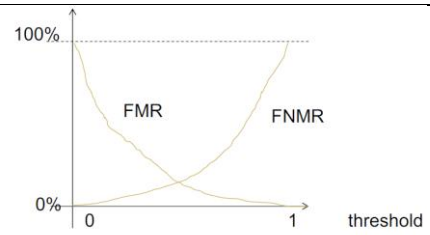
Important Formulas

Birthday Attack > 0.5	$M > 1.17 \times \sqrt{T}$ T → # total possible outcomes M → # attempts/counts
Find Probability, Generalized * Poisson Distribution * Only for collisions within 1 set	$1 - e^{\frac{-M^2}{2T}}$ T → # total possible outcomes i.e $2^{\#bits}$ M →# attempts/counts
P(Success on single trial) *Binomial Distribution	$1 - (1-p)^q$ P → probability of success Q → # attempts
Birthday Attack Variant 2 Pool attack (e.g send n spoof DNS request and n spoof replies to target DNS server) E.g digest & hashes Approx. 0.63	*Let $\mathcal{S}$ 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 $\mathcal{T}$ of m n-bit binary strings. It can be shown that, the probability that $\mathcal{S}$ has non-empty intersection with $\mathcal{T}$ is more than* $1 - 2.7^{-km2^{-n}}$ $1 - 2.7^{-km2^{-n}}$ K → number of distinct elements (each element is a n bits binary string) M → # attempts ** Need to satisfy km = 2^n → $km/2^n = 1$ quick approximation to find m simply by setting $k \cdot m = 2^n$ Usually, we find m → answer, cus we are calculating the number of attempts

	accept	reject
genuine attempt	A	C
false attempt	B	D



how to set the threshold? Depend on application.

lower threshold => more relax in accepting, higher threshold => more stringent in accepting

False Match Rate (FMR)	$FMR = \frac{\#\ successful\ false\ matches(B)}{\#\ attempted\ false\ matches\ (B+D)}$
False NonMatch Rate (FNMR)	$FNMR = \frac{\#\ rejected\ genuine\ matches(C)}{\#\ attempted\ genuine\ matches\ (C+A)}$

When key length for symmetric is 112, the corresponding recommended length for digest is at least 224

QNS	Why the digest length is twice larger?
ANS	Birthday Attacks. • Hash requirement is always x2 regular requirements. • This is because for birthday attack the $T^{1/2}$ component • *** For same level of entropy/level of security as a X bit symmetric key, the hash digest has to be 2X bits. This is to account for birthday attacks.

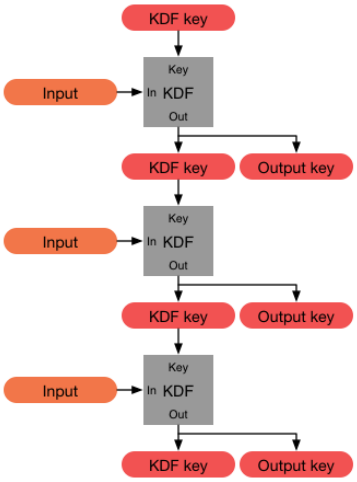
Conversions	**n-Factor Authentication (2FA) and Multi-Step Verification**
• 2^{10} = kb • 2^{20} = Mb • 2^{30} = Gb	• At least 2 different authentication factors used: 1) Something you know : Password, Pin 2) Something you have : Security Token, Smart Card, Phone, ATM card 3) Who you are : Biometrics

Tools/Commands	Purpose	Involved Protocol:
Wireshark	• Packet analyser • Listens to interactions between OS and network driver - It is a MITM between OS and network card. • Note: The header added by the network OR modifications made by the network card may not be captured by Wireshark.	
Nmap	• Used for port scanning. → determine which ports are open in the network. • Test which ports are open/closed.	
dig	Performs a DNS query	DNS
nslookup	Similar to`dig`, performs DNS query	DNS
telnet	Connects to a server on port	Telnet
ping	Uses ICMP to check reachability of host ** does NOT cause a DNS query to be issued	ICMP
traceroute (or tractrt)	Traces the route of the packets take to reach the host	ICMP (for probes)
Ifconfig/ dnscherker.org	Check MAC	

Password Strength

Entropy per symbol for different symbol sets $\log_2(N)$

Symbol set	Symbol count *N*	Entropy per symbol *H*
Arabic numerals (0–9) (e.g. PIN)	10	3.322 bits
hexadecimal numerals (0–9, A–F) (e.g. WEP keys)	16	4.000 bits
Case insensitive Latin alphabet (a–z or A–Z)	26	4.700 bits
Case insensitive alphanumeric (a–z or A–Z, 0–9)	36	5.170 bits
Case sensitive Latin alphabet (a–z, A–Z)	52	5.700 bits
Case sensitive alphanumeric (a–z, A–Z, 0–9)	62	5.954 bits
All ASCII printable characters except space	94	6.555 bits
All ASCII printable characters	95	6.570 bits
All extended ASCII printable characters	218	7.768 bits
Binary (0–255 or 8 bits or 1 byte)	256	8.000 bits
Diceware word list	7776	12.925 bits



