

# CS2107 Introduction to Information Security

AY23/24 Sem 2, github.com/gerteck

## 0. Introduction

CS2107: Introductory module, illustrates fundamentals of how systems fail due to malicious activities and how they can be protected.

### Lecture Notes notation

- **Textbook:** Security in Computing (5th ed). Prentice Hall. Reference [PFx.y] refer to chapter x section y of this book.
- Links with “read”: Part of lecture, required.
- Links with “see”: Optional good to browse references.

## Internet Security Threat Report Exc. Summary

Persistent threats are threats that often operate within the shadows, outside of attention focused.

- Highly targeted, often use combination of social engineering and software vulnerabilities to establish footholds within the targeted enterprise.
- Network protection not enough to mitigate threats, comprehensive monitoring program required to scan all internal and external network traffic. Identifying, securing data is also key to protecting assets.
- Recent attack types: Formjacking, Ransomware, Living off the Land (LoL),

## Introduction to Computer/Info Security

Systems may fail due to operator mistakes, hardware failures, poor implementation etc. Many systems robust against typical noise. Security is about intentional failures inflicted by deliberate human actions.

### Security Definitions: C-I-A triad

- **Confidentiality:** Prevention of unauthorized disclosure of information.
- **Integrity:** Prevention of unauthorized modification of info / processes.
- **Availability:** Prevention of unauthorized withholding of info / resources.
- Other requirements may include (Confidentiality: anonymity, covert channel), (Integrity: Non-repudiation (digital signature)), (Other: Accountability, traitor-tracing (printout w hidden watermark)) etc.
- Importance of understanding security requirements before adopting mechanisms. Do not adopt mismatched protection mechanisms.

## Difficulties in achieving Security

- **Security not considered** (in early design stage). Lack of adversarial thinking in design / trade-off in security with ease-of-use, performance / cost.
- Difficult to formulate requirements /design / implementation bugs.
- Difficult to operate/manage (Human error).
- **Known Vulnerabilities: CVE (Common vulnerabilities & Exposures)**. Repository of discovered vulnerabilities. Significant portion considered “implementation bugs”.
- **Zero-day Vulnerabilities:** Unpublished vulnerabilities. If attackers deploy attacks on zero-day v., victims have “zero-day” to react. Not easy to get.

## 1. Encryption

### Key Summary & Takeaways

- **Encryption designed for confidentiality.** (Not necessarily integrity).
- Formulate attack scenario by defining attacks it can prevent.
- Notions of “Oracle”: Encryption, Decryption, Padding Oracle.
- **Key strength:** Quantifying security by equivalence of best-known attack to exhaustive search.
- No known efficient attacks on modern schemes under “original” threat models, but there are pitfalls. Such as implementation error (wrong mode, wrong random source, mishandle of IV), side channel attack, implicit require integrity, padding oracle attack.
- Design of various symmetric key encryption schemes:
  - One-time pad, stream cipher (xor’ing with “pseudo-random” string)
  - Block cipher (mode of operations: CBC, ECB, CTR, GCM)
- **Crucial role of IV.** (Need randomness for indistinguishability). Make encryption probabilistic, how it is deployed, why it is important.

### Definitions

- **Symmetric Key Encryption Scheme:** Two algorithms: encryption and decryption. Meets correctness property, and must be secure.
- **Correctness:**  $(D_k(E_k(x)) = x)$ . **Security:** Informally, “difficult” to derive useful info of the key k, and plaintext x. Ciphertext resemble random sequence of bytes.
- **Cryptography:** Study of techniques in securing communication in present of adversaries with access. Encryption just a primitive (other includes crypto hash, digital signature etc. Common placeholders: Alice, Bob, Eve (“eavesdropper”), Mallory (malicious, modify message), etc.

### Attack Model

Aka: Threat / Adversary / Security Model, Attack Scenario.

Measuring **security of a system**: Through attack classes it can prevent. Secure w.r.t these classes of attacks. Attack models are application-dependent. Attack models described by:

- Attacker’s knowledge (info / service exposed) and computing resources
- Attacker’s goal

**Types of information we assume access to:** (Access to info can be formulated to accesses to an “Oracle”).

- **Ciphertext only attack:** Adversary given collection of ciphertext c. May know some properties of the plaintext, for e.g. the plaintext is an English sentence. (adv. can’t choose plaintext).
- **Known plaintext attack:** Adversary given collection of plaintext m and corresponding ciphertext c. (adv. can’t choose plaintext)
- **Chosen plaintext attack (CPA):** Access to blackbox (i.e. the oracle). Can choose, feed any plaintext m, obtain corresponding ciphertext c (all encrypt with the same key), reasonable large number of times. Can see ciphertext and choose next input. (*black-box called encryption oracle*).
- **Chosen ciphertext attack (CCA2):** Same as chosen plaintext attack, but adv. chooses ciphertext and blackbox outputs plaintext. (*black-box called decryption oracle*).
- **Note:** (strange to assume attacker has power to decrypt, but good reason is that in some practical scenario, attacker does have some but not full decryption capability, e.g. *padding oracle*). To cater for all scenarios, in formulation and design of encryption, we consider oracle with full decryption capability.)

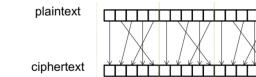
### Adversary’s Goals

- **Total Break:** Attacker wants to find the key.
- **Partial Break:** Few definitions, e.g. decrypt ciphertext, determine coarse information about plaintext, etc.
- **Indistinguishability (IND):** ) 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”). (Equivalently, unable to distinguish the ciphertext and a randomly sequence).
- **Note:** total break most difficult goal, since achieves all. Distinguishability weakest goal, design cryptosystem preventing weakest goal.

## 1.1 Classical Ciphers (Symmetric)

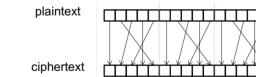
### 1.1.1 Substitution Cipher

- **Plaintext, ciphertext:** String over a set of symbols U.
- **Key:** Substitution table S, 1-1 onto function from U to U.
- **Key space:** Set of all possible keys (e.g. 27!). **Space Size** is total number of possible keys (factorial, 27!). **Key Size** is number of bits required to represent a key. ( $\log_2(27!)$ , since  $27!$  unique)
- **Exhaustive Search (Brute-force): Can be infeasible in worst case**
- **Known-plaintext-attack:** Access to pairs of ciphertext and corresponding plaintexts: *Sub. cipher “not secure / totally broken under known plaintext attack”*. (Possible in practice, e.g. certain words in header of email “From, Subject”, protocols bytes fixed header information etc.)
- **Ciphertext only attack:** Vulnerable to frequency analysis attack, when plaintexts English sentences.



### 1.1.2 Permutation Cipher (aka transposition cipher)

- Groups plaintext into blocks of  $t$  characters, applies secret permutation (1-1 onto function). Fails miserably on known-plaintext attack.



- S & P cipher not secure, performing substitution multiple times no use. However, by interleaving them (S&P), attacks become more difficult. Many modern encryption scheme (e.g. AES) designed using rounds of S & P.

### Terminology

- **Cryptosystem:** A system for encryption and decryption.
- **Plaintext:** Original form of message.
- **Ciphertext:** Encrypted form of message.
- **Perfect Secrecy:** A cryptosystem has perfect secrecy if for any distribution  $X$ , for all  $x, y$ :  
$$Pr(X = x|Y = y) = Pr(X = x)$$
- For any ciphertext  $y$  and plaintext  $x$ , the chances attacker correctly predicts  $x$  before knowing  $y$ , and after knowing  $y$ , are the same.
- **Work Factor:** Difficulty of breaking an encryption (Amount of effort necessary).
  - (E.g. determine time it would take to test single password, multiply by total possible passwords).



## Role of Unique IV

- Recall, IV appended to front of  $c$  to form ciphertext. This IV must be different for every message. IV need not be secret.
- Sequence is derived from IV and secret key. IV needs to be unique for every message! Otherwise, leaks information.
- Unique IV:** If IV different, two pseudorandom sequences will be different. Two ciphertexts of same plaintext will be different. Can just randomly choose the IV for each message.
- Hence, xor'ing two ciphertexts will not cancel out pseudosequences.
- IV makes encryption “probabilistic.”**

### Why IV? What if the IV is always the same?

Suppose there isn't an IV (or the IV is always set to be a string of 0's)

Consider the situation where the same key is used to encrypt two different plaintexts

$$X = x_1, x_2, x_3, x_4, x_5$$

$$Y = y_1, y_2, y_3, y_4, y_5$$

Further suppose that an attacker eavesdropped and obtained the two corresponding ciphertexts, U, V.

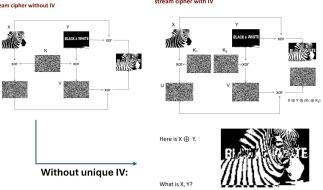
The attacker can now compute

$$U \oplus Y = (X \oplus K) \oplus (Y \oplus K)$$

By associative and commutative property of xor

$$U \oplus Y = (X \oplus Y) \oplus (K \oplus K) = X \oplus Y$$

So, from U and V, the attackers can obtain information about  $X \oplus Y$ , i.e. the following sequence

$$(x_1 \oplus y_1), (x_2 \oplus y_2), (x_3 \oplus y_3), (x_4 \oplus y_4), (x_5 \oplus y_5)$$


- IV also needed in **CBC mode**, to make encryption non-deterministic. If encryption deterministic (without IV), it will leak information on whether plaintext of two ciphertext are the same, if  $C_1 == C_2$ . Could be crucial piece of information.

