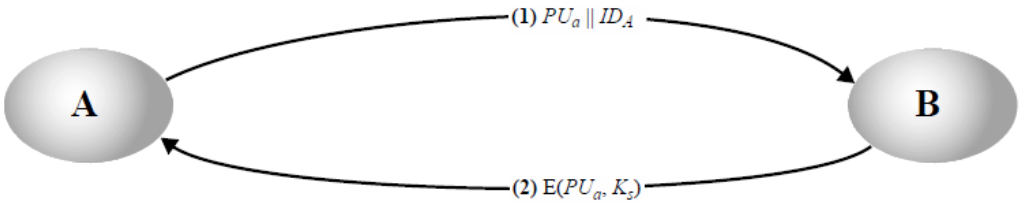
5.2.4 Transparent key control scheme

For providing end-to-end encryption in connection-oriented protocols.

1. Host sends packet to request connection
2. Security service buffers packet; asks KDC for session key
3. KDC distributes session key to both hosts
4. Buffered packet is transmitted

5.3 Asymmetric private key distribution

5.3.1 Symmetric key distribution using asymmetric encryption



This scheme is vulnerable to "man-in-the-middle" attacks → same scheme but with an attacker in the middle[1].

5.3.2 Diffie-Hellman key exchange

Enables 2 users to securely exchange a key (eg for use in subsequent symmetric encryption).
It is based on the difficulty to calculate discrete logarithms. The exponent i is the discrete logarithm of b for base a mod p: $d\log_{a,p}(b) = i \Leftrightarrow b \equiv a^i \bmod p$.
Parties share the publicly known pair (q, α) where q is a prime number and α is an integer and prime root of q, meaning a unique exponent i exists for every integer $b < q$ such that $b \equiv \alpha^i \bmod q$.
Private keys are random number X_A and X_B both smaller than q. The public keys are $Y_A = \alpha^{X_A} \bmod q$ and $Y_B = \alpha^{X_B} \bmod q$. The shared secret key $K = (Y_B)^{X_A} \bmod q = (Y_A)^{X_B} \bmod q$ can be calculated.
To an attacker, q, α, Y_A and Y_B are available. This means that the private key of B can be calculated as $X_B = d\log_{\alpha,q}(Y_B)$. The security of Diffie-Hellman lies in the fact that it is easy to calculate exponentials modulo a prime number but that it is hard to calculate discrete logarithms.
The "man-in-the-middle" attack is still possible.

5.4 Secure distribution of public keys

There are 4 general techniques for public key distribution:

1. Public announcement: broadcast public keys to anyone
2. Public directory: dynamic centralized directory of keys

[1] Duh

3. Public key authority: tightly controlled central directory

4. Public key certificates: decentralized secure key exchange

Both (1) and (2) are not safe, (3) has scalability issues due to the central nature. Hence, (4) is the preferred solution due to better scalability.

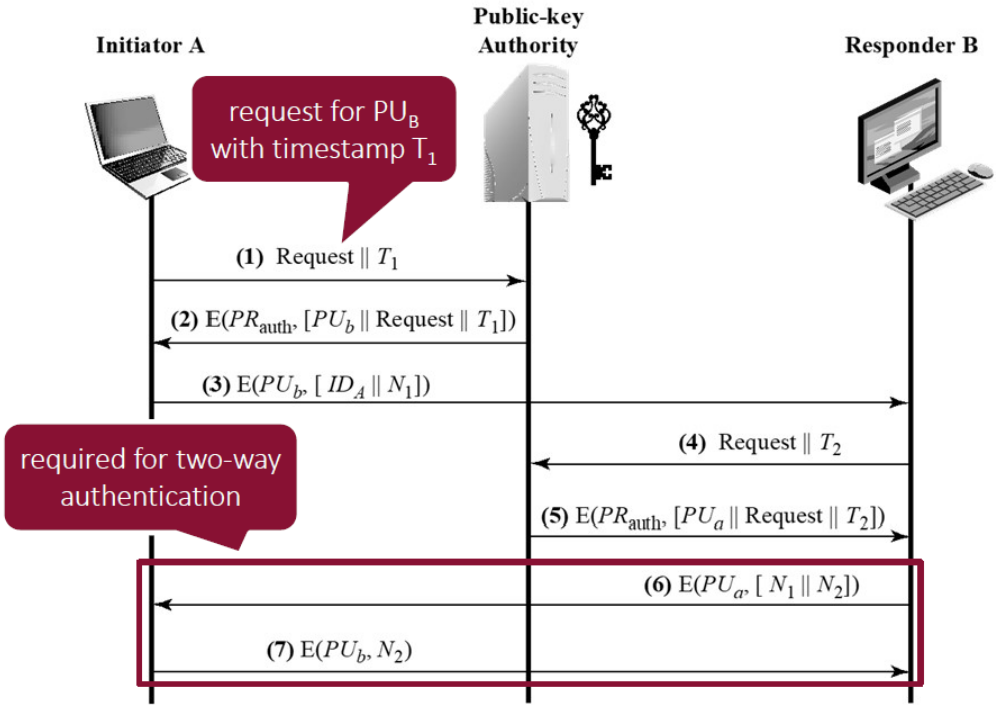
5.4.1 Public announcement

Since there is no form of control, anyone can forge an announcement and pretend to be A, spreading their own public key.

5.4.2 Public directory

Keys are registered in person or using secure authenticated communication. If the directory's private key is stolen, an attacker can counterfeit public keys.

5.4.3 Public key authority



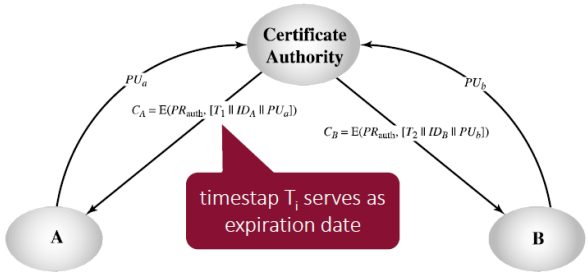
5.4.4 Certificates

Central authority only used to generate a certificate that consists of the public key, the ID of the owner the trusted third party's signature. A certificate must adhere to following requirements:

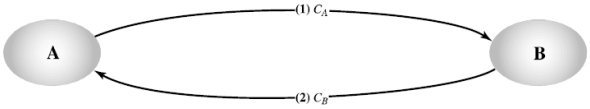
- Any participant can determine the name and public key of the owner
- Any participant can verify that the certificate originated from the certificate authority and is not counterfeit
- Only the certificate authority can create and update certificates
- Any participant can verify the time validity of the certificate

Certificates: secure key exchange without an authority

Step 1: obtain a certificate for your public key

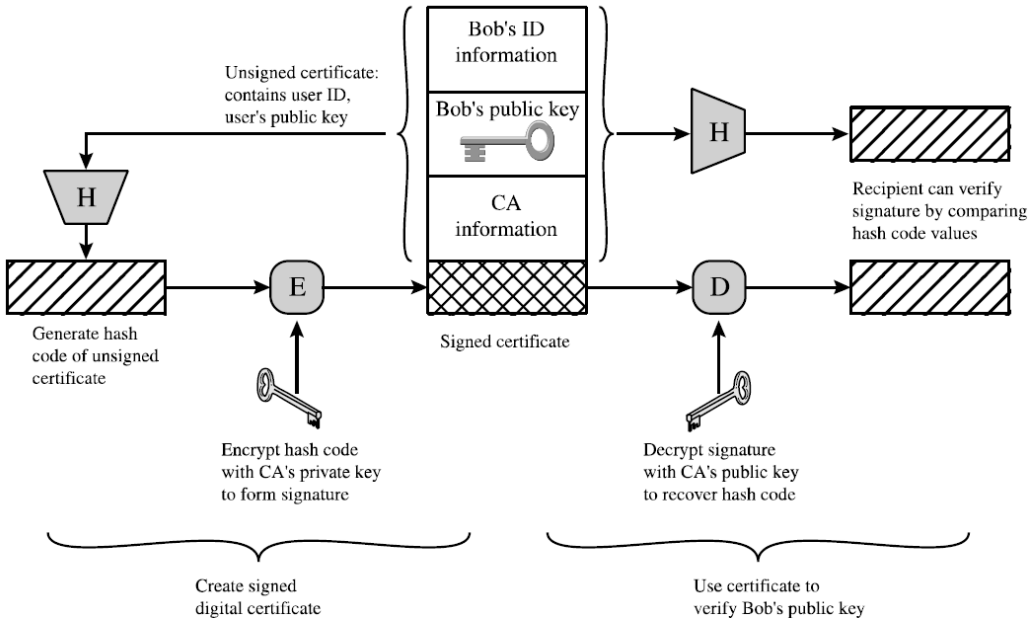


Step 2: exchange public key certificates

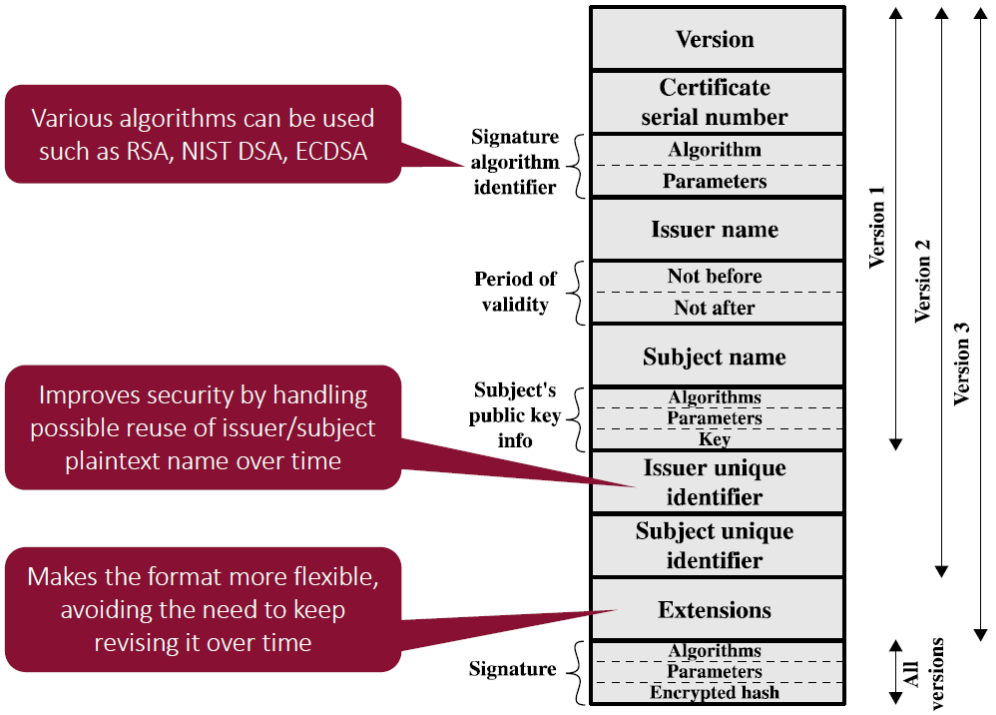


5.5 X.509 certificates

5.5.1 Certificate generation/working



5.5.2 X.509 format



5.5.3 CA organization

If the user community is large, eg the internet, then multiple CA's are needed.

$Y << X >>$: certificate of user X is issued by CA Y.

User A trusts CA X and user B trusts CA Y.
How does A get the key of B?

- Step 1: X owns Y<<X>> and Y owns X<<Y>>
- Step 2: A owns X<<A>> and B owns Y<<B>>
- Step 3: A validates key of B via chain of certificates: X<<Y>> Y<<B>>

An arbitrarily long chain of CA certificates can be used to validate a key

Two types of certificates stored by CA X:

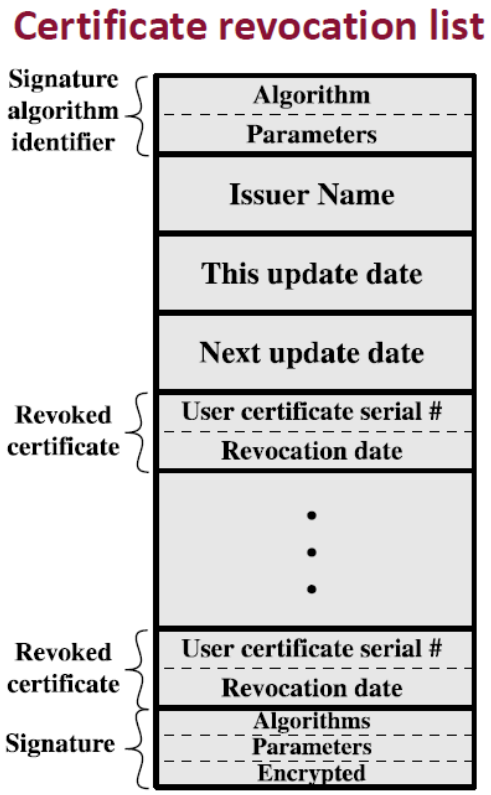
- Forward certificate: Of X generated by other CA
- Reverse certificate: Of other CA generated by X

5.5.4 Certificate revocation

Revocation may be needed if:

- User's private key is possibly compromised
- User is no longer certified by CA
- CA's certificate is possibly compromised

Each CA maintains a CRL (certificate revocation list) with all non-expired but revoked certificates.