- Online: At least 29/30 bits entropy
- Offline: At least 96/128 bits entropy
  - Using Key Derivation Function (KDF), offline dictionary attacks will be more difficult, and 96 bits will be sufficient.
  - The KDF forces intensive computation but causing delay during legitimate usage.
    - ➢ Passwords are hashed X times → Slows down speed of exhaustive search by X
- Alphanumeric = 62 symbols total.
  - A password with 10 alphanumeric characters = $\log_2 62^{10}$ = 59.5 bits of entropy

Date	Minimum security level (in bits)	Symmetric algorithm	RSA key size (in bits)
2010 (Legacy)	80	3DES with 2 keys	1,024
2011–2030	112	3DES with 3 keys	2,048
> 2030	128	AES-128	3,072
>> 2030	192	AES-192	7,680
>>> 2030	256	AES-256	15,360

- To mitigate exhaustive search on the password, it is common to deploy a hash function that is intentionally designed to be very slow. Such hash function is aka Key Derivation Function (KDF)
- Most wifi access point deploy WPA2 to secure the wireless connection. There are variants of WPA2 that uses LEAP or PEAP (LEAP is vulnerable to offline attack and PEAP is secure against offline attacks).
  - Fortunately, NUS uses PEAP.

- To prevent the offline attack, PEAP first makes sure that the server is authentic, and then all communication is protected by a key generated by "authenticated key-exchange" .
- The password (or hash of it) can then next send via the secured channel. For this, we need the client to know the server's public key, and thus the need of "certificate".

Encryption:

Ciphertext only Attack	• The adversary is given a collection of ciphertext c. • The adversary may know some properties of the plaintext, for e.g. the plaintext is an English sentence. (adversary can't choose the plaintext).
Known Plaintext Attack	• The adversary is given a collection of plaintext m and their corresponding ciphertext c. (assume that adversary can't choose the plaintext)
Chosen Plaintext Attack (CPA)	• The adversary has access to an oracle. • He can choose and feed any plaintext m to the oracle and obtain the corresponding ciphertext c (all encrypt with the same key). • He can see the ciphertext and then choose the next input. • We call this black-box an encryption oracle.
Chosen Ciphertext Attack (CCA2)	• Similar to chosen plaintext attack, but here, the adversary chooses the ciphertext and the blackbox outputs the plaintext. • We call the black-box a decryption oracle. • In some practical scenario, the attacker indeed has some but not full decryption capability, e.g. padding oracle. • Because we must cater for all scenarios, in formulation and design of encryption, we consider oracle with full decryption capability.

- In public key cryptography, the encryption oracle is always available.

Adversary's Goals:

Total Break	If an attacker wants to find the key, we call this goal total break.
Partial Break	• The attacker may satisfy with a partial break. • There are a few definitions for that. For instance, the adversary may want to decrypt a ciphertext (but not interested in knowing the secret key), or the adversary may want to determine some coarse information about the plaintext (e.g., whether the plaintext is a jpeg image or a C program).
Indistinguishability	• Indistinguishability (IND) The attacker may satisfy with distinguishability of ciphertext with some "non-negligible" probability more than ½, the attacker is able to distinguish the ciphertexts of a given plaintext (say, "Y") from the ciphertext of another given plaintext (say, "N"). • Precise notion of distinguishability not required in this course. • I.e The attacker is unable to distinguish the ciphertext of any 2 plaintexts. (OR unable to distinguish the ciphertext from a randomly sequenced text)

- We want to design a "strongest" cryptosystem that can prevent attacker's "weakest" goal.

Ciphers

- Substitution Cipher (NOT SECURE)
  - Frequency analysis attack
  - Known-plaintext attack
  - Ciphertext only attack
- Permutation Cipher (NOT SECURE)
  - Known-plaintext attack
- One Time Pad
  - Leaks no information of the plaintext, even if there is infinite time to bruteforce.
  - Exhaustive search does not work with OTP.
  - Length of key = Length of plaintext (impractical in most situations)
- For a cipher to be secure, exhaustive search (brute force) must be computationally infeasible.
- Note: Modern Ciphers (AES, SEC, RC4, GSM) uses a combination/series of substitution and permutation ciphers to achieve security. (S-box, P-box)

For secure key:

Symmetric Key Algorithm	• AES (Advanced Encryption Standard): Common key lengths are **128**, **192**, and **256** bits. • 128 bits is considered secure for most applications, but 256 bits provides an extra margin of security.
Asymmetric Key Algorithm	• RSA (Rivest–Shamir–Adleman): Common key lengths range from **1024** bits to **4096** bits. As of the current best practices, key lengths of 2048 bits or higher are recommended for most applications. • ECC (Elliptic Curve Cryptography): Key lengths are typically much shorter compared to RSA for equivalent security. • For example, a **256**-bit ECC key is considered roughly equivalent in security to a **3072**-bit RSA key.

Hash (B): HMAC, KMAC, key derivation functions and random bit generation.

