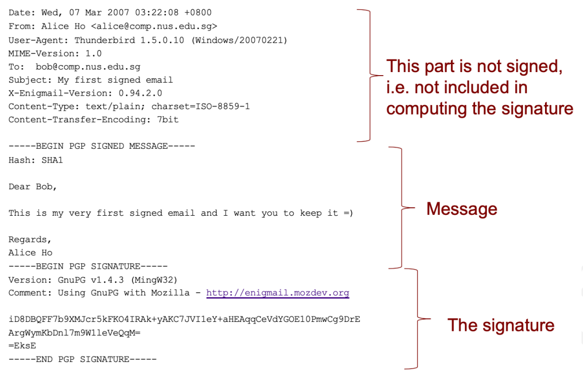
CS2107 Live Class Lec4: PKI+ Channel Security

$_/|/|U_Ch@NgRu!$

May 11, 2021

1 Example of ”signed” email using PGP public key



```
Date: Wed, 07 Mar 2007 03:22:08 +0800
From: Alice Ho <alice@comp.nus.edu.sg>
User-Agent: Thunderbird 1.5.0.10 (Windows/20070221)
MIME-Version: 1.0
To:  bob@comp.nus.edu.sg
Subject: My first signed email
X-Enigmail-Version: 0.94.2.0
Content-Type: text/plain; charset=ISO-8859-1
Content-Transfer-Encoding: 7bit

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Dear Bob,

This is my very first signed email and I want you to keep it =)

Regards,
Alice Ho
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.3 (MingW32)
Comment: Using GnuPG with Mozilla - http://enigmail.mozdev.org

iD8DBQFF7b9XMJcr5kFKO4IRAk+yAKC7JVI1eY+aHEAqqCeVdYGOE10PmwCg9DrE
ArgWymKbDnl7m9W1leVeQqM=
=EksE
-----END PGP SIGNATURE-----
```

This part is not signed, i.e. not included in computing the signature
Message
The signature

The upper part cannot be signed because that part needs to be modified by intermediate host

1.1 Man in the middle attack

Mallory sends his own public key and signed file to Bob

2 Key distribution

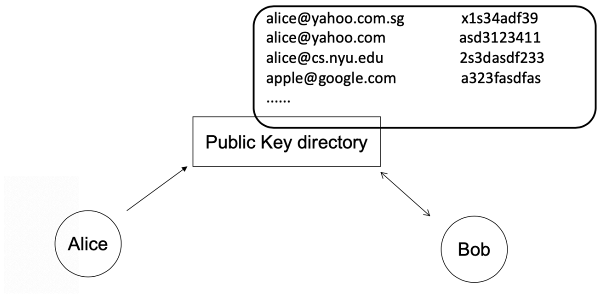
A channel is needed to distribute the public key “securely”.
There are three methods:

1. Public Announcement
2. Publishing publicly available directory
3. Public Key Infrastructure(PKI)

3 Public Announcement

The owner broadcasts her public key.For.e.g. Many owners listed their “PGP public key” in blog, personal webpage, etc. (PGP is a good standard and policy)

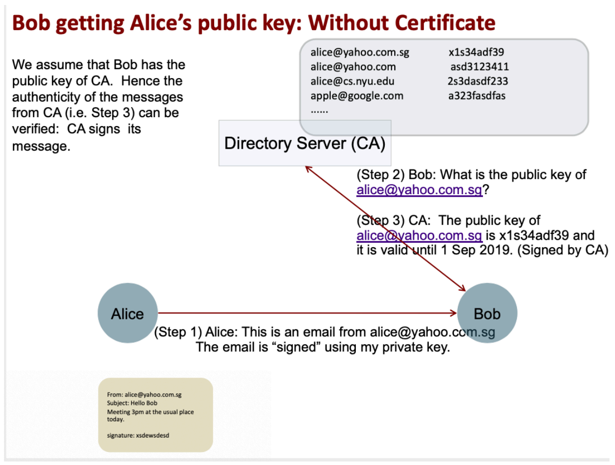
4 publicly available directory


alice@yahoo.com.sg x1s34adf39
alice@yahoo.com asd3123411
alice@cs.nyu.edu 2s3dasdf233
apple@google.com a323fasdfas
......
Public Key directory
Alice
Bob

Host can publish their public key to public available directory so that other agents can know