## 1.3 Types of Attacks

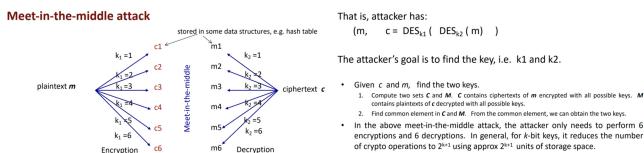
### 1.3.1 Meet-In-The-Middle Attack (Double / Triple DES)

#### Double / Triple DES

- Double / Triple DES:** To improve DES security, (DES weak as key length of 56 bits short), do multiple repeated encryption using different keys. A reason for this may be to utilize already existing suitable hardware to encrypt.
- DES does not form a group ( $E_{k1}(E_{k2}(x)) \neq E_{k2}(x)$ ) for some  $k3$ , so makes sense to use multiple encryption.
- Double DES:** Consider double encryption, we expect key-strength to be  $112$ . ( $2^{56+56}$ ). However, attacker, with storage space, can reduce key strength to below 57.

#### Meet-In-The-Middle Attack

- MITM:** Meet-in-the-middle Attack, a well-known plaintext attack, is a generic space-time tradeoff cryptographic attack against encryption schemes that perform multiple encryption operations in sequence.
- Breaking two-part encryption from both sides simultaneously.
- Primary reason why **Double DES not used**, and why **Triple DES key** (168-bit) can be brute forced by attacker with  $2^{56}$  space and  $2^{122}$  operations.
- Mechanism:** Assume attack has a pair  $(m, c)$  of plaintext and corresponding ciphertext.
- Remedy:** Use triple encryption, but with 2 keys. (E.g. 3DES, TDES, 3TDES etc.)



### 1.3.2 Padding Oracle Attack

#### Padding Format

- For fixed size blocks, e.g. block size of AES is 128bits (16 bytes), padding is needed to fit plaintext into last block.
- Many ways to fill value, but important piece of information encoded: **number of padded bits**. If information missing, receiver will not know length of plaintext.
- E.g. **PKCS#7 Padding Standard**.

#### PKCS#7 Padding Standard

- PKCS#7 is a padding standard.
- Read [https://en.wikipedia.org/wiki/Padding\\_\(cryptography\)#PKCS7](https://en.wikipedia.org/wiki/Padding_(cryptography)#PKCS7)

- The following example is self-explanatory.  
Suppose the block size is 8 bytes, and the last block has 5 bytes (thus 3 extra bytes required), padding is done as follow:

DD DD DD DD DD DD DD    DD DD DD DD DD **03 03 03**

- In general, the paddings are:

01  
02 02  
03 03 03  
04 04 04 04 etc.

- If the last block is full, i.e. it has 8 bytes, an extra block of all zeros is added.

#### Oracle

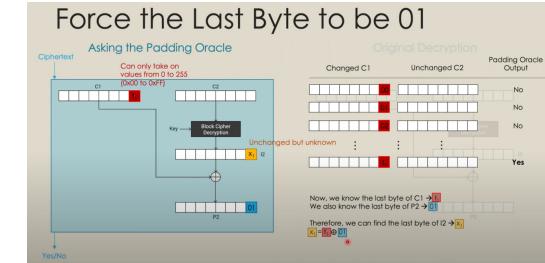
- Security Analysis:** Need to know 1. **information attackers have**, 2. attacker's goals.
- Oracle:** Query-answer system. Attacker can send in multiple queries, Oracle will output the answer. E.g.
  - Encryption Oracle:** Output ciphertext for given plaintext, of s. key  $k$ .
  - Decryption Oracle:** Output plaintext given ciphertext, of s. key  $k$ .
- Padding Oracle:** Attacker can query a ciphertext (encrypted using some secret key  $k$ , padding oracle knows  $k$ ). Oracle outputs yes/no if plaintext is in correct “padding” format.
- Padding Oracle can come in many forms, e.g. query response behavior, response time, subtle differences.
- Padding Oracle Attack Model:**
  - Attacker has ciphertext including iv:  $(iv, c)$
  - Attack's goal: plaintext of  $(iv, c)$ .

#### Padding Oracle Attack (on AES CBC Mode)

- AES CBC mode not secure against padding oracle attack. (In particular, when done with PKCS#7).
- Easily extend algorithm to find all plaintext. Attack is practical as there are protocols between client and server which performs this. *Now, if attack obtained ciphertext, attack can interact with server to get plaintext.*
- Prevention of Padding Oracle Attack:**
  - Deny access to such Oracle. Might not be feasible all the time.
  - Change padding standard to mitigate attack. However, may be smarter way to attack new padding.
  - Using CTR mode might avoid padding. In practice, bit strings have to be padded in one way or another.
  - Padding Oracle weaker form of Decryption oracle. If scheme secure in presence of decryption oracle, scheme also secure against padding oracle attack. GCM believed to be IND-CCA2 secure.

### Padding Oracle Attack Algorithm

- Algorithm:**
  - Force last X bytes to be padding
  - Do this by modifying byte of previous ciphertext/IV (Loop till YES)
  - Here, by XOR'ing padding and input, we get intermediate byte.
  - By XOR'ing this int. byte with original IV / ciphertext, find plaintext!
- Easily extend algorithm** to find all plaintext, by using increasing length of padding, decrypt from end to start. (Right to Left, repeat process).



#### Back to Decryption!



## 1.4 Cryptography Pitfalls

- Any secure encryption scheme can be vulnerable if not implemented or adopted properly. Examples include:
  - Predictable Randomness:** Secret key generated predictable, compromise security
  - Modify/Design own encryption scheme:** “Don’t roll your own crypto /security protocol”. Use standard library.

#### Wrong choice of IV

- IV Generation:** IV's must not be same for two different ciphertexts.
- E.g. To encrypt file F, IV derived from filename/meta-data.
- E.g. **Microsoft RC4 [implementation] Flaw**, in both Word and Excel, where same IV used when document modified, causing part of documents recovered with negligible amount of computation.
- When using AES under CBC mode, IV should be unpredictable to prevent certain attack. (E.g. IV = 1, 2, 3). BEAST attack exploits this.
- Reusing one-time pad:** E.g. US Verona Project against Soviets.

#### Reliance on Obscurity: Kerchoff's Principle

- Principle:** System should be secure even if everything about system, except secret key, is public knowledge.
- Security through Obscurity:** To hide design of system to achieve security. Not advisable to reveal certain settings (network structure, settings, algorithm, implementation, usernames etc). Obscurity as additional layer in defense-in-depth strategy. Deter, discourage novice attackers.
- E.g. **Against relying on Obscurity:** RC4 algorithm was trade secret, but was anonymously posted. MIFARE Classic smartcard also reversed-engineered.

## 2. Authentication Credential

- **Authentication Credential** as any data (PIN, digi. certificate, password) or device that is issued to individual / system used to authenticate identity for purposes of facilitating access.
- **Authentication:** Process of assuring communicating entity or origin of piece of information is one it claims to be.
- **Authenticity implies integrity.**



### • Authentication Process:

- For **data-origin auth**, one way is to use crypto scheme such as Signature, or MAC (message auth. code).
- For **communication entity auth**, need some authentication protocol employing above crypto primitives.
- **Credential:** Unforgeable info required for authentication. (E.g. password is an authentication credential.)

## 2.1 Password

### Password System

1. **Bootstrapping:** User, server establish common password, server keeps file recording identity (*userid, username*) & *password*.  
- (Bootstrapping, establish user chosen /or default password.)
2. **Authentication:** Server authenticates entity. Entity who gives correct password to claimed identity authentic.
3. **Password Reset:** Many reasons to reset password.

### Weak Authentication System, Replay Attack

- Simple sending of identity and password protocol is "**weak authentication**" system.
- Such protocol subjected to simple "**replay attack**".
- Information sniffed from communication channel, replay to impersonate.
- Under **strong authentication**, info sniffed cannot be used to impersonate.

### Attacks on Password System

- **Attack bootstrapping:** Attacks make use of default passwords.
- **Attack password reset process:** *Security-Cost Usability tradeoff*:  
- **Fallback authentication:** Security Qn / Pw. reset.  
- Enhance usability, reduce cost but reduce security.  
- **Danger:** Social engineering + pw. reset.
- **Search for correct password:**
  - **Exhaustive search:** Test all combinations
  - **Dictionary Attacks:** Online & Offline dictionary attacks.
- **Dictionary attacks** Two scenarios in dictionary attacks:
  - **Online dictionary attack:** an attacker must interact with the authentication system during the searching process. In other words, attacker must be online. (e.g. choose a password and ask the system (oracle) whether it is authentic)
  - **Offline dictionary attack:** There are two phases.
    1. The attacker obtains some information **D** about the password, possibly via some interactions. (e.g. somehow the hash of the password is sent over the protocol.)
    2. Next, the attacker carries out the searches using **D** without interacting with the system. e.g. testing hash of words in dictionary.
- **Stealing of password:**
  1. **Sniffing:** Shoulder surfing (look-over-shoulder attack), Sniffing communication (uncommon now, sniffing unencrypted password over public network in clear).
  2. **Viruses, Keylogger:** Computer viruses as key-loggers, or hardware keyloggers. Send captured data back to attack via "covert channel".
  3. **Phishing:** Trick to voluntarily send password to attacker. Social engineering attack.
    - **Spear Phishing:** targeted attack against particular small group of users.
    - **Phishing, Pharming, Vishing, Smishing**
    - **Prevention:** User training, blacklisting.
  4. **Cache:** Shared workstation where information is cached.
  5. Lost of password file
- **Password Strength**
- **Password Entropy:** Measure of (randomness) password strength.
- **Remark: Password Entropy**
  - We often encounter this term "entropy" when quantifying strength of password. Entropy is a measurement of randomness. In this class we won't go into the definition of entropy. We can use the following example to have a sense of its meaning.
  - Suppose a set **P** contains **N** unique passwords. Alice chooses her passwords by randomly & uniformly picking a password from the set **P**. Every password in **P** has an equal chance to be chosen (i.e. 1/**N**). In this case, by definition, the entropy of Alice's password is:  $(\log_2 N)$  bits
  - What if Alice doesn't choose the passwords uniformly, for e.g., the probability that she picks a word starting with letter "A" is higher than the probability that she picks a word starting with "Z"? In such cases, the entropy is not  $(\log_2 N)$ . By the definition of entropy, it is  $\sum_{i=1}^N p_i \log_2 p_i$   
where  $p_i$  is the probability that Alice picks the *i*-th word in **P**. (If we put  $p_i = 1/N$ , then we get  $\log_2 N$ .)
  - It can be shown that, for the entropy to be highest for a set of **N** items,  $p_i$  must be  $1/N$ . In other words, uniform choices.
  - (omit if this further confuse you) Another way to measure randomness is min-entropy, which is  $\min_i(-\log_2 p_i)$