Date	Security Strength	Symmetric Algorithms	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash (A)	Hash (B)
Legacy (1)	80	2TDEA	1024	160	1024	160	SHA-1 (2)	
2019 - 2030	112	(3TDEA) (3) AES-128	2048	224	2048	224	SHA-224 SHA-512/224 SHA3-224	
2019 - 2030 & beyond	128	AES-128	3072	256	3072	256	SHA-256 SHA-512/256 SHA3-256	SHA-1 KMAC128
2019 - 2030 & beyond	192	AES-192	7680	384	7680	384	SHA-384 SHA3-384	SHA-224 SHA-512/224 SHA3-224
2019 - 2030 & beyond	256	AES-256	15360	512	15360	512	SHA-512 SHA3-512	SHA-256 SHA-512/256 SHA-384 SHA-512 SHA3-256 SHA3-384 SHA3-512 KMAC256

All key sizes are provided in bits. These are the minimal sizes for security.

Summary of 4 main Mode of Operations (For CS2107)

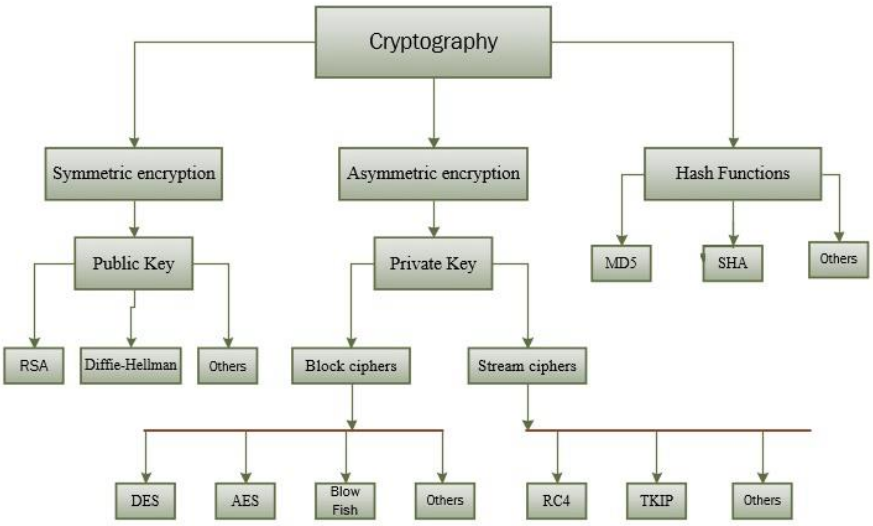
Mode	Description	Pros	Cons
Electronic Code Book (**ECB**)	Divides the plaintext into blocks and then applies the block cipher to each block, **all using the SAME KEY**. Block Cipher	• Simple and parallelizable. • No error propagation	• All blocks uses the same key. • ECB is not secure and leaks information. • ECB is deterministic. • No IV involved. DON'T USE
Cipher Block Chaining (**CBC**)	Each block of plaintext is XORed with the previous ciphertext block before encryption Block Cipher • Used for Authentication	• Provides diffusion (disrupts patterns). Suitable for variable-length messages. • IV can be randomly chosen → non-deterministic. • Secure against CPA.	• Sequential processing (not fully parallelizable). • Sensitive to bit errors, as a change in one block affects subsequent blocks. • Vulnerable against padding oracle, BEAST attack. • Has error propagation.
Counter (**CTR**)	• Uses a counter as a nonce, and the block cipher encrypts the counter value. • Stream Cipher • Typically used in high-speed systems such as IPSec, ATM.	• Fully parallelizable, enabling efficient encryption/decryption of large data sets. • Secure against CPA • Uses IV. • No error propagation	• Requires a unique counter value for each block, and must not reuse counters with the same key. • Vulnerable against padding oracle, if padded. • No integrity and easily changed (Malleable)
Galois/Counter Mode (**GCM**)	• Combines Counter mode with Galois field multiplication.	• Provides confidentiality, integrity, (and authentication). • Efficient and parallelizable. • Secure against CCA2 • Secure even with decryption oracle.	• Requires unique IVs (Initialization Vectors) for each key. • Sensitive to nonce reuse (can compromise security).

- If the cryptoalgo is secure against CCA2, then it is probable non-malleable.
- For CS2107, take it that only GCM is non malleable, the others are malleable.

If we combine these modes of operation with MAC, then they will be non-malleable. → Attacker will not be able to know what the ciphertext is and hence can predictable modify the ciphertext to affect the plaintext. (Property of MAC)

Stream Cipher vs Block Cipher

	Stream Cipher	Block Cipher
Operation	Encrypts data bit by bit or byte by byte Stream Cipher = Plan text bit + Pseudorandom bits (However, it is hard to decipher which is plaintext/ random bits without the key)	Encrypts data in fixed-sized blocks
Encryption Seed	Faster, especially for streams of data	Slower, especially for short messages
Key Usage	Uses a key as input to pseudorandom generator to produce a key stream	Operates on fixed-size blocks of plaintext using a key
Error Propagation	Errors in transmission affect only the corresponding bits in the keystream.	Errors can propagate to multiple blocks, affecting more of the ciphertext.
Security	• Susceptible to certain attacks like known-plaintext attacks if key reuse occurs. • Low Diffusion • More susceptible to malleability attacks, where an attacker can modify the ciphertext in a predictable way by altering the plain/ciphertext • Stream cipher does not provide authenticity.	• Generally considered more secure due to the complexity of the encryption algorithm and the block chaining methods used. • High Diffusion • Less susceptible to malleability attacks due to fixed-sized blocks and chaining modes used. However, ECB is vulnerable.s
Memory Usage	Requires less memory as it processes data bit by bit.	May require more memory due to the need to store entire blocks during encryption/decryption.
Use Cases	Often used in applications where a continuous flow of data needs to be encrypted/decrypted in real-time, like streaming media.	Suitable for encrypting fixed-size blocks of data, such as files or messages.