5 Certificate Authority

The CA issues and signs digital certificates
A trusted authority that manage a directory of public keys.
Anyone can send queries to search the directory
An entity can request adding its ublic key to the public directory.
CA also has it’s own public-private key(The CA’s public key should be securely distributed to all entities involved)(A secure channel is needed to distrubute the CA’s public key)
(Most OSes and browswes have a few pre-loaded CAs’ public keys: they are known as the ”root” CAs. Not all CAs’ public keys are preloaded)


Bob getting Alice's public key: Without Certificate

We assume that Bob has the public key of CA. Hence the authenticity of the messages from CA (i.e. Step 3) can be verified: CA signs its message.

alice@yahoo.com.sg	x1s34adf39
alice@yahoo.com	asd3123411
alice@cs.nyu.edu	2s3dasdf233
apple@google.com	a323fasdfas
……	

(Step 2) Bob: What is the public key of alice@yahoo.com.sg?

(Step 3) CA: The public key of alice@yahoo.com.sg is x1s34adf39 and it is valid until 1 Sep 2019. (Signed by CA)

(Step 1) Alice: This is an email from alice@yahoo.com.sg The email is "signed" using my private key.

From: alice@yahoo.com.sg
Subject: Hello Bob
Meeting 3pm at the usual place today.

signature: xsdewsdesd

In the example above, the CA's reply can be guaranteed from CA(because Bob know CA's public key)

5.1 Components of certificates

1. The identity of owner, e.g. "alice@yahoo.com"
2. The public key of the owner
3. The time window that this certificate is valid(when the certificate is expired)
4. The signature of CA

Some other possible information: function of the entity; whether the owner has a single or group of entity; Usage of the certificate

5.1.1 Why the certificate needs an expired date

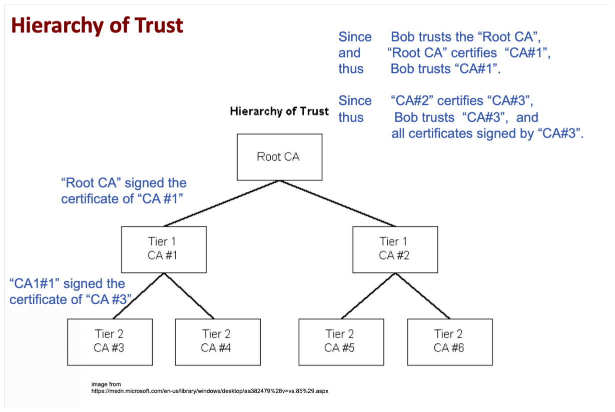
The DNA may be changed, the onwer of the identity may also be changed as time goes by

5.2 Responsibility of CA

1. issuing certificate
2. verify that the information is correct(the CA should check that the applicant indeed own the above domain name. This may involve manual checking and thus it could be costly.)

6 Certificate Chain

There are many CA’ s.
Most OS, browsers already have a few CA’ s public key pre-loaded. These are the “root CA” .
There is a trust hierarchy like DSA survey



Hierarchy of Trust

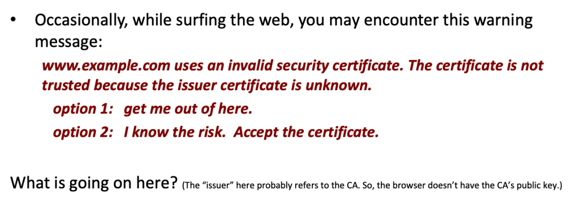
Since Bob trusts the “Root CA”,
and “Root CA” certifies “CA#1”,
thus Bob trusts “CA#1”.

Since “CA#2” certifies “CA#3”,
thus Bob trusts “CA#3”, and
all certificates signed by “CA#3”.

“Root CA” signed the certificate of “CA #1”

“CA1#1” signed the certificate of “CA #3”

7 Question



- Occasionally, while surfing the web, you may encounter this warning message:

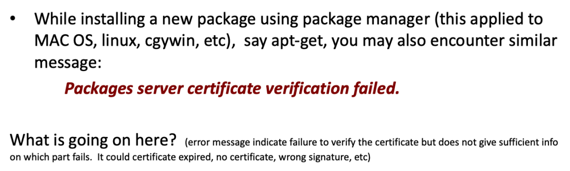
  www.example.com uses an invalid security certificate. The certificate is not trusted because the issuer certificate is unknown.

  option 1: get me out of here.

  option 2: I know the risk. Accept the certificate.