Suppose with probability 0.5, Alice picks her password as "Alice", and for probability 0.5, she uniformly chooses from **P**. That is, each word in **P** has probability  $1/|D|N$  being chosen, and "Alice" has 0.5. Now, the entropy is roughly  $(\log_2 N - 1)$ , which is high. However, there is good chance in correctly guessed her password. So, entropy might not be a good measure of password strength. Note that, in this case, the min-entropy is low and is 1, correctly reflects the poor choice. (Note, the using "Alice" is not a problem.)
- Truly random password high "entropy", difficult to remember. User selection biased. Human generated 20-digit password likely to attain entropy much less than  $20 \log_2(10)$ .
- RFC 4086 suggests password at least 29 bits of entropy to secure against **online attacks**.
- If cryptographic keys generated from password, **offline attacks** possible, password should have at least 128 bits of entropy.
- **Online vs. Offline attack:** Need to communicate with server not under control to check whether password is correct.
- E.g. WPA2 personal vulnerable to offline dict. attack. Some "hash" of password sent in clear.
- **Enhance password system:**
  - Make online attack difficult by adding intentional delay / lock.
  - Make offline attack difficult by applying KDF to password.
  - System check for weak password / regular changes of password.
- **Password vs. Secret Key:**
  - KDF: key derivation function.
  - Password as source for crypto secret key.
  - Using cryptographic hash, hash multiple times to form key, increase cost of exhaustive search. Tradeoff utility.

### • Password Files:

#### Hashing:

- File stores userid, password. Add additional layer of protection.
- Password file should be "hashed" and "salted!"
- (Not "encrypted", don't want way to decrypt and get back)
- To authenticate, store and compare hash.

#### Salt:

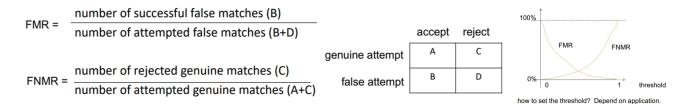
- Need same password hash to two diff. values, for two diff. userid.
- (Rainbow table: precomputed table caching outputs of crypto hash function to crack hashes)
- Achieve this by salting. New salt randomly generated, concat to pw., output different hash. Salt and hash stored in database.

## ATM Skimming

- Demonstrates password stealing.
- Authentication: User presents card, and PIN.
- Data encoded into magnetic strip using well-known standards, easy to "copy" card by reading and writing to spoofed card.
- ATM Skimmer: card-reader, camera overlooking keypad / spoofed keypad, some means to record, transmit.
- **Measures:** Anti-skimmer devices, shielding keypad, increase awareness, change to unforgeable smartcard.

## 2.2 Biometric

- **Biometric data as password.** For identification, or verification.
- **Enrollment:** template of user biometric data captured, stored (bootstrapping). **Verification:** capture biometric data, compare using matching algorithm.
- Inevitable noise capturing biom. data, leading to error in matching.
  - **FMR (False Match Rate), FNMR (False non-match rate).**
  - Adjust threshold to adjust FMR & FNMR.
  - **EER (Equal Error Rate, FNMR = FMR)**
  - **FER (False-to-enroll rate, users' data uncaptureable, e.g. injury)**
  - **FTC (Failure to capture rate, uncaptureable during auth, e.g. finger dirty**



- **Fingerprint as biometric:** performance depends on quality of scanner, EER range from 0.5% to 5%.
- Biometric data easily spoofable, include *liveness detection* to verify entity.

## 2.3 Multi-factor / Multi-step Authentication

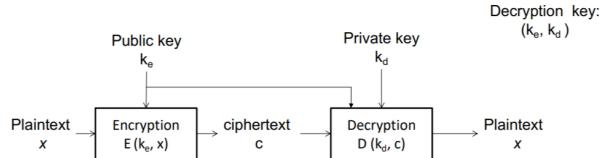
- **n-factor Authentication:** Require *n* authentication factors.
- **"Factors": Know (Pw, PIN), Have (card, phone, token), Are (Biometric).**
- Must be distinct factors for multi-factor (Know + Have, Have + Are) etc.
- One Time Password token: **Time-based /or Sequence-based**.
- **2-Step Verification:** Both same factor, e.g. (email + password, both "what-you-know", cannot be called "2-factor").
- **out-of-band:** In 2-step verification, two communication channels, main and separate for add. authentication. Non-main called "out-of-band" channel, assume attacker unable to compromise both channels.

### 3. Authenticity (Data Origin)

#### 3.1 Crypto Primitive: Public Key Cryptography

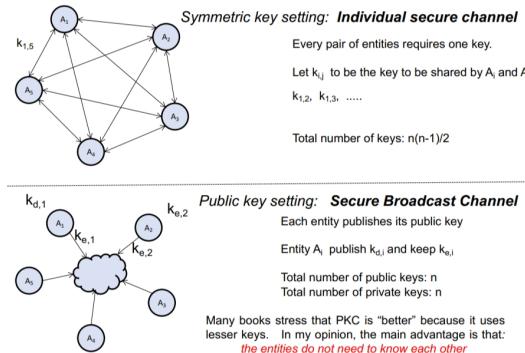
Public key crypto primitive includes public key encryption and signature. Goal is for confidentiality, and authenticity.

- Symmetric-key encryption uses same key for encrypt, decrypt.
- **Public Key (Asymmetric-key) scheme:** Uses different keys for encryption and decryption.
- Decryption algo also needs public key. When said "using private key to decrypt", this means using both.



- **Security Requirement:** Given public key & ciphertext, difficult to determine private key & plaintext. (Encryption oracle accessible.)
- **(PKC) Public Key Crypto:** Handles scenario of sharing symmetric key via secure channel. PKC only requires secure broadcast channel to distribute the public key. However, building secure broadcast channel to distribute public key is challenging.
- **PKC:** Useful for encryption, important application is in authentication (e.g. HTTPS). Limitations compared to symmetric in speed.

#### Keys Distribution



- **Remark:** Many PKC represent data as integers, algo used are some arithmetic operation on integers, represented using binary.
- A 1024-bit key is represented under binary representation. An algorithm exhaustively searches 1024-bit integers ( $2^{1024}$ ) is infeasible.
- For this course, we will consider "**classroom**" RSA, most basic form RSA.
- **In Practice:** Padded RSA (to destroy some property), choose strong primes, fast secure ways to generate primes, secure implementation to guard against side-channel attack, "e" is fixed mostly at 65537. Considerations of quantum computer.

#### Popular PKC Schemes

**RSA:** Key size - 2048 bits.

**ElGamal:** Exploit techniques in Elliptic Curve Cryptography (ECC), reduce key size to - 300 bits.

**Paillier:** Partial homomorphic with respect to addition.

#### Public Key: RSA ("Classroom RSA")

##### "Classroom RSA" - setup

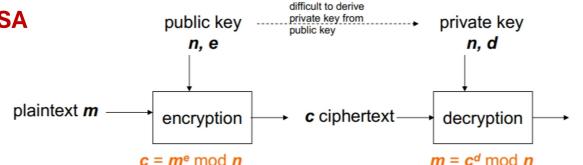
1. Owner randomly chooses 2 large primes  $p, q$  and computes  $n = pq$ .
  2. Owner randomly chooses an encryption exponent  $e$  s.t.  $\gcd(e, (p-1)(q-1)) = 1$  (i.e.  $e$  does not have common factor with  $(p-1)$  or  $(q-1)$ )
  3. Owner finds the decryption exponent  $d$   
where  $d e \bmod (p-1)(q-1) = 1$
- There is an algorithm that finds  $d$ , when given  $e, p$  and  $q$ . We won't get into the details.  
The term  $(p-1)(q-1) = \phi(n)$  is aka the Euler's totient function, which is the number of co-primes  $< n$ .
4. Owner publishes  $(n, e)$  as public key, and safe-keep  $(n, d)$  as the private key. (note that owner doesn't need to keep  $p, q$ )

##### Encryption, Decryption

public key $(n, e)$	private key $(n, d)$
• Encryption: Given $m$ , the ciphertext $c$ is $c = m^e \bmod n$	• Decryption: Given $c$ , the plaintext $m$ is $m = c^d \bmod n$
	$n = pq$ $\phi(n) = (p-1)(q-1)$

$$d = e^{-1} \bmod \phi(n)$$

#### RSA



Optional remark: Consider the relationship:  $c = m^e \bmod n$

- (RSA problem) Given  $c, n, e$ , find  $m$ , (called the  $e$ -th root of  $c$ )
- (discrete log problem): Given  $c, n, m$ , find  $e$ , (called the discrete log of  $c$ )

- **Correctness of RSA:** For any positive  $m < n$ , and any pair of public/private keys, that  $\text{Decrypt}(\text{Encrypt}(m)) = m$ .
- $(m^e)^d \bmod n = m$
- Correctness depends on the property of modulo.

##### Example RSA

There is an efficient algorithm that computes  $d$  from  $p, q, e$ . Details omitted.

- Suppose  $m = 9$
- $p=5, q=11, n=55$
- $(p-1)(q-1) = 40$
- Suppose  $e = 3$ , then  $d = 27$   
note that indeed  $3 \cdot 27 \bmod 40 = 81 \bmod 40 = 1$

Suppose  $m = 9$

Encrypt:  $c = m^e \bmod n = 9^3 \bmod 55 = 14$

Decrypt:  $c^d \bmod n = 14^{27} \bmod 55$

There is an efficient algorithm that computes  $d$  from  $p, q, e$ . Details omitted.

$$= 14^{(2+1)(1+1)} \bmod 55$$

$$= (36 \cdot 16 \cdot 9) \bmod 55$$

$$= 9$$

- **Interchangeable role of encryption & decryption key:** Swap role of  $d$  and  $e$ , public can decrypt, owner can encrypt. Usually does not hold in other PKC. This property useful in designing **signature scheme**.