** Generally: Plaintext length = Ciphertext Length – Padding Length

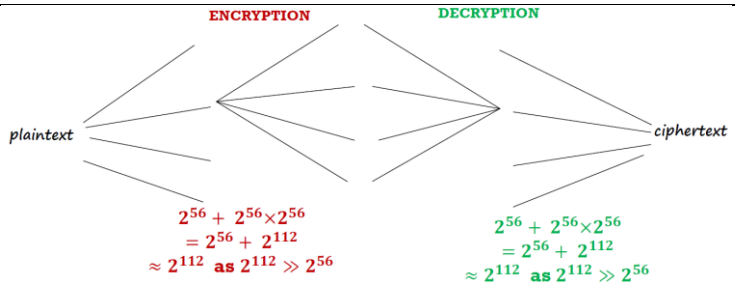
	ECB	CBC	CTR	GCM
Ciphertext Only	❌	✅	✅	✅
Known Plaintext	❌	✅	✅ *	✅
CPA	❌	✅	✅	✅
CCA2	❌	❌	❌	✅

* For CTR mode, using the same counter value (IV) with known plaintext can leak information. Use a unique IV for each encryption.

Encryption

- Encryption is designed to provide confidentiality
  - Encryption MIGHT NOT achieve authenticity
  - GCM → Achieves confidentiality + Integrity
  - DON'T Encrypt + MAC (i.e don't reinvent GCM → leaks data)
- Substitution & Permutation Ciphers are NOT secure
- ECB → Lacks Diffusion
- CDC mode of operation → High Diffusion
- Substitution Cipher. Given n chars → key space = n! (e.g not case sensitive alphabets = 27!)
  - A substitution cipher being carried out consecutively is still equivalent to a substitution cipher carried out once. Its just a different mapping.
  - Substitution Cipher insecure under known-plaintext attack and ciphertext-only attack
- Given n bits key length, key space = 2^n
- CBC Mode of Operation:
  - CBC encryption CANNOT run in parallel.
    - The encryption at round i to produce the ciphertext body blocks y_i take it in as its input the ciphertext block y_{i-1} that is generated only in the previous round i-1.
  - CBC decryption CAN run in parallel
    - The decryption at round I to recover the plaintext block, x_i, depends only on the ciphertext blocks y_i and y_{i-1} which are both readily available from the sent ciphertext.
- Stream Cipher fails badly when IV is the same
  - Same IV & same secret key → Same pseudorandom sequence.
  - "malleability" refer a property of an encryption scheme whereby it is possible to transform (without knowledge of the secret key) a ciphertext to another ciphertext of a related plaintext.
  - Stream cipher is malleable
  - Streram cipher does not provide authenticity.
- Compression:
  - Compression algorithm exploits repeating patterns in the bits.
  - Encrypted file resembles a random sequence
  - It is difficult to find patterns for compression if we encrypt then compress.
  - Hence, it is better to compress then encrypt.

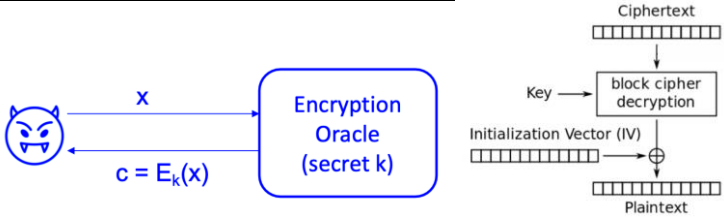
Attacks

<table>
<tr>
<td>Meet-in-the-Middle</td>
<td>


By increasing the number of keys used, it might not add a lot of security

Odd +1 = Even → +1 key entropy. E.g From 2^{56} to 2^{57}
Even + 1 = Odd → x2 key entropy. E.g From 2^{56} to 2^{112}

In general, for k-bit keys, it reduces the number of crypto operations to 2^{k+1} using approximately 2^{k+1} units of space.