What is going on here? (The “issuer” here probably refers to the CA. So, the browser doesn’t have the CA’s public key.)

The https needs to verify the public key, but the CA is not known by the browser and the root CA



- While installing a new package using package manager (this applied to MAC OS, linux, cgywin, etc), say apt-get, you may also encounter similar message:

  Packages server certificate verification failed.

What is going on here? (error message indicate failure to verify the certificate but does not give sufficient info on which part fails. It could certificate expired, no certificate, wrong signature, etc)

The https needs to verify the public key, but the CA is not known by the browser and the root CA

8 Certificate Revocation

Non-expired certificates can be revoked for different reasons:

1. Private key was compromise
2. Issuing CA was compromise
3. Entity left an organization
4. Business entity closed

A verifier needs to check if a certificate in question is still valid, although the certificate is not expired yet

8.1 Two way for certificate revocation

1. Certificate revocation list(CRL): CA periodically signs and publishes a revocation list
   This is favored in Firefox 28 and Mozilla
2. Online certificate status protocol(OCSP):OCSP Responder validates a cert in question
   OCSP problems:
   (a) Privacy: OCSP responder knows which certificate you are validating
   (b) Soft-fail validationL some browsers proceed in the event of no reply to an OCSP request(no reply is a ”good” reply)

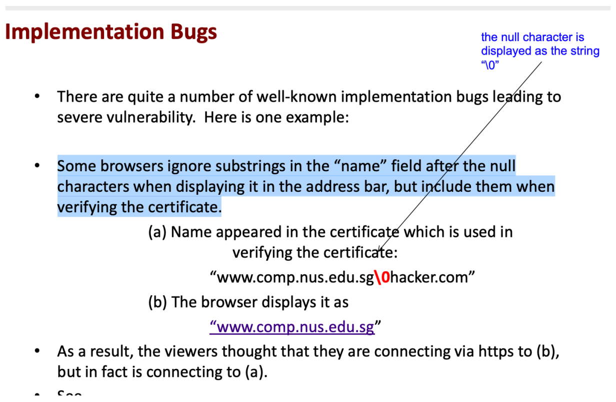
   Solution to the shotcome of OCSP:
   (a) OCSP stapling: allows certificate to be accompanied or ”stapled” by a (time-stamped) OCSP response signed by CA
   (b) Part of TLS handshake: client do not need to contact CA or OCSP responder

   The above solution however increase network cost

9 Limitation/attacks on PKI

9.1 Implementation Bugs

e.g.


Implementation Bugs

- There are quite a number of well-known implementation bugs leading to severe vulnerability. Here is one example:
- Some browsers ignore substrings in the "name" field after the null characters when displaying it in the address bar, but include them when verifying the certificate.

  (a) Name appeared in the certificate which is used in verifying the certificate:

  "www.comp.nus.edu.sg\0hacker.com"

  (b) The browser displays it as

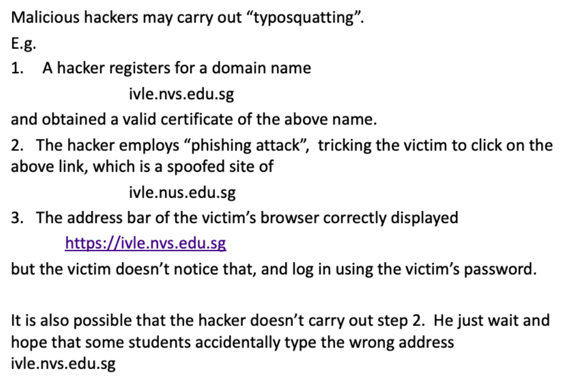
  "www.comp.nus.edu.sg"
- As a result, the viewers thought that they are connecting via https to (b), but in fact is connecting to (a).

The Common Vulnerabilities and Exposures (CVE) system provides a reference-method for publicly known information-security vulnerabilities and exposures.

9.2 Abuse by CA

There are so many CA' s. One of them could be malicious. A rogue CA can practically forge any certificate. Here is a well- known incident.

9.3 Social Engineering


Malicious hackers may carry out "typosquatting".

E.g.

1. A hacker registers for a domain name

   ivle.nvs.edu.sg

and obtained a valid certificate of the above name.