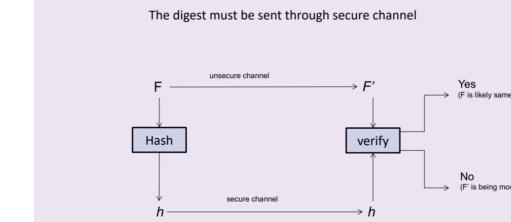
- **Algorithmic Issues:** There is efficient algo to compute exponentiation, hence, efficient way to encrypt, and decrypt (given  $n, m, e$ , find  $m^e \bmod n$  for encrypt, given  $c, d, n$ , computer  $c^d \bmod n$ ).
- **Step 1 setup:** Find random prime, randomly pick number, test if prime. (Primality Test). **Step 3 setup:** Value of  $d$  efficiently computable from  $e$  and  $n$  using *extended Euclidean algorithm*.

#### Security of RSA

- Getting RSA private key from public key as difficult as factorizing  $n$ .
- However, not known whether the problem of getting plaintext from ciphertext as difficult as factorization.
- **post-Quantum cryptography:** Generally refers to PKC that are secure against quantum computer.
- Significant efforts to migrate current systems to post-quantum crypto (Quantum possibility to render RSA algo insecure by 2030). From 2016, NIST plans to choose standard by 2024. Four candidates for Rd 4 (2022).

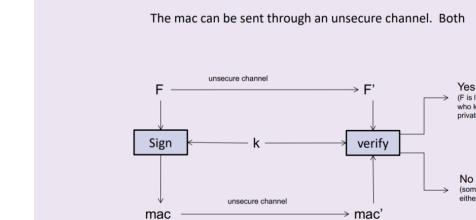
#### Summary of Data Authenticity

##### Digest (Hash)



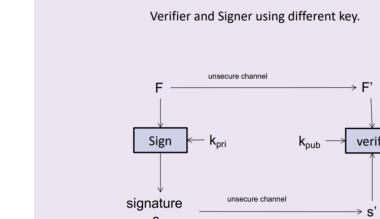
Security Requirement: Different to find a pair  $F, F'$  with the same digest

##### MAC (Message Authentication Code)



Security Requirement: Without knowing  $k$ , it is difficult to forge a mac, even after seen many pairs of messages and their valid mac.

##### Signature



Security Requirement: Without knowing the private  $k_{pri}$ , it is difficult to forge a signature, even after seen many pairs of messages and their valid signatures.

## 3.2 Data Authenticity (Hash): Unkeyed Hash

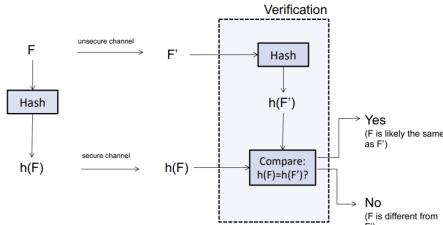
**Follow Kerckhoff's Principle:** Adversary knows all the algorithms.

### Hash

- **A (cryptographic) hash** is a function that takes an arbitrary large message as input, and outputs a fixed size (say 160 bits) digest.
- **Collision-resistant:** It is difficult for an attacker to find two different messages  $m_1, m_2$  that “hash” to the same digest.
- **One-way:** A hash that is collision-resistant is also one-way, that is, given a digest  $d$ , it is difficult to find a message  $m$  s.t.  $h(m) = d$ .
- Examples of non-secure hash algorithms: (taking selected bits from data, CRC checksum).
- **Popular Hash:**
  - **SHA-0, SHA-1/2/3:** SHA-1 popular standard producing 160-bits message digest, employed in SSL, SSH etc. In 2017, successful collision attack on SHA-1 found.
  - **MD5:** Produces 128-bit digest. However, algorithm that can find collisions have been found.

### Application Scenario of Unkeyed Hash (no secret keys)

- **Example:** Alice downloaded a software vlc-2.2.8-win32.exe from the web. Is the downloaded file authentic?
- Since the website is hosted with HTTPS protocol, Alice is being assured that the content displayed on the browser is from VLC and authentic.
- However, the downloading site is a 3rd-party, channel not secure. Possibility 3rd party website is malicious and giving out virus infested software.
- To verify that the file indeed original, after download, check the integrity by matching the “hash” of the file with the “SHA-256 checksum” displayed in the browser. If they match, file is intact. If not, file is corrupted.

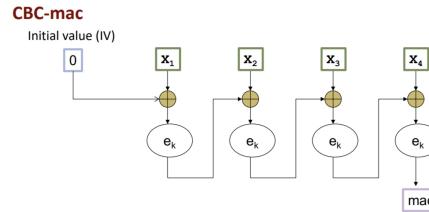


- For the attacker, to find a  $F'$  that has the same hash as  $F$ , this is known as **2nd preimage problem**.
- Although practically, some part of program needs to be same to be workable, we focus on modest goal of finding any file. If easiest goal preventable, so is harder goal. Hence, for hash, **easiest attack goal is “collision”**.
- **A Hash function H() is called collision-resistant** if it is computationally difficult to solve collision attack. More specifically, it is collision-resistant if no method can significantly outperform birthday attack

## 3.3 Data Authenticity (MAC): Keyed Hash

- **MAC:** Message Authentication Code.
- **A keyed-hash** is a function that takes an arbitrary large message and a **secret key** as input, and outputs a fixed size (say 160 bits) MAC.
- **Forgery-resistant:** Even with multiple valid pairs of messages and mac, difficult for attacker to forge the mac of message not seen before.
- **Popular Keyed-Hash (MAC)**

### – CBC-MAC: (based on AES under CBC mode)



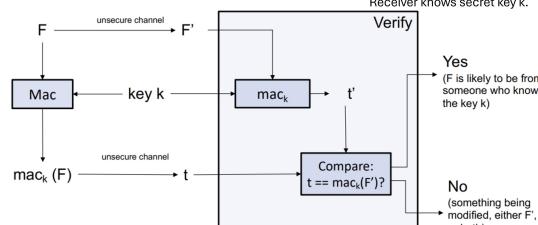
### – HMAC : based on SHA (optional)

## Application Scenario for MAC (keyed hash)

- In the previous example (on vlc), we assume secure channel to send the digest.
- In scenarios without secure channel to deliver the digest, we can protect the digest with the help of some secrets.
  - In the symmetric key setting, it is called the mac.
  - In the public key setting, it is called the digital signature

### mac (Message Authentication Code)

In this setting, the mac might be modified by attacker. If such case happened, it can be detected with high probability. MAC typically appended to  $F$ , to be used to carry out verification. Receiver knows secret key  $k$ .



**Security Requirement:** Without knowing the key  $k$ , it is difficult to forge a mac.

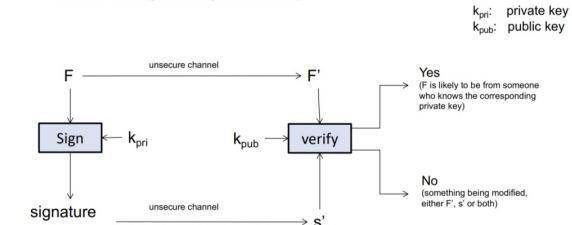
- Attacker would need to forge valid pair of (message, mac).
- Typically, the mac is appended to  $F$ . Stored as a single file / transmitted together. (Aka authentication tag / code). Later, entity can carry out the verification process using secret key.

## 3.4 Data Authenticity (Signature): Asymmetric Key

**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.
- So, anyone can verify the authenticity of the data, but only the person who know the private key can generate the signature.

Verifier and Signer using different key.



**Security Requirement:** Without knowing the private  $k_{pri}$ , it is difficult to forge a signature.

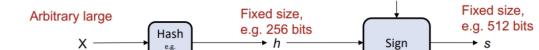
- Similarly, computed signature appended to  $F$  and stored as a single file.
- “Alice signs file  $F'$ ”: signature is computed using Alice’s private key and  $F$ , then appended to  $F$ .
- **Authenticity**’ of  $F$  can be verified by anyone, using the public key. The valid signature can only be computed by someone who knows the private key. So, if it is valid, then  $F$  must be authentic.
- **Non-repudiation:** Assurance that someone cannot deny previous commitments or actions.
- Can be seen as digital signature, certified authentic, on top of ease of key management.
- **Popular Signature Scheme:**

- A popular group of schemes use RSA for the sign/verify component: RSASSA-PSS, RSASSA-PKCS1: signature scheme based on RSA
- DSA (Digital Signature Algorithm) is another popular standard whose security depends on discrete log.

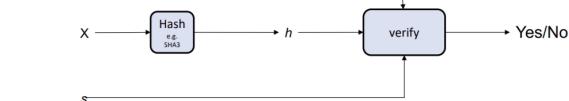
## Design of Signature Scheme

**Signature** need two things: **Unkeyed hash, sign/verify algorithm.**

Generation of signature



Verification of signature



• Hashing done not only for purpose of efficiency.

- For RSA-based signature, to be secure, hash must be carried out, e.g. if message very short, only 1 byte.
- For RSA, essentially, signature is the “encrypted” digest. During verification, decrypt to obtain digest and compare.
- Note: Here, we use private key to encrypt. Previously, we used public key to encrypt. Role flipped.

## 3.5 Attacks & Pitfalls

### Birthday Attack on hash

Similar to “exhaustive search” in encryption. Birthday attack applicable to all hash functions, similar to how exhaustive search is on all encryption schemes.