</td>
</tr>
<tr>
<td>Padding Oracle Attack</td>
<td>


</td>
</tr>
</table>

	$v = IV \oplus \langle 0,0,0,0,0,0,0,0,t,08,08,\texttt{F7},\texttt{0C},\texttt{0C},\texttt{0C},\texttt{0C}\rangle$ and output $t \oplus 08$. That is, (a) For t= 0 to FF (b) let $v = IV \oplus \langle 0,0,0,0,0,0,0,0,t,08,08,\texttt{F7},\texttt{0C},\texttt{0C},\texttt{0C},\texttt{0C}\rangle$. (c) send $v\|c$ to oracle. (d) If yes, output $t \oplus 08$. ***Method 1 - Linear Probing/Checking***: Find out the number of the padding bytes as follows (*note*: 16 is the block size). (1) For i in $[1,2,\ldots,16]$ (2) Change the i-th byte of y_1 to any other value, and send $(IV \parallel y_1 \parallel y_2)$ to Pading Oracle. (3) If the reply is "no", then declare the number of padding bytes is $17-i$ and break from this loop. ***Method 2 - (Improved) Binary Probing/Checking***: Use *binary search* to find the i value that goes from "yes" (valid padding) to "no" (invalid padding). Declare the number of padding bytes as in Method 1.
Bootstrap Attack	• Attacker uses default passwords on IOTs. (E.g wifi router, IP camera etc) • To combat this: Force users to change password once used.
Replay Attack	• Mallory replay the sniffed information to impersonate the user • In a strong authetication setting, sniffed information CANT be used to impersonate a user.
Dictionary Attack	• Use a pre-defined list of commonly used password • Can be done offline/online
Side Channel Attack	• Attacker use an indirect approach to obtain user's information. • E.g via a vulnerability • Attack the vulnerabilty of another system e.g bad software/hardware to gain access to user's data

Padding Oracles:

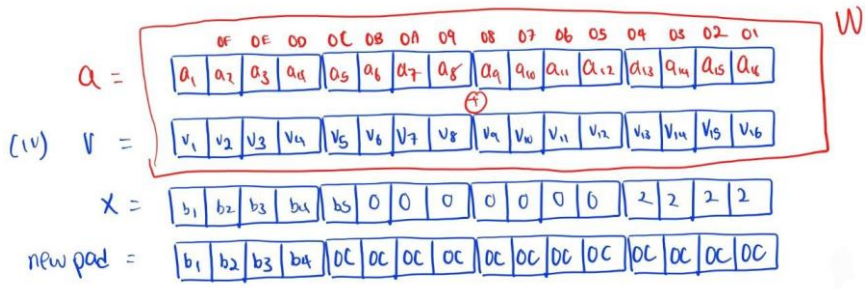
Suppose the attacker knows that a 16-byte plaintext is the string (hexadecimal representation):

($\boldsymbol{b}_1$, $\boldsymbol{b}_2$, $\boldsymbol{b}_3$, $\boldsymbol{b}_4$, $\boldsymbol{b}_5$,0,0,0, 0,0,0,0, 2, 2, 2, 2)

The attacker does not know the value of the $\boldsymbol{b}_i$'s. The attacker has access to the padding oracle and has the ciphertext of the above plaintext. Let $\boldsymbol{v}$ be the one-block IV and $\boldsymbol{c}$ be the one-block ciphertext. The attacker wants to send a query ($\boldsymbol{w}$, $\boldsymbol{c}$) to the oracle where:

$\boldsymbol{w} = \boldsymbol{v} \oplus (a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, a_{15}, a_{16})$

so that the oracle outputs "Correctly padded" iff $\boldsymbol{b}_5 = 1$.



- Note: It is safe to just ignore the IV here.
  - Since $IV \oplus d_k(c) = x$
  - $(IV \oplus \text{a}) \oplus d_k(c) = x \oplus \text{a}$
  - So, we can just evaluate using $x \oplus$ a 😊

- Just like ordinary padding oracle qn, our goal is to change the padding to (b1, b2, b3, b4, 0C, 0C, 0C, 0C, 0C, 0C, 0C, 0C, 0C, 0C, 0C, 0C)
  - The above will be the new padding
  - Since we move to the next left bit it's b5 which is 12th position from the back so 0C.
  - The question says that b_5 is correctly padded if $b_5 = 1$

$$x_5 \oplus a_5 = 0C$$

  - Since $x_5 = b_5 = 1$, → $a_5 = 0C \oplus 1 = 0D$
- Note: (a1, a2, a3, a4, a5, a6, a6, a8, a9, a10, a11, a12, a13, a14, a15, a16) should be the value when we XOR the original padding with the new padding.
  - To find any a_i, simply do: Original Padding ⊕ New Padding
    - ➢ i.e $x_i \oplus newpad_i$

14. For the above method to work, what is the value of a_{10}?

A. 02 B. 0A
C. 0B D. 0C
E. any of the above will work.

Q14.
a_{10} should be $0 \oplus 0C = 0C$
since the original padding at a_{10} is 0
the new padding at a_{10} is 0c

15. For the above method to work, what is the value of a_{16}?

A. 0A B. 0B
C. 0C D. 0E
E. any of the above will work.

Q15.
original padding at a_{16} = 2
new padding at a_{16} = 0C
$0C \oplus 2 = 0E$

16. For the above method to work, what is the value of a_5?

A. 0A B. 0B
C. 0C D. 0D
E. any of the above will work.

Q16. (As seem from example above)
original padding at a_5 = 1
new padding at a_5 = 0C
$0C \oplus 1 = 0D$

17. For the above method to work, what is the value of a_1?

A. 02 B. 0B
C. 0C D. 0D
E. any of the above will work.

Q17.
original padding at a_1 = something but doesnt matter
new padding at a_1 = also doesnt matter since we are just changing padding b5 (inclusive) onwards