2. The hacker employs "phishing attack", tricking the victim to click on the above link, which is a spoofed site of

   ivle.nus.edu.sg

3. The address bar of the victim's browser correctly displayed

   https://ivle.nvs.edu.sg

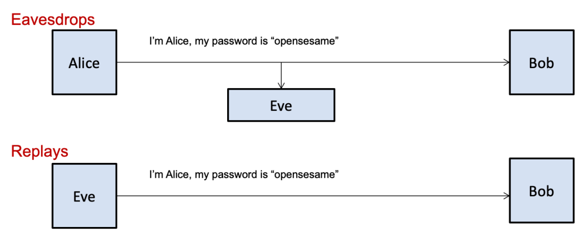
but the victim doesn't notice that, and log in using the victim's password.

It is also possible that the hacker doesn't carry out step 2. He just wait and hope that some students accidentally type the wrong address ivle.nvs.edu.sg

10 Strong authentication

10.1 weak authentication

Password is sent in clear, an eavesdropper can get the password and replay it


Eavesdrops
Alice
I'm Alice, my password is "opensesame"
Bob
Eve
Replays
Eve
I'm Alice, my password is "opensesame"
Bob

10.2 Strong authentication: Challenge-response

Suppose Alice and Bob have a shared secret k, and both have agreed on a message authentication code

1. Alice sends to Bob a hello message: ”hi, I am Alice”
2. (Challenge)Bob **randomly** picks a plaintext m and sends m to Alice
3. (Response) Alice computes t= mac_k(m). Alice sends t to Bob
4. Bob verifies that the tag received is indeed the mac of m. If so, accpets, otherwise rejects(Only the entity who knows k can produce the mac, and hence must be Alice or Bob)

By property of mac, even if Eve can sniff the communication between Alice and Bob, Eve still can’t get the secret key k, and can’t forge the mac for messages that Eve has not seen before.(Confidentiality)
Eve also can’t replay the response. This is because the challenge is **randomly** chosen and likely to be different in the next authentication session. The challenge m ensures freshness of the authentication process.

10.2.1 unilateral authentication and mutual authentication

This protocol only authenticates Alice. That is, authenticity of Alice is verified. Hence it is call unilateral authentication. There are also protocols to verify both parties, which are called mutual authentication.

10.2.2 Repair attack

Bob **randomly** picks a plaintext, if not random, the attacker may see the plaintext is given before and he can know the cyphertext

10.3 Unilateral authentication using PKC

1. (Challenge) Alice chooses a random number r and send to Bob:”Bob, here is your challenge”,r$(Response)Bobuseshisprivatekeytosignr.Bobalsoattacheshiscertificatesign(r),Bob'scertificate$

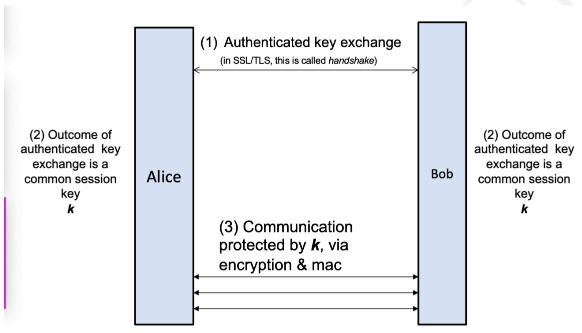
2. Alice verifies Bob's certificate, extracts Bob's public key from the certificate, and verifies that the signature is correct

10.3.1 Analysis

1. Confidentiality: by property of signature, the eavasdropper can't drive Bob's private key and replay the response. If Alice already knwos Bob;s public key, the certificate can be omitted
2. The value r is also known as the cryptographic nonce(or simply nonce)

10.3.2 Key-exchange and authenticated key-exchange

After the authentication is made, attack may also comes out and play the role in one side. Though the current authentication protocol assumes that the adversary is unable to interrupt after the authentication is made. If the attacker is able to interrupt, as a defence method, the authentication process should establish new shared secret k(**session key**) and the subsequent phase will be protected using k(symmetric)
The process establishing a secret(with or without authentication) between Alice and Bob is called **key-exchange or key-agreement**
If the process is incorporated with authentication, then it is called Authenticated key-exchange. A well-known scheme is called station-to-station protocol


(1) Authenticated key exchange
(in SSL/TLS, this is called *handshake*)
(2) Outcome of authenticated key exchange is a common session key *k*
Alice
Bob
(2) Outcome of authenticated key exchange is a common session key *k*
(3) Communication protected by *k*, via encryption & mac

