CS2107 Cheatsheet AY22/23

Security Requirements

A system can fail due to various reasons:

- Operator mistakes
- Hardware failures
- Poor implementation
- Deliberate human actions desgined to cause failure

Cyber security is concerned with **intentional failures**:

- **Assets:** Hardware, Software, Data and Information, Reputation → which is intangible
- **Threat:** Set of circumstances that has the potential to cause loss or harm
- **Vulnerability:** Weakness in the system → in procedures / design / implementation that might be exploited to cause loss/harm
- **Control:** Control/countermeasure/security mechanism meant to counter threats. It is an action/device/producedure or technique that removes/reduces a vulnerability

There is a **threat agent** that gives rise to a **threat** that exploits a **vulnerability** that leads to a **risk** that can damage an **asset** and cause an **exposure**, all of which can be counter measured by a **safeguard** that directly affects the **threat agent**.

C-I-A Triad

We can describe a class of attacks by giving the attacker's goals and also the attacker's resources. This is also known as the attack/adversary/security model.

- Confidentiality: Prevention of unauthorized disclosure of information.
- Integrity: Prevention of unauthorized modification of information/process.
- Availability: Prevention of unauthorized withholding of information/resources.

There are 2 more properties added by ISO 7498-2 [ISO89] that are desirable, particularly in communication networks.

- **Authenticity/Authentication**: Ability of a system to confirm the identity of a sender.
- **Non-repudiation/Accountability**: Ability of a system to confirm that a sender cannot convincingly deny having sent something.

Security, Functionality and Ease-of-Use

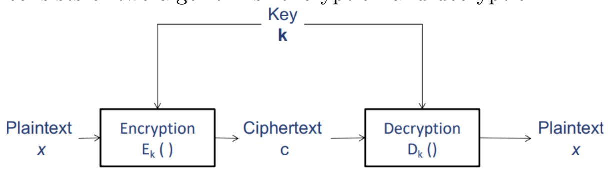
The more secure something is, the less usable and functional it becomes.

- **(ease-of-use)** → Security mechanisms intefere with working patterns users are familiar with.
- **(performance)** → Security mechanisms consumes more computing resources
- **(cost)** → Security mechanisms are expensive to develop

Encryption

Symmetric Key Encryption

A symmetric-key encryption scheme (also known as cipher) consists of two algorithms: encryption and decryption.



There are 2 requirements:

- **Correctness:** For any plaintext: x, key: k, $D_k(E_k(x))$ = x. It also must be secure. Formal formulation of security is difficult.
- **Security:** Given a ciphertext, it should be computationally difficult to derive useful information about the key k and plaintext x. The ciphertext should resemble a random sequence of bytes.

Attack Models: Adversary's Capabilites

If attacker wants to find the key → ***total break***. However attacker may be satisfy a ***partial break***.

- **Ciphertext only attack:** adversary is given a collection of a ciphertext c, they may know some properties of the plaintext (e.g. plaintext is an english sentence).
- **Known plaintext attack:** adversary is given a collection of plaintext m and corresponding ciphertext c, can be used to guess the key and further decrypt other ciphertexts.
- **Chosen plaintext attack (CPA):** adversary has access to a blackbox (an *oracle*). They may choose and feed any plaintext m to the blackbox and obtain the corresponding ciphertext a reasonably large number of times c (all encrypted with the same key).
- **Chosen cipher attack CCA2:** same as chosen plaintext attack, but the adversary chooses the ciphertext and the blackbox outputs the plaintext.

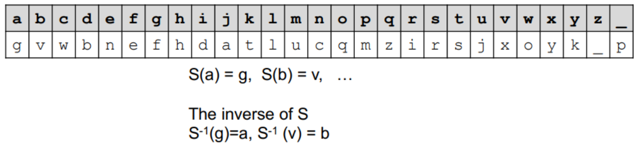
Indistinguishability (IND) The attacker may satisfy with distinguishability of ciphertext: with some "non-negligible" probability more than $\frac{1}{2}$, the attacker is able to distinguish the ciphertexts of a given plaintext (say, "Y") from the ciphertext of another given plaintext (say, "N").

Classical Ciphers

Substitution Cipher

Plaintext and ciphertext:

- a string over a set of symbols U.
- Let U= { "a", "b", "c" . . . , "z", "_" }
- Plaintext may be "hello_world"



a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	_
g	v	w	b	n	e	f	h	d	a	t	l	u	c	q	m	z	i	r	s	j	x	o	y	k	_	p

S(a) = g, S(b) = v, ...

The inverse of S
$S^{-1}(g)=a$, $S^{-1}(v) = b$

The key space is the set of all possible keys. The key space size or size of key space is the total number of possible keys. The key size or key length is the number of bits required to represent a key. Here, the key space size is (27!) key size is approximately 94 bits.

Attacking Substitution Cipher

The goal for attackers is to figure out the key. If the key can be found, then the plaintext can be obtained. The converse is also true. If the attacker can get information on the plaintext or even the complete plaintext, they can also easily obtain the key. Hence there are 2 levels of access to information:

- **ciphertext only** – large number of ciphertexts all encrypted using the same key
- **plaintext** pairs of ciphertext and the corresponding plaintext

Known Plaintext Attack: Substitution Cipher:
This attack occurs when the attacker has access to pairs of ciphertexts and their corresponding plaintexts. Attacker can figure out the entries in the key that are used in both ciphertext and plaintext. With a sufficiently long ciphertext, the entire key can be determined. If the adversary can derive the key, we call the scheme **"insecure under known plaintext attack"** or **"broken undder known plaintext attack"**.

First Few Bytes
Not unreasonable for attacker to obtain at least one pair of ciphertext and plaintext as only a small number of bytes is required. Some of the following can be guessed directly:

- Email data: e.g. certain headers or words are fixed ("From", "Subject" . . .)
- Network Protocols: similar to email, as they have fixed headers.

Exhaustive Search – Brute Force

- Assume attacker knows a ciphertext C and the corresponding plaintext X.
- For S in all possible substition tables:
  Compute X' = D_S(C), if X' == X: break
- Since a key in the above table can be represented by a sequence of 27 symbols, size of the key space is 27!

Frequency Analysis

Suppose the plaintexts are English sentences. **Letter frequency distribution** in english is non-uniform. The adversary is able to guess (given a sufficiently long plaintext) by:

- Mapping the frequently-occuring letters in the ciphertext to the frequently-occuring letters of english.
- Carry out frequency analysis.

⋆ Substitution cipher is **not secure under ciphertext-only attack** either.

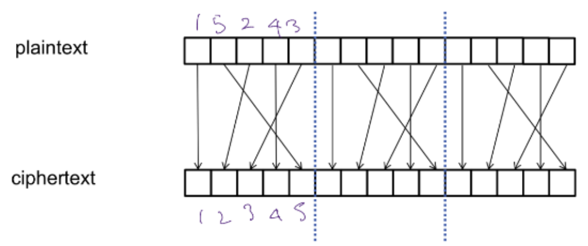
Permutation Cipher – transposition cipher

The encryption first groups the plaintext into blocks of t characters, and the applied a secret "permutation" to each block by shuffling the characters. The key is the secret "permutation", which is an 1-1 onto function e from 1,2,..,t to 1,2,...,t. The size t could be part of the key, that is, t is also kept secret. We can write the permutation p as a sequence

$$\underline{p} = (p_1, p_2, p_3, \dots, p_t) \quad (1)$$

which shift the character at position i to the position p_i

Given the plaintext and the key ***t***=5, ***p***=(1,5,2,4,3):



⋆ Permutation cipher fails miserably under known-plaintext attack. Given a plaintext and a ciphertext, it is very easy to determine the secret key. Permutation cipher is also easily broken under ciphertext only attack if the plaintext is English text

One-Time Pad

Given an *n-bit* plaintext $(x_1\ x_2 \dots x_n)$ and an *n-bit* key $(k_1\ k_2 \dots k_n)$ we can output the ciphertext, C:

$$C = (x_1 \oplus k_1)(x_2 \oplus k_2)\dots(x_n \oplus k_n) \quad (2)$$

The condition is that the key and the plaintext must be of the same length. We can then decrypt the ciphertext by XORing the cipher with the key once more to get the plaintext X:

$$X = (c_1 \oplus k_1)(c_2 \oplus k_2)\dots(c_n \oplus k_n) \quad (3)$$

For this to work, we need to be able to transfer the key securely!

What is $\oplus$ or XOR?

- Commutative: $A \oplus B = B \oplus A$
- Associative: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$
- Identity element: $A \oplus 0 = A$, where 0 is the *identity element*
- Self-inverse: $A \oplus A = 0$

⋆ From a pair of ciphertext + plaintext, attack can derive the key, but since it is only used one time, this key is useless. One-time pad leaks no information of the plaintext, even if the attacker has arbitrary running time.

Perfect Secrecy

Attacker's prior knowledge of the unknown plaintext will not change even after they have seen the ciphertext y. Meaning the chances that the attacker correctly predicts x before knowing y, and after knowing y, are the same.
Definition: A crypto system has ***perfect secreccy*** if for any distribution of X, for all x, y:

$$\boxed{\Pr(X = x | Y = y) = \Pr(X = x)}$$

Modern Ciphers

Designs of modern ciphers take into consideration of known-plaintext attack, frequency analysis and other known attacks

- DES (Data Encryption Standard, 1977) → Key length too short
- RC4 (Rivest's Cipher 4, 1987) → Broken in some adoptions
- A5/1 (used in GSM, 1987)
- AES (Advanced Encryption Standard, 2001)

Exhaustive Search, Key Length and Work Factor

If the key length is 32 bits, there are 2^{32} possible keys. Hence, the exhaustive search needs to "loop" for 2^{32} times in the worst case.

We can quantify the security of an encryption scheme by the length of the key. Consider a scheme **A** with 64-bit keys and a scheme **B** with 54-bit keys. Scheme **A** is more secure w.r.t. exhaustive search.

DES - Data Encryption Standard

Key length of DES is 56 bits (intentionally short key size). While exhaustive search on 56 bits seemed infeasible in the 70s, very soon, it is possible using distributed computing or a specialized chip.

AES - Advanced Encryption Scheme

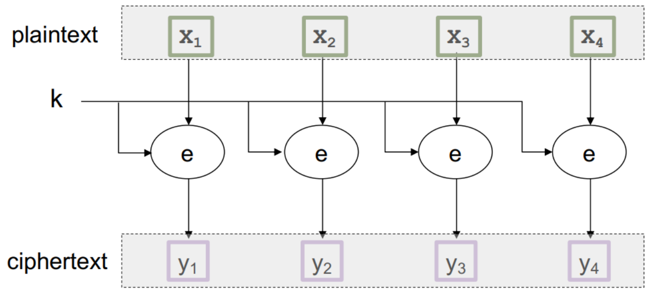
AES has a block length of 128, and key length can be 128, 192, or 256 bits. It is a symmetric block cipher developed comprising of the following: Byte Sub, Shift Row, Mix Columns, Add Round key all repeated n times.

Block cipher & Mode-of-Operations

For a large plaintext, it is divided into blocks (of equal size) before the block cipher is applied. DES and AES are also known as "Block Cipher". Block cipher are designed for some fixed size input/output.

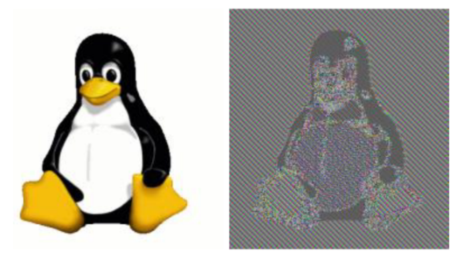
ECB Mode - Electronic Code Book

ECB divides the plaintext into blocks and then applies block cipher to each block, all with the same key.
→ BUT, ECB leaks information!!!



⋆ An encryption scheme is "deterministic" in the sense that, the encryption algorithm will always produce the same output (i.e the ciphertext) when given the same input (i.e. the key and plaintext).
⋆ In contrast, a "probabilistic" encryption scheme produces different ciphertext even with the same input (key, plaintext).

Image below is divided into blocks, and encrypted with some deterministic encryption scheme* using the same key. Since it is deterministic, any two blocks that are the same (for e.g. blocks in the white background) will be encrypted to the same ciphertext.

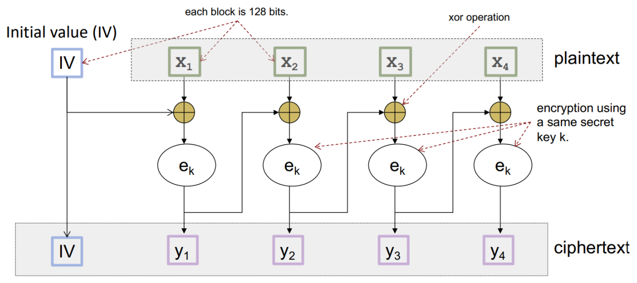


To prevent leakages of information, there are the following solutions:

- Use a separate IV for each block (CBC mode) → less adopted as final ciphertext would be twice that of plaintext.
- Linking of 2 blocks such that upon the encryption of one, we would use that to XOR for the next block.
- CTR (Counter) mode, used as a stream cipher.

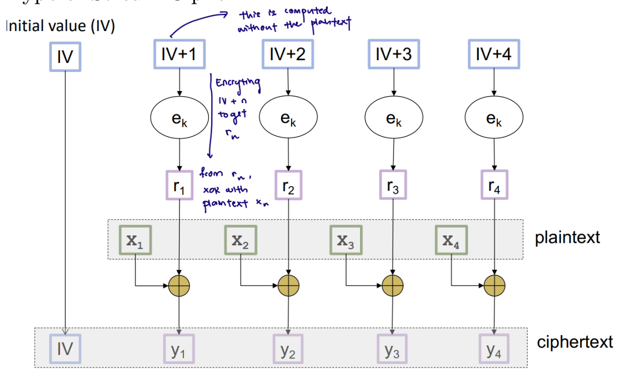
CBC - Cipher Block Chaining on AES (mode-of-op)

The Initial Value (IV) is an arbitrary value chosen during encryption. (different in different encryptions of the same plaintext) → $\boxed{y_0 = IV,\ y_i = E_k(x_i \oplus y_{i-1}) \text{ for } i > 0}$



CTR - Counter Mode on AES (mode-of-op)

Type of Stream Cipher.