Authentication Credentials

- Weak authentication is easily subjected to replay attacks.
- Passwords should be HASHED, not encrypted.
  - Store cipher text in password file, NOT the plaintext!
  - Best practice: Salt + Hash.
- Use Multifactor Authentication → Even if attacker can get your password & username, cannot log in

Authenticity:

RSA:

- For RSA, the $\varphi(n)$ MUST NOT be made public, otherwise security will be compromised.
  - Recall $\varphi(n) = (p-1)(q-1)$
  - With $\varphi(n)$ leaked, we can reverse to find the composites and find the primes p & q.
- In the same vein, the attacker cannot decrypt ciphertexts since deriving $\varphi(n)$ from n requires performing prime factorization. (allegedly)

Encryption	Decryption
Public Key(n, e) Given m, the ciphertext c is: $c = m^e \ mod \ n$	Private key(n, d) Given c, the plaintext m is: $m = c^d \ mod \ n$

- Interchangeable role of Encryption & Decryption.
  - d and e can be swapped.

Diffie Hellman:

- Parties A & B agree on a large prime p, a generator g.
- A & B chooses a secret exponent a, b for their own private keys respectively:

A's key: $A_k = g^a \bmod p$	B's key: $B_k = g^b \bmod p$
$k = B_k^a \bmod p$ $k = (g^b \bmod p)^a \bmod p$ $k = g^{ab} \bmod p$	$k = A_k^a \bmod p$ $k = (g^a \bmod p)^b \bmod p$ $k = g^{ab} \bmod p$

- Important to keep private exponents secret, if not attacker can reverse it.

Public Key Encryption

- Goal → Obtain Confidentiality
- Signature's Goal → Obtain Authenticity
- RSA Key size ≈ 2048 bits
  - Prime Number Theorem: Probability of a randomly chosen 1024-bit number being a prime is around ln(21024)~ (1/710). So likely takes about 710 trials to get a prime number.

- The "classroom" RSA needs to be modified to prevent some attacks. Many attacks exploit one nice property of RSA: homomorphic property.
  - This property is a double-edged sword: it is useful (e.g. blind signature), but is also a vulnerability.
    - ➢ It turns out that the classroom RSA leaks one-bit of information about the plaintext!
    - ➢ If adversary knows the module n, and knows the ciphertext c, the adversary can derive one "bit" of information regarding the plaintext.
    - ➢ This is because RSA encryption preserves the "Jacobi symbol", i.e. the Jacobi symbol of the plaintext and ciphertext is the same.

$$\left(\frac{a}{p}\right) := \begin{cases} 0 & \text{if } a \equiv 0 \pmod{p}, \\ 1 & \text{if } a \not\equiv 0 \pmod{p} \text{ and for some integer } x\text{: } a \equiv x^2 \pmod{p}, \\ -1 & \text{if } a \not\equiv 0 \pmod{p} \text{ and there is no such } x. \end{cases}$$

- RSA is very slow (vs AES) ** of equivalent key strength

Algorithm	MiB/Second	Cycles Per Byte	Microseconds to Setup Key and IV	Cycles to Setup Key and IV
AES/GCM (2K tables)	102	17.2	2.946	5391
AES/GCM (64K tables)	108	16.1	11.546	21130
AES/CCM	61	28.6	0.888	1625
AES/EAX	61	28.8	1.757	3216
AES/CTR (128-bit key)	139	12.6	0.698	1277
AES/CTR (192-bit key)	113	15.4	0.707	1293
AES/CTR (256-bit key)	96	18.2	0.756	1383

In different order of magnitude.

Operation	Milliseconds/Operation	Megacycles/Operation
RSA 1024 Encryption	0.08	0.14
RSA 1024 Decryption	1.46	2.68
RSA 2048 Encryption	0.16	0.29
RSA 2048 Decryption	6.08	11.12

-
-

Other PKC Schemes:

Discrete log-base PKC	ElGamal Encryption • ElGamal is a "Discrete Log-based" encryption, whereas RSA is "factorization-based". • There are many choices of Algebraic groups for discrete log-based encryption, e.g. Elliptic Curve. • Those using Elliptic Cure are often called Elliptic Curve Cryptography (ECC). Certain choices of ECC reduce the key size. - E.g. ~300 bit for equivalent of 2048-bit RSA. Paillier Encryption • Paillier Encryption is also discrete-log based. • ElGamal can be easily modified so that it is homomorphic w.r.t. multiplication, whereas Paillier is homomorphic w.r.t. addition. If an algorithm can efficiently solve the Discrete Log problem, then the algorithm can easily break the discrete-log based encryption such as Paillier and ElGamal. (more in key exchange.)
Lattice PKC	• Currently, there are active research on getting Lattice-based PKC. (details omitted). • **Lattice-based cryptography**: Based on this hard problem-- Given the "basis" of lattice, it is computationally hard to find the shortest (or approx) lattice point. Many proposals, no clear winner yet.