5.5.5 Certificate extensions field

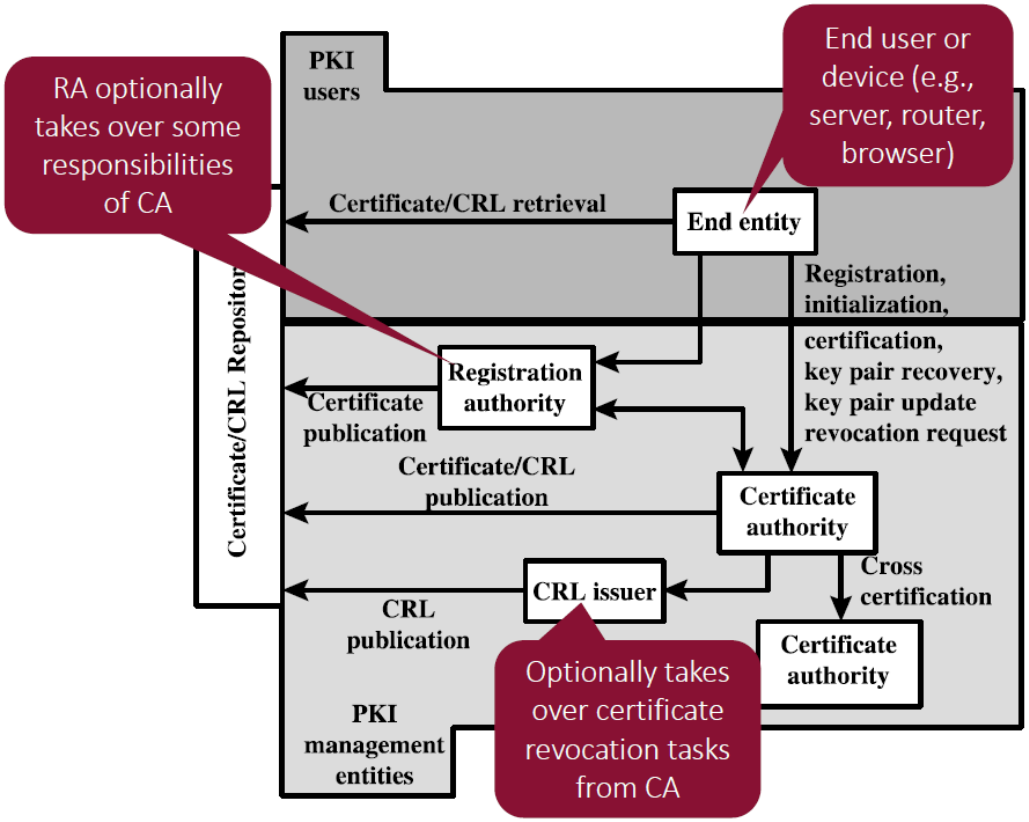
X.509 version 3 introduces 3 types of optional extension fields

- Key and policy information
  - Authority key identifier: allows CA to use multiple public keys
  - Subject key identifier: allows user to use multiple public keys
  - Key usage: indicates for which security purposes the key may be used
  - Private-key usage period: may invalidate private key earlier than public key
  - Policy mappings: Indicate equivalence between CA policies
- Certificate subject and issuer attributes
  - Subject alternative name: defines aliases for the user
  - Issuer alternative name: defines aliases for the CA
  - Subject directory attributes: for use with X.500 directories
- Certificate path constraints
  - Basic constraints: indicates if the subject may act as a CA themselves
  - Name constraints: indicates namespace restriction for hierarchical CAs
  - Policy constraints: restrictions on policy mapping on remainder of path

5.5.6 Public Key Infrastructure (PKI)

Set of hardware, software, people, policies and procedures needed to create, manage, store, distribute and revoke digital certificates based on asymmetric cryptography.
PKI objective: enable secure, convenient and efficient acquisition of public keys.
→ PKI X.509: formal, generic model based on X.509 to deploy certificate-based architecture on the internet.

PKIX architecture model



PKIX management functions

Management functions provided by one of two protocols:

- RFC 2510: certificate management protocols (CMP)
- RFC 2797: certificate management messages over CMS (CMC)

Functions:

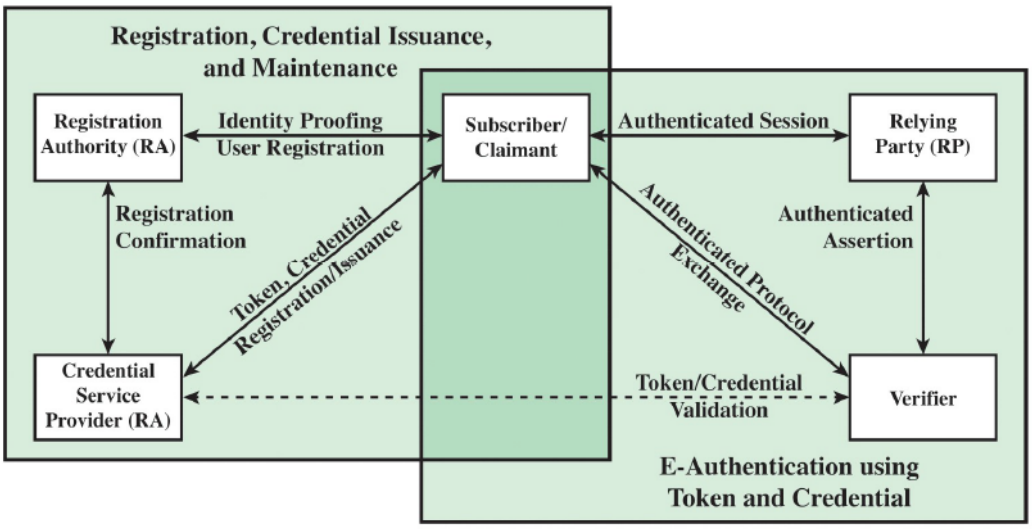
- Registration: User makes itself known to CA and exchanges shared secret keys
- Initialization: Initialize the different public and private keys
- Certification: CA issues a certificate for a user's public key and posts it
- Key-pair recovery: Recover decryption keys when keying material is not available
- Key-pair update: Replace keys and certificates over time for extra security
- Revocation request: Revoke certificate when abnormal behaviour is detected
- Cross-certification: Two CAs can exchange information and mutual certificates

6 Access Control

6.1 Authentication and authorization

Access control refers to the combination of both authentication and authorization.

Authentication: verify identity of a system entity (eg user). Authorization: verify entity's right to perform specific actions.



The above scheme depicts how authentication happens:

- In a first step, a subscriber will proof its identity to the Registration Authority (RA). The RA will then interface with the Credential Service Provider (CSP) to confirm the registration and CSP will provide the user with a token or credentials.
- In the second step, the user contacts a verifier via an authenticated protocol exchange. This verifier will then verify the credentials with the CSP and if this proves successful the verifier will relay to relying party to authenticate the user.

The means of authentication include what the user:

- knows: password, pin, questions, ...
- has: key, token generator, ...

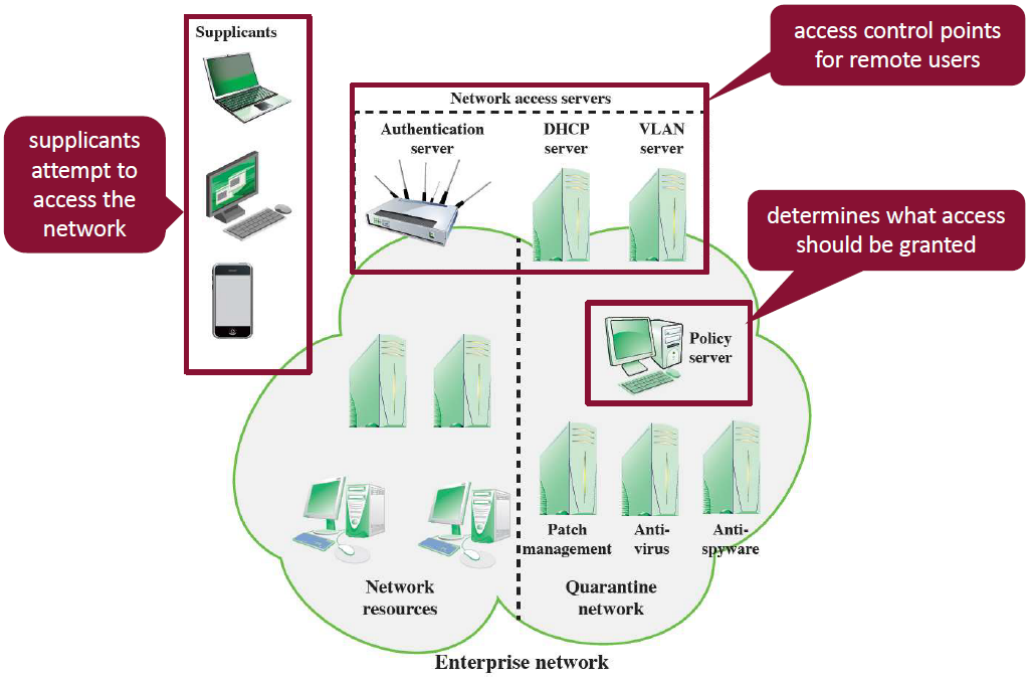
- is: fingerprint, iris, ...

Multifactor authentication combines multiple methods to improve security.
Access control:

- Functions
  - Authenticates users login into the system
  - Determines what data logged in users can access
  - Determines what actions logged in users can perform
  - Examines the health of the user's device
- Enforcement methods
  - IEEE802.1x
  - Firewalls
  - VLANs
  - DHOP management
- Related technologies and support systems
  - Intrusion detection
  - IPSecurity

6.2 Network Access Control (NAC)

6.2.1 Network access control



As we can see there are 3 actors:

- Supplicants: attempts to access the network, can be any IP based device
- Policy server: determines what access should be granted to supplicant, relies in back end services such as patch management, anti-virus, anti-spyware, user directory
- Network access server: access control points for remote users, can be an authentication server or DHCP/VLAN server

6.2.2 Extensible authentication protocol (EAP)

Provides a set of protocol messages that can encapsulate various authentication methods to be used between a supplicant and an authentication server. As such, it enables authentication over a network between hosts in the network. It can also operate over a variety of link layer facilities and technologies

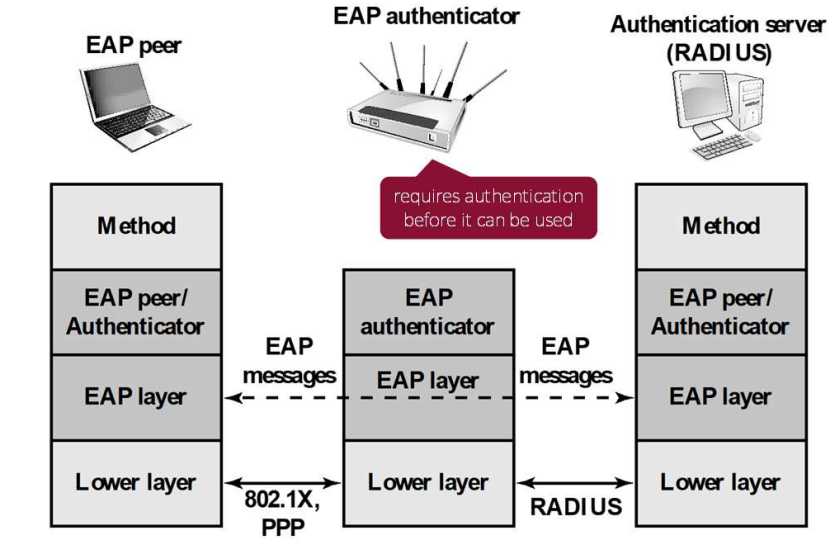
such as point-to-point links, LAN, WIFI, ...

Common EAP authentication methods

EAP-TLS	Uses the handshake protocol of TLS (not encryption) for authentication using certificates
EAP-TTLS	Like EAP-TLS, but the server authenticates first, then client authentication using any EAP method
EAP-GPSK	Authentication and session key derivation using a pre-shared key (PSK), based on private keys
EAP-IKEv2	Mutual authentication and session key establishment using the Internet Key Exchange protocol

[2]

EAP protocol exchange



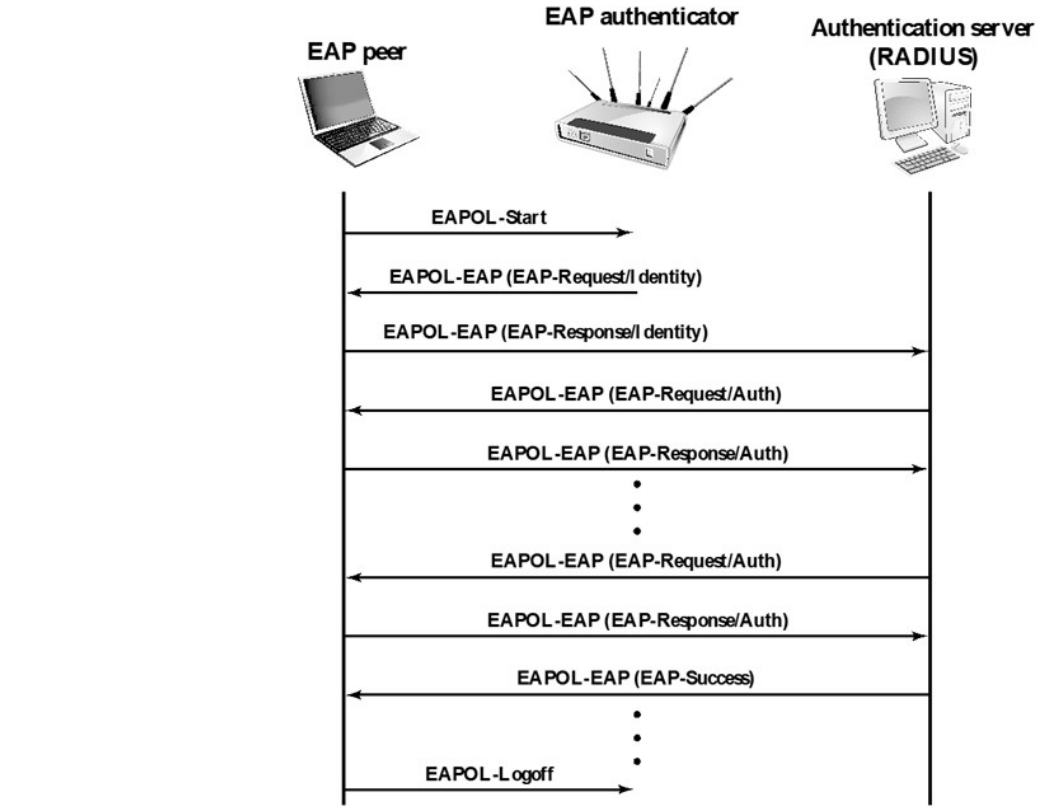
[2]IMPORTANT

6.2.3 802.1x port based NAC

Enables access control in LANs, such as ethernet and wifi, and is based on a concept of port-based network access control. This means that it was a concept of controlled and uncontrolled ports. These ports are logical entities defined by the authenticator and refer to physical network connections.
An uncontrolled port allows exchange of data between the supplicant and the authentication server regardless of the authenticated state. This means that control messages and messages of authentication exchange such as EAP protocol messages are always allowed into the network by th network access point, but only once the authenticator has authenticated the supplicant the controlled ports are opened allowing the exchange of actual data traffic between the external supplicant and internal network.

6.2.4 802.1x EAP over LAN (EAPOL)

Enables the message exchange between a supplicant, an authenticator and authentication server.



6.3 Firewalls

Design goals:

1. All traffic in and out must pass through the firewall
2. Only authorized traffic is allowed to pass through
3. The firewall is immune to penetration

Firewalls can use traffic filtering properties to determine whether to allow traffic or not (eg source and dest IP, port numbers, protocols, ...).