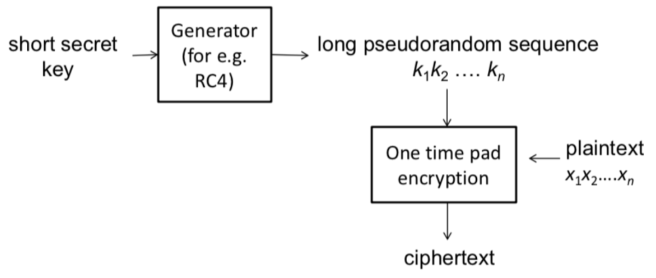

Initial value (IV)
this is computed without the plaintext
IV
IV+1
IV+2
IV+3
IV+4
Encryting IV + n to get rn
ek
ek
ek
ek
from rn, xor with plaintext xn
r1
r2
r3
r4
x1
x2
x3
x4
plaintext
IV
y1
y2
y3
y4
ciphertext

GCM mode - Galois/Counter (mode-of-op)

It is an authenticated-encryption, ciphertext consists of an extra tag for authentication. Secure in presence of decryption oracle.

Stream Ciphers

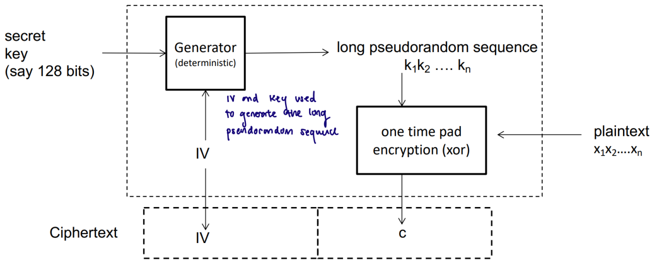
Inspired by one-time-pad. Suppose the plaintext is 2^{20} bits, but the secret key is only 256 bits. Stream cipher generates a 2^{20}-bit sequence from the key and treats the generated sequence as the secret key in one-time-pad. The generator must be carefully designed, so that it gives **cryptographically secure pseudorandom sequence**.



Initialization Vector (IV)

Most ciphers have an IV, which can be randomly chosen or from a counter.

- **Encryption:** Long pseudorandom sequence generated from: secret key + IV. Final ciphertext contains the IV followed by the output of the one-time-pad encryption
- **Decryption:** IV extracted from ciphertext. Plaintext can be obtained by: getting the same pseudorandom sequence generated from key + IV.

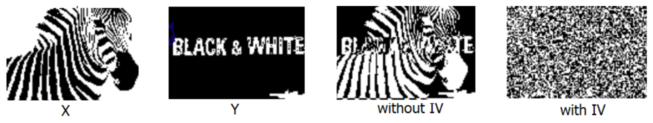

secret key (say 128 bits)
Generator (deterministic)
long pseudorandom sequence
$k_1k_2 \dots k_n$
IV and key used to generate the long pseudorandom sequence
IV
one time pad encryption (xor)
plaintext
$x_1x_2 \dots x_n$
Ciphertext
IV
c

Why must we use IV for Stream Ciphers?

Without the IV, if an attacker can obtain two ciphertexts, they can simply XOR the 2 ciphertexts together. Let's say we have ciphertext *U* & *V*, which are obtained from plaintexts X and Y respectively without using IV.

$$\boxed{U \oplus V = (X \oplus K) \oplus (Y \oplus K) = (X \oplus Y) \oplus (K \oplus K) = \underline{(X \oplus Y)}}$$

The above is obtained after doing some manipulation as a result of the associative and commutative properties of XOR = X⊕Y. Let us take X and Y as black and white images, where every pixel corresponds to a bit.



They look different because K_1 and K_2 are different due to it being produced by different IVs, thus there is no cancelling out when XOR both images. IV makes an encryption probabilistic

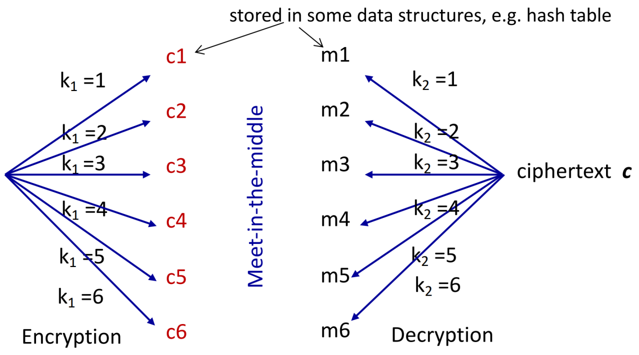
Triple DES

DES not secure w.r.t. today computing power. We can improve it by encrypting multiple times → using different keys. DES doesn't form a group in the sense that, $E_{k1}(E_{k2}(x))$ is not the same as $E_{k3}(x)$ for some *k3*.

Meet-in-the-middle Attack

Known plaintext attack, where the attacker's goal is to find the keys *k1* and *k2*.

- Compute two sets of $\boldsymbol{C}$ and $\boldsymbol{M}$. $\boldsymbol{C}$ contains ciphertexts of $\boldsymbol{m}$ encrypted with all possible keys. $\boldsymbol{M}$ contains plaintexts of $\boldsymbol{c}$ decrypted with all possible keys.
- Find common element $\boldsymbol{C}$ and $\boldsymbol{M}$, thereby obtaining both keys.
- In general, for *k-bit* keys, it reduces the number of crypto operations to 2^{k+1} using approx 2^{k+1} units of storage space.



⋆ Remedy – Use Triple Encryption, but with 2 keys. Both below believed to have the same level of security:

- $E_{k1}(E_{k2}(E_{k1}(x)))$
- OR $E_{k2}(E_{k1}(E_{k2}(x)))$

Padding Oracle Attack

Attacker has a ciphertext including (IV, c) and access to a padding oracle. Attacker's goal is to get the plaintext of (IV, c) The cipher text is encrypted with a secret key *k* and the oracle knows *k*.

Padding - PKCS#7

Suppose a block size is 8 bytes, and the last block has 5 bytes only (thus 3 extra bytes required), the padding will be done as follows (if the last block is full, an extra block of all zeroes are added.):

DD DD DD DD DD DD DD DD	DD DD DD DD DD 03 03 03

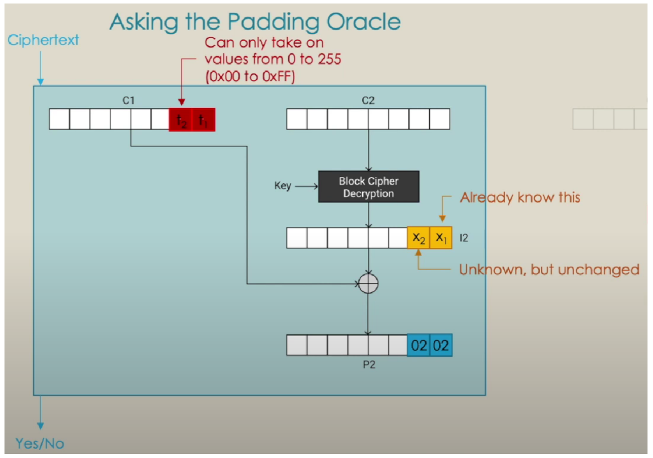
Padding oracle attack takes advantage of the deterministic natrue of a Block Cipher, if the plaintext is unchanged, the resultant ciphertext (and vice versa) would also remain the same. Padding oracle attack also takes advantage of XOR operations:

$$\boxed{\text{for any A, B, C if } A \oplus B = C,\ B \oplus C = A, \text{ then } A \oplus C = B.}$$

Padding Oracle Attack - Example (on CBC)

Suppose the attacker wants to find out what is t_2.

- First, force the plaintext to have a padding of 2 (02 02) – allow reverse engineering of the XOR operation to find out the original plaintext in that byte position
- Next, use exhaustive search to find out which values of t_2 from the previous block together with the ciphertext from block C2, when XORed, will produce a "YES" from the padding oracle.
  - If "NO", repeat exhaustive search.
  - If "YES", this means that the padding is correct, and that means this value of t_2 is the correct value for that byte.
- Finally, we XOR the value t_2 with that of the ciphertext from P2 to obtain the intemediary x_2.
- From the intemediary, we can use the unchanged key we already know to get the original plaintext.



Cryptography Pitfalls

Wrong choices of IV

Some applications overlooked IV generation. As a result, under some situations, the same IV is reused.

- e.g. To encrypt a file F, the IV is derived from the filename. It is quite common to have files with the same filename → Schneier on Security, Microsoft RC4 Flaw:
  - The stream cipher RC4 with key length up to 128 bits is used in Microsoft Word and Excel to protect documentss. But when an encrypted document gets modified and saved, the initialization vector remains the same, thus the same keystream generated from RC4 is applied to encrypt different versions of that document.
- When using AES under the "CBC mode", the IV should be unpredictable in order to prevent a certain type of attack. (So, it is vulnerable to choose IV as 1,2,3,....)
  - The Browser Exploit Against SSL/TLS Attack (BEAST) exploits this, as when encrypting multiple packets, the IV of a packet would be the last ciphertext block of the previous packet, which is visible to anyone.

Reusing One-Time-Pad Key

Similar to reusing IV, if the one-time-pad key is reused and attacker notices it, it is possible to decrypt entire messages. Project Venona saw the decryption of encrypted messages from the Soviet Union by the US.

Predictable Secret Key Generation

When progamming, we may use packages such as `java.util.Random` despite the fact that it is not actually secure. `java.security.SecureRandom` is preffered because:

- **Size:** `Random` class only has 48 bits, whereas `SecureRandom`, can have up to 128 bits, smaller chance of repeated `SecureRandom`.
- **Seed Generation:** `Random` uses system clock to generated the seed, attacker can easily replicate if they know when the seed was produced. `SecureRandom` takes random data from OS as most OSes collect and store these data in files (interval between keystrokes etc.) and use that as the seed
- **Breaking the code:** `Random` requries 2^{48} tries (practically possible with compute resources today), `SecureRandom` requires 2^{128} attemps, may take many years.
- **Generating Function:** Standard Oracle JDK 7 implementation uses a Linear Congruential Generator to produce random values in `java.util.Random`. `java.security/SecureRandom` implements SHA1PRNG algorithm, using SHA1 to generate pseudorandom numbers.
- **Security:** `java.util.Random` must not be used for either security-critical applications/protecting sensitive data.

Designing your own crypto/cipher

Don't design your own crypto, or even make slight modification to existing scheme... unless you have in-depth knowledge on the topic.

Kerckhoffs's principle

A system should be secure even if everything about the system, except the secret key, is public knowledge.

Arguments against obscurity

- RC4 introduced in 1987 with its algorithm as a trade secret. In 1994, a description of its algorithm was anonymously posted in a mailing group.
- MIFARE Classic is a contactless smartcard widely used in Europe. It uses a set of proprietary protocols/algorithms. However, they are reverse-engineered in 2007. It turns out that the encryption algorithms are already known to be weak (with 48-bit keys) and breakable.

Security through Obscurity

To hide the design of the system in order to achieve security.

- obscurity can be used as an addition layer in the **defense-in-depth** strategy.
- NOT advisable to reveal the computer network structure and settings (for example, location of firewall and the firewall rules), although these are not "secrets".
- Although it is advisable to make the algorithm public, it is not advisable to publish the actual program used in a smart-card. By publishing the program/code, advisory may able to identity implementation flaw that was previously unaware of, or carry out side-channel attacks.
- Usernames are not secret. However, it is not advisable to publish all the usernames

Authentication (Password)

Authentication: The process of assuring that the communicating entity, or origin of a piece of information, is the one that it claims to be.
There are 2 types of authentication:

1. **Entity authentication:**
   - For connection-oriented communication
   - Verifying authenticity of entities involved in a connection
   - Mechanisms: password, challenge and response, cryptographic protocol
2. **Data-origin authentication:**
   - For connectionless communication
   - Verifying the origin of a piece of information
   - Mechanisms: Crypto primitives such as MAC or digital signature.

Authenticity vs Integrity

Authentic is an adjective to say that the claimed enity/origin is assured by supporting evidence. Authenticity is the condition of being authentic.
Authenticity and integrity are thus related. In the context of an insecure channel, we can say that a message that has been modified in transit means that it no longer comes from its original source.
In other words, a message whose integrity is compromised also means that its authenticity is compromised. As such **data-origin authenticity implies data integrity**. But **data integrity does not imply data-origin authenticity**. Authenticity is thus a stronger requirement than integrity.

Password System

A password is part of an authentication system that usually consists of:

- Identity (Identification): need not be kept secret. It can be a username in a system, bank account number, customer ID etc.
- Password (Authentication): password is kept secret, only the authentic user & server knows it. The fact that an entity knows the password **implies that it is either the server or the authentic user.**

1. *Bootstrapping*
   - Password is to be established during Bootstrapping, done by:
     - Server/user chooses a password and sends it to the user/server through another communication channel.
     - Default password.
2. *Password Authentication*
   - Protocol:

     User → Server: My name is **Alice**
     Server → User: What is your password
     User → Server: **OpenSesame**
     Server verifies whether password is correct.
   - Authentication can also be carried out without interactions: User sends sms to server: UserId: **Alice@nus.edu.sg** Password: **OpenSesame**. Instructions: Unsubscribe from your mailing list. No more junk mail.
   - This protocol is **"weak authentication"**, subjected to replay attack (packet sniffing/eavesdropper in communcation channel).
3. *Password reset*
   - Only authorized entities can reset the password. How do we verify the entity is authentic?
   - Need to authenticate entity before allowing entity to change password

Attacks on the Password System

Attacker may intercept the password during bootstrapping. They can also target the use of default passwords (e.g. IP Security Cameras). **No individual passwords:** Cost increase for device manufacturer, print passwords on each equipment/manual/case.

Searching for the Password

Dictionary Attack

Ability to test → feed a guess into login screen:

- **Exhaustive search** test all combinations to guess passwords
- **Dictionary attack** tests passwords stored in a "dictionary" (contain words from English dictionary, known compromised passwords, most passwords are generated by Humans).
- Dictionary attack could also exhaustively test combinations of words in dictionary. Exhaustively try all possible capitalization of words ("a" substitute by "@").

Two scenarios in dictionary attacks:

- **Online dictionary** attack: an attacker must interact with the authentication system during the searching process. (attacker must be online)
- **Offline dictionary** attack: There are 2 phases
  1. Attacker obtains some information **D** about password (sniff hash from interactions, or steal password file).
  2. Next, attacker carries out the searches using **D** without interacting with the system.

Guessing the password from social information

Attacker gathers social information about user to guess password.

Stealing the password

Sniffing

Shoulder sniffing (look-over-the shoulder), or **Sniffing the communication**, not as common now, some older systems sent passwords over public network (not encrypted).

Viruses, Keylogger

Key-logger records keystrokes, sends information back to attacker.

- (software) Some computer viruses are designed as a **key-logger**
- (hardware) Hardware key-logger physically connected to device

Phishing

A type of **social engineering**. Victim tricked to voluntarily send the password to the attacker. Phishing attacks ask for passwords under some false pretense. Typically, it tricks the user to visit a website, which is spoofed login page.

Spear Phishing

Phishing can be targeted to a particular small group of users (for example, NUS Staff). These are known as *spear phishing* which is an example of *targeted attacks*.