- ElGamal: ElGamal can exploit techniques in Elliptic Curve Cryptography (ECC), reducing the key size to ≈ 300 bits.
- Paillier: Partial homomorphic with respect to addition.
- We assume it is feasible to carry out 2^{60} operations
- Pseudo random is deterministic
  - The same seed → same sequence generated

```
#include <time.h>
#include <stdlib.h>
   srand(time(NULL));
   int s = rand();
```

  - The above are not cryptographic secure.
    - ➢ If attacker knows the time, the attacker can derive the key.
    - ➢ If attacker knows the approximate time, the attacker can also exhaustively search.
    - ➢ But even if attacker has 0 clue about the time, the attacker can still brute force and to find the value of s.
      - ➔ Why? s is an integer. It is either 2 bytes (16bits) or 4 bytes (32 bits). The most you need to brute force is 2^{32} times, which is feasible.

Post Quantum Cryptography

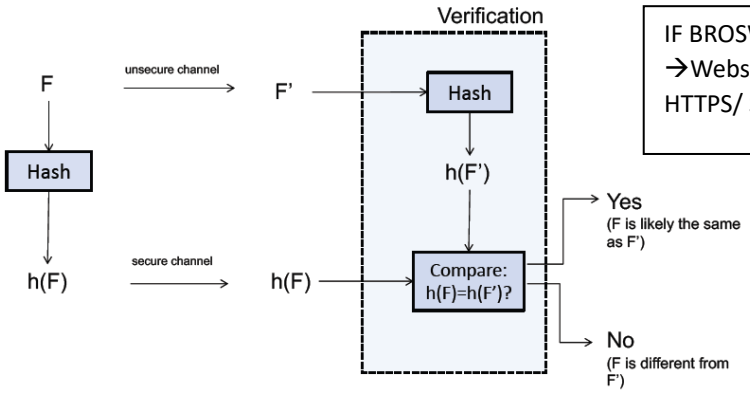
- Shor's Algorithm:
  - Used to attack asymmetric cryptogrphy
  - Using Qubits, shor's algorithm can reduce bruteforce time from 317 millioin to 10 seconds.
  - RCC, ECC no longer safe.
- Grover's Algorithm:
  - Used to attack symmetric cryptography
  - Conventional attacks take $O(2^n)$ time.
  - Grover's algorithm takes $O(2^{n/2})$ time.
  - Can overcome this by doubling the key size.

Data Authenticity (Digest), unkeyed

- From here, we follows Kerckhoff's principle, where we assume that the adversary knows all the algorithms.

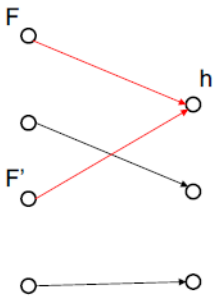
Kerckhoff's Principle
Kerckhoff's Principle states that the security of a cryptosystem must lie in the choice of its keys only; everything else (including the algorithm itself) should be considered public knowledge.

Hashes (Unkeyed, no secret key)(No secret involved, the adversary knows the algorithm)



IF BROSWER SHOWS PADLOCK →Website is authentic (Assuming HTTPS/ Secure channel is used)

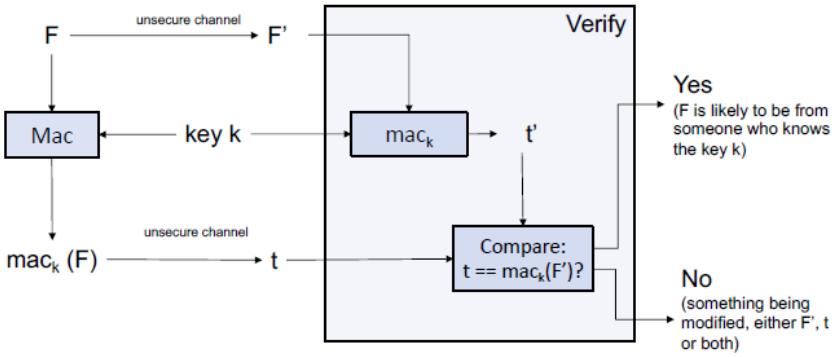
- For cryptographically secure hash, it is infeasible to recover the password from the hashed value. In fact, to be secure, we don't want to have a way to recover the password
- A (cryptographic) hash is a function that takes an arbitrary large message as input, and outputs a fixed size (say 160 bits) digest. [Most cryptographic application need at least 160 bits]
- **Hash Security Requirements:**
  - The hash function should be collision resistant.
  - A hash that is collision-resistant is also called one-way.
    - That is, given a digest **d**, it is difficult to find a message **m** such that h(m) = d
    - It is **difficult** for an attacker to find 2 different messages, say m_1 and m_2 that hash to the same digest i.e $h(m_1) = h(m_2)$
- There are many ways an attacker can attempt to solve:
  - The problem of 2nd pre-image attack on h()
    - Given a F, find a F' such that h(F) = h(F'), and F ≠ F' (input if F)
  - The problem of one-way on h()
    - Given a h, find a F' such that h(F') = h (input is h)
  - The problem of Collision attack on h:
    - Find F and F' such that h(F) = h(F'), and F ≠ F' (no input)



- Comparing ease of hacking, collision will be the easiest out of the 3.