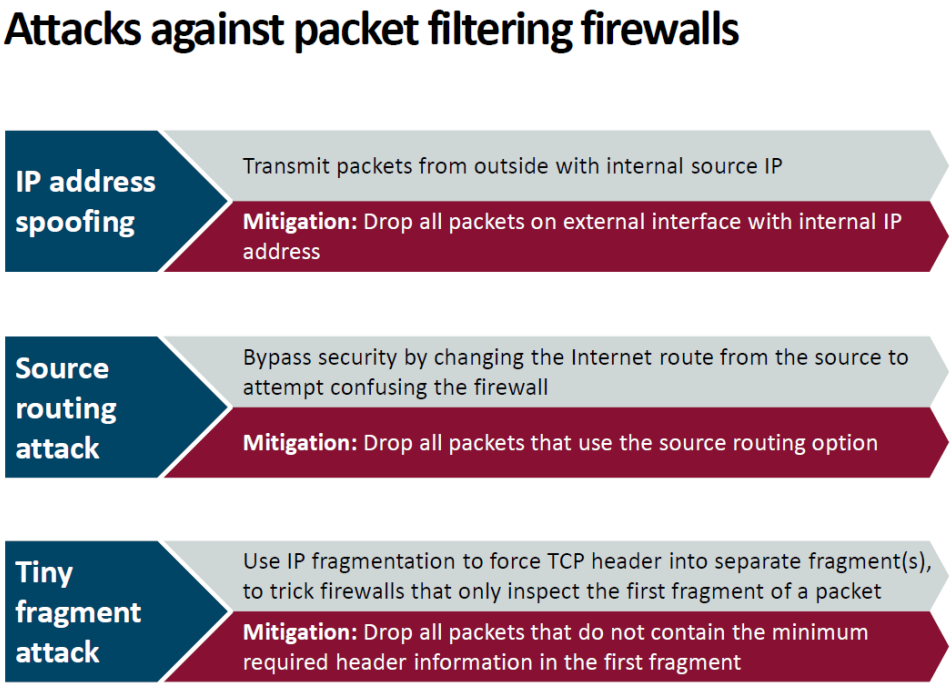
6.3.1 Firewall types

In increasing order of effectiveness:

1. Packet filtering type: network access rules are applied to each packet separately, which is then forwarded or discarded. The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken. Two default policies are possible:

   - Default = discard: That which is not expressly permitted is prohibited (whitelisting).
   - Default = forward: That which is not expressly prohibited is permitted (blacklisting).

   The default discard policy is more conservative. Initially, everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are more likely to see the firewall as a hindrance. However, this is the policy likely to be preferred by businesses and government organizations. Further, visibility to users diminishes as rules are created. The default forward policy increases ease of use for end users but provides reduced security; the security administrator must, in essence, react to each new security threat as it becomes known. This policy may be used by generally more open organizations, such as universities. This kind of filtering works on the transport layer.



2. Stateful inspection: keeps track of outbound TCP connections, so only currently active TCP ports have to be opened.
   A stateful packet inspection firewall reviews the same packet information as a packet filtering firewall, but also records information about TCP connections. Some stateful firewalls also keep track of TCP sequence numbers to prevent attacks that depend on the sequence number, such as session hijacking. Some even inspect limited amounts of application data for some well-known protocols like FTP, IM and SIPS commands, in order to identify and track related connections.

3. Application proxy: acts as a termination point for TCP connections, relaying traffic after user authentication.
   Application-level gateways tend to be more secure than packet filters. Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the TCP and IP level, the application-level gateway need only scrutinize a few allowable applications. In addition, it is easy to log and audit all incoming traffic at the application level.
   A prime disadvantage of this type of gateway is the additional processing overhead on each connection. In effect, there are two spliced

connections between the end users, with the gateway at the splice point, and the gateway must examine and forward all traffic in both directions.

4. Circuit-level proxy: similar to (3), but doesn't examine content after connection establishment.
   As with an application gateway, a circuit-level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.
   A typical use of circuit-level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application-level or proxy service on inbound connections and circuit-level functions for outbound connections. In this configuration, the gateway can incur the processing overhead of examining incoming application data for forbidden functions but does not incur that overhead on outgoing data.

6.3.2 Firewall placement and demilitarized zone (DMZ)

The DMZ contains the servers that need to be externally accessible.
An external firewall is placed at the edge of a local or enterprise network, just inside the boundary router that connects to the Internet or some wide area network (WAN). One or more internal firewalls protect the bulk of the enterprise network. Between these two types of firewalls are one or more networked devices in a region referred to as a DMZ (demilitarized zone) network. Systems that are externally accessible but need some protections are usually located on DMZ networks. Typically, the systems in the DMZ require or foster external connectivity, such as a corporate Web site, an e-mail server, or a DNS (domain name system) server.

6.4 Intrusion detection

We distinguish 3 types of intruders:

- Masquerader: outsider who penetrates the system's access control to exploit a legitimate user's account.
- Misfeasor: legitimate user who accesses data, programs or resources to which he is not authorized.
- Clandistine user: individual who seizes supervisory control of the system and uses the control to evade auditing or access control.

6.4.1 Intrusion detection systems

Intrusion detection is based on the assumption that the behavior from intruders differs from that of a legitimate user in quantifiable ways. Since there is not a clear distinction, some overlap will occur leading to false positives/negatives.
Motivation:

- If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised, or the damage can be contained and recovery can be faster.
- An effective detection can also work preventive.
- Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

6.4.2 Types of intrusion detection

1. Statistical anomaly detection[3]: over a period of time, legitimate user behavior data is collected after which statistical tests[4] are applied to observed behavior in order to determine, with a high level of confidence, whether or not that behavior is legitimate user behavior.
    - threshold detection: involves defining thresholds, independent of user, for the frequency of occurrence of various events

[3]Foundation of this approach is in analysis of audit records
[4]Mean and standard deviation, multivariate, Markov process, time series, operational model

- profile based: a profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts

2. Rule based detection (=signature detection): involves an attempt to define a set of rules or attack patterns that can be used to decide that a given behavior is that of an intruder

    - Rule based anomaly detection: historical audit records are used to automatically generate rules of proper conduct, to which new actions are matched
    - Rule based penetration detection: defines rules specific to the machine or OS, that identify attacks against known weaknesses and vulnerabilities in the system

6.4.3 Distributed intrusion detection

Consists of 3 main components:

- Host agent module: an audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security related events on the host and transmit them to the central manager.
- LAN monitor agent module: operates the same way as a host agent module, except that it analyzes LAN traffic and report the results to the central manager.
- Central manager module: receives reports from LAN monitor and host agents, and processes and correlates these reports to detect intrusion.

6.4.4 Honeypots

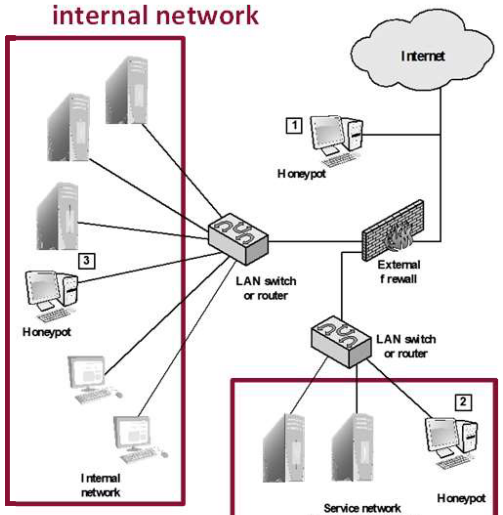
Honeypots

A **honeypot** is a decoy system designed to lure potential attackers away from actual critical systems, in order to:

- Divert an attacker from critical systems
- Collection information about attackers
- Enable early detection by administrators

Can be deployed in different locations:

1. Outside the external firewall to detect connection to unused IP addresses
2. In the DMZ, behind the firewall but among externally accessible services
3. In the internal network, allowing to catch internal attacks or firewall misconfigurations



A compromised honeypot in the DMZ or internal network may compromise the entire network!

6.5 IP Security (IPSec) and VPN

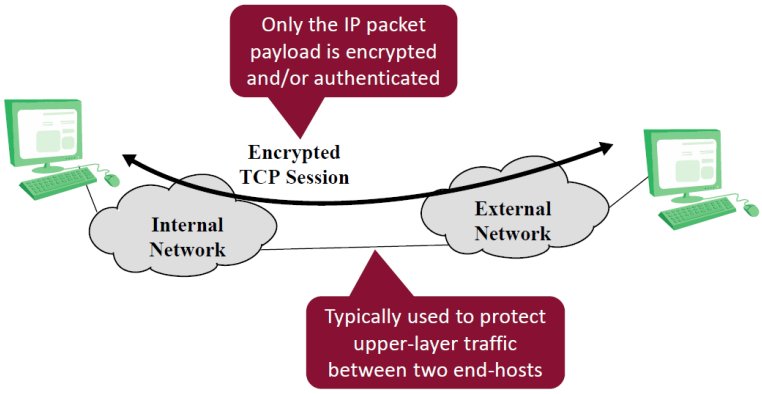
IPSec encompasses 3 functional areas:

- Authentication: assures that a received packet was in fact transmitted by the party identified as the source in the packet header.
- Confidentiality: enables communicating nodes to encrypt messages to prevent eavesdropping by third parties.
- Key management: secure exchange of keys.

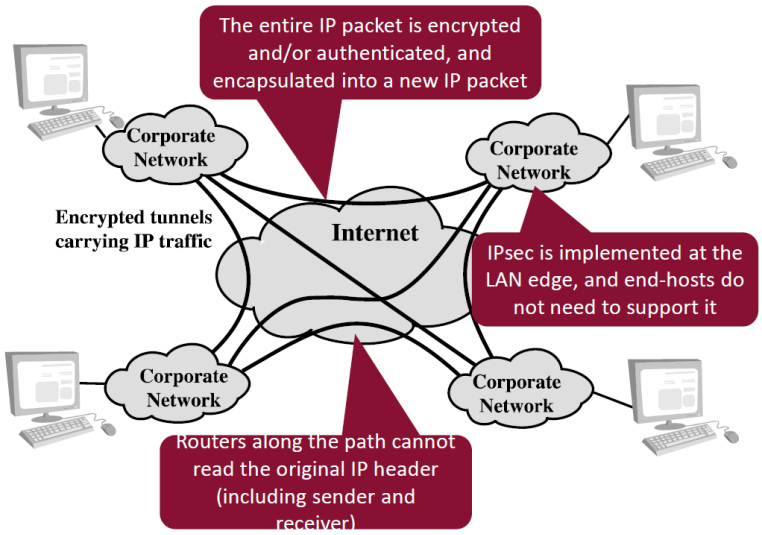
6.5.1 Modes of operation

1. Transport mode: provides protection primarily for upper-layer protocols → transport mode protection extends to payload of an IP packet,

source and dest IP are not kept confidential, used for end-to-end communication between two hosts.



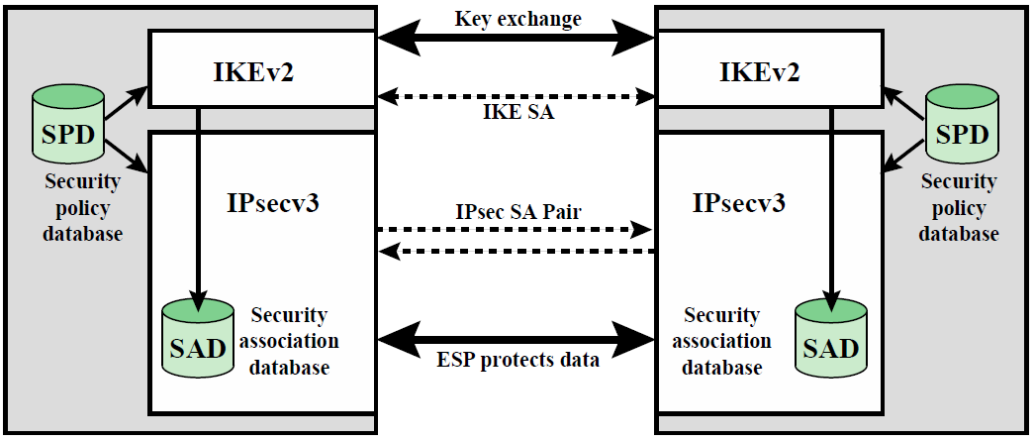
2. Tunnel mode: provides protection to entire IP packet. After the AH er ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new outer IP packet with a new outer IP header. Used when one or both ends of a security association (SA) are a security gateway, such as a firewall or a router that implements IPSec.



6.5.2 IPSec architecture

IPSec is based on concept of a security policy applied to each IP packet that transits from source to dest. Policy is determined primarily by interaction of

2 DBs: security association DB (SAD) and security policy DB (SPD).



In each IPsec implementation, there is a nominal **Security Association Database (SAD) that defines the parameters associated with each SA**. The key management mechanism that is used to distribute keys is coupled to the authentication and privacy mechanisms only by way of the **Security Parameters Index (SPI)**. Hence, authentication and privacy have been specified independent of any specific key management mechanism.

The **means by which IP traffic is related to specific SAs is the** nominal **Security Policy Database (SPD)**. In its simplest form, an SPD contains entries, each of which defines a subset of IP traffic and points to an SA for that traffic. In more complex environments, there may be multiple entries that potentially relate to a single SA or multiple SAs associated with a single SPD entry. **Each SPD entry is defined by a set of IP and upper-layer protocol field values, called selectors. In effect, these selectors are used to filter outgoing traffic in order to map it into a particular SA.** Possible selectors include: remote IP address, local IP address, next layer protocol, local port, and remote port.

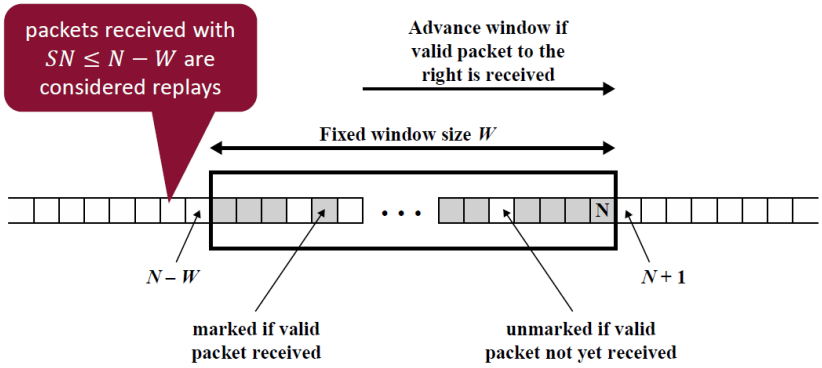
IPsec is executed on a packet-by-packet basis. When IPsec is implemented, each outbound IP packet is processed by the IPsec logic before transmission, and each inbound packet is processed by the IPsec logic after reception and before passing the packet contents on to the next higher layer (e.g., TCP or UDP).

6.5.3 Encapsulating Security payload (ESP)

One of two alternative security mechanisms in IPSec, ESP provides confidentiality and optionally authentication. The alternative Authentication Header (AH) only provides authentication. ESP can be used in combination with a wide range of encryption and authentication algorithms.
Security services include packet confidentiality, data origin authentication, connection-less integrity, anti-replay service, (limited) traffic flow confidentiality.

Anti-replay function

Non-trivial as IP packets may arrive out of order, so simply comparing subsequent sequence numbers (SNs) is not enough



6.5.4 Internet key exchange

Goal: determination and distribution of secret keys for confidentiality and integrity.

- Key exchange methods:
  - Manual: sysadmin manually configures each system with the keys.
  - Automated: on demand creation of keys for SAs.
- Default automated key exchange methods:
  - Internet Security Association and Key Management Protocol: protocol for the negotiation of security attributes.
  - Oakley: key exchange protocol based on Diffie-Hellman.