Cache

Using shared workstation, information keyed in could be cached. Next user still has access to cache.

Insider attack

Malicious system admin steals password file. OR System admin's account compromised, lost of password file.

Preventive Measures

1. User Training:

Workshop, reminders. Embedded Phishing Exercise (Authorized entities send out "phishing" emails to employees).

2. Blacklisting

Repository site keeps list of phishing site, blacklisted by browser or firewall.

3. Using Strong Password

Truly random password, using automated password generator. High "Entropy" difficult to remember.

- User selection:
  - Mnemonic Method: Pbmbval!
  - Altered Passphrases: Dressed*2*tge*9z
  - Combining/Altering Word: B@nkC@mera
- Usability: Strong passwords are difficult to remember. Difficult to enter alphanumeric passwords on mobile devices.

3a. Password Ageing

Recommended for users to regularly change passwords.

Online vs Offline Attack

- **Online:** To check whether a password is correct, the attacker needs to communicate with a server not under his control. E.g. - Attacker obtained a list of 1000 valid nusnet id, to find the password for some of them. The attacker writes an automated script that attempt to login to Canvas using guessed passwords for each of these 1000 valid nusnet id.
- **Offline:** To check whether a password is correct, the attacker can execute some algorithm without connecting to a server. - Attacker has an AES encrypted pdf file. The key is derived from a password. The attacker wants to find the password. - In some password authentication protocols, a "hash" of the password is sent in clear. The attacker first obtained the hash by eavesdropping a valid login session. Next, the attacker went offline and searched for the password. WPA2 personal is vulnerable to this form of offline dictionary attack.

Enhancing Password System

- Online dictionary attacks more difficult, there may be intentional delay into login sessions or account locking after a few failed attempts.
- Offline dictionary attacks more difficult, KDF applied to password, forces intensive computation, but has delay during legitimate usage.

Additional Protection to password files

During authentication, password entered by entity is hashed, then compared with value stored in password file. To verify whether password **P** belongs to a user **U**:

1. Compute d = Hash(**P**)
2. If ¡**U**, d¿ is in the password file, accept, else reject.

We cannot have the same password being hashed to the same value for 2 different usernames. Allows attackers to obtain all un-hashed passwords of the same value simply by comparing hashed values. To prevent this, we add a **salt**: random string of characters to the front of the password before hashing. Salt randomly generated for all users, stored in password file.

Self-help Password Reset

1. Security Question

Security-Cost-Usability Tradeoff: viewed as a mechansim for **fallback authentication**

- Enhance usability: user can still login if password is lost
- Reduce cost: reduces oeprating cost of helpdesk
- Weak security: attackers have another mean to obtain access

2. Recovering email's account

1. User → System: User id is X, I want to reset
2. System → User: Send email to user, URL contains an OTP
3. User → System: OTP is OTP1, This is my new password
4. System checks if OTP correct, if correct, reset

ATM Skimmer

For authentication, there is a (1) Card and (2) Pin. Card contains magnetic strip, storing user account id, PIN is the password. AN ATM Skimmer steals victim's account id + password.

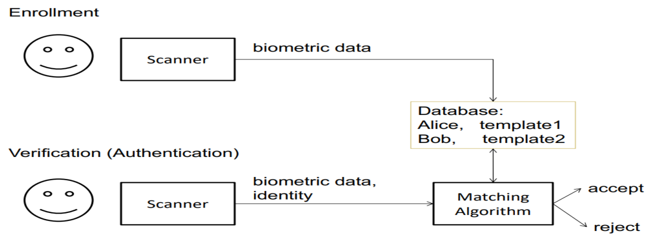
1. card-reader attached on top of existing ATM reader;
2. camera overlooking the keypad/spoofed key-pad on top of existing keypad;
3. means to record/transmit the information back to attacker.

Measures against ATM Skimmers

1. Anti-skimmer device: prevents external card reader attachment
2. Sheilding of keypad
3. Awareness among users

Biometric

- **enrollment**, a template of user's biometric data is captured and stored. (same as bootstrapping)
- **verification**, biometric data of person-in question captured and compared with template using matching algorithm (algo decides accept or reject).



There is inevitable noise in capturing biometric data, leading to error in making matching decision. **False Match Rate(FMR)** and **False Non-Match Rate (FNMR)**

$$FMR = \frac{number\ of\ successful\ false\ matches\ (B)}{number\ of\ attempted\ false\ matches\ (B+D)} \quad (7)$$

$$FNMR = \frac{number\ of\ rejected\ genuine\ matches\ (C)}{number\ of\ attempted\ genuine\ matches\ (A+C)} \quad (8)$$

	Accept	Reject
Genuine attempt	A	C
False attempt	B	D

- **Equal-Error Rate (EER):** FNMR = FMR
- **False-to-enroll Rate (FER):** Some user's biometric data cannot be captured e.g. due to injury.
- **Failure-to-capture (FTC):** User's biometric data may fail to be captured during transactions e.g. dirty fingers.

Attack of biometric system

Some biometric data can be easily spoofed. Other systems include **liveness detection** to verify if the entity being scanned is indeed live or instead of something spoofed.

n-Factor Authentication (2FA)

Require 2 different authentication factors:

1. Something you know: Password, PIN
2. Something you have: Security token, mobile phone, ATM Card
3. Who you are: Biometric

OTP Token

Hardware generates one time password, each token and the server share some secrets.

1. **Time-based:** Based on shared secret and current time interval, generate password ***K***.
2. **Sequence-based:** An event (user pressing button) triggers change of OTP.

Example: Password + SMS
Registration: User gives server mobile phone number and password.
Authentication:

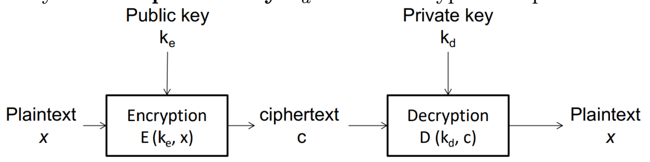
1. User sends password and username to server
2. Server verifies password is correct, then sends OTP to user through SMS
3. User receives SMS, enters OTP
4. Server verifies OTP is correct

- OTP generated from a factor.
- Password is long term and can be the same for a long duration. OTP is valid for a short period of time/for a specific transaction.
- If attack somehow managed to eavesdrop session and steal OTP with password, password can still be used as password + OTP insufficient for replay attacks of further sessions.
- OTP does not expose (non-password) factor during authentication. Even if session compromised, confidentiality still preserved.

Authenticity

Public Key Encryption

Public-key scheme uses different keys for encryption + decryption. Only with the **private key** $\mathbf{k}_d$ can we decrypt the ciphertext.

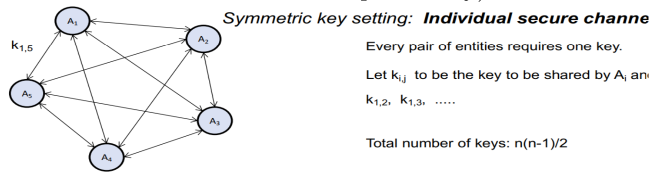


Security Requirements

Given the public key and ciphertext (not private key) it is difficult to determine the plaintext.

ADVANTAGES
In PKC, if there are multiple entities A_1, ... A_n each has their own <private key, public key>. Each entity only needs to broadcast their public key, keeping private *secret*.

If we do not use PKC, then any 2 entities must share a symmetric key via a secure channel, requiring both entities to know one another before actual communication. (PKC still needs a secure broadcast channel to distribute the **public key**).



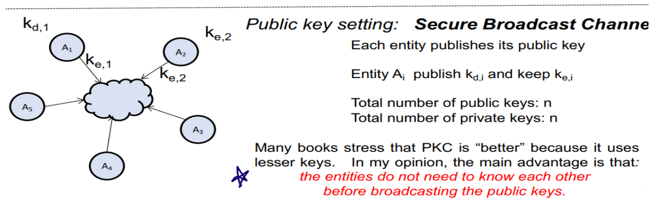
Symmetric key setting: ***Individual secure channe***

Every pair of entities requires one key.

Let $k_{i,j}$ to be the key to be shared by A_i an

$k_{1,2}$, $k_{1,3}$, ...

Total number of keys: n(n-1)/2



Public key setting: ***Secure Broadcast Channe***

Each entity publishes its public key

Entity A_i publish $k_{d,i}$ and keep $k_{e,i}$

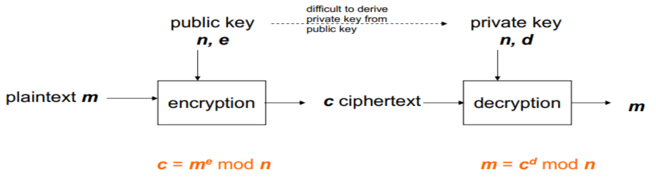
Total number of public keys: n
Total number of private keys: n

Many books stress that PKC is "better" because it uses lesser keys. In my opinion, the main advantage is that:
the entities do not need to know each other before broadcasting the public keys.

RSA – Rivest–Shamir–Adleman

1. Randomly choose 2 large primes **p** and **q**, computes **n = pq**
2. Randomly choose encryption exponent **e**:
$$gcd(e, (p-1)(q-1))$$
3. Find decryption exponent **d**, where
$$d\ e\ mod (p-1)(q-1) = 1$$
4. Published (**n**, **e**) as public key, safe-keeps (**n**, **d**) as private key

$$\boxed{\begin{aligned} n &= pq \\ \phi(n) &= (p-1)(q-1) \\ d &= e^{-1} mod \phi(n) \end{aligned}}$$



Property of RSA
We can also use the decryption key **d** to encrypt and encryption key **e** to decrypt. This does not hold in other public key schemes.

Algorithmic issues

- (step 1): How to find random prime? Rnadomly pick a number and test whether it is a prime.
- (step 3): Value of **d** can be efficiently computed from **e** and **n** using the Euclidean algorithm.
- Encryption of plaintexts at different times give the same ciphertexts. Some form of IV or padding is needed to introduce an extra element of randomness.

Issues that RSA faces

POOR EFFICIENCY AND PERFORMANCE

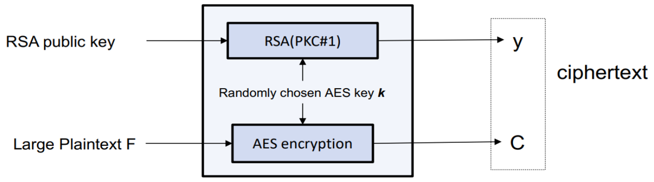
- RSA is significantly slower than AES (10,000x slower)
- A 128-bit AES has the same key strength as a 3072-bit RSA

To overcome this issue:
When a large file is to be encrypted under the public key setting, for efficiency, the following steps can be carried out:

1. Randomly choose an AES key k
2. Encrypt F using AES with k as the key to produce the ciphertext C
3. Encrypt k using RSA to produce the ciphertext q
4. The final ciphertext consists of 2 components: (q, C)

The reverse is to be done for decryption:

1. Decrypt q using RSA to produce the key k
2. Decrypt C using AES with k as the key to produce plaintext F



SECURITY OF RSA
RSA not necessarily 'more secure' than AES. It can be shown that getting the private key from public key is as difficult as factorization. Unknown if the problem of getting the plaintext from ciphertext and public key is as difficult as factorization.

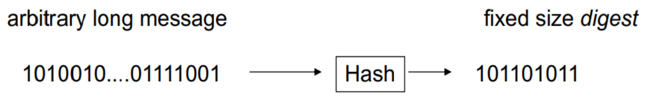
The 'textbook' RSA has to be modified so that difficult encryptions of the same plaintext to lead different ciphertexts, and such modifications are not straightforward (e.g. PKCS#1).

Strengths of PKC

Main strength is the public key setting, allowing the entity in public to perform encryption without a pre-establihed pair-wise secret key. This secret-key-less feature is also useful in providing authentication. PKC is rarely used to encrypt a large file.

Cryptographic Hash

A hash is a function that takes in an arbitrarily long message as input and outputs a fixed size **digest**.



Security Requirements:

- Preimage Resistant or One-way: Given a digest d, it is difficult to find a m such that h(m) = d. (Difficult to reverse engineer)
- Second-preimage resistant: Given m_1, difficult to find a second preimage $m_1 \neq m_2$ such that $h(m_1) = h(m_2)$.
- Collision-resistant: Difficult to find 2 different menssages $m_1 \neq m_2$ that hashes into the same digest $(h(m_1) = h(m_2))$

Popular Hashes: SHA-0, SHA-1, SHA-2, SHA-3

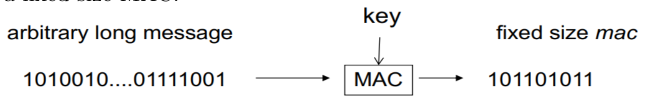
- SHA-0 was published by NIST in 1993. It produces a 160-bits digest. It was withdrawn shortly after publication and superseded by the revised version SHA-1 in 1995.
- SHA-1 is a popular standard. It produces 160-bits message digest. It is employed in SSL, SSH, etc.
- In 1998, an attack that finds collision of SHA-0 in 261 operations was discovered. (Using the straight forward birthday attack, collision can be found in 2160/2 = 280 operations). In 2004, a collision was found, using 80,000 CPU hours. In 2005, Wang Xiaoyun et al. (Shandong University) gave attack that can finds collision in 239 operations.
- In 2001, NIST published SHA-224, SHA-256, SHA-384, SHA-512, collectively known as SHA-2. The number in the name indicates the digest length. No known attack on full SHA-2 but there are known attacks on "partial" SHA-2, for e.g. attack on a 41-rounds SHA-256 (the full SHA-256 takes 64 rounds)
- In 2005, Xiaoyun Wang et al gave a method of finding collision of SHA-1 using 269 operations, which was later improved to 263 . A collision was found in 2017. It took 110 GPU years, completed 263 SHA1 operations.
- In Nov 2007, NIST called for proposal of SHA-3. In Oct 2012, NIST announced the winner, Keccak (pronounced "catch-ack").

Popular but obselete: MD5

- Designed by Rivest. MD, MD2, MD3, MD4, MD5, MD6.
- MD6 was submitted to NIST SHA-3 competition but did not advance to the second round of the competition.
- MD5 was widely used. It produces 128-bit digest.
- In 1996, Dobbertin announced a collision of the compress function of MD5.
- In 2004, collision was announced by Xiaoyu Wang et al. The attack was reported to take one hour.
- In 2006, Klima give an algorithm that can find collision within one minute on a single notebook.