11 key exchange

Request: even the eavesdropper can sniff the channel between the two sides, he cannot extract any information of the established key
Two common method:

1. PKC

(a) Authentication has been made

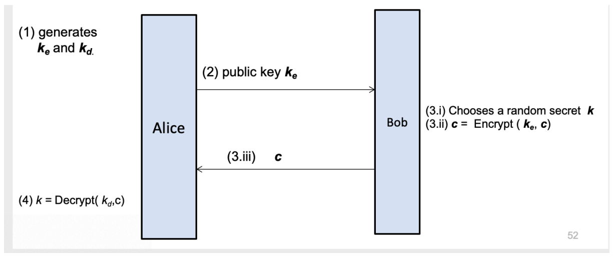
(b) Alice generates a pair of private/public key

(c) Alice sends the public key to Bob

(d) Bob select a secret k, and encrypt k with the public key sent by Alice into E_k

(e) Bob send the encrypted E_k to Alice

(f) Alice decrypts the E_k with private key and get the k


(1) generates k_e and k_d.
Alice
(2) public key k_e
Bob
(3.i) Chooses a random secret k
(3.ii) c = Encrypt (k_e, c)
(3.iii) c
(4) k = Decrypt(k_d,c)
52

2. Diffie-Hellman key exchange(can achieves additional security)

(a) We assume Alice and Bob have agreed on two public parameters: a generator g and a large prime(e.g. 1000 bits) prime p. Here both g and q is known to public

(b) Alice randomly chooses number:a and compute x=g^a mod p

(c) Bob randomly choose a number:b and compute y=g^b mod p

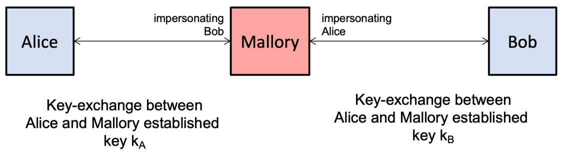
(d) Alice send the x to Bob and Bob send the y to Alice

(e) Alice compute k=y^a mod p

(f) Bob compute k=x^b mod p

The eavesdropper cannot get any information of k from the x and y

11.1 Vulnerability of PKI


Alice
impersonating Bob
Mallory
impersonating Alice
Bob
Key-exchange between Alice and Mallory established key k_A
Key-exchange between Alice and Mallory established key k_B

The Mallory can personate Bob and send his public key k_A to Alice; send decrypt using his private key, and encrypt the key with Bob's public key, hence the Mallory can know the key

11.2 Expired time of the session key

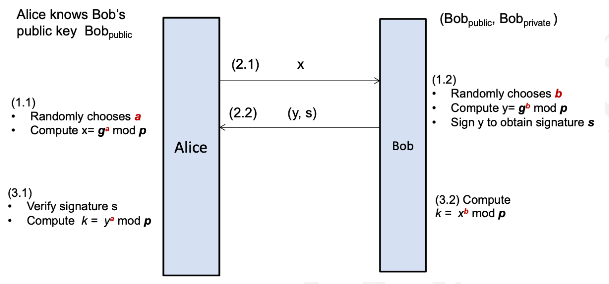
Even if the key should is protected by authentication and encryption, it needs a expired time, so that if one party's system is hacked or the session key is leaked in some way, re-authentication is needed and the damage is limited

12 Authenticated key-exchange

1. public key version
   (a) mutual: need to know each other public key (this can be achieved using PKI and certificate)
   (b) unilateral authentication, only one party need to have public key

   After the protocol has completed, a common key (i.e the Session key) is established.

2. DH key exchange with PKC this special case of authenticated key-exchange is also known as the Station-To-Station Protocol (STS).
   The unilateral authentication key exchange


Alice knows Bob's public key Bob_{public}
(1.1)
Randomly chooses a
Compute $x = g^a \bmod p$
(3.1)
Verify signature s
Compute $k = y^a \bmod p$
Alice
(2.1) x
(2.2) (y, s)
Bob
$(Bob_{public}, Bob_{private})$
(1.2)
Randomly chooses b
Compute $y = g^b \bmod p$
Sign y to obtain signature s
(3.2) Compute $k = x^b \bmod p$