7 Secure Network Protocols

7.1 Wifi security basics

Wireless networks, and the wireless devices that use them, introduce a host of security problems over and above those found in wired networks. Some of the key factors contributing to the higher security risk of wireless networks compared to wired networks include the following:

- Channel: Wireless networking typically involves broadcast communications, which is far more susceptible to eavesdropping and jamming than wired networks. Wireless networks are also more vulnerable to active attacks that exploit vulnerabilities in communications protocols.
- Mobility: Wireless devices are, in principal and usually in practice, far more portable and mobile than wired devices.
- Resources: Some wireless devices, such as smartphones and tablets, have sophisticated operating systems but limited memory and processing resources with which to counter threats, including denial of service and malware.
- Accessibility: Some wireless devices, such as sensors and robots, may be left unattended in remote and/or hostile locations. This greatly increases their vulnerability to physical attacks.

Security threats to wireless networks:

- Accidental association: Company wireless LANs or wireless access points to wired LANs in close proximity (e.g., in the same or neighboring buildings) may create overlapping transmission ranges. A user intending to connect to one LAN may unintentionally lock on to a wireless access point from a neighboring network.
- Malicious association: In this situation, a wireless device is configured to appear to be a legitimate access point, enabling the operator to steal passwords from legitimate users and then penetrate a wired network through a legitimate wireless access point.
- Ad hoc networks: These are peer-to-peer networks between wireless computers with no access point between them. Such networks can pose a security threat due to a lack of a central point of control.

- Nontraditional networks: Nontraditional networks and links, such as personal network Bluetooth devices, barcode readers, and handheld PDAs, pose a security risk in terms of both eavesdropping and spoofing.

- Identity theft (MAC spoofing): This occurs when an attacker is able to eavesdrop on network traffic and identify the MAC address of a computer with network privileges.

- Man-in-the middle attacks: This attack involves persuading a user and an access point to believe that they are talking to each other when in fact the communication is going through an intermediate attacking device. Wireless networks are particularly vulnerable to such attacks.

- Denial of service (DoS): In the context of a wireless network, a DoS attack occurs when an attacker continually bombards a wireless access point or some other accessible wireless port with various protocol messages designed to consume system resources. The wireless environment lends itself to this type of attack, because it is so easy for the attacker to direct multiple wireless messages at the target.

- Network injection: A network injection attack targets wireless access points that are exposed to nonfiltered network traffic, such as routing protocol messages or network management messages. An example of such an attack is one in which bogus reconfiguration commands are used to affect routers and switches to degrade network performance.

IEEE 802.11i

1997 — **Wired Equivalent Protection (WEP)**
- Uses RC4 steam cipher for confidentiality and CRC-32 for integrity
- Successfully cracked in 2001 and fully deprecated in 2004

2003 — **Wi-Fi Protected Access (WPA)**
- Based on a preliminary draft of the IEEE 802.11i standard
- Uses RC4 stream cipher and Temporal Key Integrity Protocol (TKIP)

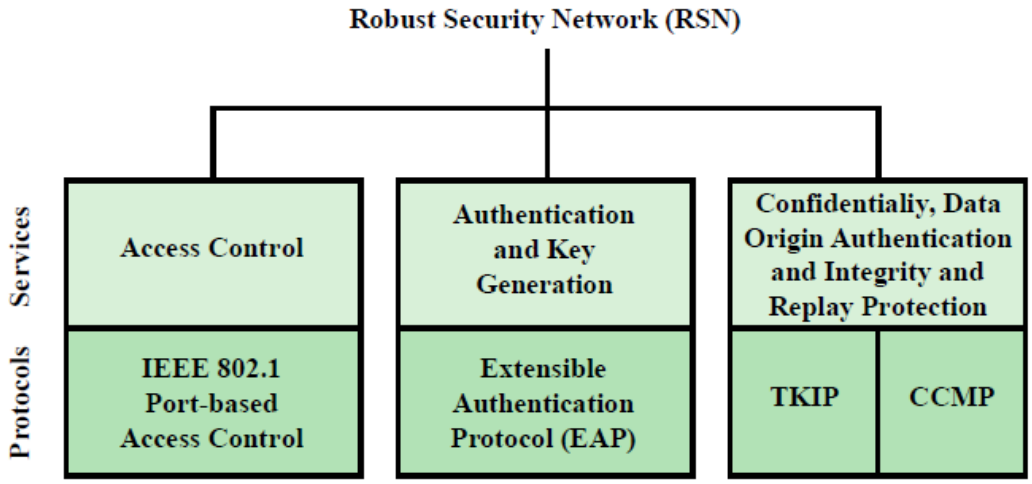
2004 — **Wi-Fi Protected Access II (WPA2) or Robust Security Network (RSN)**
- Implements the completed final IEEE 802.11i standard
- Uses strong AES encryption for confidentiality

2018 — **Wi-Fi Protected Access III (WPA3)**
- Uses stronger and up-to-date cryptography algorithms
- More secure key exchange using Simultaneous Authentication of Equals (SAE) to replace WPA2 pre-shared keys

5

[5]WEP sucks, WPA is outdated

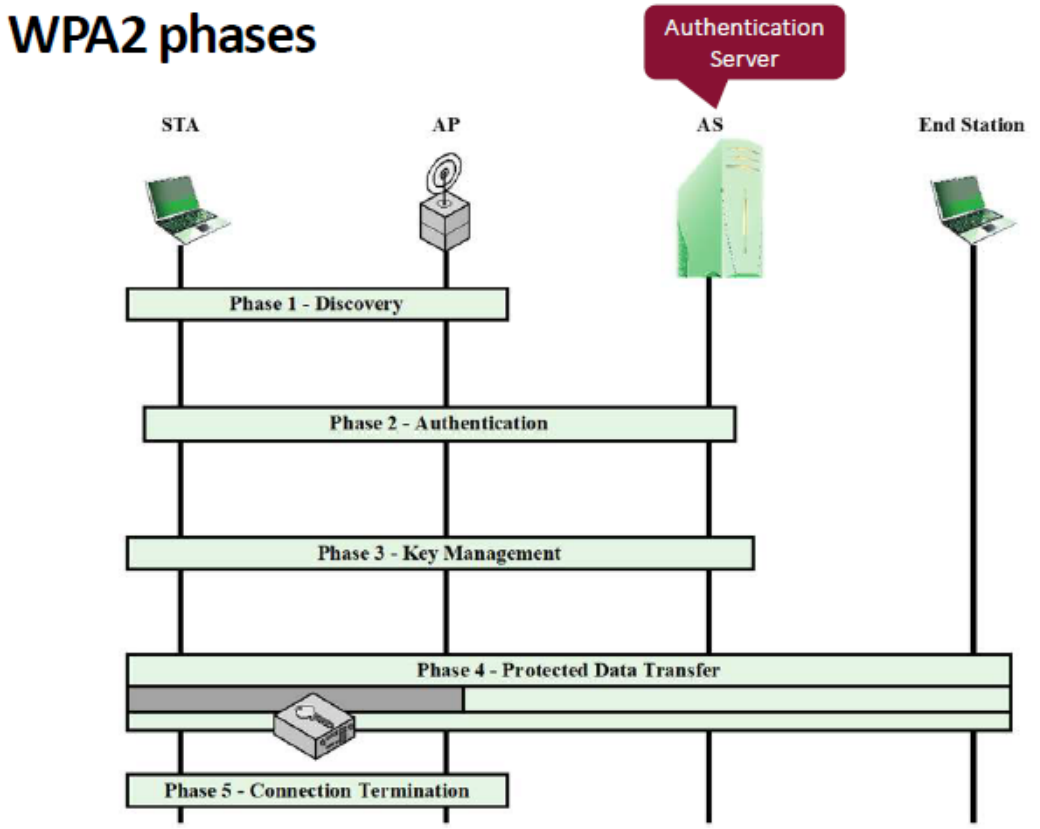
WPA2 security services and protocols



TKIP = Temporal Key Integrity Protocol
CCMP = Counter Mode with Cipher Block Chaining MAC Protocol

The 802.11i RSN security specification defines the following services:

- Authentication: A protocol is used to define an exchange between a user and an AS that provides mutual authentication and generates temporary keys to be used between the client and the AP over the wireless link.
- Access control: This function enforces the use of the authentication function, routes the messages properly, and facilitates key exchange. It can work with a variety of authentication protocols.
- Privacy with message integrity: MAC-level data (e.g., an LLC PDU) are encrypted along with a message integrity code that ensures that the data have not been altered.



The five phases are defined as follows:

1. Discovery: An AP uses messages called Beacons and Probe Responses to advertise its IEEE 802.11i security policy. The STA uses these to identify an AP for a WLAN with which it wishes to communicate. The STA associates with the AP, which it uses to select the cipher suite and authentication mechanism when the Beacons and Probe Responses present a choice. The discovery phase consists of three exchanges:

    - Network and security capability discovery: During this exchange, STAs discover the existence of a network with which to communicate. The AP either periodically broadcasts its security capabilities (not shown in figure), indicated by RSN IE (Robust Security Network Information Element), in a specific channel through the Beacon frame; or responds to a stations Probe Request through a Probe Response frame. A wireless station may discover available access points and corresponding security capabilities by either

passively monitoring the Beacon frames or actively probing every channel.

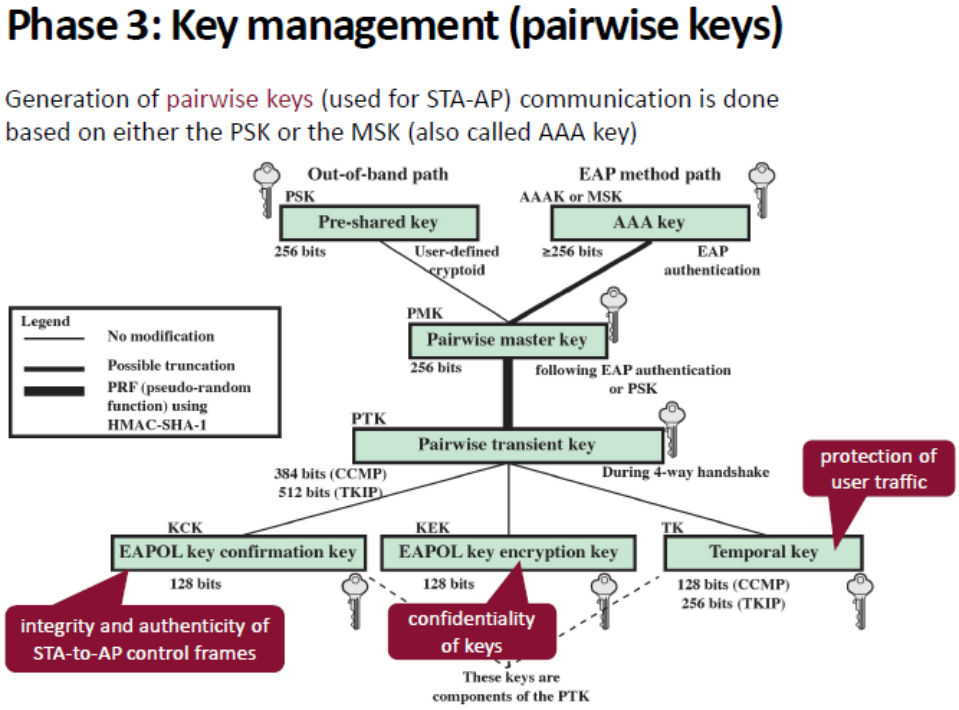
- Open system authentication: The purpose of this frame sequence, which provides no security, is simply to maintain backward compatibility with the IEEE 802.11 state machine, as implemented in existing IEEE 802.11 hardware. In essence, the two devices (STA and AP) simply exchange identifiers.
- Association: The purpose of this stage is to agree on a set of security capabilities to be used. The STA then sends an Association Request frame to the AP. In this frame, the STA specifies one set of matching capabilities (one authentication and key management suite, one pairwise cipher suite, and one group-key cipher suite) from among those advertised by the AP. If there is no match in capabilities between the AP and the STA, the AP refuses the Association Request. The STA blocks it too, in case it has associated with a rogue AP or someone is inserting frames illicitly on its channel.

2. Authentication: During this phase, the STA and AS prove their identities to each other. The AP blocks non-authentication traffic between the STA and AS until the authentication transaction is successful. The AP does not participate in the authentication transaction other than forwarding traffic between the STA and AS. We can think of authentication phase as consisting of the following three phases:

   - Connect to AS: The STA sends a request to its AP (the one with which it has an association) for connection to the AS. The AP acknowledges this request and sends an access request to the AS.
   - EAP exchange: This exchange authenticates the STA and AS to each other. A number of alternative exchanges are possible, as explained subsequently.
   - Secure key delivery: Once authentication is established, the AS generates a master session key (MSK), also known as the Authentication, Authorization, and Accounting (AAA) key and sends it to the STA. As explained subsequently, all the cryptographic keys needed by the STA for secure communication with its AP are generated from this MSK. IEEE 802.11i does not prescribe a method

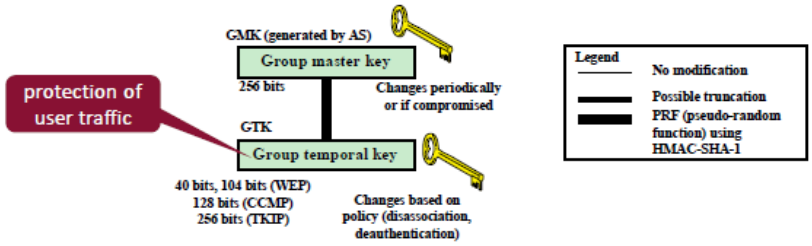
for secure delivery of the MSK but relies on EAP for this. Whatever method is used, it involves the transmission of an MPDU containing an encrypted MSK from the AS, via the AP, to the AS.

3. Key generation and distribution: The AP and the STA perform several operations that cause cryptographic keys to be generated and placed on the AP and the STA. Frames are exchanged between the AP and STA only.

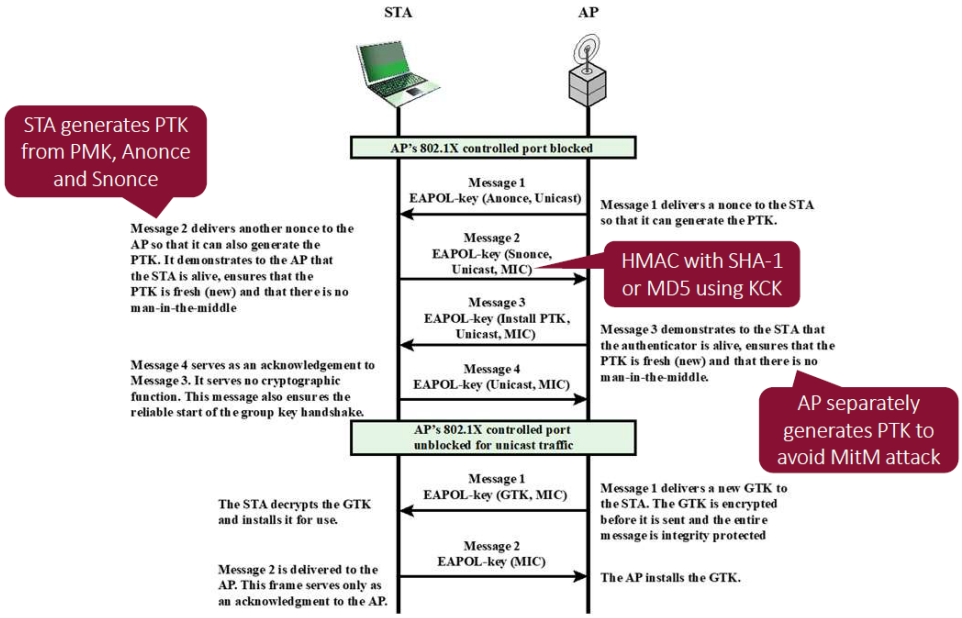


Phase 3: Key management (group keys)

- Group keys (for multicast communication) are generated by the AS
- Group temporal key (GTK) is shared by all stations in the same multicast group
- GTK changes whenever a station in the group leaves the network



Phase 3: 4-way handshake to exchange keys



Note: Message Authentication Code (MAC) is referred to as MIC in the context of 802.11, to avoid confusion with the Media Access Protocol

4. Protected data transfer: Frames are exchanged between the STA and the end station through the AP. As denoted by the shading and the encryption module icon, secure data transfer occurs between the STA and the AP only; security is not provided end-to-end.