- Hashes are designed to be **collision-resistant**, hard to find two diff. messages give same digest. However, subjected to birthday attack.
- We want to design a hash so that known attacks cannot do better than birthday attack.
- **Birthday attack** is an attack that occurs when someone exploits the mathematics behind the birthday problem in probability theory to launch a cryptographic attack.
- The birthday problem states that in a group of 23 people, there's a 50% chance that at least two will have the same birthday.

### Birthday Attack Formula (Single Common Pool)

- Suppose we have  $M$  **messages**.
- Each value tagged with value randomly chosen, range of values =  $[1, T]$ . ( $T$  is **no. of values**)

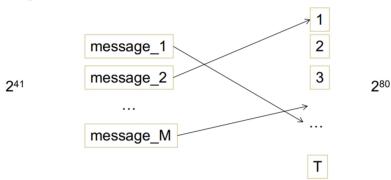
• If:

$$M > 1.17 * T^{0.5}$$

- There is a probability higher than 0.5 that there is a pair of messages tagged with the same value.
- i.e.  $P(\text{collision}) > 0.5$ .

#### Birthday Attack Example

- Suppose the digest of a hash is 80 bits ( $T = 2^{80}$ ). An 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, two of them give the same digest.



- In general, for a **set of  $T$  values**, where  $M$  **values chosen randomly**,
- the probability **at least one value chosen more than once** (aka collision) approximately:

$$\begin{aligned} p(M; T) &\approx 1 - e^{-M(M-1)/(2T)} \\ &\approx 1 - e^{-M^2/(2T)} \end{aligned}$$

- NIST recommendation of key length: (2019-2030)  
Security Strength (key length for symmetric key): 112  
Discrete Logarithm Key (length of digest): 224.
- The length of truncated message digests used shall be at least twice the desired security strength required for the digital signature. (Due to birthday attack)

### Variant of Birthday Attack (Two Pools)

The first version in lecture notes takes a pair from a common pool.

- Consider where there are two pools. One item in the pair is drawn from a pool, and another item drawn from another pool.
- Let  $S$  be a set of  $k$  distinct elements where each element is an  $n$  bits binary string. Now, let us independently and randomly select a set  $T$  of  $m$   $n$ -bit binary strings.
- It can be shown that, the probability that  $S$  has non-empty intersection with  $T$  is more than

$$1 - 2.7^{-km2^{-n}}$$

## 3.6 Additional Notes

### 3.6.1 Design flaw: Using encryption for authenticity

- Encryption is designed to provide confidentiality. It does not necessarily guarantee integrity and authenticity.
- In example, XYZ wants to achieve “authenticity”, but wrongly employed encryption to achieve that. (Some encryption schemes also provide authenticity, but not all.)
- Note that a lot of details are omitted. Simply adding mac to the instructions is not sufficiently secure. To prevent “replay” attack, “nonce” is required. Essentially, problem is about authenticity a communicating party, not just authenticating the data source. Hence a secure implementation would employ TLS (to be covered) first, and then send the instruction via TLS. Secure but less efficient.

### 3.6.2 Password Vs. Secret Key in Crypto

- Passwords are generated by human to be remembered by humans.
- Secret keys (e.g. 128-bit AES encrypt key, 1024-bit RSA key) machine generated, truly random, or derived based on data.
- However, sometimes password used as source for secret key. (E.g. use password to encrypt file).
- Transformation is called **key derivation function**, (KDF)

### 3.6.3 Hash Application: (Password file protection)

- Password file stores userid, password.
- This file could be leaked (insider / accidental leakage, hack). Many well-known incidents where weak/unprotected password files leak, large number of passwords compromise. Desirable to add additional layer of protection on file. (See 2.1).
- **Password file should be “hashed” and “salted”**.
- During authentication, password entered is hashed, and compared with value stored in password file.
- **Salt is important**: If no attacker can just hash a common password, and check to see if it matches any of the 100,000 leaked passwords. Salt (not secret) forces attacker to test one by one.
- Due to one-way property of hash, password file if leaked, is not as disastrous.

## 4. PKI and Channel Security

### PKI and Channel Security Summary

#### Summary & takeaways

##### Part 1: PKI

- PKC requires a "secure broadcast channel" to securely broadcast the public keys: PKI. (know the consequence of having the wrong public key).
- Certificate: a piece of document that binds a "name" to a "public key" & certified by an authority, called CA. A Certificate contains:
  - name, public key, expiry date, usages,
  - meta info (type of crypto, name of CA, etc),
  - CA's signature
- PKI: infrastructure to broadcast the key. Comprise of
  - Certificate Authority (CA)
  - The processes on issuing, verification, revocation of certificates.
  - The mechanism of chain-of-trust. (A root CA can certificate other CA. Root CA's public keys are pre-installed or "manually" installed.)
- The "Public PKI" usually refers to the one for Internet (often simply called "PKI"). "Private PKI" use different sets of CAs. Public PKI's limitations: Too many root CAs.

#### Summary & takeaways

##### Part 2: Channel security

- Protocols: Authentication, Key Exchange, Authenticated Key Exchange.
  - Authentication. *Adversary attack and later impersonate*. (no key exchange. Assuming each entity remains the same throughout the session). Unilateral, Mutual.
  - Key exchange: *Adversary sniffs and wants to steal the session key*. No authentication. (PKC, DH)
  - Authenticated key exchange. *Adversary is Mallory*. (can sniff, spoof, modify, and thus can take over session) and wants to impersonate and/or steal the session key.
- Putting all together. With crypto primitives, we can obtain a secure (*w.r.t. authenticity & confidentiality, and against Mallory*) channel on top of an underlying unsecure public channel.
  - Method:
    - (1) Use *long term key for Authenticated key exchange* to get session key
    - (2) Use *session key* to protect confidentiality (encrypt) & integrity (mac) of subsequent messages via *authenticated encryption*.
  - Why we need a separate session key? (1) More efficient. (2) Forward secrecy.

## 4.1 Public Key Distribution

In order to use digital signature (compared to symmetric key, or unkeyed digest, or MAC, which require some secret key between sender and receiver), we still need a **secure way for receiver to obtain sender's public key**.

### Why require Public Key Distribution?

- While both need secure channel to distribute keys, it is easier to securely "broadcast" public key than to "establish" different symmetric key for every pair.
- With public key distributed, can use for encryption (**confidentiality**) and signature verification (**authenticity**).
- **Public Key:** Entity broadcasts public key to public billboard only once, and no need know existence of receiver while broadcasting.
- **Symmetric Key:** Entity needs different symmetric key per pair, and both entities need to interact to establish key.

### Key Distribution Methods

- Public Announcement on existing mechanism (social media, name card)
- Hard coded in verification programs
- Publish in publicly available directory
- Public Key Infrastructure (PKI)

First few solutions have potential issues. How to verify information submitted is authentic, not everyone trusts the server, server may be overwhelmed, cannot handle load. Hence, **PKI** as a distributed way to broadcast the keys.

## 4.2 Public Key Infrastructure (PKI)

PKI is a standardized system that distribute public keys. To address limitation of the previous methods.

A main objective is to be deployable on a large scale.

- **Two main components: Certificate, and Chain of trust of Certificate Authority (CA).**
- "Public PKI" refer to the PKI we adopted in Internet for domain name, emails address etc. In short, "PKI".
- "Private PKI" are systems for other applications, has own set of "CA". (E.g. possible for company to set up a PKI for own usage).

### 4.2.1 Certificate Authority (CA)

- CA is a trusted authority that manages a directory of public keys, issues certificates. An entity can request to add its public key. CA also has its **own public-private key**.
- Anyone can send queries to search the directory. (Certificates facilitate checking/querying to be done in an offline manner.)
- Assume some CA's public keys securely distributed via other means. (so, we still need a secure channel to distribute the CA's public key, e.g. pre-loaded in OS, known as "root CAs").
- Not all CAs' public keys are preloaded. Other CA's public keys can be added through the chain-of-trust.

### 4.2.2 Certificate

- **Certificate** simply document signed by CA, containing an identity, associated public key, valid time window, and signature of CA. It binds the public key to the identity, and is certified by the signature.
- **Rationale: Certificate for distributing public keys:**

- 'Lazy CA': A certificate is simply a precomputed "reply". CA "signs" message beforehand, and pass to sender, and sender sends this signed message to receiver. This is called the certificate.
- Since no one except CA can produce the valid signature, authenticity of information in certificate as good as coming directly from the CA.
- **A certificate** is a digital document that contains at least the following 4 main items:
  1. The name(s), for e.g. alice@yahoo.com or bbc.com or \*.bbc.com (note that \*.bbc.com is a set of names)
  2. The public key of the owner.
  3. The time window that this certificate is valid. (Expiry)
  4. The signature of the CA

#### • Other important info:

- Usage of certificate (type of name email or domain name, whether name can take role of a CA (chain of trust))
- Digest (Fingerprint), for verification without using CA's public key
- Meta data such as algo involved in verifying signature. (E.g. ECC / RSA, key length)
- Hence, from certificate, receiver can obtain sender's public key, and verify authenticity without connection to CA.
- **Assumption:** Receiver trusts CA.
- **Standard:** E.g. ITU-T X.509 standardisation body, specifies format for certificates, certificate revocation lists, validation algorithm.
- Self-signed certificate by a CA is also called "root-certificate"

### 4.2.3 Certificate Authority & Trust Relationship

- **Responsibility of CA:** CA needs to verify information is correct, which could be costly. Getting certificate signed by reputable CA is not free.
- **Certificate Chain-of-trust:** There might be too many CAs. Root CA (preloaded public key) to CA#1, to CA#2 etc. is called **chain of trust**.

### 4.2.4 Certificate Revocation

- Non-expired certificates may be revoked due to different reasons:
  - Private key compromised, vulnerability in choosing key, admin left organization, Business entity closed, Issuing CA compromised.
- Verifier needs to check if certificate is still valid, although the certificate not expired yet. (conflicting requirement!!)
- **Two different approaches to certificate revocation:**

- **Certificate Revocation List (CRL):** CA periodically signs and publishes a revocation list. Need an online CRL Distribution Point.