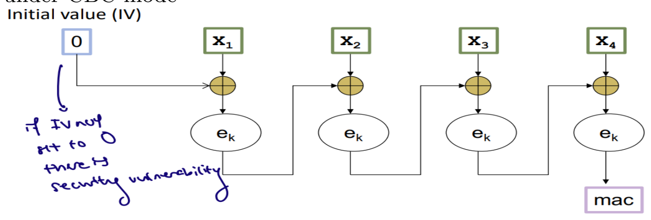
Message Authentication Code (MAC)

Also known as the Keyed-hash, it is a function that takes in an arbitrarily long message and a secret key as an input, and outputs a fixed-size MAC.



Security Requirements: Without knowing the key, it is difficult to forge the MAC. After seen multiple valid pairs of messages and their corresponding mac, it is difficult for the attacker to forge the mac of a message not seen before.

Popular keyed-hash (MAC): CBC-MAC, based on AES operated under CBC mode



HMAC (—— means concatenation)

$\text{HMAC}_k(x) = \textbf{SHA-1}\,(\,(K \oplus opad\,)\ ||\ \textbf{SHA-1}\,(\,(K \oplus ipad\,)\ ||\ x)\,)$

where

opad =	3636...36	(outer pad)
ipad =	5c5c...5c	(inner pad)

(the above are in hexadecimal)

Data Integrity

How do we know if an email we received or software we downloaded is authentic?

Unkeyed Hash for Integrity Protection Let us assume that there is a secure channel to send a short piece of information. We can then carry out the following steps:

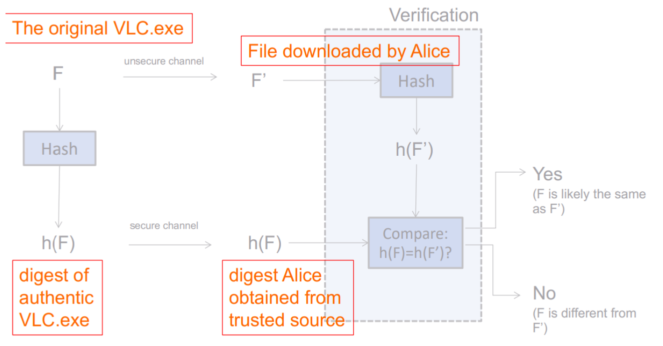
- Let F be the original file
- We obtain the digest h(F) from the secure channel
- We then obtain the file F' who origin claims that it is F
- We can then compute and compare the two digests h(F), h(F')

> – If h(F) = h(F'), then F = F' – with very high confidence
> – Else if h(F) $\neq$ h(F'), then integrity of file is compromised

What would an attacker do: attacker's goal is to make Alice accept any file other than F. To trick Alice, the attacker needs to have a F' such that h(F') = h(F) and F $\neq$ F'.

With the digest, the verifier can be assured that the data is authentic, thus the authenticity of the data origin is acheived. Nonetheless, when there is no secret key involved, the hash function only provides INTEGRITY not AUTHENTICTY.

This is because a man-in-the-middle attack can potentially edit the original message before rehashing it using the same hash function, then send the new digest over. This is the reason for the digest to be sent separately via some secure channel.



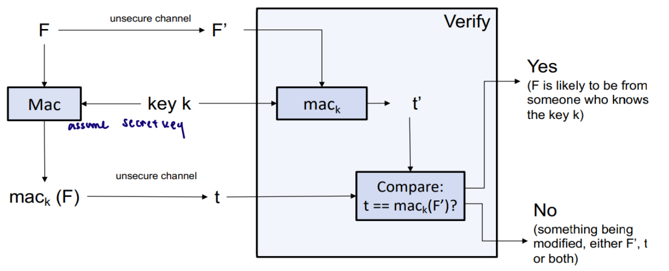
Data-Origin Authenticity

What can we do if we do not have a secure channel to deliver the digest.

Message Authentication Code

MAC can help to ensure integrity and authenticity as only the sender will know of the secret key used for the MAC. Comparing MAC_k(F) (which can be sent through an insecure channel) and MAC_k(F'), we can easily determine if the file is from the right person. If the MAC matches, F is likely to be from someone who knows the key k. Else something has been modified, either F', the MAC_k(F) sent through the insecure channel, or both.

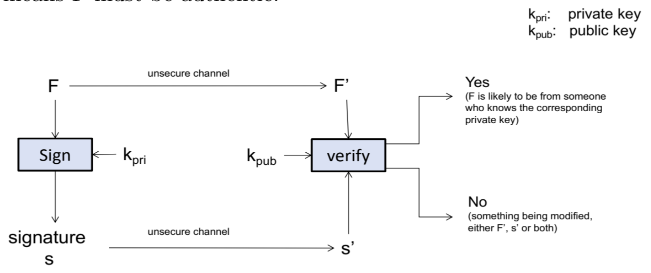
In this setting, the mac might be modified by attacker. If such case happened, it can be detected with high probability.



What would the attacker do: Forge a valid pair of (message, mac). Typically the MAC is appended to F, then stored as a single file or transmitted through the communication channel together. There is no issue on confidentiality, and the data F can be sent in the clear. Later, an entity who wants to verify the authenticity of F, can carry out the verification process using the secret key.

Data-Origin Authenticity (Signature), Asymm key

The public key version of MAC is called Signature. Here, the owner uses the **private key** to generate the signature. The public can use the **public key** to verify the signature. Anyone can verify the authenticity of the data, but only the person who know the private key can generate the signature. If the signature is valid, it means F must be authentic.



What is special about signature comapred to MAC?
Ease of key management. We can view the digital signature as the counterpart of the handwritten signature in legal document. A legal document is authentic or certified, if it has the correct handwritten signature. No one, except the authentic signer, can generate the signature. Signature scheme achieves Non-repudiation.

> Non-Repudiation: Assurance that someone cannot deny previous commitments or actions.

Birthday attacks

Similar to exhaustive search in encryption. When designing hashes, we want to do so such that attacks cannot do better than a birthday attack. All hash functions are subjected to **birthday attack**.

- In a class of 25 students, probability more than 0.5 that there is a pair of students with the same birthday.
- Suppose M messages, each message tagged with a value randomly chosen from {1, 2, 3, ..., T }.
- If M > 1.17 $T^{0.5}$ then with probability more tha 0.5, there is a pair of messages tagged with the same value.
- Suppose the digest of hash is 80 bits (T = 2^{80}) and the attacker wants to find a collision.
- If the attacker randomly generates 2^{41} messages (M = 2^{41}), then M > 1.17 $T^{0.5}$. Hence with probability more than 0.5, among the 2^{41} messages, 2 of them give the same digest.

This has serious consequence on the digest length requried by a hash function to be collision resistant. When the key length of a symmetric key is 112, the recommended length for digest is at least 224.

Using Encryption for Authenticity

Common for people to claim that their communication channel is secure as they use a certain encryption scheme that provides a high level of security. False sense of security, as encryption scheme merely provide confidentiality. However it does not provide INTEGRITY and AUTHENTICITY required for communication channels.

Example
The mobile phone and a server share a secret 256-bit key k. The server can send instructions to the mobile phone via sms.The format of the instruction is: **X P**

> where X is an 8-bit string specifying the operation, and P is a 120-bit string specifying the parameter. So, an instruction is of size 128 bits. If an operation doesn't need a parameter, P will be ignored. There is a total of 15 valid instructions.

- An instruction is to be encrypted using AES CBC-mode with 256-bit key, encoded to readable characters and sent as sms. (recap: block size of AES is 128 bits).
- After a mobile phone received a sms, it decrypts it. If the instruction is invalid, it ignores the instruction. Otherwise, it executes the instruction.
- This is not secure as the SMS does not provide any form of authenticity and integrity if it simple encrypts the message. A secure design could use a MAC instead of encryption.
- **Opportunity for replay attacks.** If intercepted one message, even with MAC appended, attacker can simply repeatedly send the message they intercepted, which will result in some command or instruction repeatedly executing. (can use a cryptographic nonce to prevent this instead).

Public Key Infrastructure

Public Key Distribution

There needs to be a way to transfer public key securely. If public key is distributed insecurely, there may be a possibility of facing a man-in-the-middle attack.

Public key vs Symmetric key distribution

Both need secure channel to distribute keys. Easier to securely 'broadcast' compared to 'establish' a different symmetric key for every pair.

Public Key	Symmetric key
only broadcasted once (linear)	securely establish different symmetric key with each of the rest (quadratic)
does not need to know existence of receiver while broadcasting	both entities needs to interact to establish the key

3 Different methods for Public Key Distribution

1. Public Annoucement

Owner broadcasts her public key (email, social media or physical namecard etc.) Many owners list their 'PGP public key' in blog, personal webpage etc.

Limitations:

Not standardized, no systematic way to search/verify the public key

2. Publicly Available Directory

List all names/email addresses and public keys in a public-key directory server.

Limitations:

Anyone can post their public keys in the server. Not clear how to verify the information. How does server verify information is authentic. Not everyone might trust the server as well.

3. PKI

Standardized system to distribute public keys. Addresses limitations of previous 2 methods. Aims to be **deployable** on a large scale. They are centered around **Certificate** and **Chain of trust of Certificate Authority (CA)**

Certificates and Trust

Certificate Authority

CA issues + signs digital certificates. Cryptography involves that of digital signatures. It keeps a directory of public keys, and also has its own public-private key pair. We assume that CA's public key has been securely distributed to all entities involved.

Without Certificate:

1. Alice sends email + public key to Bob, email signed with private key.
2. Bob asks the CA for what is the public key of Alice's email.
3. CA sends the public key of Alice back to Bob in a message. (This message is signed by CA).
4. Bob verifies that both public keys received are the same.

With Certificate:

1. Alice sends email + certificate, email signed with private key and public key is listed in certificate.
2. Bob verifies that signature in certificate is indeed signed by CA.
3. Since no one except CA can produce valid signature, authenticity of information in certificate is as good as coming directly from CA.

Most OSes/Browsers have preloaded CA's public keys, aka root CAs.

What is a Certificate?

A **certificate** is a digital document that contains at least the following 4 items:

1. name (e.g. *alice@yahoo.com* or *bbc.com* or **.bbc.com*)
2. public key of owner
3. time window that this certificate is valid
4. signature of CA $\rightarrow$ computed from CA's private key

There is additional information based on the intended purpose of the certificate.

- Usage of certification: (1) type of 'name' (email/domain name) or if it can take the role of CA (chain of trust)
- Digest (Fingerprint) for verification without CA's public key
- Meta data such as type of algorithm (ECC/RSA/key length)

Standard: X509:

- ITU-T X.509: Specifies formats for certificates, certificate revocation lists, and a certification path validation algorithm
- The Public-Key Infrastructure (X.509) Working Group (PKIX): IETF working group that creates Internet standards on issues related PKI based on X.509 certificates
- Structure:
  - Certificate:
    - Version Number
    - Serial Number
    - Signature Algorithm ID
    - Issuer Name
    - Subject Name
    - Subject Public Key Info: Public Key Algorithm, Subject Public Key
    - Validity period: Not Before, Not After
    - Issuer Unique Identifier (optional)
    - Unique Identifier (optional)
    - Extensions (optional)
  - Certificate Signature Algorithm
  - Certificate Signature

How do I get a certificate

Get a cert from a CA ($10 - $50 per year).
Let's Encrypt provides (basic) TLS certs at **no charge**

- Launched in 2016, valid for 90 days
- Renewal can take place at anytime
- Automated process of cert creation, validation, signing, installing and renewal

Self-signed certificate

Self-signed certificate is signed by its owner, said to be verified using the 'public key' listed in the certificate.

- Convenient in manual installation of public key. If user wants to include a binding of name and public key of an entity D, but D does not have a certificate signed by CA. D can 'self-sign' a certificate and pass to the user.
- User can manually accept the certificate (provided they trust the source). By accepting the self-signed cert user instructs machine to accept the binding of **D's name** and **their public key**. User takes responsibility in ensuring they are correct

Certificate Authority and Trust Relationship

Certificate Chain-of-trust

- Suppose Alice's certificate **issued**, signed by CA#1, but Bob does not have public key of CA#1
- Alice anticipating Bob might not have the pulic key of CA#1 can send her email, certificate and CA#1 certificate to Bob, allowing him to verify:
  - CA#1's certificate using root CA's public key
  - Alice's certificate using CA#1's public key
  - Alice's email using Alice's public key
- If Alice does not attach CA#1's cert, Bob can obtain it from other sources

Revocation

Non-expired certificates to be revoked for different reasons:

- Private key compromised
- Entity left an organization
- Business entity closed
- Issuing CA compromised

A verifier needs to check whether a certificate in question is *still valid*, even when not expired. Recommended for users (e.g. browsers) to periodically update its local cache of revocation list

2 different approaches:

- Certificate Revocation List (CRL): CA periodically signs and publishes a revocation list
- Online Certificate Status Protocol (OCSP): OCSP Responder validates cert

Limitations/Attacks on PKI

Implementation Bugs

Some browsers ignore substrings in the 'name' field after the null character when displaying it in address bar but include them when verifying certificate.

1. Name appeared in certificate: `www.comp.nus.edu.sg\0.hacker.13525.com` OR `*.hacker.13525.com`
2. Browser displays as `www.comp.nus.edu.sg`

Cause viewers to thought that they are connecting to 1., but in fact is connecting to 2.

Abuse by CA

Many CAs, one could be malicious. Rogue CA can practically forge any cert.

- Trustwave issued a "subordinate root certificate" (i.e. the receipt can now issue certificate) to an organization for monitoring the network. With this certificate, the organization can "spoof" X.509 certificates and hence is able to act as the man-in-the-middle of any SSL/TLS connection.

- see ComputerWorld, Trustwave admits issuing man-in-the-middle digital certificate; Mozilla debates punishment, Feb 8 2012.
- OR Lenovo's SuperFish scandal

Social Engineering

Attacker first rightfully register domain name that resembles targetted domain name. Next, use registered domain to confuse the victim in phishing attack. (Domain spoofing (typosquatting), URL spoofing, Fake URL).

- Method 1 (typosquatting):
  1. An attacker registered for a domain name: `luminus.nus.edv.sg` and obtained a valid certificate of the above name. No one has registered **edv.sg** and thus the attacker is about to get it.
  2. The attacker employed "phishing attack", tricking the victim to click on the above link, which was a spoofed site of luminus.nus.edv.sg
  3. The address bar of the victim's browser correctly displayed https://luminus.nus.edv.sg but the victim didn't notice that and logged in using the victim's password
- Method 2 (sub-domain):
  1. A more commonly deployed method uses sub domain. E.g. Attacker is the rightful owner of `134566.com`.
  2. Attacker creates a sub.domain: `luminus.nus.edu.sg.134566.com`.
  3. Since attacker is the owner of `134566.com`, it can get a valid certificate of `luminus.nus.edu.sg.134566.com` or `*.134566.com`

Authentication – Symmetric Key / Public Key

An entity wants to convince Bob that she is indeed Alice. The entity does so by convincing Bob that she knows some 'secrets'. Eve can sniff the communicatoin between the authentic Alice and Bob, and use stolen information to impersonate Alice.