Phase 4: Protected data transfer

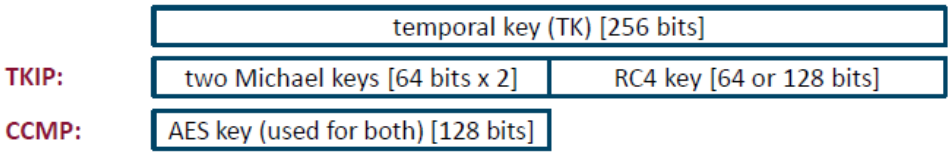
Temporal Key Integrity Protocol (TKIP)

- Hardware compatible with WEP devices, requiring only software changes
- Integrity: 64-bit MIC is generated using the Michael algorithm
- Confidentiality: MPDU+MIC are encrypted using RC4

Counter Mode-CBC MAC Protocol (CCMP)

- For newer devices, with dedicated hardware to perform the calculations
- Integrity: Provided by CBC-MAC
- Confidentiality: CTR block cipher mode combined with AES-128

Key generation



5. Connection termination: The AP and STA exchange frames. During this phase, the secure connection is torn down and the connection is restored to the original state.

Attacks against WPA

Dictionary and brute-force attacks

- Weak or predictable pre-shared keys (PSKs) are vulnerable
- A truly random 20 character passphrase selected from the 95 permitted characters provides the best protection

Attacks against WPA-TKIP

- Some attacks against WEP can be applied to WPA-TKIP as well
- Vanhoef and Piessens demonstrated an attack that can inject arbitrary packets with a payload of at most 112 bytes in 2013
- WPA-TKIP is currently deemed insecure, but still widely used

Attacks against WPA2

- Vanhoef demonstrated the Key reinstallation attack (KRACK) in 2017, which forces key and nonce reuse by repeating messages 3 and 4 of handshake

WPA3 improvements

Simultaneous Authentication of Equals (SAE)

- Extends the Dragonfly handshake procedure (RFC 7664)
- Protects against offline dictionary attacks (i.e., weak passwords)
- Offers forward secrecy (if an attacker learns the password, they cannot use that to decrypt old traffic)

Other improvements

- Wi-Fi device provisioning protocol (DPP) replaces insecure WPS to simplify configuration of devices without displays (e.g., sensors)
- Unauthenticated encryption protects open Wi-Fi networks against passive eavesdropping attacks (e.g., public hotspots)
- Increased session key sizes (e.g., AES-GCM256, HMAC-SHA384)

7.2 Transport layer security (SSL/TLS)

Two important TLS concepts are the TLS session and the TLS connection, which are defined in the specification as follows:

- Connection: A connection is a transport that provides a suitable type of service. For TLS, such connections are peer-to-peer relationships. The connections are transient and every connection is associated with one session.

- Session: A TLS session is an association between a client and a server and is created by the Handshake Protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

TLS is designed to make use of TCP to provide a reliable end-to-end secure service. **TLS is not a single protocol but rather two layers of protocols**. The **TLS Record Protocol** provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, **can operate on top of TLS**. **Three higher-layer protocols are defined as part of TLS:** the Handshake Protocol; the Change Cipher Spec Protocol; and the Alert Protocol. These TLS-specific protocols are used in the management of TLS exchanges and are examined later. A fourth protocol, the Heartbeat Protocol, is defined in a separate RFC and is also discussed subsequently.
The TLS Record Protocol provides two services for TLS connections:

- Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of TLS payloads.

- Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

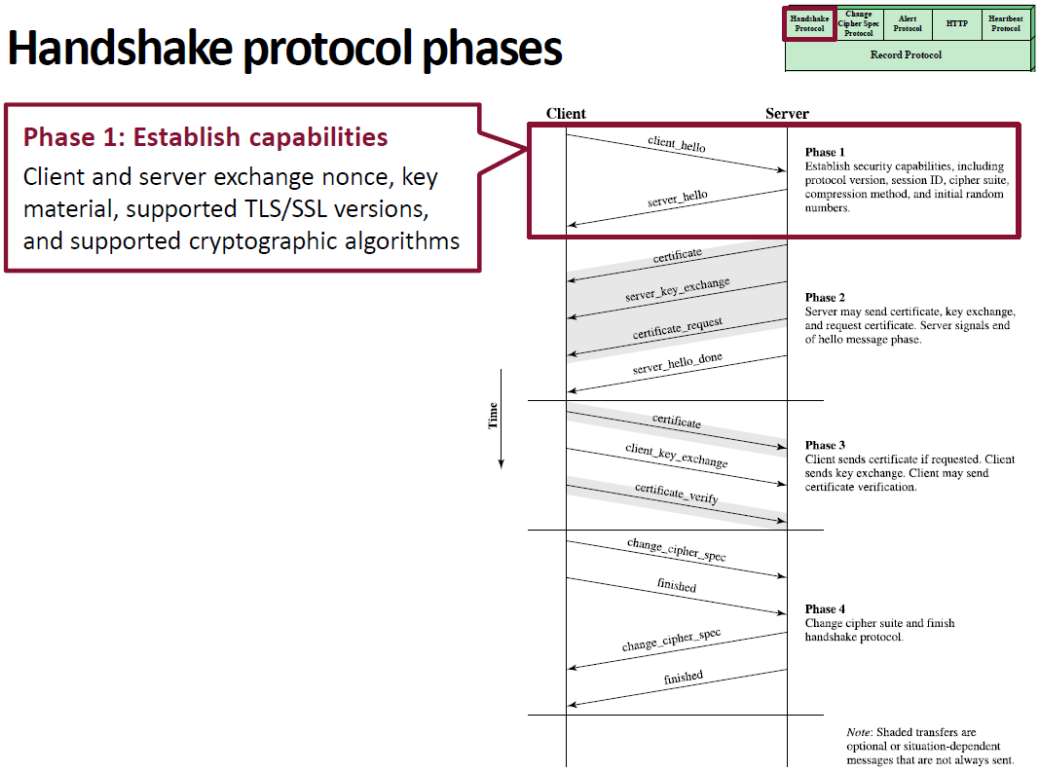
The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higherlevel users.

The first step is fragmentation. Each upper-layer message is fragmented into blocks of 2^{14} bytes or less. Next, compression is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes. In TLSv2, no compression algorithm is specified, so the default compression algorithm is null. The next step in processing is to compute a message authentication code over the compressed data.
Next, the compressed message plus the MAC are encrypted using symmetric encryption. Encryption may not increase the content length by more than 1024 bytes, so that the total length may not exceed 214 + 2048. For stream encryption, the compressed message plus the MAC are encrypted. For block encryption, padding may be added after the MAC prior to encryption. The padding is in the form of a number of padding bytes followed by a one-byte indication of the length of the padding.
The final step of TLS Record Protocol processing is to prepend a header consisting of the following fields:

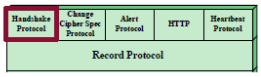
- Content Type (8 bits): The higher-layer protocol used to process the enclosed fragment.
- Major Version (8 bits): Indicates major version of TLS in use. For TLSv2, the value is 3.
- Minor Version (8 bits): Indicates minor version in use. For TLSv2, the value is 1.
- Compressed Length (16 bits): The length in bytes of the fragment. The maximum value is 214 + 2048.

7.2.1 Handshake protocol phases

The Handshake protocol consist of 4 phases.

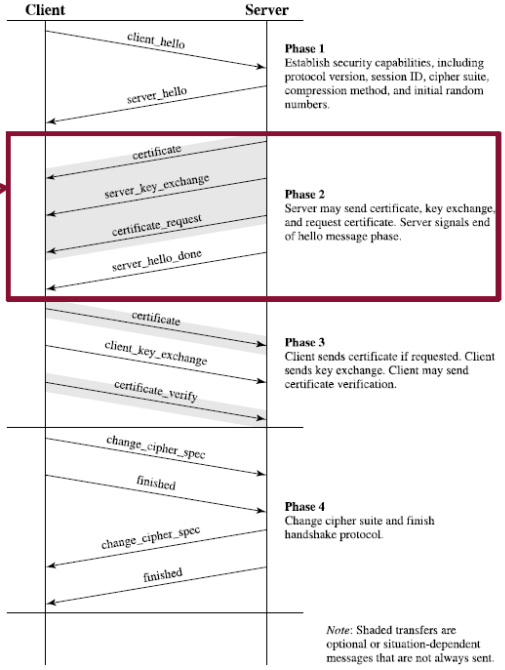


Handshake protocol phases

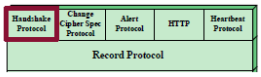


Phase 2: Server authentication

Depending on selected public key exchange method (RSA, Diffie-Hellman), server sends certificate, public key, and/or client certificate request

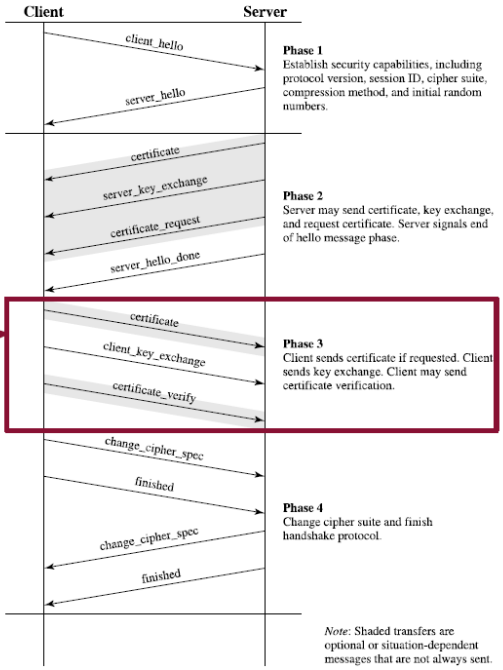


Handshake protocol phases

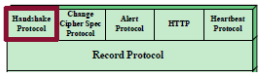


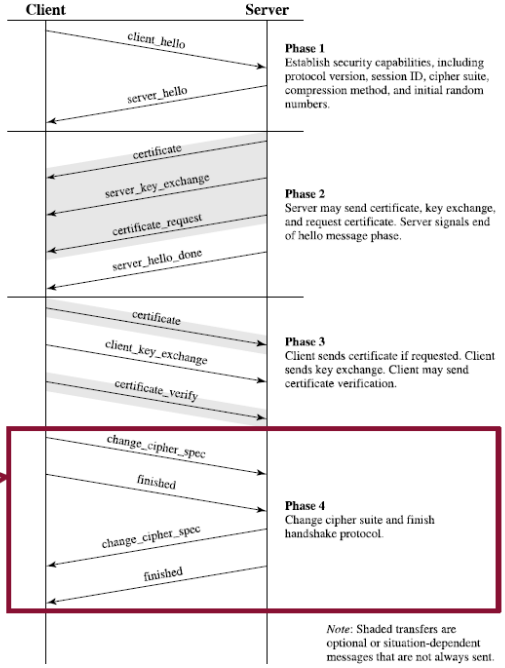
Phase 3: Client authentication

Client sends a certificate and verification and its own public key (obligatory)



Handshake protocol phases





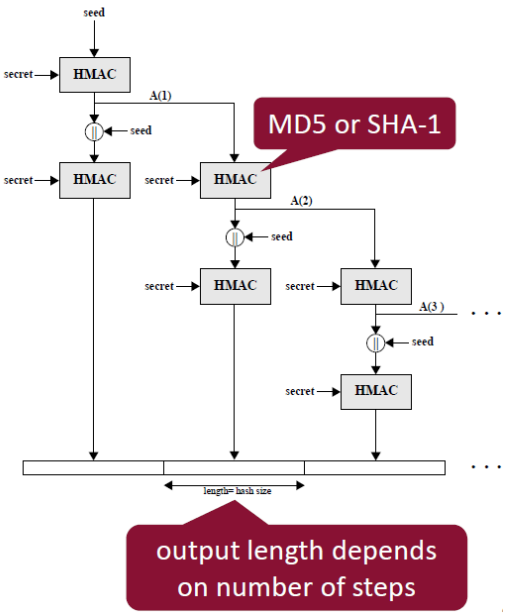
Phase 4: Finish

- Part of change cipher spec protocol
- Client and server send a message encrypted with the new algorithms and keys for verification

7.2.2 Creation of symmetric keys

Creation of symmetric keys

- pre_master_secret is generated by client (RSA) or client and server (Diffie-Hellman) and exchanged
- master_secret is calculated using PRF using pre_master_secret (until 48 bytes are produced)
- cryptographic parameters (MAC keys, session keys, IVs) are generated using PRF using master_secret until enough bytes are generated



The creation is in two stages. First, a pre_master_secret is exchanged. Second, the master_secret is calculated by both parties. For pre_master_secret exchange, there are two possibilities: RSA of Diffie-Hellman.
Both sides now compute the master_secret as
master_secret = PRF(pre_master_secret, master secret, ClientHello.random || ServerHello.random)
where ClientHello.random and ServerHello.random are the two nonce values exchanged in the initial hello messages.
The algorithm is performed until 48 bytes of pseudorandom output are produced. The calculation of the key block material (MAC secret keys, session encryption keys, and IVs) is defined as:
key_block = PRF(SecurityParameters.master_secret, key expansion, SecurityParameters.server_random || SecurityParameters.client_random)
until enough output has been generated.
CipherSpecs require a client write MAC secret, a server write MAC secret, a client write key, a server write key, a client write IV, and a server write IV,

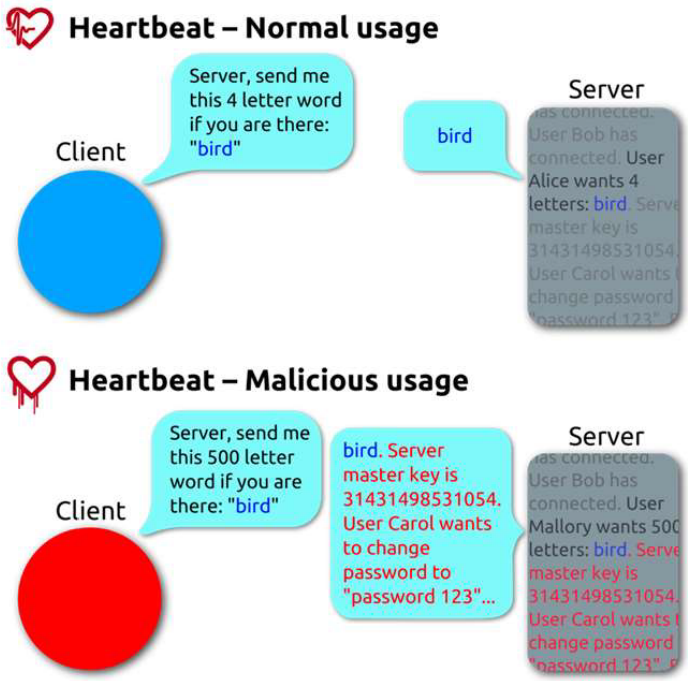
which are generated from the master secret in that order. These parameters are generated from the master secret by hashing the master secret into a sequence of secure bytes of sufficient length for all needed parameters.
The generation of the key material from the master secret uses the same format for generation of the master secret from the pre-master secret as
`key_block =`
`MD5(master_secret || SHA(A || master_secret || ServerHello.random || ClientHello.random)) ||`
`MD5(master_secret || SHA(BB || master_secret || ServerHello.random || ClientHello.random)) ||`
`MD5(master_secret || SHA(CCC || master_secret || ServerHello.random || ClientHello.random)) ||`
`...`
until enough output has been generated. The result of this algorithmic structure is a pseudorandom function. We can view the master_secret as the pseudorandom seed value to the function. The client and server random numbers can be viewed as salt values to complicate cryptanalysis.
TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation. The objective is to make use of a relatively small, shared secret value but to generate longer blocks of data in a way that is secure from the kinds of attacks made on hash functions and MACs. The data expansion function makes use of the HMAC algorithm with either MD5 or SHA-1 as the underlying hash function.