- **Online Certificate Status Protocol (OCSP):** OCSP Responder validates a cert in question. Need an online OCSP Responder.

- Recommendation is for a user (e.g. browser) to periodically (say weekly) update its local cache of revocation list. The user does not need to online check whenever the user wants to verify a certificate.

## 4.3 Limitations/Attacks on PKI

### 4.3.1 Implementation Bugs

- Same as many other secure designs, it is vulnerable if not implemented correctly
- **E.g.** Some browsers ignore substrings after null characters when displaying in address bar but include them when verifying certificate. (www.nus.sg\0.hacker123.com → displayed as www.nus.sg). Viewer thought connected via https to a, but in fact connected to b).

### 4.3.2 Abuse by CA

- Many CA's. One could be malicious. Rogue CA can forge any cert.
- **E.g.:** Trustwave issue man-in-middle digital cert, Lenovo's SuperFish scandal.

### 4.3.3 Social Engineering

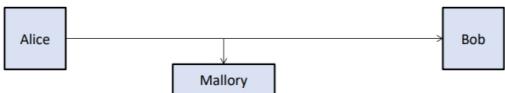
- Social engineering exploits **confusing naming convention**, so that the verifiers are tricked into using a wrong name.
- **E.g.** an attacker first rightfully registered for a domain name that resembles a targeted domain name. Next, used the registered domain to confuse the victim in phishing attack.
- A few techniques to confuse the verifier: "Domain typosquatting", "Homograph attack", using sub-domain name. These attack are aka Domain spoofing, URL spoofing, Fake URL, etc
- **Typosquatting:** Register for domain name (e.g. one letter off), employ phishing attack. Rely on typos made by users when inputting website address, lead to any URL.
- **Sub-Domain:** E.g. rightful owner of 123.com can get sub.domain nus.edu.sg.123.com (\*.123.com)

## Authentication Protocols

Consider 3 types of protocols: (basic) authentication, key-exchange, authenticated key-exchange.

- **Attack Model:** Attack-then-impersonate
- **Types:** Unilateral, Mutual
- **Authentication Method:** Challenge-response

## 4.4. (basic) Authentication Protocol



- **Scenario:** Entity wants to convince Bob that she is indeed Alice. The entity does so by convincing Bob that she knows some "secrets".
- **Attack Model:** Attack-then-impersonate. Attacker (Mallory) can sniff and modify the communication between the authentic Alice and Bob. Attacker uses stolen info to impersonate Alice. Can simply "replay" password sent to impersonate.

## Authentication: Challenge-response

Suppose Alice and Bob have a shared secret  $k$ , both agreed on a message authentication code (MAC). An entity who knows  $k$  is either Alice or Bob. Entity P wants to convince Alice that he is Bob. (P: "Prover").

1. P sends to Alice a hello message.
  2. **Challenge:** Alice randomly picks message  $m$  and sends  $m$  to P.
  3. **Response:** P computes  $t = \text{mac}_k(m)$ . P sends  $t$  to Alice.
  4. Alice verifies tag received is indeed mac of  $m$ . If so, accept, else reject.
- By property of **mac**, even if Eve (third party) sniff communication and obtain multiple pairs of valid  $m, t$ , Eve still (1) can't get the secret key  $k$ , (2) can't forge the mac for messages that Eve has not seen before!
  - Eve can't replay response as challenge randomly chosen, likely different in next authentication session. Challenge  $m$  ensures **freshness** of authentication process. It is also known as the cryptographic **nonce**.
  - This protocol only authenticates Bob. That is, authenticity of Bob is verified. Hence, called **unilateral authentication**. There are also protocols to verify both parties, which are called mutual authentication.

## Authentication: Challenge-response using PKC

Public key version using signature (Unilateral Authentication). Alice wants to authenticate entity P who claims to be Bob.

1. (Challenge) Alice chooses a random message  $r$  and sends to P: ("Bob, here is your challenge",  $r$ )
  2. (Response) P uses his private key to sign  $r$ . P also attaches a certificate: ( $\text{sign}(r)$ , certificate)
  3. Alice verifies certificate belongs to Bob and valid. Next, extracts public key from certificate and verify signature  $\text{sign}(r)$  is correct. If so, accept.
- If Alice already knows Bob's public key, the certificate can be omitted.
  - Assume Attacker observes multiple interactions. By security property of signature, eavesdropper cannot derive Bob's private key, can't forge response.
  - Nonce  $r$  ensure freshness.

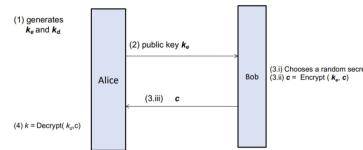
## 4.5 Key-exchange

Two communicating entities want to establish a shared key between them. Consider attack Eve who can sniff, goal to steal established key.

### PKC-based Key-exchange

#### PKC-based Key-exchange

- Here is a key-exchange that uses a PKC.
1. Alice generates a pair of private/public key.
  2. Alice sends the public key  $k_e$  to Bob.
  3. Bob carries out the following
    - i. Randomly chooses a secret  $k$ .
    - ii. Encrypts  $k$  using  $k_e$ .
    - iii. Sends the ciphertext  $c$  to Alice.
  4. Alice uses her private key  $k_d$  to decrypt and obtain  $k$ .

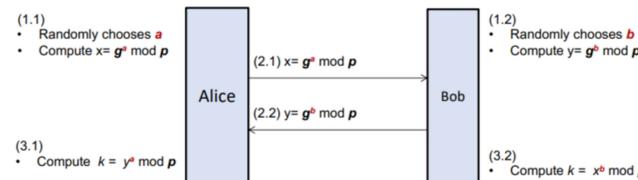


- Attacker (Eve) can only obtain the public key  $k_e$  and the ciphertext  $c$ .
- By security of PKC, from the public and ciphertext, attacker can't get any information of the plaintext, which is the key  $k$ .

### Diffie-Hellman Key-exchange

#### 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.

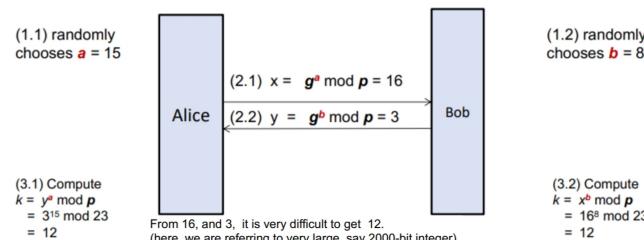
*Computational Diffie-Hellman CDH assumption:*  
Given  $g, p, x = g^a \text{ mod } p, y = g^b \text{ mod } p$ , it is computationally infeasible to find  $k = g^{ab} \text{ mod } p$ .

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.

### Diffie-Hellman Key-exchange Example

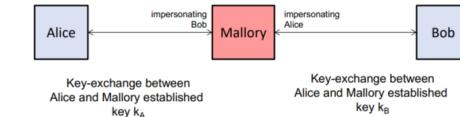
Eg.  $g = 2, p = 23$



## 4.6 Authenticated Key-exchange Protocol

Consider **malicious Mallory**, who can sniff, spoof and modify message.

- Mallory first allows Alice and Bob carry out strong authentication. After, Mallory interrupts and takes over channel. Mallory pretends to be Bob! Even if employ challenge-response, Mallory still succeeds.



- **Authenticated Key-Exchange:** Protect subsequent interactions.

1. Authenticated key-exchange, outcome new shared secret session key  $k$ .
2. All subsequent communication protected (encrypted + mac) using  $k$ .

#### Summary: Mutual Authenticated key exchange

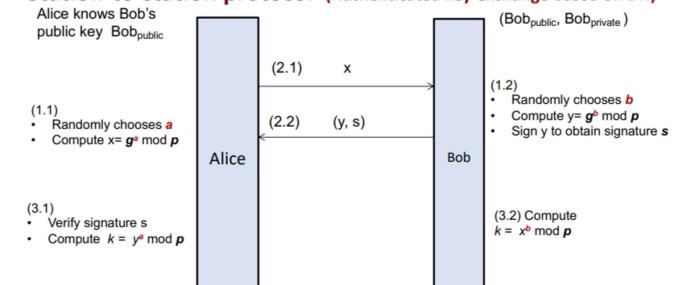
- *Before the protocol:*
  1. Alice has a pair of public, private key ( $A_{\text{public}}, A_{\text{private}}$ ).
  2. Bob has a pair of public, private key ( $B_{\text{public}}, B_{\text{private}}$ ).
  3. Alice knows Bob public key and vice versa. These two sets of keys are known as the **Long-term 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:*
  1. Both A and B obtain a shared key  $k$ , known as the **Session key**.
- Security Requirement.
  1. (Authenticity) Alice is assured that she is communicating with an entity who knows  $B_{\text{private}}$ .
  2. (Authenticity) Bob is assured that he is communicating with an entity who knows  $A_{\text{private}}$ .
  3. (confidentiality) Attacker unable to get the session key.

- Turns out authenticated key-exchange easily obtained from existing key-exchange, (either PKC-based key-exchange or DH-based key-exchange). Done by signing all communication using the private key.
- Special name for authenticated key-exchange that uses DH: it is known as **Station-To-Station Protocol (STS)**.

## Station-to-Station Protocol (based on DH)

Assume both Alice and Bob agreed on two public parameters, generator  $g$  and large (e.g. 1000 bits) prime  $p$ . Both  $g$  and  $p$  are not secret and known to the public. Consider unilateral authentication. Alice want to authenticate Bob. Can be easily extended to mutual authentication.