Weak Authentication

Password is a type of secret. Sending the secret over is a simple method, but Eve can simply 'replay' it to impersonate.

Authentication: Challenge-response

Suppose Alice and Bob have a shared secret k, both agreed on a MAC. An entity who knows k is either Alice or Bob. Now, an entity **P**, prover wants to convince Alice that he is Bob. An attacker first sniffs all interactions between Alice and Bob. Attacker than communicates with Alice to try to trick Alice that it knows the secret k. A protocol is secure if this attacker can be prevented.

1. **P** sends Alice a hello message
2. (Challenge) Alice randomly picks message m, sends m to **P**.
3. (Response) **P** computes $t = \text{mac}_k(m)$. **P** then sends t to Alice.
4. Alice verifies the tag received is indeed the mac of n. If so accept, else reject.

- By property of mac, even if Eve sniffs, and obtain multiple valid pairs of m and t, Eve still (1) cannot get secret key k and (2) cannot forge mac for messages Eve have not seen before.
- Cannot replay response since challenge is randomly chosen. Challenge m ensures **freshness** of authentication process. It is also known as a *cryptographic nonce*.
- This protocol is called ***unilateral authentication*** as authenticity of Bob is verified only.
- Protocols that verify both parties are called ***mutual authentication***

Unilateral Authentication: PKC

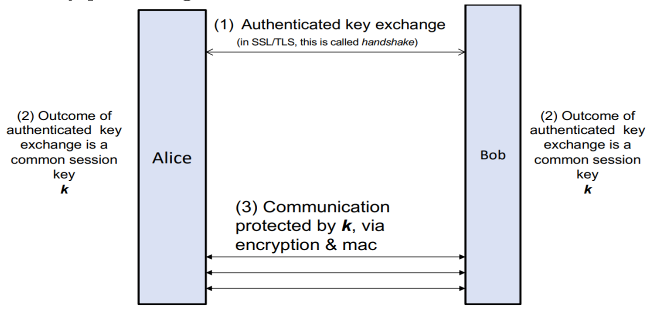
We can also have a public key version using signature. Suppose Alice wants to authenticate an entity **P** who claims to be Bob.

1. (Challenge) Alice chooses a random message r and sends to **P**: $\langle$ "Bob, here is your challenge", r $\rangle$
2. (Response) **P** uses his private key to sign r. **P** also attaches a certificate $\langle$sign(r), certificate $\rangle$
3. Alice verifies the certificate indeed belong to Bob and is valid. Next, extracts the public key from the certificate and verifies that the signature sign(r) is correct. If so, accept.

If Alice already knows Bob's public key, certificate can be omitted. Similarly, we assume an attacker can observe multiple interaction between Alice and the real Bob. After that, the attacker try to convince Alice that it knows the private key. By security property of signature, the eavesdropper can't derive Bob's private key and can't forge the response. Nonce r ensures **freshness**.

Handling Mallory using Authenticated Key Exchange

Consider a malicious Mallory, they can impersonate Bob and can sniff, spoof and modify the message. After Alice is convinced that **P** is Bob, Mallory can interrupt and take over the channel, thereby pretending to be Bob.


(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

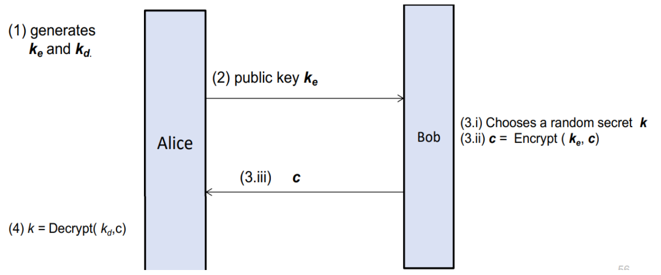
- Authentication key-exchange: Outcome is a new shared secret k known as **session key**.
- Subsequently, all communication will be protected (encrypted + mac) using k.

Key-exchange —— Sniff then Steal session key

Alice and Bob want to establish a common key. Key can be used to protect subsequent communication. The interaction could be eavesdropped by Eve, after sniffing, Eve wants to guess the established key.

PKC-based Key-exchange

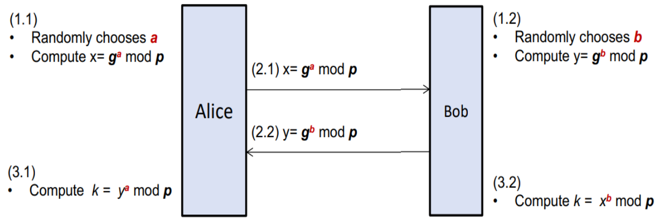
1. Alice generates a pair of private/public key
2. Alice sends public key $\mathbf{k}_e$ to Bob
3. Bob carries out the following:
   - Randomly chooses secret **k**
   - Encrypts **k** using $\mathbf{k}_e$
   - Sends the ciphertext **c** to Alice
4. Alice uses her private key $\mathbf{k}_d$ to decrypt and obtain **k**.


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

- Attacker (Eve) can obtain public key $\mathbf{k}_e$ and the ciphertext **c**.
- By security of PKC, from the public and ciphertext, attacker cannot get any information of the plaintext, which is the key **k.**

Diffie-Hellman Key-exchange

We assume both Alice and Bob have agreed on two public parameters, a generator g and a large (e.g. 1000 bits) prime p. Both g and p are not secret and known to the public



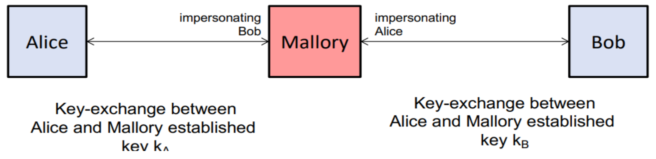
Security relies on the CDH assumption.

Computational Diffie-Hellman CDH assumption:
Given g, p, x= g^a mod p, y= g^b mod p, it is computationally infeasible to find k=g^{ab} mod p.

Remarks:
1. Step (1.1)&(1.2), (2.1)&(2.2), (3.1)&(3.2) can be carried out in parallel.
2. The assumption seems self-fulfilling. Nonetheless, there are many evidences that it holds.
3. The operation of "exponentiation" can be applied to any algebraic group, i.e. not necessary integers. CDH doesn't hold in certain groups. The crypto community actively searches for groups that CDH holds. E.g. Elliptic Curve Cryptography ECC is based on elliptic curve group where CDH believed to hold.

Authenticated Key-exchange

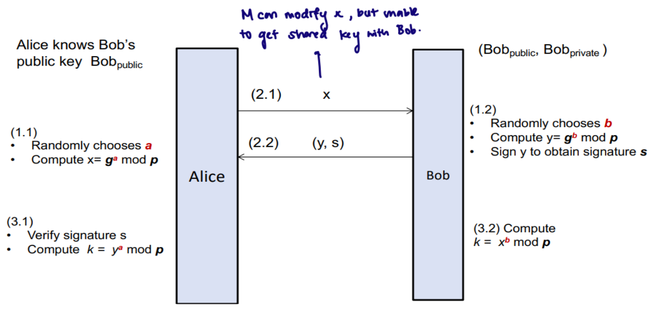
What if the adversary is malicious? Example, a man-in-the-middle? In this case, Alice mistaken that Mallory is Bob. Communication from Alice is encrypted using k_A. Mallory can decrypt using k_A and re-encrypt using k_B. Hence, Mallory can see and modify the message.


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

- Key-exchange protocol assumes that the adversary can only *sniff* but not malicious.
- Authenticated key-exchange can easily be obtained from existing key-exchange. (either PKC-based key-exchange or DH-based key-exchange). This can be done by signing all communication using private key.

Station-to-Station protocol STS

Authenticated key-exchange based on DH. We assume both Alice and Bob have agreed on two public parameters, a generator g and a large (e.g. 1000 bits) prime p. Both g and p are not secret and known to the public. Here, we consider unilateral authentication. Alice want to authenticate Bob.


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

- (Asymmetric) Previous authenticated key-exchange protocols such as Station-to-station are based on public key. That is, an entity is considered authentic if it can convince the other that it know the private key of the associated public key.
- (Symmetric) There are also symmetric key version, i.e both entities share a symmetric key. An entity is authentic if it can prove to the other that it knows the key.
- (Password) A special case of symmetric key is when the key is a password. Password usually has very low entropy, and thus potentially could be vulnerable to "offline" dictionary attack (see tutorial). There are secure protocols that, even if the entropy of password is low, it is still secure against offline dictionary attack. These are called "Password-Authenticated Key agreement" (PAKE).

Summary: Mutual Key-exchange

- Before the protocol:
  1. Alice has a pair of public, private key (A_{public} , $A_{private}$).
  2. Bob has a pair of public, private key (B_{public} , $B_{private}$).
  3. Alice knows Bob public key and vice versa. These two sets of keys are known as the ***Long-term key*** or ***Master key***.
- They carry out Authenticated key exchange protocol (e.g. STS). If an entity is not authentic, the other will halt.
- After the protocol: Both A and B obtain a shared key **k**, known as the ***Session key***
- Security Requirement:
  - (Authenticity) Alice is assured that she is communicating with an entity who knows $B_{private}$.
  - (Authenticity) Bob is assured that he is communicating with an entity who knows Aprivate.
  - (Confidentiality) Attacker unable to get the session key

Securing Communciation Channel —— TLS

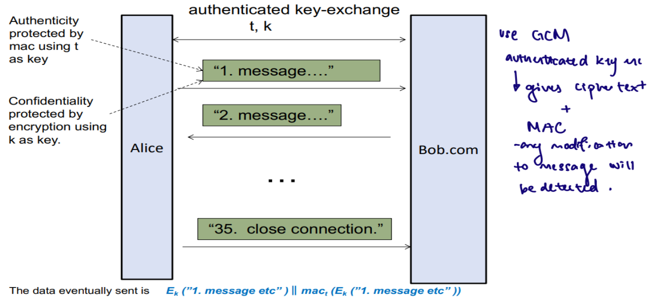
Public channel facilitates communication, but not secure with presence of Mallory. We can use crypto primitive on the messages, so that it is as secure as a 'private channel'.

Alice wants to visit a website `Bob.com`. ALice using the free wifi in a cafe called Mallory. How to secure the communication under the presence of malicious Mallory. In TLS/SSL:

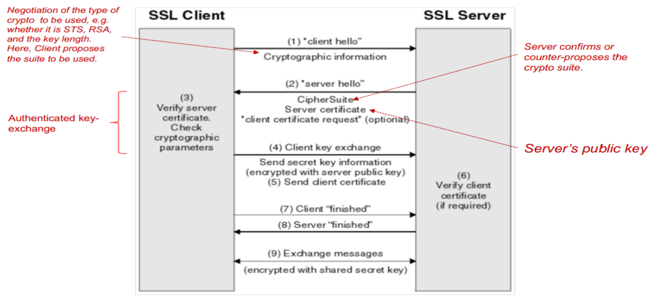
1. Using **long-term keys (i.e. Bob's public and private key)**, carry out authenticated key-exchange (aka handshake in TLS). Outcomes are:
   - Alice is convinced that she is interacting with Bob.
   - Both Alice and Bob have a shared session key. Mallory unable to get the key.
2. Subsequent communication protected by the session key.

TLS

1. Alice obtains `Bob.com`'s public key. This is done by having Bob sending his certificate to Alice.
2. Alice and Bob.com carry out **unilateral authenticated key exchange** protocol with Bob's private/public key. After the protocol, both Bob and Alice obtain two shared keys $\boldsymbol{t}$, $\boldsymbol{k}$ where $\boldsymbol{t}$ is the secret key of the MAC, and $\boldsymbol{k}$ is the secret key of the symmetric-key encryption, say AES. They are called the ***session keys***. From Alice's point of view, the protocol is secure in the sense that, only an entity who knows Bob's private key can complete the protocol. So, Alice is convinced that the entity who now holds $\boldsymbol{t}$, $\boldsymbol{k}$ is Bob. Here, Bob doesn't care about Alice's authenticity.
3. Subsequent interactions between Alice and Bob.com will be protected by $\boldsymbol{t}$, $\boldsymbol{k}$ and a sequence number. Suppose m_1, m_2, m_3, ... are the sequence of message exchanged, the actual data to be sent for m_i is
$$E_k(i||m)||mac_t(E_k(i||m))$$
where i is the sequence number. This is known as 'encrypt-then-MAC'. There are other variants: 'mac-then-encrypt' and 'mac-and-encrypt'. Using the wrong variant might leak info. When in doubt, use "authenticated encryption" such as AES GCM mode. GCM is fairly new and was not established when TLS was designed.


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…."
"2. message…."
. . .
"35. close connection."
The data eventually sent is E_k ("1. message etc") || mac_t (E_k ("1. message etc"))
use GCM
authenticated key enc
gives ciphertext
+
MAC
-any modification to message will be detected.

SSL and Transport Layer Security (TLS) are protocols that secure communication using cryptographic means. SSL is the predecessor of TLS and HTTPS is built on top of TLS.


Negotiation of the type of crypto to be used, e.g. whether it is STS, RSA, and the key length. Here, Client proposes the suite to be used.
SSL Client
SSL Server
(1) "client hello"
Cryptographic information
Server confirms or counter-proposes the crypto suite.
(2) "server hello"
CipherSuite
Server certificate
"client certificate request" (optional)
Authenticated key-exchange
(3) Verify server certificate. Check cryptographic parameters
Server's public key
(4) Client key exchange
Send secret key information (encrypted with server public key)
(5) Send client certificate
(6) Verify client certificate (if required)
(7) Client "finished"
(8) Server "finished"
(9) Exchange messages
(encrypted with shared secret key)

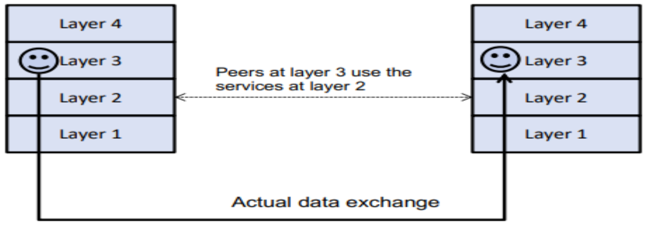
Network Security

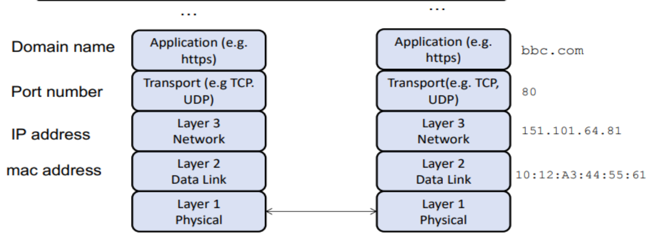
Computer Network establishes communicating connections between entities. To share networking resources and enhance robustness, instead of having dedicated lines between any 2 nodes, **packet switching** is deployed.