7.3 Attacks against SSL and TLS

7.3.1 Heartbleed attack

Caused by a minor programming error in the OpenSSL implementation: `memcpy(bp, pl, payloadsize)`. If `pl` array is smaller than what `payloadsize` makes believe, old data in memory alloted to `bp` array is not overwritten and returned to the attacker.

Heartbleed: The Internet's worst vulnerability to date



The heartbleed fix

```
/* Read type and payload length first */
if (1 + 2 + 16 > s->s3->rrec.length)
      return 0; // silently discard

hbtype = *p++;
n2s(p, payload);

if (1 + 2 + payload + 16 > s->s3->rrec.length)
      return 0; //silently discard per RFC 6520 sec. 4

pl = p;
```

protect against zero-length heartbeats

make sure heartbeat length is correct

7.3.2 POODLE

POODLE is an attack against TLS.

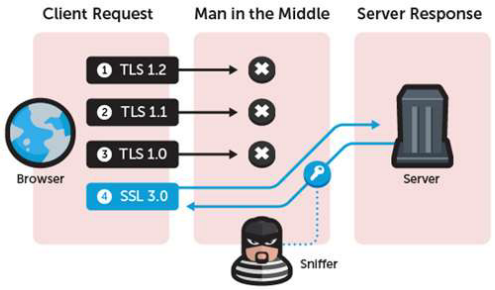
POODLE: Padding Oracle On Downgraded Legacy Encryption

Two stage attack

- Trick the server into reverting to SSL 3.0 compatibility mode
- Perform a Padding Oracle man-in-the-middle attack

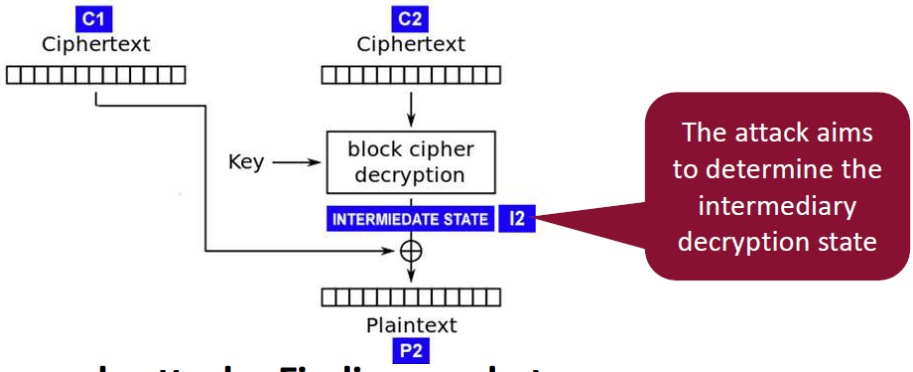
Step 1: trick the server into reverting to SSL 3.0

- Assume an attacker controls the network between the client and server
- Attacker can interfere during TLS handshake when client offers TLS 1.0 or later
- Attacker only allows successful handshake when client offers SSL 3.0 downgrade



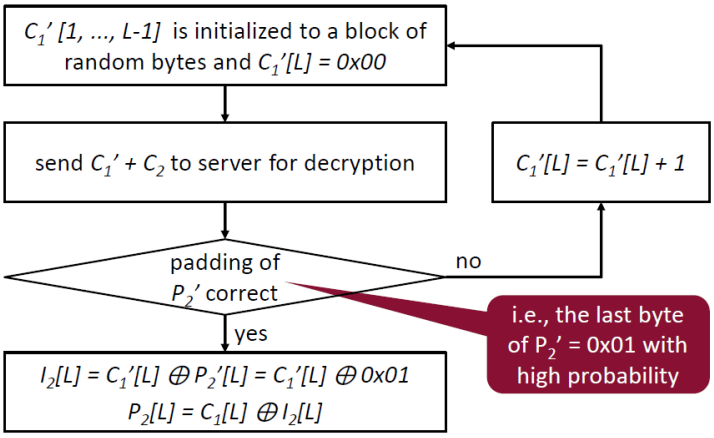
Padding oracle attacks

- In Cipher Block Chaining (CBC) mode, each block of plaintext is XORed with the previous ciphertext block before encryption
- Knowing whether or not a given ciphertext produces plaintext with valid padding is enough to break CBC encryption
- In PKCS7 padding, each padded byte equals the length of the padding (e.g., a 13-byte block is padded with 3 0x03 bytes to form a valid full 16-byte block)

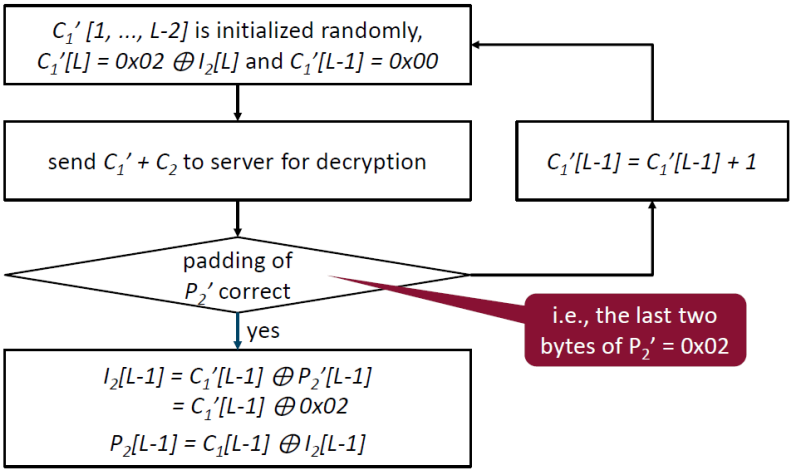


Padding oracle attacks: Finding one byte

Given a block length L, the last byte of the intermediary state I_2 can be determined as follows:



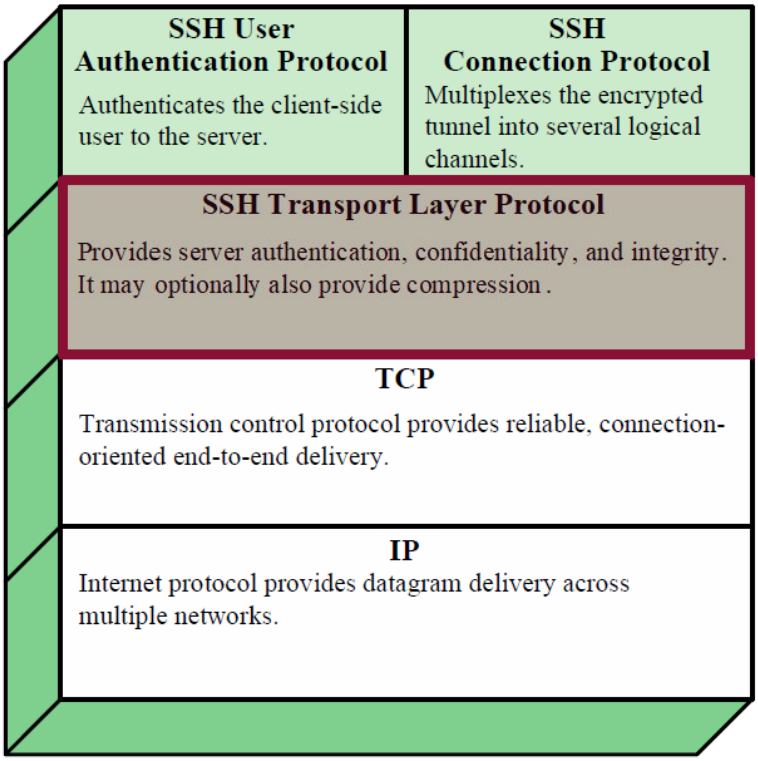
Padding oracle attacks: finding the previous byte



7.4 Secure SHell

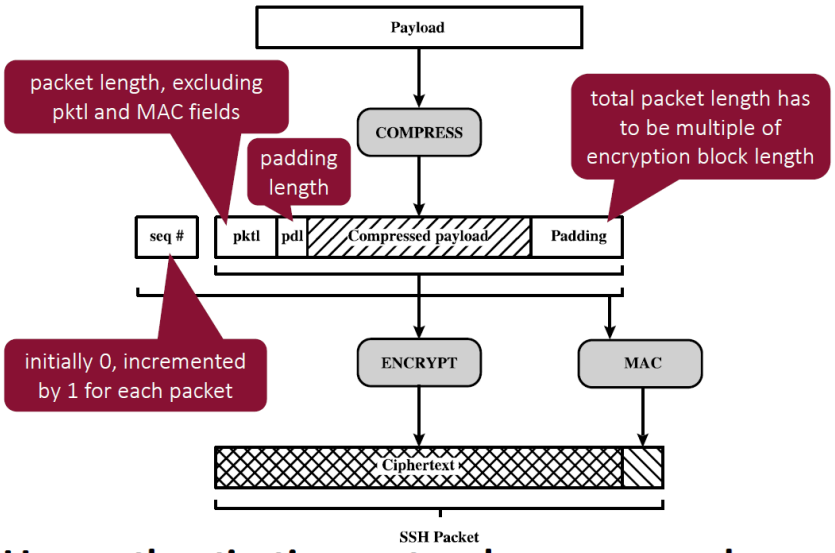
Secure Shell (SSH) is a protocol for secure network communications designed to be relatively simple and inexpensive to implement, it is used for both secure remote logon facilities and network functions as file transfer and email.

SSH protocol stack

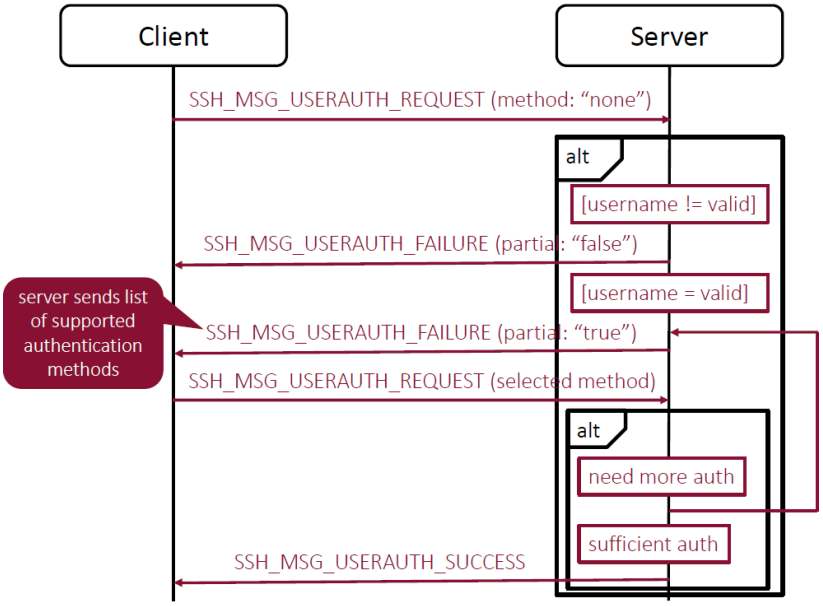


If you want details on the handshake protocol, do go to the slides (it's too much and too detailed for here).

Transport layer protocol: packet format

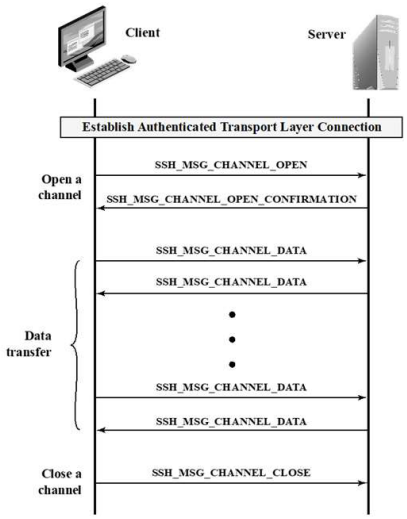


User authentication protocol: message exchange



Connection protocol

Multiplexes a number of logical channels over the secure authenticated SSH connection (also called tunnel)

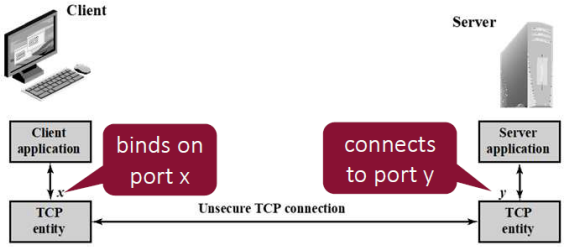


Channel types

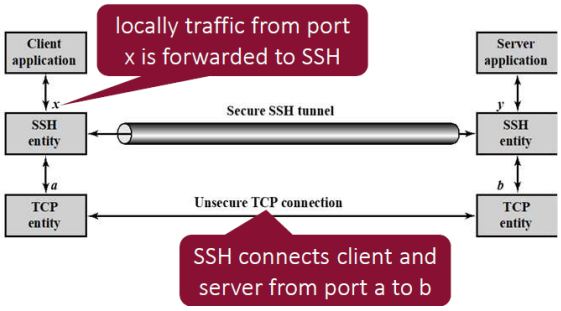
- session: remote execution of a program
- x11: remote GUI forwarding using the X Window System
- port forwarding: Convert insecure TCP connection to secure SSH connections

SSH port forwarding

insecure TCP connection



secure TCP via SSH tunnel



Port forwarding: traffic is intercepted on the local host (client)/remote host (server) and forwarded to the given host and port via the remote host (server)/local host (client).
Command: `ssh <-L/-R> x:remotehost:y user@sshserver`

8 Application layer security

8.1 Email security

Email security threats

Threat	Impact on Purported Sender	Impact on Receiver	Mitigation
Email sent by unauthorized MTA in enterprise (e.g. malware botnet)	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	Unsolicited bulk email (UBE) and/or email containing malicious links may be delivered into user inboxes	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email message sent using spoofed or unregistered sending domain	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered into user inboxes	
Email message sent using forged sending address or email address (i.e. phishing, spear phishing)	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered. Users may inadvertently divulge sensitive information or PII.	
Email modified in transit	Leak of sensitive information or personal identifiable information (PII).	Leak of sensitive information, altered message may contain malicious information	Use of TLS to encrypt email transfer between server. Use of end-to-end email encryption.
Disclosure of sensitive information (e.g. PII) via monitoring and capturing of email traffic	Leak of sensitive information or PII.	Leak of sensitive information, altered message may contain malicious information	
Unsolicited Bulk Email (i.e. spam)	None, unless purported sender is spoofed.	UBE and/or email containing malicious links may be delivered into user inboxes	Techniques to address UBE.
DoS/DDoS attack against an enterprises' email servers	Inability to send email.	Inability to receive email.	Multiple mail servers, use of cloud-based email providers.