   (a) Alice and Bob have agreed on two public parameters, a generator g and a larger(more than 1000 bits) prime p. Both g and p are known to public.
   (b) Alice randomly chooses number a and compute x=g^a mod p
   (c) Bob randomly chooses b and compute y=g^b mod p; sign y to obtain signature s
   (d) Alice sent x to Bob and Bob send(y,s) to Alice
   (e) Alice verify signature s and compute k=y^a mod p
   (f) Bob compute k=x^b mod p

   The mutual authentication key exchange

(a) Alice and Bob share a number g and a large prime number(more than 1000 bits) p

(b) Alice has a pair of $k_{privateA}$ and $k_{publicA}$

(c) Bob has a pair of $k_{privateB}$ and $k_{publicB}$

(d) Alice chooses a number a, and sign a

(e) Alice compute x=g^a mod p

(f) Alice sign x with her $k_{privateA}$ to sA and send (x,sA, certificateA) to Bob

(g) Bob receives x and verify it with sA and certificate

(h) Bob select number b, and compute y=g^b mod p

(i) Bob sign y with his own $k_privateB$ to sB

(j) Bob send(y, sB, certificateB) to Alice

(k) Alice verify y with sB and certificate B

(l) Alice compute secrete=y^a mod p

(m) Bob compute secrete=x^b mod p

Analysis

(a) Authentity: Alice is assured that she is communicating with an entity who knows $B_{private}$

(b) Authenticity: Bob is assured that he is communicating with an enitity who knows $A_{private}$

(c) confidentiality: Attacker unable to get the session key

12.0.1 Difference between mutual authentication and unilateral authentication

In the unilateral example, only one party get authenticated while the other side does not care about the identity who it is communicating with(server-client relationship)

12.1 Password authenticated key exchange

Authenticated key exchange can be used in symmetric key setting. Where the eavesdropper and man-in-middle cannot brute-force the password without further interuction with two parties; where strong authentication can be made even with short password(Important: simply mac cannot achieve this)
Example:

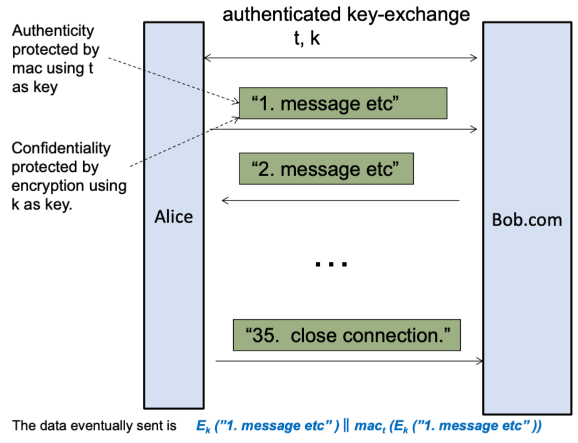
(a) Encrypted Key Exchange(EKE)

(b) PEAP(Protected Extensible Authentication protocol): secure against offline dictionary attacks.

(c) LEAD(Light weight extensive authentication protocol): which is vulnerable to offline dictionary attacks

13 TLS

https uses TLS



authenticated key-exchange
t, k
Authenticity protected by mac using t as key
Confidentiality protected by encryption using k as key.
Alice
Bob.com
"1. message etc"
"2. message etc"
...
"35. close connection."
The data eventually sent is E_k("1. message etc") ‖ mac_t (E_k ("1. message etc"))

The Alice and Bob handshake/authenticated key exchange and get key pair (k,t)
The data eventually sent after handshake is $E_k(''1.messageetc'')||mac_t(E_k(''1.messageetc''))$
(The above is called Encrypt-and-MAC)
the $E_k(''1.messageetc'')$ ensure confidentiality
the $mac_t(E_k(''1.messageetc''))$ ensure integrity
IMPORTANT: this is **encrypt and mac**
In AES, this is called authenticated encryption
There is a sequence number: i.e.1,2,3,4,5,6,7, without which some of the pkt may be dropped by accident or maliciously

13.1 MAC-then-encrypt, encrypt-then-MAC, Encrypt-and-MAC and Authenticated encryption

13.1.1 Encrypt-then-MAC

Encrypte the plaintext, then compute the MAC on the ciphertecxt, and append it to the ciphertext(In that case, we do not forget to include the initialization vector(IV) and the encryption method identifier into MACed data)
Characters:

1. Provides integrity of cipher text. Assume the MAC shared secret has not been compromosed, we ought to be able to deduce whether a given ciphertext is indeed authentic or has been forged(NOTE: encrypt-then-hash is insecure especially when the cipher scheme is malleable)
2. Plaintext integrity
3. even if the encryption method is malleable, we should not worries because the mac can filter out this invalid ciphertext
4. The MAC does not provide any information on theplaintext since, assuming the output of the cipher appears random, so does the MAC. In other words, nothing about the plaintext can be found from the MAC

13.1.2 MAC-then-Encrypt

Compute the MAC on the cleartext, append it to the data, and then encrypt the whole(What the TLS does)
reference: https://tools.ietf.org/html/rfc7366

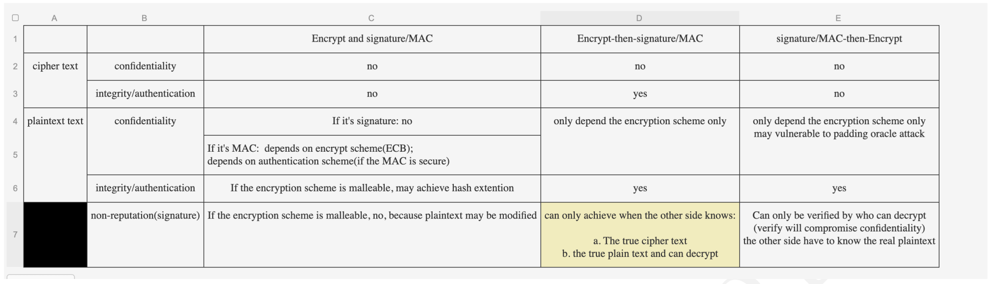
1. Does not provide any integrity on the ciphertext, since we have no way of knowing until we decrypt the message whether it was indeed authentic or spoofed
2. Plaintext integrity, here the mac cannot provide any information of the plaintext either because the plaintext is encrypted
3. If the cipher scheme is malleable, it may be possible to alter the message to appear valid and have a valid MAC. This is a theoretical point. In reality, the MAC serete should provide protection.

13.1.3 Encrypt-and-MAC

Compute the mac on the plaintext, encrypt the plaintext and then append the mac at the end of the ciphertext(What SSH does)

1. No integrity on the ciphertext, becasue the mac is taken against the plaintext. This opens the door to some chosen cipher text attacks on the ciphertext attacks on the cipher
2. The integrity of the plaintext can be verified the cipher scheme is malleable, the contents of the ciphertext could well be altered, but on decryption, we can find the plaintext is invalid.
3. may reveal information about the plaintext in the mac. This occurs if the plaintext msg are repeated and nouce is not applied

Reference: https://crypto.stackexchange.com/questions/202/should-we-mac-then-encrypt-or-encrypt-then-mac



		Encrypt and signature/MAC	Encrypt-then-signature/MAC	signature/MAC-then-Encrypt
cipher text	confidentiality	no	no	no
	integrity/authentication	no	yes	no
plaintext text	confidentiality	If it's signature: no If it's MAC: depends on encrypt scheme(ECB); depends on authentication scheme(if the MAC is secure)	only depend the encryption scheme only	only depend the encryption scheme only may vulnerable to padding oracle attack
	integrity/authentication	If the encryption scheme is malleable, may achieve hash extention	yes	yes
	non-reputation(signature)	If the encryption scheme is malleable, no, because plaintext may be modified	can only achieve when the other side knows: a. The true cipher text b. the true plain text and can decrypt	Can only be verified by who can decrypt (verify will compromise confidentiality) the other side have to know the real plaintext

13.2 Authenticated encryption

Authenticated encryption (AE) and authenticated encryption with associated data (AEAD) are forms of encryption which simultaneously assure the confidentiality and authenticity of data.

1. The encryption scheme is semantically secure under a chosen plaintext attack.
2. The MAC function is unforgeable under a chosen message attack.
3. can provide security against chosen ciphertext attack

13.3 SSL and TLS

SSL and Transport layer security(TLS) are protocols that secure communication using cryptographic mean
SSL is the predecessor of TLS, https is built on top of TLS

14 Forward secrecy

If the attacker can hack into one system and get the private key, if the public key version authenticated key-exchange is used, all the history message can be decrypted

But if DH key exchange is applied, because the private key is only used to sign, the attacker still cannot decypt.