#### Station-to-station protocol (Authenticated key-exchange based on DH)

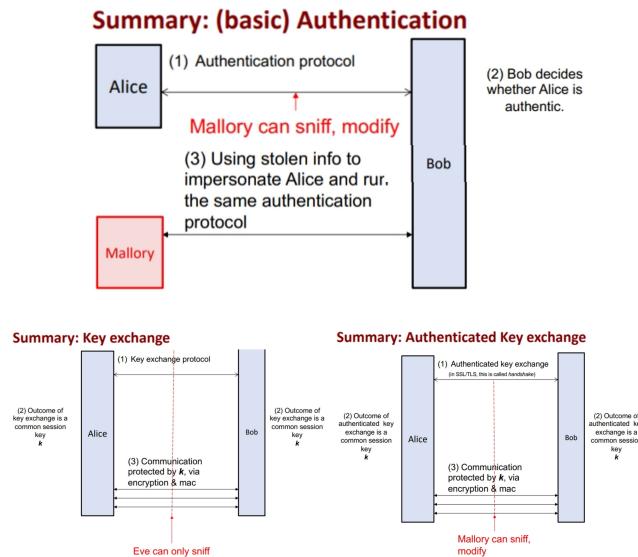


Remark: This is unilateral authentication. Can extend it to mutual by making Alice signs her messages in step (2.1).

## Password based Encrypted Key Exchange

(Asymmetric) Previous authenticated key-exchange protocols such as Station-to-station are based on 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, thus potentially vulnerable to “offline” dictionary attack (see tutorial). There are secure protocols that, even if entropy of password low, still secure against offline dictionary attack (to guess password correct, attacker forced to interact with online server). These are called “**Password-Authenticated Key agreement**” (PAKE).



## 4.7 Securing Communication Channel (Ensuring Channel Security)

Given a “public channel” that facilitates communication, but presences of Mallory. Use unsecure public communication as carrier, add layer of crypto primitives on the messages, so channel is as secure as a “private channel”.

- **Scenario:** Alice wants to visit website Bob.com. Alice uses free wifi called Mallory. Everyone can access, thus public channel. Secure public channel using crypto.

TLS/SSL secure the public channel in this way: ([https://www.ibm.com/support/knowledgecenter/en/SSFKSJ\\_7.1.0/com.ibm.mq.doc/sy10660.htm](https://www.ibm.com/support/knowledgecenter/en/SSFKSJ_7.1.0/com.ibm.mq.doc/sy10660.htm))

- (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**.
- (2) Subsequent communication protected by the session key.

## TLS: Transport Layer Security

**Alice wants to visit Bob . com: How TLS does it.**

**[Step 0]** Alice obtains Bob . com's public key securely. This is done by having Bob sending his certificate to Alice.

**[Step 1]** 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 a shared key, which could come in the form of 2-tuple  $(t, k)$  where  $t$  is the secret key of the MAC, and  $k$  is the secret key of the symmetric-key encryption, or a single key  $k$  when authenticated encryption (e.g. GCM) is used. These keys are called the **session keys**. By property the protocol, Alice is convinced that only Bob and herself know the session key. Here (unilateral authentication), Bob doesn't care about Alice's authenticity.

**[Step 2]** Subsequent interactions between Alice and Bob . com will be protected by the session keys and a sequence number. Suppose  $m_1, m_2, m_3, \dots$  are the sequence of message exchanged, the actual data to be sent for  $m_i$  is

$$E_k(i \parallel m) \parallel mac_t(E_k(i \parallel m))$$

where  $i$  is the sequence number.

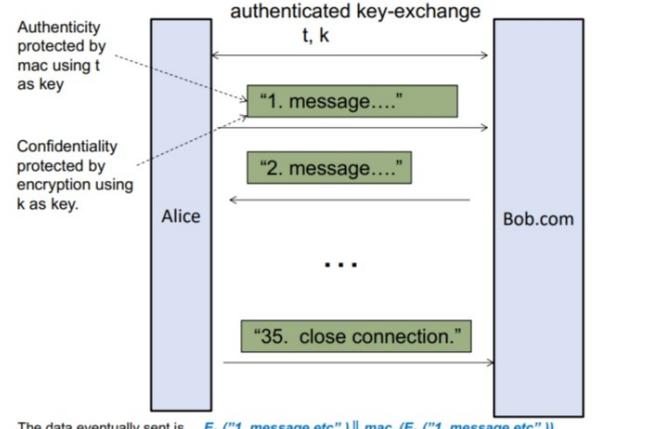
For GCM mode or other authenticated encryption, the message to be sent is simply

$$E_k(i \parallel m)$$

•  $\parallel$  refer to string concatenation.

• The above 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.

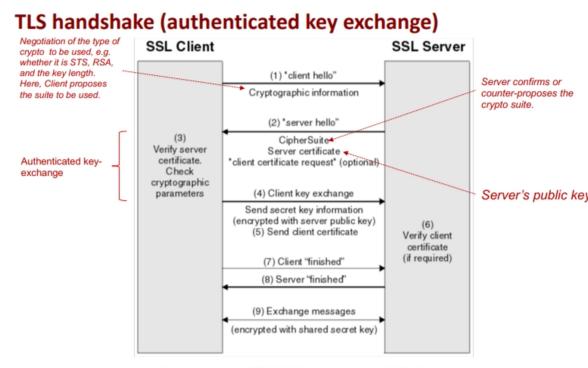


**Question:** What is the role of the sequence number 1,2,3, ... and the last message “close connection”?

Role of sequence number and last message is to make sure mallory is not able to add messages in between, or after the connection is closed.

### Relationship among TLS/SSL/https

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



## 5. Network Security

Cryptography and PKI establishes end-2-end security (w.r.t. Confidentiality & authenticity) even in the presence of MITM. However, there are other issues in Networking, where Availability not addressed and Routing needs protection. We want to mitigate MITM as much as possible (e.g. concerned on implementation flaws, side-channel leakage, unprotected channels, etc.).

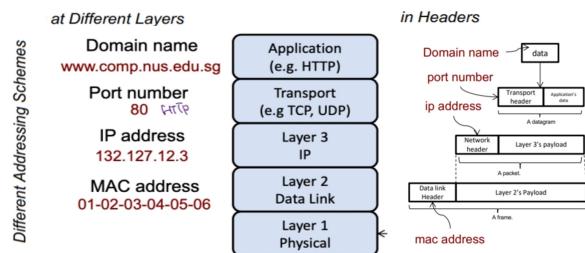
- **Routing:** Layering. Intermediate nodes need to see and modify routing info at different “layers”. Protection at different layers. MITM in different layers
- **Monitoring and segmentation:** Firewall, Intrusion Detection, DMZ, Server’s zone.
- **Specific Attacks:** Name resolution attacks (poisoning, spoofing, ARP, DNS), Flaw in protocol (e.g. TLS renegotiation attack), Port Scanning, DDOS, botnets.
- **Popular Protocols:** SSL/TLS (application), IPSEC (network), WPA (link)
- What does padlock, https in browser mean? What can a MITM (in link layer) get? What can an attacker obtain via DNS spoofing?
- Tools: Wireshark, nmap.

## 5.1 Background on Networking

### 5.1.1 Brief Overview

Computer Network establishes communication between large number of nodes. For robustness and resource sharing, use packet switching.

- **Network Security:** Attackers among intermediate nodes. Steal, modify (traffic routed to wrong places), disrupt. Guess who is talking to who.
- **Multiple hops:** At each intermediate node, routing data read and modified.
- **Network Layers:** Partitions complex communication system into several abstraction layers. E.g. view layer-N protocol built upon a virtual connection at layer N-1. Passes down the message, level below abstracted.
- **Ports:** Each node has total 65535 ports. If no process “listening” port to process data, port is “closed”.
- **Addressing Schemes (different Layers):** A single node has different names in different layers.



### 5.1.2 MITM: Man-In-The-Middle

A MITM sits in-between two 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 which part of the header the MITM has access to.

- **MITM in Layer x:** we mean a MITM along the virtual connection in layer x. The MITM can see and modify data unit in that layer. For IP layer, the MITM has access to the IP packet, including the header and payload.
- E.g. Evil cafe owner free wifi can MITM in Data Link layer, sniff, spoof, modify, drop. Anyone in cafe tapping in wirelessly is MITM in physical layer who can sniff, spoof, but not drop, nor modify.

## 5.2 Name Resolution and Attacks

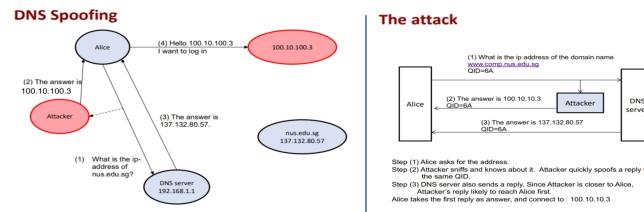
Each peer entity has a name. A single node may have different “name” at different layer. Initial design of resolution protocols did not take security into account, easy for attackers to manipulate outcome.

- **Domain Name System (DNS):** maps domain names to their respective IP addresses. It uses a hierarchical decentralized naming system. An attacker can thus target the association of the domain name with the IP address.
- **Address Resolution Protocol (ARP):** associates or maps IP addresses (logical addresses) with MAC addresses (physical addresses). Uses broadcast mechanism on a local network. An attacker on the local network can target the association.

### Domain Name System (DNS) Attack

**Name Resolution:** Given a domain name (e.g. www.comp.nus.edu.sg), its IP address found by looking up locally stored host table, or querying DNS server.

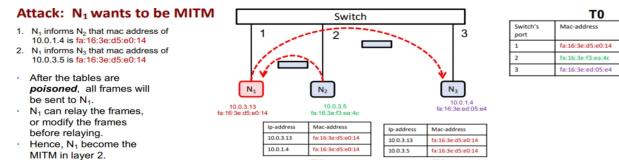
- The entity (aka client) initiating query (UDP protocol) is resolver. If address found, domain name is resolved.
- Each query contains 16-bit number, Query ID (QID). Response from name server must contain QID. If QID in response no match query QID, client rejects answer.
- Note no encryption or MAC involved, QID was probably not meant for authentication, but as efficient way to match multiple queries.
- **DNS server resolves domain name**, can be the “single-point-of-failure” for network.
- A DoS attacks, instead of attacking a Web server, could attack the DNS server instead.