- Messages route via multiple switches and routers → Messages are broken into packets/frames
- Network security focuses on effects on attackers among intermediate nodes. Attacker wants to steal, modify, disrupt CIA.
- **Routing**:
  1. Route traffic to nowhere: Availability
  2. Know who is talking to who.
  3. Route traffic to attacker, become MITM.

Multiple hops and netowrk layers

To handle different types of intermediate nodes, network protocols are abstracted as layers. Conceptually, layer (N-1) provides a 'virtual' channel for entities in layer N. The *peer entities* in the N layer communicate using the virtual channel in layer (N-1). In turn, layer N provide yet another virtual channel for layer N+1.

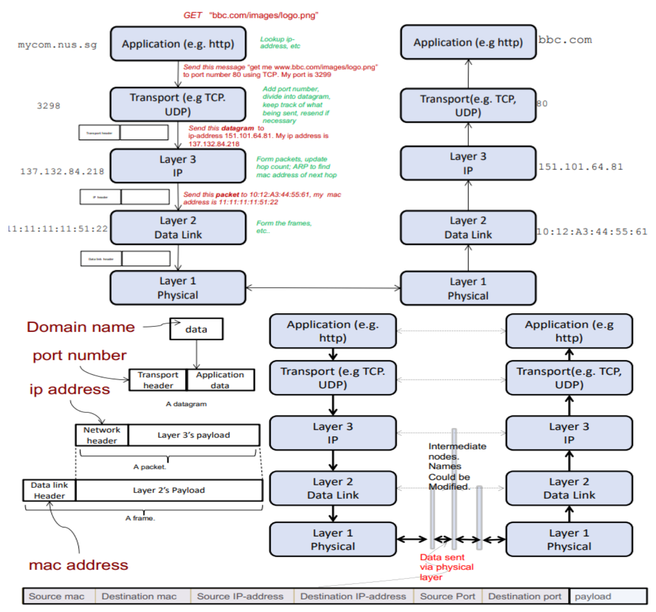




- Domain name: `bbc.com`
- Port Number: 60
- IP-address: 151.101.64.81
- MAC-address: 10:12:A3:44:55:61

Data units generated by layers

- Layer-N channel invoked to send message **n**, protocol in layer N might transform **m** into pieces of 'payload'. Each payload has a header generated by protocol. Both header and payload forms a data unit in layer N to be sent by layer N-1.
- Layer N at receiver from a series of received data units, reconstruct back to **m**.
  - Transport layer: datagram
  - Network layer: packet
  - Datalink layer: frame
- Each header contains 2 pieces of information: (1) src address, sender's address (at layer N) and (2) dest address, receiver's address (at layer N)



Intermediate nodes

Data are routed through multiple hops. Intermediate nodes could be owned by different third parties, e.g. Internet service provider (ISP), company's firewall. To facilitate routing, intermediate nodes see routing/header information, and might also change them (e.g translate the address, add "hop count").

Man-In-The-Middle —— MITM

MITM sits between 2 communicating parties. Unless otherwise stated, the MITM can sniff, spoof, modify, drop the data. Very often, when mentioning a MITM, it is clear from context what info the MITM has access to.

- **MITM sits in Layer 3**, MITM can see input (e.g. datagram, transport header) to Layer 3 and decide what the output of Layer 3 (packet, network header etc.) and know all internal info stored in Layer 3 (e.g. secret key).
- **MITM sits between Layer 2 and 3**, or just above layer 2, just below layer 3, we mean that the MITM can see and modify output of Layer 3, but do not have access to internal data in Layer 3.

Challenges in Network Security

- (Intermediate nodes and layers) There are many intermediate nodes, each handling routing-related information at a different layer.
- (Security Requirements)
  - **Availability** main concern in networking
  - **Confidentiality and integrity of Routing**:
    - ∗ Modifying routing process would break connections (availability) redirect traffic to adversary (confidentiality) ——could lead to side-channel leakages or implementation flaws if attacker can see ciphertext still.
    - ∗ Leakage of routing information could reveal connectivity information like A talking to B → Anonymity and Privacy.
  - (Legacy and security tradeoff) Initial design of netowrking protocols did no consider intentional attacks. (e.g. DNS does not employ strong protoection mechanisms for performance sake.)
  - (Management) There is a need to isolate and control data flow (firewall).

TCP/IP and UDP/IP

Transport Layer + IP

- Transport layer and IP layer often treated as a single layer: **ip-address** and **port number**.
- Each node in the network has a total of 65535 ports
- Communication channel between 2 nodes established by connecting 2 ports. e.g. between `11.11.1.1:2` and `55.55.5.5:65535`

UDP/IP

- `DatagramSend(srcPort, destIp, destPort, message)`
- `DatagramSend` library constructs an *IP datagram* and then an *IP packet* and passes to **Data Link layer**.
- There is a limit on the size of the array `message` → 65,000 bytes.
- `DatagramSend` does not return a result indicating whether the destination received the packer. There is a possibility that the packet is lost, arrived late or the recipient does not exist.
- UDP protocol is *unreliable*, data might get lost of arrive in a different order.

TCP/IP: reliable communication

- Application make library calls in the following form:

```
P = open_connection(2, "33.43.100.2", 65535)
    send(P, out_message) // can have multiple
    read(P, in_message)  // rounds of send/read
close_connection(P)
```

- `open_connection` carries out some form of handshake-protocol (TCP 3-way handshake) to make sure recipient is listening.
- `send` constructs *ip packets* of the following form and pass them to datalink layer. If message too long, form multiple ip packets.
- Protocol also has mechanism for *re-sending*, *re-ordering*, *acknowledgement* to ensure destination did in fact receive the message.

Reliability does not imply Security

- TCP/IP reliable but not <u>secure</u>. Intermediate nodes along communication route can still modify data in header or payload.
- Malicious intermediate node might act as MITM in IP layer.
  - spoof IP packet to inform one node to close the connection
  - reorder the packets

Listening to a port / Closed port

- Behind certain ports, there are applications waiting to process data coming via the respective port. → node/process is listening to the port
- If port is not listening → closed port, data sent to closed port will be dropped.

Name Resolution and Attacks: DNS/ARP

To use virtual connection in layer (N-1), need to know receiver name in layer (N-1) Method of finding the corresponding name is called *resolution* protocol.

- **Domain Name System (DNS):** resolution of Domain name to IP address
- **Address Resolution Protocol (ARP):** resolution of IP address to MAC address

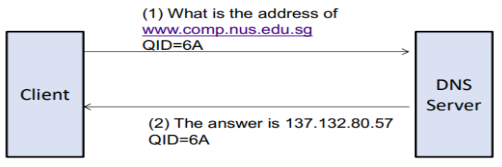
Domain Name System (DNS)

Given a domain name e.g. www.comp.nus.edu.sg, ip address can be found by querying a remote DNS server. The client who initiates the query is called the *resolver*. If the address is found, we say domain name is *resolved*.

DNS resolution:

(1) Client sends a query to DNS Server (using UDP protocol)

(2) DNS sends the answer back (UDP protocol)


(1) What is the address of www.comp.nus.edu.sg QID=6A
Client
DNS Server
(2) The answer is 137.132.80.57 QID=6A

- The query contains a 16-bit number, known as QID (query ID).
- The response from the server must also contains a QID.
- If the QID in the response doesn't match the QID in the query, the client rejects the answer.

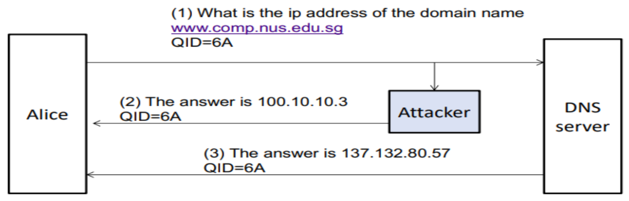
DNS Spoofing: Attack scenario

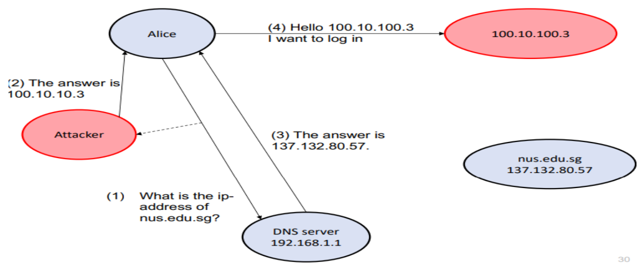
WHEN NOT UNDER ATTACK

- Alice using cafe wifi to surf web and wants to visit and login to <u>nus.edu.sg</u>. She types domain name into browser address bar.
- Browser makes query to DNS server to determine ip address (`nslookup`) Browser obtains and connects to ip address.

ATTACK

- Consider an attacker who is also in the cafe, since wifi no protected, attacker can
  - sniff data from communication channel
  - inject spoofed data into communication channel
- Attacker cannot remove/modify data sent by Alice (sits below physical layer).
- Attacker owns a webserver at `100.10.10.3` which is a spoofed NUS website. ATTACK:
  1. Alice ask for address.
  2. Attacker sniffs and knows about it, quickly spoofs a reply with the same QID.
  3. DNS server also sends a reply, since attacker closer to Alice, their reply likely reaches Alice first.
  4. Alice takes first reply as answer and connects to spoof NUS website `100.10.10.3`


(1) What is the ip address of the domain name www.comp.nus.edu.sg QID=6A
Alice
(2) The answer is 100.10.10.3 QID=6A
Attacker
DNS server
(3) The answer is 137.132.80.57 QID=6A



- DNS important component as it resolves domain name, can be a <u>single-point of failure</u> of network.
- DDOS attacks on web service, instead of directly attacking server, could conduct DOS to DNS server instead. → When DNS server downed, web service no longer reachable.

Poisoning Attack on ARP Table

A switch connects a few nodes. There is one node N_0 that acts as a gateway, which is a virtual node in the switch device that connects to the Internet.

- Switch is similar to telephone switchboard, it connects 2 ports.
- Switch does not understand ip-addresses and does not store ip-addresses (router), it conencts ports based on MAC-addresses.
- Switch keeps a table that maps port to MAC-addresses.
- Resolution of ip-address to MAC-address is done by nodes, each node also has a switch table.
- Nodes update each other using protocols on information on ip/MAC-addresses

ARP Poisoning is an attack that modifies (poison) the tables so as to gain MITM access

ARP Poisoning attack

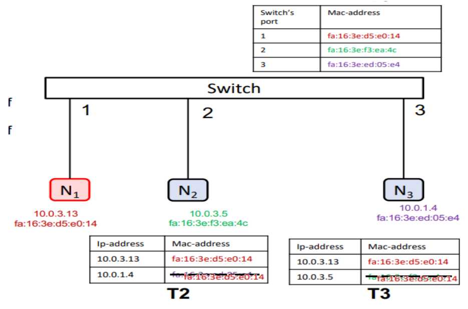
Under normal circumstances,, these are carried out when N_2 sends a packet to 10.0.1.4.

1. N_2 looks up the table **T2**, resolved to <u>fa:16:3e:ed:05:e4</u>.
2. N_2 sends the frame to switch, with destination <u>fa:16:3e:ed:05:e4</u>.
3. Switch looks up table **T0**, redirect frame to port 3.

Suppose N_1 wants to be MITM between 10.0.3.5 and 10.0.1.4

1. N_1 informs N_2 that MAC address of 10.0.1.4 is **<u>fa:16:3e:d5:e0:14</u>**.
2. N_1 informs N_3 that MAC address of 10.0.3.5 is **<u>fa:16:3e:d5:e0:14</u>**.

After tables are **poisoned**, all frames will be sent to N_1, which can relay the frames, modify the frames before relaying. N_1 becomes MITM in layer 2.



Switch's port	Mac-address
1	fa:16:3e:d5:e0:14
2	fa:16:3e:f3:ea:4c
3	fa:16:3e:ed:05:e4

Ip-address	Mac-address
10.0.3.13	fa:16:3e:d5:e0:14
10.0.1.4	~~fa:16:3e:ed:05:e4~~ fa:16:3e:d5:e0:14

Ip-address	Mac-address
10.0.3.13	fa:16:3e:d5:e0:14
10.0.3.5	~~fa:16:3e:f3:ea:4c~~ fa:16:3e:d5:e0:14

Denial of Service Attack

DOS is an attack on availability.

- **Availability:** The property of being accessible and usable upon demand by an authorized entity.
- **Denial of service (DOS):** The prevention of authorized access to resources or the delaying of time-critical operations.
- Many successful DOS attacks simply flood the victims with overwhelming requests/data.

Example of DOS attack

E.g. MyDoom worm which targeted SCO's website. Attacks start on Feb 12, 2004, sending a large number of DNS requests to the DNS server. For DOS to be effective, large number of attackers required. When DOS is carried out by large number of attackers, called DDOS: **Distributed Denial of Service**.

Reflection Attack

Reflection attack is a type of DOS in which the attackers send requests to intermediate nodes, which in turn send overwhelming traffic to the victim. Indirect, and thus more difficult to trace.

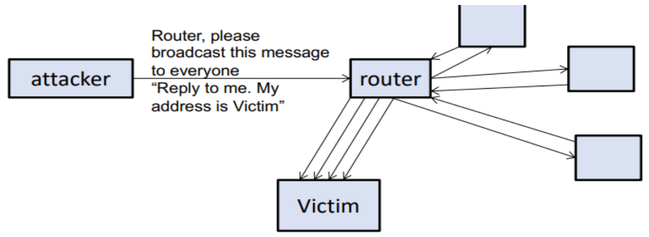
Amplification Attack

Refection attack mechanism can be measured by its amplification factor, which is the size of traffic the victim received over the size of traffic sent by the attacker. A single request could trigger multiple responses from the intermediate nodes. Reflection attack aka amplification attack.

Example of Reflection attacks: ICMP/Smurf Flood

This attacker is no longer effective, as routers are configured to not broadcast by default.