Email security mechanisms

STARTTLS	SMTP security extension that provides digital signatures and encryption with TLS over SMTP
S/MIME or PGP	Provides digital signatures and encryption of the entire email message body transferred within SMTP messages
DNSSEC	Provides authentication and integrity of DNS data, used as a basis for various email security protocols
DANE	DNS-based authentication of named entities provides an alternative to certification authorities using DNSSEC
SPF	Sender Policy Framework uses DNS to associate domain names with specific valid message sending IP addresses
DKIM	DomainKeys Identified Mail signs the header and body of messages to provide source domain validation
DMARC	Domain-based Message Authentication, Reporting and Conformance validates SPF and DKIM policies

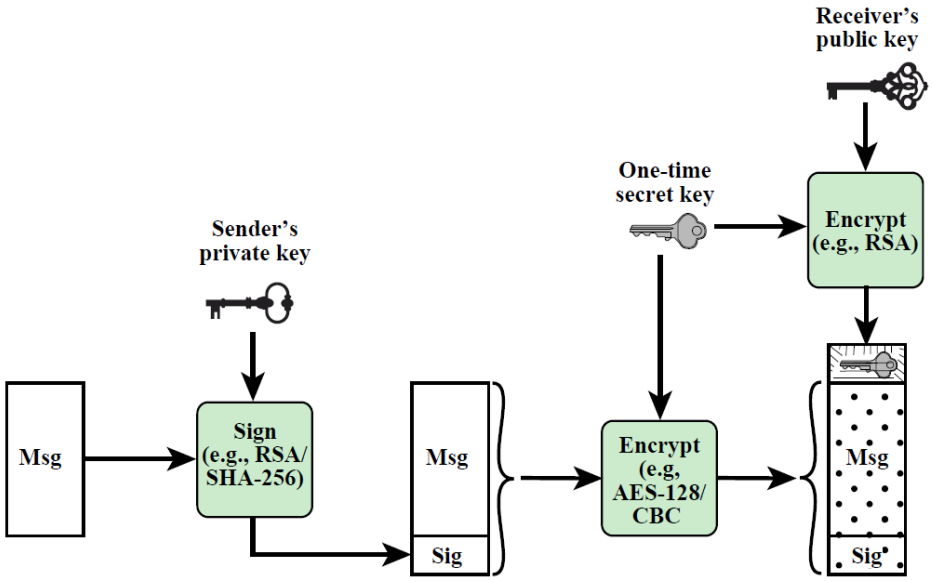
8.1.1 SMIME

Secure MIME (S/MIME)

- Compatible with MIME (RFC 2045—2049)
- Standardized in many RFCs (e.g., 5750, 5751, 2634)

Functions	Typical Algorithm	Typical Action
Digital signature	RSA/SHA-256	A hash code of a message is created using SHA-256. This message digest is encrypted using RSA with the sender's private key and included with the message.
Message encryption	AES-128 with CBC	A message is encrypted using AES-128 with CBC with a one-time session key generated by the sender. The session key is encrypted using RSA with the recipient's public key and included with the message.
Compression	unspecified	A message may be compressed for storage or transmission.
E-mail compatibility	Radix-64 conversion	To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.

Simplified S/MIME message sending



8.1.2 PGP

An alternative email security protocol is Pretty Good Privacy (PGP), which has essentially the same functionality as S/MIME. The actual operation of PGP, as opposed to the management of keys, consists of four services: authentication, confidentiality, compression, and e-mail compatibility.
Both authentication and confidentiality services may be used for the same message. First, a signature is generated for the plaintext message and prepended to the message. Then the plaintext message plus signature is encrypted using CAST-128 (or IDEA or 3DES), and the session key is encrypted using RSA (or ElGamal). When both services are used, the sender first signs the message with its own private key, then encrypts the message with a session key, and finally encrypts the session key with the recipients public key.
As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage.
The signature is generated before compression for two reasons:

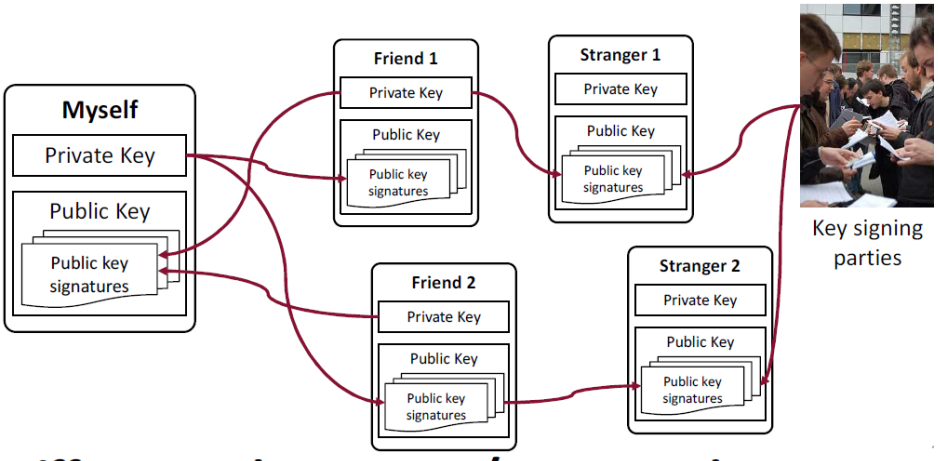
1. It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.

2. Even if one were willing to generate dynamically a recompressed message for verification, PGPs compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed forms. However, these different compression algorithms are interoperable because any version of the algorithm can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm.

Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

PGP uses Web of Trust to validate public keys

- It relies on self-signed certificates, that can in turn be signed by your circle of trusted friends
- This creates a web of people that trust each other (in)directly



Differences between S/MIME and PGP

Key certification

- S/MIME uses X.509 certificates and CAs to certify public keys
- In PGP, users generate their own public-private key pairs, trust is gained by signing them by someone already trusted by the recipient (Web of Trust)

Key distribution

- S/MIME attaches the sender's public key and the certificate to each message
- PGP puts the responsibility of distributing keys with the user, which often post them to TLS-protected websites (e.g., https://pgp.mit.edu/)

➔ Due to greater confidence in the CA system over a Web of Trust, NIST recommends the use of S/MIME over PGP

8.2 Web security

The Open Web Application Security Project (OWASP) is a 501(c)(3) worldwide not-forprofit charitable organization focused on improving the security of software. OWASP is in a unique position to provide impartial, practical information about AppSec to individuals, corporations, universities, government agencies and other organizations worldwide. Operating as a community of like-minded professionals, OWASP issues software tools and knowledge-based documentation on application security.
OWASP top 10 web application vulnerabilities:

1. Injection: Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

2. Broken Authentication: Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

3. Sensitive Data Exposure: Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

4. XML External Entities (XXE): Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

5. Broken Access Control: Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as

access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

6. Security Misconfiguration: Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

7. Cross-Site Scripting (XSS): XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with usersupplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

8. Insecure Deserialization: Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

9. Using Components with Known Vulnerabilities: Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

10. Insufficient Logging & Monitoring: Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

SQL injection prevention mechanisms

- Parameterize your SQL statement
- Encode and validate input
- Prevent untrusted input from being interpreted as part of query

Parameterization example:

```
txtNam = getRequestString("CustomerName");
txtAdd = getRequestString("Address");
txtCit = getRequestString("City");
txtSQL = "INSERT INTO Customers
(CustomerName,Address,City) Values(@0,@1,@2)";
db.Execute(txtSQL,txtNam,txtAdd,txtCit);
```

SQL engine checks parameters and ensures that they are correct, as well as treated literally

XML External Entity (XXE)

- Poorly configured XML parsers are vulnerable
- Exploits the parser's desire to blindly process external DTD entities
- Allows various attacks, such as accessing local files, server-side request forgery (SSRF), remote file includes and XML DoS attacks

Prevention:

- Easiest way is to completely disable external entities

Example (accessing local files):