### Poisoning attack on ARP table

A switch connects a few nodes. (Note that switch handles mac-addresses, router handles ip-addresses.) E.g. our home wifi “router”, although often called as a “router”, it also functions as a switch. It contains a gateway which performs routing.

- Switch connect ports based on mac-addresses, has ARP table associating port to mac-addresses.
- Resolution of IP to MAC address done by nodes (e.g. phone) with own ARP table. Nodes update each others (ARP query mechanism).
- **ARP poisoning** is an attack that modifies (aka “poisons”) the tables so as to gain MITM access.



## 5.3 Denial of Service Attack

DoS attacks target **availability**, preventing some service from being accessible and usable upon demand by an authorised entity, delaying time-critical operations. Many successful DOS attacks simply flood the victims with overwhelming requests/data. When DOS carried out by large number of attackers, **DDoS: Distributed Denial of Service**.

### Reflection and Amplification attack

- **Reflection:** Reflection attack type of DOS where attackers send requests to intermediate nodes, which in turn send overwhelming traffic to the victim.
- Indirect, more difficult to trace.
- **Amplification:** Reflection 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.

### Types of DoS Attacks

	Stopping Service	Exhausting Resources
Local Attack (Easily detected and punished/mitigated)	Process killing Process crashing System reconfiguring	Spawning processes Filling up the file system
Remote Attack (over the Internet)	Sending malformed packet attacks Requires vulnerabilities, which are easily patched up	Packet flooding, as the system cannot tell if the request is valid or invalid

### DoS Example 1: ICMP/Smurf Flood Attack

“smurf”: Attack can bring down big targets. Attack makes use of public servers to target a specific victim IP address. This is done via:

1. An attacker sends the “ICMP PING” request to router, instructing router to broadcast request.
2. The request’s source IP address spoofed, replaced with victim IP address.
3. The router thus broadcasts this request.
4. Every entity that receives request replies, sends “Echo reply” to source, which is the victim.
5. The victim is overwhelmed with “Echo reply’s” from the entire network.
6. The attacker thus takes advantage of the amplification effect to attack the victim.

- Attack no longer effective, as most routers are now configured to not broadcast the requests. To prevent this attack, the measure is to simply disable a feature that was previously thought to be useful.

### DoS Example 2: Application-Layer DoS Attack (HTTP Get)

The trick is to simply flood a web server with HTTP requests. Effective if large number of attackers, since each attacker can only send requests at a low rate. Distributed Denial of Service (DDoS).

- DDoS is normally achieved via a **botnet**.
- A bot, or zombie, is a compromised machine, and a botnet, or zombie army, is a large collection of connected bots, communicating via covert channels.
- **Covert channels:** to prevent the owner of the zombie computer from noticing. Owners unaware system used.
- **A botnet has a command-and-control mechanism**, can be controlled by a single individual to carry out a Distributed Denial of Service attack.
- **Other Botnet Usages:** Vulnerability Scanning, Anonymising HTTP Proxy, Email Address Harvesting, Cipher Breaking etc.

## 5.4 Useful Tools

### Wireshark (packets analyzer)

Wireshark is a popular, free and open-source network packet analyser. It generally performs capturing at the link layer. This depends on the operating system and hardware, however. It essentially captures “interactions” between the operating system and the network card driver.

Some things that Wireshark can do: View list of packets, View packet details, View packet bytes, Filter for packets, Follow TCP stream.

### Nmap (Port Scanner)

A port helps a server decide which application process to handle an incoming packet. By saying that a process or service is “listening” to a particular port, we mean that the process is running and ready to process packets with that particular port number. We also say a port is “open” when there exists such a process running in the server.

- **Port scanning:** process of determining which ports are open on hosts in a network.
- **Port scanner:** tool to perform port scanning. Useful for both attackers and network administrators to scan for vulnerabilities. Nmap is a very popular port scanner.
- **Nmap** is a full featured port-scanning tool: Command-line tool with a GUI frontend.
- Installation: sudo apt-get install nmap, zenmap
- Usage: nmap [ScanType(s)] [Options] {target specification}
- Examples: TCP ACK scan (a stealthier scan): nmap -sA, OS fingerprinting: nmap -O, Service/version detection: nmap -sV.

Well-known port numbers:	
- 1: TCP Port Service Multiplexer	- 43: WHOIS Protocol
- 7: Echo Protocol	- 53: Domain Name System (DNS)
- 17: Day of the Week	- 80: Hypertext Transfer Protocol (HTTP)
- 19: Character Generator Protocol (CHARGEN)	- 220: Internet Message Access Protocol (IMAP), version 3
- 20: File Transfer Protocol (FTP) data transfer	- 443: Hypertext Transfer Protocol over TLS/SSL (HTTPS)
- 21: File Transfer Protocol (FTP) control	- 465: Authenticated SMTP over TLS/SSL (SMTPS)
- 22: Secure Shell (SSH), secure logins, file transfers (scp, sftp) and port forwarding	- 515: Line Printer Daemon (LDP), print service
- 23: Simple Mail Transfer Protocol (SMTP), used for email routing between mail - servers	- 666: Doom, first online first-person shooter

## 5.5 Network Protection: Cryptography

There are several cryptographic techniques that can help us achieve confidentiality (via encryption) and authenticity (via MAC, PKI, and Strong Authentication) over a public communication channel, even if the adversary can sniff and spoof the data.

There are various security protocols that essentially achieve that, but operates at different layers. Some well-known protocols are: TLS/SSL, WIFI Protected Access II (WPA2), Internet Protocol Security (IPsec).

### Note: Discussion of Security protocols and attacks

Security protocol considers the layer it aims to protect.

- **Complication:** Some protections span across multiple layers, or do not provide full protection of the targeted layer.
- When analysing an attack, insightful to figure out at which layer the attacker resides or if attacks span across multiple layers. Hard to pinpoint the layer.
- **General Guideline:** A security protocol that protects layer k would protect information from that layer and above against an attacker sitting at layer k-1 and below.
- E.g. if attacker resides at layer 1, security protocol protects layer 3. Security protocol protects information generated in layer 3 and above, but not the information generated in layer 2.

### 5.5.1 Secure Sockets Layer / Transport Layer Security (SSL/TLS)

The SSL/TLS sits on top of the transport layer. Application passes data, destination IP address to SSL/TLS. SSL/TLS protects the data using encryption (for confidentiality) and MAC (for authenticity), instructs transport layer to send protected data.

- **End-to-end Encryption:** End-to-end encryption performed. Data stays encrypted from one end of its journey to the other.
- **Link Encryption / Hop-by-Hop Encryption.** Encryption occurs at the Data Link and Physical layers, and the packet is decrypted at every device between the two ends.

#### Attack Scenario 1: Attacker at the Physical Layer

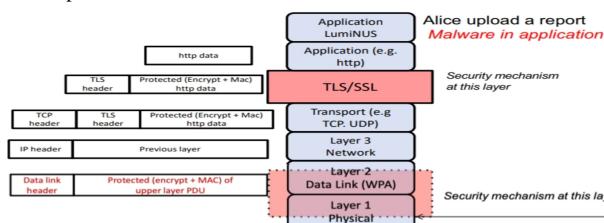
Suppose attacker at physical layer, can sniff/spoof message at layer. Alice uploads her report in café using free and open WIFI, no WPA protection. Anyone in café has access to physical layer.

- **Cannot learn application data:** protected by SSL/TLS in application layer, protected from attacker in lower layers
- **Can learn website IP address:** IP headers from network layer, not protected by TLS/SSL (on transport layer) from attacker below at data link/physical layer.

#### Attack Scenario 2: Attacker at the Application Layer

Adversary at the application layer. E.g. malicious JavaScript code injected into LumiNUS, executed by Alice’s computer.

- **Can learn/spoof application data:** SSL/TLS cannot protect info from attacker at its layer and above layer.
- **Cannot learn website IP address:** MAC address defined at layer 2. Attacker at application layer unable to attack “downwards”, only can attack upwards.



### 5.5.2 WIFI Protected Access II (WPA2)

WPA2: popular protocol, common in home WIFI access points, more secure than Wired Equivalent Privacy (WEP) (broken), and WPA. Provides protection at layer 2 (Link) and layer 1 (Physical). However, not all information in layer 2 are protected.

#### Attack Scenario: Attacker at the Physical Layer

Attacker at the physical layer who is able to sniff and spoof information, and Alice uploads a report.

- **Cannot learn application data / website IP address:** Data from the upper layers have been encrypted with AES.
- **Can learn MAC address:** MAC addresses assigned in the link layer, and is never encrypted.

Anyone within the range of network might see traffic, but scrambled with up-to-date encryption standards. WPA2 uses AES and keys that are 64 hexadecimal digits long. WPA3 was announced in January 2018.

### 5.5.3 Internet Protocol Security (IPsec)

IPsec provides **integrity** and **authenticity** protection of IP addresses, but not confidentiality. Attackers unable to “spoof” source IP addresses, but can learn source and destination IP addresses of sniffed packets. Mechanism whose goal is to protect the IP layer.

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.

- 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.
- Uses cryptographic security services to protect communications over Internet Protocol (IP) networks.
- 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.

#### Which Layer to Protect?

Seems protecting lowest layer, can protect information in all layer. However, not feasible as intermediate node needs to access some higher layer info, e.g. ip-address, and sit in higher layer. Hence, a malicious intermediate node could be a MITM in higher layer.

- Typical implementation use TLS/SSL + WPA2.
- IPSEC most expensive to implement, difficult to deploy. (“touches” OS).

## 5.6 Network Protection: Firewall, Intrusion Detection System

## 5.7 Network Security Management

There is a need to continuously monitor and adjust network characteristics. Best Practices:

- **Security Operations Center (SOC):** Centralised unit in an organisation that monitors the IT systems and deals with security issues.
- **Security Information and Event Management (SIEM) “SIM”:** Approach that aims to provide real-time analysis of security alerts generated by network hardware and network applications. May include capabilities of Data aggregation and correlation, - Event alerting, Compliance report generation, Forensic analysis.