1. Attacker sends request `ICMP PING` to router, instructing router to broadcast request. Source ip-address of request is spoofed with victim ip address.
2. Router broadcast this request.
3. Each entity who has received this request, replies to it by sending an 'echo reply' to the source which is the victim.
4. Victim's network overwhlemed with the echo reply.


attacker
Router, please broadcast this message to everyone
"Reply to me. My address is Victim"
router
Victim

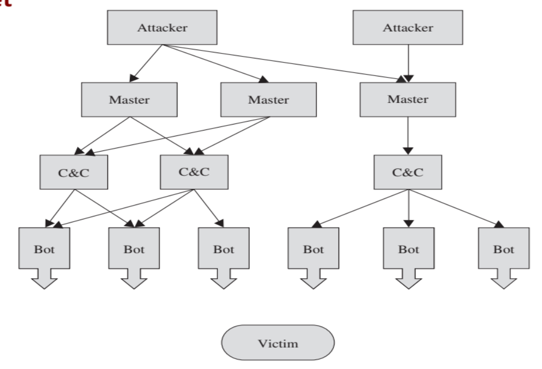
Example of Reflection attacks: DNS reflection attack

During a DNS amplification attack, the perpetrator sends out a DNS query with a forged IP address (the victim's) to an open DNS resolver, prompting it to reply back to that address with a DNS response. With numerous fake queries being sent out, and with several DNS resolvers replying back simultaneously, the victim's network can easily be overwhelmed by the sheer number of DNS responses. (e.g. Github using Memcache on 5 Mar 2018.)

Botnet

- **Bot**: aka *zombie* is a ccompromised machine
- **Botnet**: aka *zombie army* is a large collection of connected bots, communicating via covert channels.
- A botnet has a command-and-control mechanism, and thus can be control by an individual to carry out DDOS.
- Possible usages of a botnet:
  - DDoS flooding, vulnerability scanning, anonymizing
  - HTTP proxy, email address harvesting, cipher breaking

Botnet



- there is a new trend of using 'non-PC' such as IoT devices as bots in launching DDOS.

Useful tools

Wireshark —— Packet Analyzer

- Wireshark listens to "interactions" between the OS and the network card driver. (In other words, it is a MITM between OS and network card).
- Hence, header added by the network card, or modification made by the network card, may not be captured by Wireshark. This depends on the OS and the hardware. (usually Wireshark operates in layer 2)

Nmap —— Port Scanning

- Multiple processes running in a server. When the server receives a packet, base on the port number, it will decide which process handle that packet. So, by saying that a process/service is **"listening"** to a particular **port**, we mean that the process is running and ready to handle arriving packets with that particular port number.
- When a port is **"open"**, there exist such a process running in the server. When a port is **"closed"**, no process is listening to that port.
- If a port is **"closed"**, attacker is unable to feed malicious data to that port.
- **Port scanning:** process of determining which ports are open in a network
- **Port scanner:** tool used for port scanning. **Port scanning** is a useful tool for attacker, and network administrator to scan for vulnerabilites.

Protection —— Secure communication channel

Very often, when referring to a security protocol, we indicate the "layer" the protocol targets to protect. When analyzing an attack, it is also insightful to figure out at what layer the attacker resides.

A security protocol that protects layer k, would protect information in that layer and above. Hence, if an attacker resides at layer 1, and there is a security protocol that protect layer 3, then information generated in layer 3 and above will be protected, but information generated in layer 2 would not be protected by the security protocol

SSL/TLS

SSL/TLS sit on top of transport layer. When an application wants to send data to the other end point, it first passes the data and the ip address to SSL/TLS. SSL/TLS first 'protects' the data using encryption (confidentiality) and MAC (authenticity) and then instructs the transport layer to send the protected data.

Suppose Alice uses LumiNUS to upload a report `a.pdf` to LumiNUS server, LumiNUS uses HTTPS (HTTP on top of TLS.) Alice's machine carries the following:

1. The LumiNUS application passes the file `a.pdf` to https, and then to TLS.
2. TLS protects the data by encryption and mac.
3. TLS passes the protected data to the transport layer.

LumiNUS's server carries out the following:

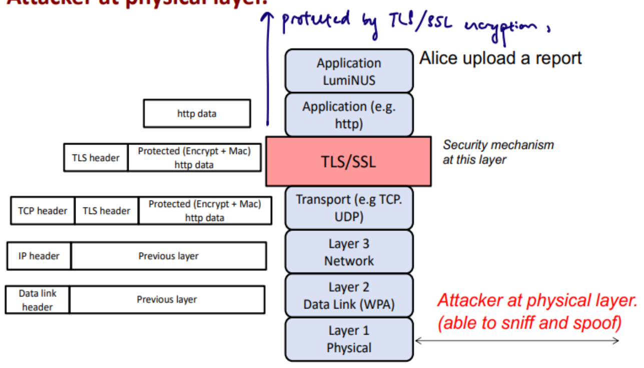
1. The transport layer passes the protected data to TLS.
2. TLS decrypt the data and verify the mac for integrity.
3. TLS passes the decrypted data to LumiNUS's application.

Scenario 1 —— Attacker @ Physical Layer

Attacker at physical layer who can sniff/spoof message at that layer. Alice uploading her report in cafe using free wifi (without WPA protectection). → anyone in the cafe has access to the physical layer and thus can sniff and spoof messages in that layer.

- Attacker cannot learn Alice's report, as data is encrypted by TLS → attacker only sees encrypted report.
- Attacker can learn the Alice is visiting LumiNUS website, since only those above TSL/SSL layer encrypted. Network layer is not encrypted, meaning attacker can still see the ip.

Attacker at physical layer.

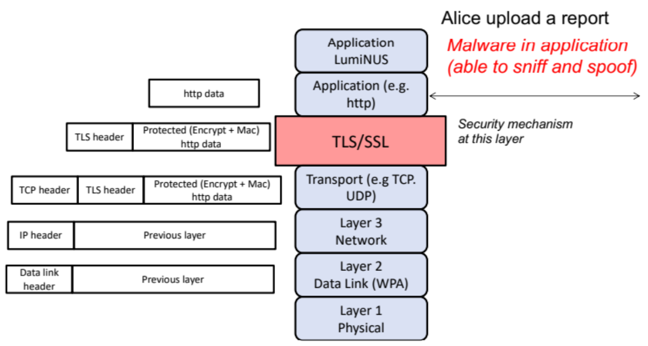


Scenario 2 —— Adversary @ Application Layer

Adversary in application layer, e.g. malicious javascript injected into LumiNUS, executed on Alice's browser.

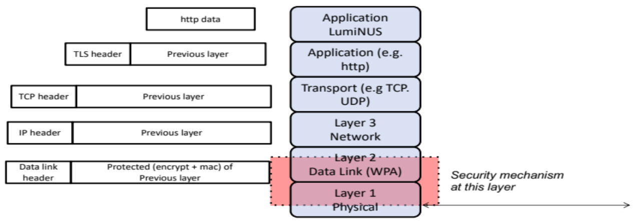
- Attacker can learn Alice's report AND can learn Alice's MAC-address.

Attacker at application layer (e.g. Malware in browser)



WPA2

Wifi Protected Access II (WPA2) is a popular protocol employed in home Wifi access point. WPA2 provides protection at layer 2 (Link) and layer 1 (Physical). Not all information in layer 2 is protected.



Scenario 3 —— Attacker @ physical layer

Attacker at physical layer and is able to sniff and spoof.

- Attacker cannot learn Alice's report and cannot learn the fact that Alice is visiting LumiNUS website, since above Layer 2 is encrypted.
- Attacker is able to determine MAC-address of Alice.

IPSec

IPSec provides "integrity/authenticity" protection of ip-address, but not confidentiality. Hence, attackers are unable to "spoof" the source ip-source, but can learn the source and destination ip-address of the sniffed packets.

- IPSec needs to modify the OS, and is a mechanism whose goal is to protect the IP Layer.
- Internet Protocol Security (IPsec) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a communication session. IPsec includes protocols for establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. IPsec can be used in protecting data flows between a pair of hosts (host-to-host), between a pair of security gateways (network-to-network), or between a security gateway and a host (network-to-host).
- Internet Protocol security (IPsec) uses cryptographic security services to protect communications over Internet Protocol (IP) networks. IPsec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.
- IPsec is an end-to-end security scheme operating in the Internet Layer of the Internet Protocol Suite, while some other Internet security systems in widespread use, such as Transport Layer Security (TLS) and Secure Shell (SSH), operate in the upper layers at Application layer. Hence, only IPsec protects any application traffic over an IP network. Applications can be automatically secured by IPsec at the IP layer."

Firewall

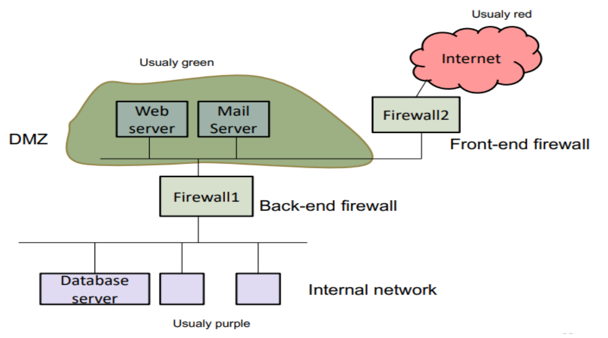
Some nodes contain more sensitive information than others. Some nodes are more secure, available patches might take some time to patch all systems and need these secure nodes to be prioritized. Certain protocols do not have protection / light-weight protection mechanisms. As such Firewall, Intrusion detection system (IDS) used to control access to the network.

Principle of least privilege and compartmentalization

- The principle of least privilege (PoLP, also known as the principle of minimal privilege or the principle of least authority) requires that in a particular abstraction layer of a computing environment, every module (such as a process, a user, or a program, depending on the subject) must be able to access only the information and resources that are necessary for its legitimate purpose.
- **Compartmentalization:** Generally, it refer to the notion of confining information within compartments.

- Firewall controls what traffic is allowed to enter the network **(ingress filtering)** or leave the network **(egress filtering)**.
- Firewalls are devices or programs that control the flow of network traffic between networks or hosts that employe differing security polices.
- **DMZ:** Demilitarized zone. A sub-network that exposes the organization's external service to the (untrusted) Internet.

A typical 2-firewall setting.



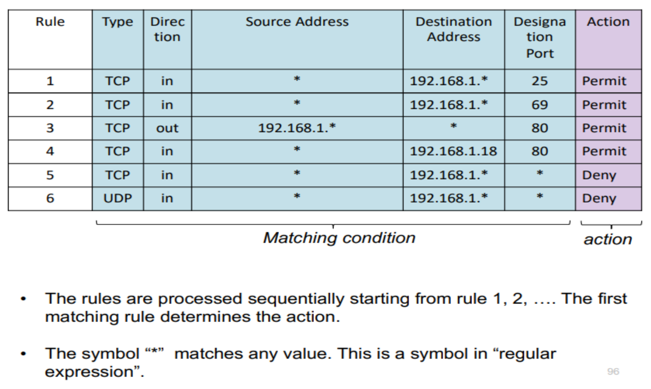
Firewall's controls are achieved by packet filtering. Filtering may occur in router, gateway/bridge or host. Packet Filtering inspects every single packet, typically only the TCP/IP packet's header information (Network and Transport layer). If payload is inspected, we call it a ***deep packet inspection (DPI)***. Actions taken after inspection:

- Allow packet to pass
- Drop packet
- Reject the packet (drop + inform sender)
- Log info
- Notify system admin
- Modify packet (advanced devices)

Firewall Rules

A firewall enforces a set of rules provided by the network administrator.

- Drop packets with "source ip-address" not within the organization's network. (To stop attacks originated within the network).
- **Whitelist:** Drop all packets except those specified in the white-list. (e.g. drop all except http, email protocol, and DNS)
- **Blacklist:** Accept all packets except those specified in the black-list. (e.g. allow https except ip-address in the blacklist).



Rule	Type	Direction	Source Address	Destination Address	Designation Port	Action
1	TCP	in	*	192.168.1.*	25	Permit
2	TCP	in	*	192.168.1.*	69	Permit
3	TCP	out	192.168.1.*	*	80	Permit
4	TCP	in	*	192.168.1.18	80	Permit
5	TCP	in	*	192.168.1.*	*	Deny
6	UDP	in	*	192.168.1.*	*	Deny

Matching condition
action

- The rules are processed sequentially starting from rule 1, 2, …. The first matching rule determines the action.
- The symbol "*" matches any value. This is a symbol in "regular expression".

96

Types of Firewalls

NIST's document groups firewalls into 3 types

1. Packet filters (inspect packet header)
2. Stateful Inspection (Deep packet inspection) —— e.g. count number of connections made in past hour
3. Proxy (Modify packets)

Intrusion Detection System

An IDS system consists of a set of "sensors" who gather data. Sensors could be in the host, or network router. The data are analyzed for intrusion. Three types of IDS:

- **Attack signature Detection:** The attack has specific, well-defined signature. For e.g. using certain port number, certain source ip address.
- **Anomaly Detection:** The IDS attempt to detect abnormal pattern. For e.g. a sudden surge of packets with certain port number.
- **Behavior-based IDS:** Can be viewed as a type of anomaly detection that focuses on human behavior. For e.g. The system might keep the profile of each user. It then tries to detect any user who deviates from the profile (e.g. start to download large files).

Management

- Management needed in order to monitor + adjust network characteristics
- **Security Operations Center (SOC):** a centralized unit in an organization that monitors the IT systems and deals with security issues
- **Security Information and Event Management (SIEM)**: Pronounced as "SIM". Approaches and tools for SOC. Popular systems: Splunk (https://www.splunk.com/), ELK Stack – Elasticsearch and Kibana (open sourced)

Access Control

We want to restrict **operations** on **objects** by **subjects**. Access control provides security perimeter which in turn facilitates segregation of accesses. Such segregation confines and localize damage caused by attacks.

Security Perimeter

Access controls setup perimiters/boundaries. With boundary, malicious activities outside of boundary would not affect resources within perimiter, and stays within boundary. Designed by the following principles:

- **Principle of Least privilege:** A camera apps shouldn't need to have access to the contact list to function. So, it is better not to grant the camera apps access to the contact list. With that, in case the camera app is malicious, confidentiality of the contact still preserved.
- **Compartmentalization:** A school website hosts two services: (1) course's fee payment and (2) exam result. With the perimeter between them, the exam result system would remain intact even if an SQL injection attack has been successfully carried out on the fee payment system.
- **Compartmentalization:** Colonial Pipeline's ransomware attack compromised the IT's system that handle client's database. If proper perimeter being setup between the IT and OT (Operation Technology) system, the failure of the client's database system should not affect OT system that manages the fuel pipeline.
- **Defence in depth:** A company deploys a firewall separating their server from DMZ. In addition, activities of server is logged and regularly inspected. So, even if attacker able to sneak in, their activities could be discovered by the monitoring team.
- **Segregation of duties:** A company keeps a backup copy of the "production" copy. The company implements a policy: a single person must not have access to both the production copy and the backup copy. Assigning different components to different person is aka Segregation of Duties. The goal is to eliminate single-point-of-failure. With that, a single rogue system admin (insider) is unable to corrupt all the data.