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
    <!ELEMENT foo ANY>
    <!ENTITY xxe SYSTEM "file:///etc/passwd" >
]>
```

Cross-Site Scripting (XSS)

- Attacker injects malicious scripts in otherwise trusted web pages
- Can occur in any web application that uses input from a user within the output it generates without validating or encoding it

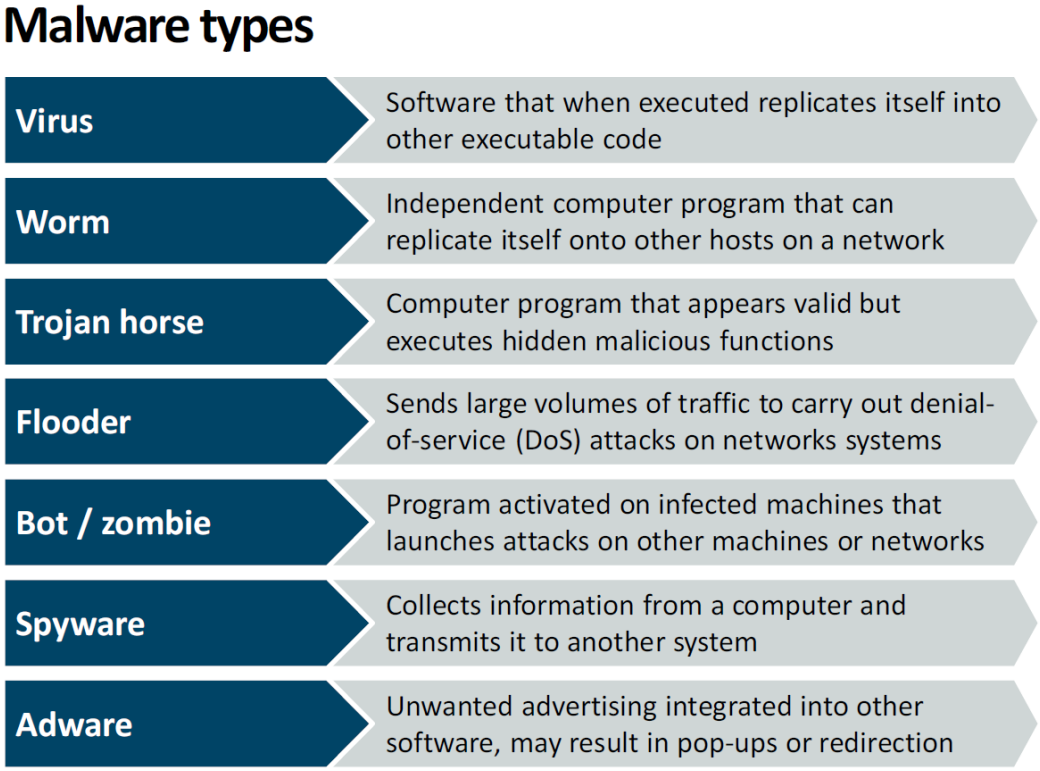
Two types of XSS attacks

- Stored XSS attacks: The injected script is permanently stored on the target servers (e.g., in a database, message forum, comment field)
- Reflected XSS attacks: The injected script is reflected off the server (e.g., in an error message, search result) and is delivered to the victim using some other way (e.g., in an email message or via another website)

XSS attack prevention

- Output encoding
- Input validation
- Content security policy
- Use security frameworks (e.g., OWASP ESAPI, AntiSamy)

8.3 Malicious software



8.3.1 Trojan horse browser: Zeus-in-the-mobile (ZitMo)

Mobile ZeuS, or Trojan-Spy.*.Zitmo, was designed for one sole purpose: to quickly steal mTAN[6] codes without mobile users noticing. The user attempts to navigate to his banks webpage and log into the system. The PC version of ZeuS registers that the victim is going to an address of interest, and modifies this webpage in the browser so that the personal data entered by the user for authentication is not sent to the bank, but to the ZeuS botnet command center. A modified authentication page would also ask the user to enter data about their mobile device (the make, model, and telephone number) in addition to their username and password. Users were told that the data was requested for the alleged purpose of certificate updates. Sooner or later, users who provided information to malicious users about their cell phones would receive a text messages asking them to install a new security certificate. This

[6]mobile transaction authentication numbers

security certificate could be downloaded via a link that was provided in the text message. However, this certificate was in fact the mobile version of the ZeuS Trojan. Then his mobile phone would be infected by ZitMo, the primary function of which is to send a text message to a malicious users phone as specified in the body of the Trojan.
The malicious users who successfully used the PC-based ZeuS to steal personal user data for online banking systems and infect victims phones with ZitMo were thus able to overcome the last barrier of online banking security systems: the mTAN code. By entering a users login and password, they were able to access their bank accounts and conduct transactions (such as transferring money from the users account to their own bank accounts). These transactions required additional authentication using a code sent by the bank via text message to the clients phone.
After the client submitted a transaction request, the bank would send the client an authentication code. The code would be sent to the ZitMo-infected handset, which immediately forwarded it to the malicious users number, who would then use the stolen mTAN to authenticate the transaction. And the victim would be none the wiser.
The attacks are generally orchestrated as follows:

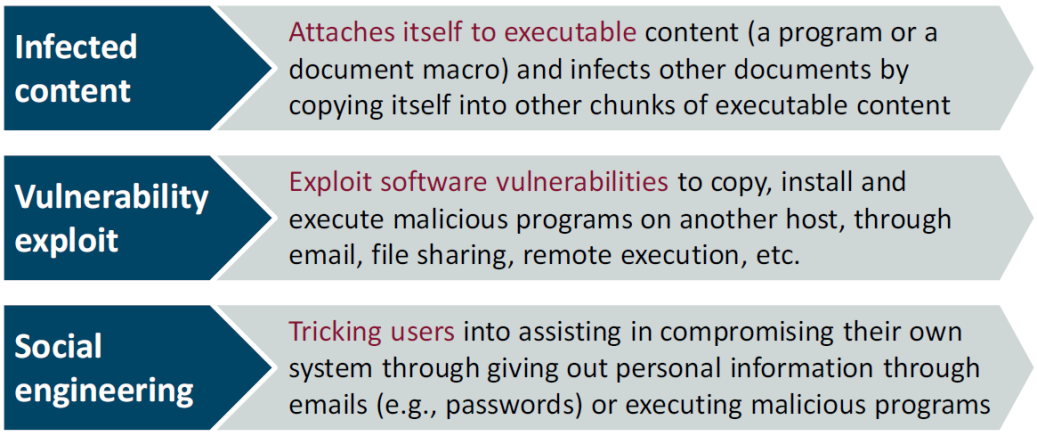
1. Cyber criminals use the PC-based ZeuS to steal the data needed to access online banking accounts and client cell phone numbers.

2. The victims mobile phone (see point 1) receives a text message with a request to install an updated security certificate, or some other necessary software. However, the link in the text message will actually lead to the mobile version of ZeuS.

3. If the victim installs this software and infects the phone, the malicious user can then use the stolen personal data and attempt to make cash transactions from the compromised account, but still needs an mTAN code to authenticate the transaction.

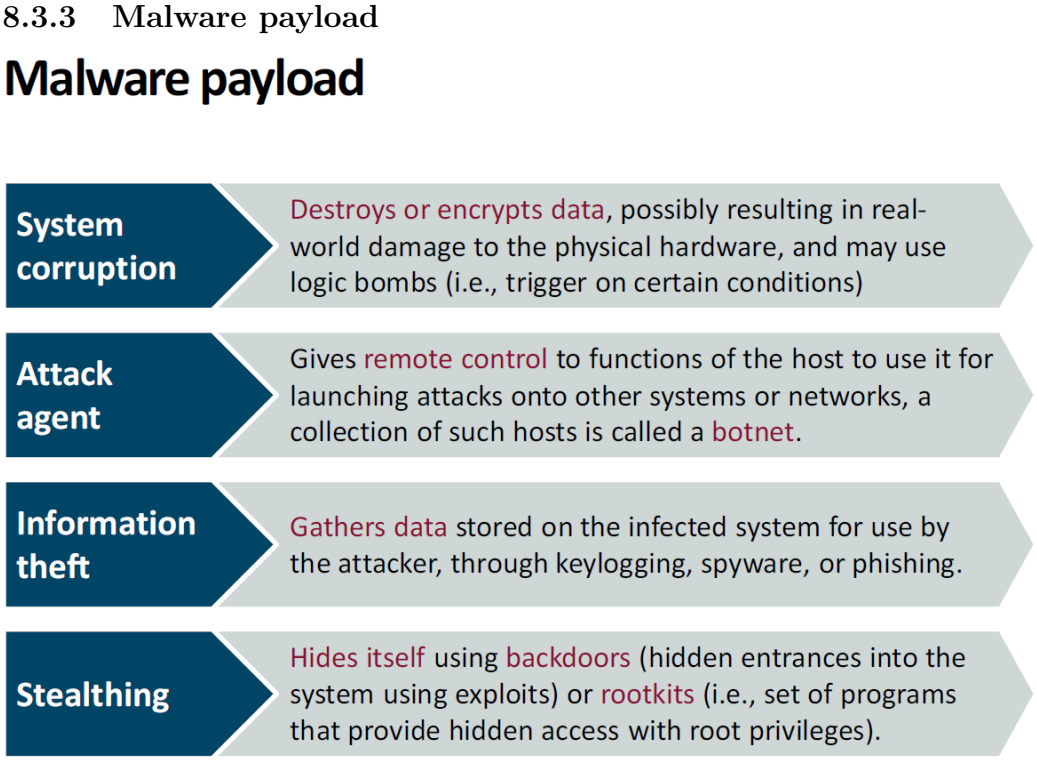
4. The bank sends out a text message with the mTAN code to the clients mobile phone. ZitMo forwards the text message with the mTAN code to the malicious users phone. The malicious user is then able to use the mTAN code to authenticate the transaction.

8.3.2 Malware propagation



The first category of malware propagation concerns parasitic software fragments that attach themselves to some existing executable content. The fragment may be machine code that infects some existing application, utility, or system program, or even the code used to boot a computer system.
A worm is a program that actively seeks out more machines to infect, and then each infected machine serves as an automated launching pad for attacks on other machines. Worm programs exploit software vulnerabilities in client or server programs to gain access to each new system. They can use network connections to spread from system to system.
The final category of malware propagation we consider involves social engineering, tricking users to assist in the compromise of their own systems or personal information. This can occur when a user views and responds to some SPAM e-mail or permits the installation and execution of some Trojan horse program or scripting code.

8.3.3 Malware payload



An early payload seen in a number of viruses and worms resulted in data destruction on the infected system when certain trigger conditions were met. A related payload is one that displays unwanted messages or content on the users system when triggered. More seriously, another variant attempts to inflict realworld damage on the system. All of these actions target the integrity of the computer systems software or hardware, or of the users data. These changes may not occur immediately, but only when specific trigger conditions are met that satisfy their logic-bomb code. As an alternative to just destroying data, some malware encrypts the users data and demands payment in order to access the key needed to recover this information. This is sometimes known as ransomware.

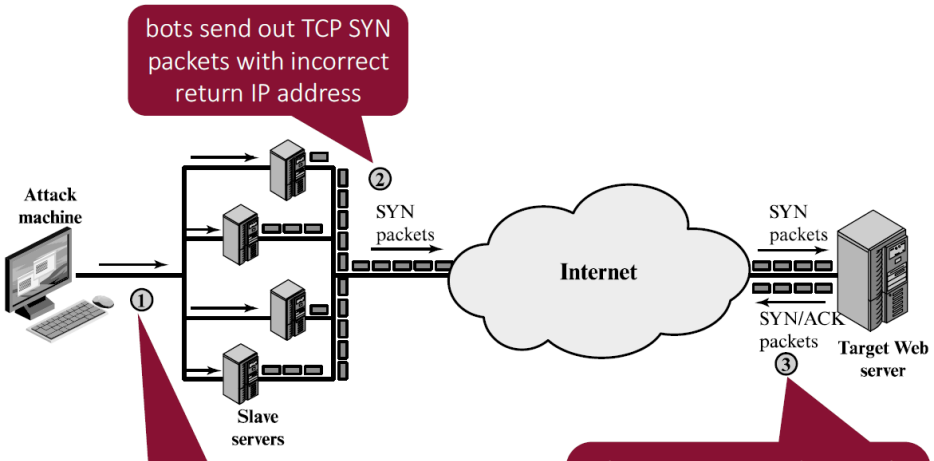
The next category of payload we discuss is where the malware subverts the computational and network resources of the infected system for use by the attacker. Such a system is known as a bot (robot), zombie, or drone, and secretly takes over another Internet-attached computer and then uses that computer to launch or manage attacks that are difficult to trace to the bots creator. The bot is typically planted on hundreds or thousands of comput-

ers belonging to unsuspecting third parties. The collection of bots often is capable of acting in a coordinated manner; such a collection is referred to as a botnet. This type of payload attacks the integrity and availability of the infected system.
We now consider payloads where the malware gathers data stored on the infected system for use by the attacker. A common target is the users login and password credentials to banking, gaming, and related sites, which the attacker then uses to impersonate the user to access these sites for gain. Less commonly, the payload may target documents or system configuration details for the purpose of reconnaissance or espionage. These attacks target the confidentiality of this information.
The final category of payload we discuss concerns techniques used by malware to hide its presence on the infected system and to provide covert access to that system. This type of payload also attacks the integrity of the infected system.

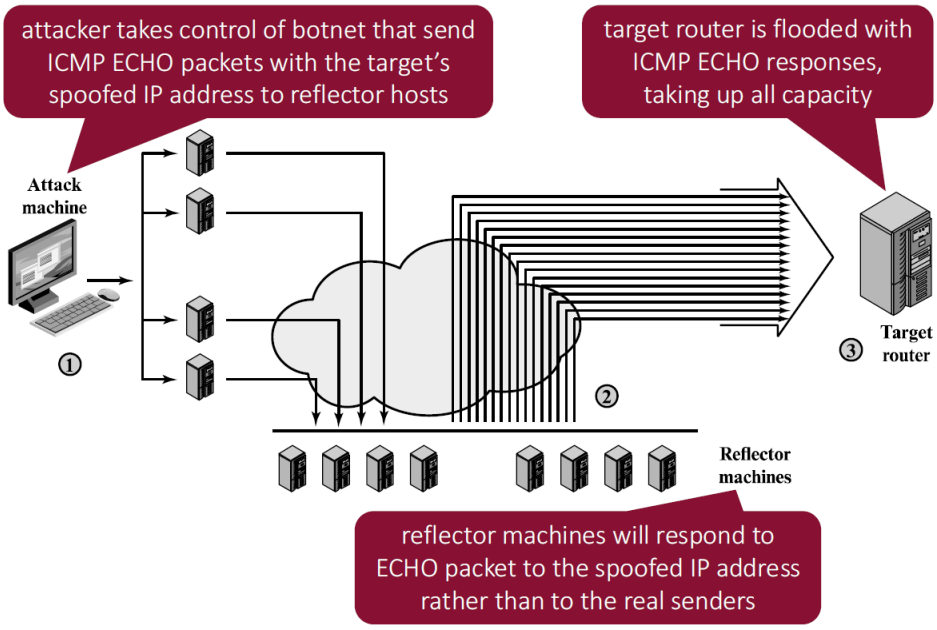
8.3.4 DDoS

A denial-of-service (DoS) attack is an attempt to prevent legitimate users of a service from using that service. When this attack comes from a single host or network node, then it is simply referred to as a DoS attack. A more serious threat is posed by a DDoS attack. DDoS attacks make computer systems inaccessible by flooding servers, networks, or even end-user systems with useless traffic so that legitimate users can no longer gain access to those resources. In a typical DDoS attack, a large number of compromised hosts are amassed to send useless packets.
A DDoS attack attempts to consume the targets resources so that it cannot provide service. One way to classify DDoS attacks is in terms of the type of resource that is consumed. Broadly speaking, the resource consumed is either an internal host resource on the target system or data transmission capacity in the local network to which the target is attacked.

DDoS: SYN flood attack



DDoS: ICMP attack



9 Guest lecture: IoT

9.1 Introduction

Security challenges in IoT:

1. Rapid Proliferation of IoT Devices
   One of the biggest IoT security challenges is keeping up with the production of IoT connected devices. The concern is not that the IoT devices are increasing each day. The issue is that these devices are not protected. Most small and medium-sized businesses continue to connect several unprotected objects, such as smart lighting, barcode readers, smart locks, security cameras, and HVAC systems to their IT networks.

2. It is Hard to Protect What You Cant See
   Hackers are proactively looking for new techniques to penetrate IoT devices, so organizations should always find vulnerabilities and fix them as soon as they occur. Predicting and preventing these threats is even more desirable. However, the primary attack vector is usually the invisibility of networks. The fact that most enterprises cant detect devices on their system means that they wont protect what they cannot see.

3. Default Passwords and Brute-Forcing
   Many manufacturers supply consumers with products that have default login information, such as the username and passwords. This information is what cybercriminals are looking for to carry out a brute-force attack on the affected devices. The Mirai botnet serves as an example of the issues that come with selling devices with default credentials and not informing the consumers to change them. The most worrying thing is that most of the default passwords are painfully obvious, such that even an ordinary user can guess them. So, with inadequate out-of-the-box security features, consumers ought to take security measures into their own hands.

4. Lack of Proper Testing and Updating
   As the need for connected devices increases, more manufacturers are working hard to produce new ones. The only challenge is that they do not pay enough attention to security. Most of these devices rarely

get updates. Sometimes, they dont even get updates at all. What it means is that these devices are only secure at the time of purchase, but remain open to attacks once hackers find security loopholes. In other words, if no one fixes the security issues on time, for both software and hardware, these products remain vulnerable to attacks.

5. Increased Attacks Targeting IoT Devices
   As the IoT technology continues to advance, so does the complexity of attacks. In fact, IoT devices were one of the main cyberattack vectors in 2018. It seems the more interconnected devices we have, the more interesting it gets for the attackers, which calls for the industry to come up with more effective security initiatives. Fortunately, most of the security vulnerabilities in 2020 will be the ones known to IT security professionals. Organizations should, therefore, stay focused on fixing the risks they already know. It could be as simple as installing the right Windows error solutions with malware protection capabilities.

6. Data Privacy and Security
   In todays interconnected world, data privacy and security are becoming increasingly challenging. The main issue is that data gets transferred between multiple devices. At one moment, it may be stored on a computer, and the next moment it is on the web or the cloud. As you know, not all devices or platforms that transmit data are secure. It gets worse when you consider that the information goes through the internet, which could leak it out to criminals. Organizations should design privacy and compliance rules that anonymize personal information before storing it. They should then apply this practice to the web, mobile, and cloud applications, including services that are used to access and process data on IoT devices.

7. Poorly Encrypted Communications
   Encryption is one of the great ways to keep attackers away from accessing crucial data. Unfortunately, it is also one of the leading IoT security challenges. Currently, most devices lack process and storage capabilities found in standard computers. What this means is that cybercriminals can easily manipulate the algorithms that are supposed to protect the devices. If no one addresses the issue, then encryption wont help much.

8. Larger and More Frequent IoT Botnets Targeting Cryptomining
   A botnet can access data, send spam, and allow access to devices without users knowledge. An example of such attacks is the 2016 Mirai, a botnet powered by over 60 login credentials. The recent rise in currency valuation, coupled with the ongoing mining spree, would make it too attractive for attackers, trying to benefit from the crypto-craze. The Blockchain framework is usually resistant to hacking, but there have been successful attempts to penetrate the system and extract usernames and passwords. Actually, the main vulnerability is the Blockchain app development, and not the Blockchain itself. So, IoT botnets and Blockchain breaches will continue to pose a high risk to IoT structures, applications, and platforms that rely on Blockchain technology.

9. IoT Automation Challenges
   As IoT devices continue to make way into our lives, organizations will reach a point where they will be dealing with thousands or even millions of devices at once. Obviously, it would be difficult for anyone to manage the massive amount of data produced by these interconnected devices, especially from networking and data collection perspectives. Network and IoT admins will, therefore, need to use AI tools to automate data management. However, the use of these tools to handle big data can pose a security threat if not managed well. A minor misconfiguration of AI tools can trigger an outage.

10. Unauthorized Home Access
    Perhaps one of the places where we might feel the impact of IoT the most is in our homes. Today, most developers fit new buildings with IoT devices. While this might be a good thing, it can also be a security concern. Not everyone is knowledgeable about the best practices that should be maintained about IoT security. Some IoT devices you use at home can expose your IP address, thus revealing your residential address. Anyone who has access to this information can use it to carry out criminal activities.