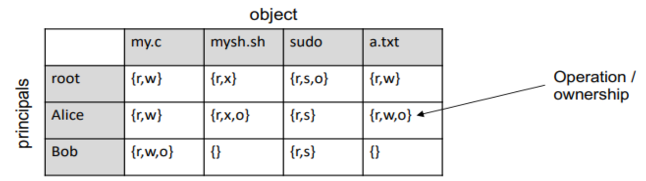
Terminologies

A *principal* (or subject) wants to access an object with some operation. The *reference monitor* either grants or denies the access. Principal → Do operation → Reference Monitor → Object

- **Principals vs Subjects:** Principals are human users, Subjects are entities in system operating on behalf of principals.
- Accesses to objects can be:
  - **Observe:** *read the file*
  - **Alter:** Writing/Deleting or changing file properties
  - **Action:** Executing a program
- **Owner:** Who decides access rights to object?
  - **Discretionary AC:** Owner object decides the rights.
  - **Mandatory AC:** System-wide policy decides, strict rules everyone follows.

Access Control Matrix

Specify access rights of a particular principal to particular object.
`r:read, w:write, x:execute, s: execute as owner, o: owner`



object
principals
Operation / ownership

	my.c	mysh.sh	sudo	a.txt
root	{r,w}	{r,x}	{r,s,o}	{r,w}
Alice	{r,w}	{r,x,o}	{r,s}	{r,w,o}
Bob	{r,w,o}	{}	{r,s}	{}

Access Control List

ACL stores access rights to an object as a list.

my.c	→ (root, {r,w}) → (Bob, {r,w,o})
mysh.sh	→ (root, {r,x}) → (Alice, {r,x,o})
sudo	→ (root, {r,s,o}) → (Alice, {r,s}) → (Bob, {r,s})
a.txt	→ (root, {r,w}) → (root, {r,w,o})

Capabilities

A subject is given a list of capabilities, where each capability is the access rights to an object.

root	→ (my.c, {r,w}) → (mysh.sh, {r,x}) → (sudo, {r,s,o}) → (a.txt, {r,w})
Alice	→ (mysh.sh, {r,x,o})→ (sudo, {r,s}) → (a.txt, {r,w,o})
Bob	→ (my.c, {r,w,o}) → (sudo, {r,s})

For ACL, it is difficult to obtain the list of objects a particular subject has access to. Conversely, for capabilities, it is difficult to get the list of subjects who have access to a particular object.

Intermediate Control

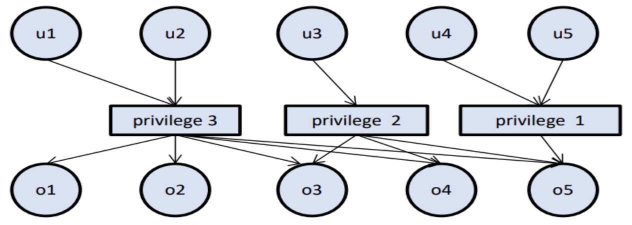
Not practical for an owner to specific each single entries in the access control matrix. "Group" the subjects/objects and define the access rights on the group.
In Unix file permission, subjects are divided into groups. Unix file permission uses ACL. For each object, the owner specific the rights for `owner`, `group`, `world` (everyone).

Role-based access control

Grouping can be determined by the role of the subject. Role associates with a collection of procedures, in order to carry them out, access rights to certain objects are required.

Privelleges

We sometime use the term privilege to describe the access right. Privilege can also be viewed as an intermediate control. It can be represented by a number, e.g. 1,2,3. if a subject can access an object, another subject with higher privilege can also access the object. (Can be viewed as a ladder, p2 subset of what p3 can access.)

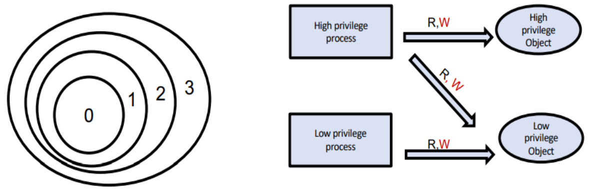


Protection Rings

In OS, "privilege" is often called protection rings. They are the same but with different name. Here, each object (data) and subject (process) is assigned a number. Whether a subject can access an object is determined by their respective assigned number. Object with smaller number are more important. If a process is assigned a number i, we say that the process runs in ring i. We call processes with lower ring number as having "higher privilege". A subject cannot access (both read/write) an object with smaller ring number.

Unix has only 2 rings, *superuser* and *user*.

e.g.



Bell-LaPadula Model —— data confidentiality

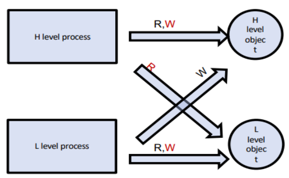
The following restrictions are imposed by the model:

- **No read up:** A subject does not have read access to object in higher level. This prevent a lower level from getting info in the higher level.
- **No write down:** A subject does not have append-right to object in lower level. This prevents a malicious insider from passing information to lower levels. (e.g. a clerk working in the highly classified department is forbidden to gossip with other staff).
- For "Confidentiality". (A subject can append to objects at higher security level. Is it possible that, by appending to an object, one could distort its original content? Yes. See e.g. in renegotiation attack.)

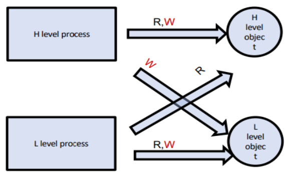
Biba Mode —— process integrity

The following restrictions are imposed by the model:

- **No write up:** A subject does not has "write" access to objects in higher level. This prevent a malicious subject from poisoning upper level data, and thus ensure that a process will not get compromised by lower level subjects.
- **No read down:** A subject does not has read access to objects lower level. This prevents a subject from reading data poisoned by lower level subjects.
- For "Integrity". If a model imposes both Biba and Bell-LaPadula, subjects can only read/write to objects in the same level (not practical).

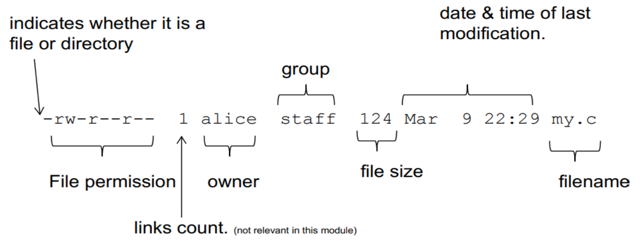


Bell-LaPadula (confidentiality)
No information coming down



Biba (Integrity)
No information going up.

Unix File System



indicates whether it is a file or directory
date & time of last modification.
group
-rw-r--r-- 1 alice staff 124 Mar 9 22:29 my.c
File permission
owner
file size
filename
links count. (not relevant in this module)

The file permission are grouped into 3 triples, that define the *read*, *write*, *execute* access for *owner, group, other (also called the "world")*.

A '-' indicates access not granted. Otherwise

r:	read
w:	write (including delete)
x:	execute (s: allow user to execute with the permission of the owner)

- Principals are user-identities (UIDs) and group-identities (GIDs).
- Information of the user accounts are stored in the "password" file
  - The file is made world-readable because some information in /etc/passwd is needed by non-root program. In earlier version of Unix, the "*" in the file was the hashed password H(pw), where H() is some cryptographic hash, and pw the password of the user. Hence, previously all users can see the hashed passwords of others.
  - The file is made world-readable because some information in /etc/passwd is needed by non-root program. In earlier version of Unix, the "*" in the file was the hashed password H(pw), where H() is some cryptographic hash, and pw the password of the user. Hence, previously all users can see the hashed passwords of others.
- `superuser (root)`: A special user is the superuser, with UID 0 and usually with the username root. All security checks are turned off for root.

Checking for file access

The objects are files. Recall that each file is associated with a 9-bit permission. Each file is owned by a user, and a group. The owner of a file, or superuser can change the permission bits.

- If the user is the owner, the permission bits for **owner** decide the access rights
- If the user is not the owner, but the user's group (GID) owns the file, the permission bits for **group** decide the access rights.
- If the user is not the owner, nor member of the group that own the file, then the permission bits for **other** decide.

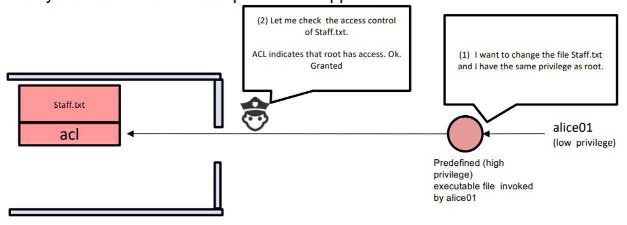
Controlled Invocation & privilege elevation

Some sensitive resources (such as network port 0 to 1023, printer) should be accessible only by the superuser. However, users sometime need those resources.

- The system provides a predefined set of applications that have access to F.
- These application is granted "elevated privilege" so that they can freely access the file, and any user can invoke the application. Now, any user can access F via the application.
- The programmer who write the application bear the responsibility to make sure that the application only performed intended limited operation. In other words, the user stay within the planned boundary when using the application.

With Controlled Escalation

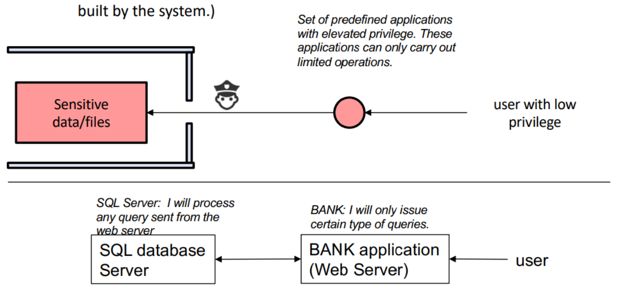
There is a set of predefined applications with "elevated" privilege. A normal user alice01 can't create applications with high privilege. However, any user can invoke these predefined applications.



(2) Let me check the access control of Staff.txt.
ACL indicates that root has access. Ok. Granted
(1) I want to change the file Staff.txt and I have the same privilege as root.
Staff.txt
acl
alice01 (low privilege)
Predefined (high privilege) executable file invoked by alice01

Bridges with Elevated Privilege

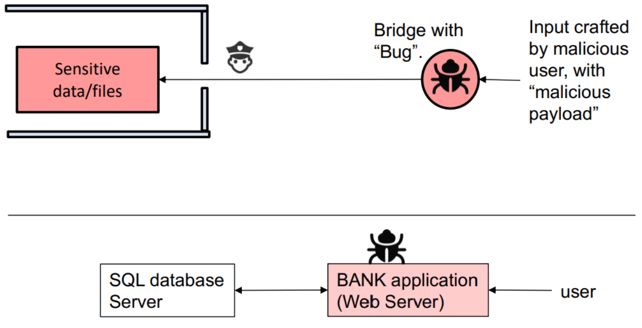
- We can view the predefined application as a predefined "bridges" for the user to access sensitive data. (note that the "bridge" can only be built by the system.)



Set of predefined applications with elevated privilege. These applications can only carry out limited operations.
Sensitive data/files
user with low privilege
SQL Server: I will process any query sent from the web server
BANK: I will only issue certain type of queries.
SQL database Server
BANK application (Web Server)
user

Firewall ensure that users can't directly send query to the SQL database server. Users can indirectly send query to SQL server via website. In this way, only certain types of queries can be sent by the users.

- "Bridge" is not implemented correctly and contains exploitable vulnerabilities.
- In some vulnerabilities, an attacker can trick the bridge to perform "illegal" operations not expected by the programmer/designer. This would have serious implication, since the process is now running with "elevated privilege".
- Known as **Privilege escalataion**



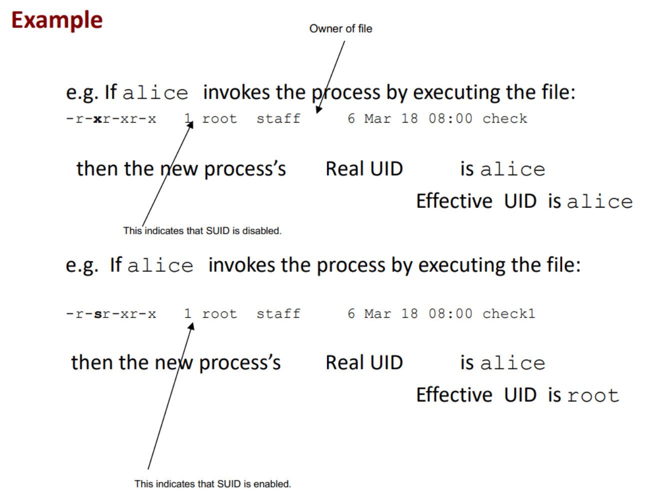
Sensitive data/files
Bridge with "Bug".
Input crafted by malicious user, with "malicious payload"
SQL database Server
BANK application (Web Server)
user

A bug in BANK that allows the user to send in arbitrary query. (SQL injection)

Controlled Invocation in Unix

A process has an identification (PID). New process can be created by executing a file or by "forking" an existing process, and associated by **Real UID** and **Effective UID**.

- **Real UID:** inherited from the user who invokes the process. For e.g. if the user is `alice`, then the real UID is `alice`.
- **Effective UID:** Processes can be created by executing a file. Each executable file has a SUID flag.
  - If the Set User ID (SUID) is disabled (the permission will be displayed as "x"), then the process' **effective UID** is same as *real UID*.
  - If the Set User ID (SUID) is enabled (the permission will be displayed as "s"), then the process' **effective UID** is inherited from the UID of the *file's owner*.
- When process wants to access a file, the **effective UID** of the process is treated as the "subject" and checked against the file permission to decide if it is granted or denied access.


Owner of file
This indicates that SUID is disabled.
This indicates that SUID is enabled.

Example

e.g. If `alice` invokes the process by executing the file:

```
-r-xr-xr-x   1 root  staff     6 Mar 18 08:00 check
```

then the new process's Real UID is `alice`
Effective UID is `alice`

e.g. If `alice` invokes the process by executing the file:

```
-r-sr-xr-x   1 root  staff     6 Mar 18 08:00 check1
```

then the new process's Real UID is `alice`
Effective UID is `root`

Need temp access to root file

- Create an executable file `editprofile` owned by `root`:
  `-r-sr-xr-x 1 root staff 6 Mar 18 08:00 editprofile`
- The program is made world-executable so that *any* user can execute.
- Furthermore, the permission is set to be "s": when it is executed, its effective UID will be "root"
- Now, if `alice` executes the file, the process' real UID is `alice`, but its **effective UID** is `root`. Following the checking rule, this process can now read/write the file `employee.txt`.

Elevated Privilege

- In this example, the process editprofile is temporary elevated to superuser (i.e. root), so that it can access sensitive data. We can view the elevated process as the interfaces where a user can access the "sensitive" information.
  - They are the predefined "bridges" for the user to access data.
  - The "bridge" can only be built by the root.
- These bridges solve the problem. However, it is important that these "bridges" are correctly implemented and do not leak more than required.