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.

MAC (Keyed Hash)



- MAC → Used for authentication
- A mac is secure against forgery.
  - Secret key, k is known to server only.
  - Attacker can generate a token but can never get the MACk(m) as the attacker does not know the secret key k. Without knowing the key k, it is difficult to forge a MAC

- Suppose not, that is, the design of authentication token is not secure. So, there is an attacker who can forge a new valid authentication token, when given some valid authentication tokens. Let's call this attacker Fiona.
- Now, let's argue that with Fiona's help, an attacker can break the MAC.
- Suppose an attack has a pair of valid mac, $\langle m, y \rangle$ where m is a message of the authentication token. We further assume that the format of m is correct (as an authentication token) and it corresponds to an user id.
- The attacker passes the authentication token $\langle m, y \rangle$ to Fiona. Recap that Fiona is able to break the authentication token. By definition of the attack model, this means that Fiona is able to forge a valid authentication token $\langle m', y' \rangle$ of another user. Fiona passes $\langle m', y' \rangle$ back to the attacker. Since the user in m and m' are different, so $m \neq m'$. Since it is a valid token, so $\langle m', y' \rangle$ is also a valid mac. This means that the attacker, with the help of Fiona, can forge another pair of valid mac.
- The above conclusion is a contradiction to the fact that the mac is secure. So, if the mac is secure, no such Fiona exists.

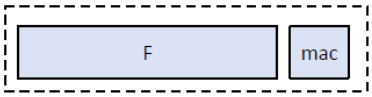
- Symmetric Key setting → MAC
- Public Key setting → Digital Signature

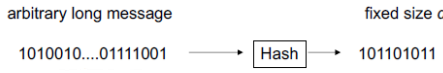
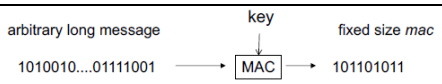
- MAC vs Randomized Token:

  In term of security, both mac-based variant and the random token variant are equivalent under reasonable scenario. Nonetheless, mac-based rely on "computational hardness" while the random token variant does not. To see the difference, note that there is no secret involved in random token variant, and thus it remain secure if the secret (where there is none in random token) is leaked.
- 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 (message authentication code).**
- **Security Requirement for Keyed-Hash** (Forgery)
  - 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.
  - Without knowing the key k, it is difficult to forge a MAC
- An authentication token (SSO) typically has an expiry date. Why?
  - If token is stolen and user is unaware of it, attackers will be forced to reauthenticate.
  - If there isn't an expiry date, the token can be used forever.
- HMAC (Hashed MAC → Based on SHA)

  $$HMAC_k(x) = SHA-1((K \oplus opad \,||\, SHA-1((K \oplus ipad \,||\, x))$$

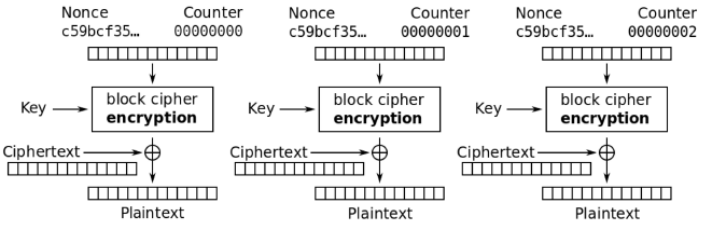
  Where:
  opad → outer padding, 0x3636…36
  ipad → inner padding, 0x5c5c…5c
- The HMAC relies on a cryptographic hash. As highlighted before, although hash function produces "pseudorandom" sequence, they are deterministic.
- So, HMAC is also deterministic.
- If two blocks of plaintext are exactly the same, then so are their HMACs (If b_n=b_m, then t_n = t_m)



Unkeyed Hash	Keyed Hash
	
A non-keyed hash produces an output that depends only on the input data. • No secret key • If it is cryptographically secure → no known attacks faster than bruteforce attack to find: - An input that hashes to a particular output; - Two inputs that hash to the same output.	A keyed hash produces an output that depends both on the input data, and a key. • Has secret key • If it is cryptographically secure → satisfies the properties of a non-keyed hash, and: - There are no known ways faster than bruteforce to find the key used, given a set of input and output pairs produced with that key; - There are no known ways better than random chance to find the correct output for any input under a particular key, without knowledge of the key.

Sign message using secret key, compute MAC and append

- Why don't we encrypt mac?



Counter (CTR) mode decryption

- Suppose we implement mac as such
- t = mac(k, m) = $Enc_k(H(m))$
- As AES CTR is used, there is a pseudorandom sequence r being generated by the stream cipher
- So t = $Enc_k(H(m))$ = r ⊕ H(m)
- Attacker can compute a message m' and obtain t' = t ⊕ H(m') ⊕ H(m)
- t' is a valid mac for m'
- t' = t ⊕ H(m') ⊕ H(m) = r ⊕ H(m) ⊕ H(m') ⊕ H(m)
- = r ⊕ H(m') ⊕ H(m) ⊕ H(m) = r ⊕ H(m') ⊕ 0
- = r ⊕ H(m')
- which itself is $Enc_k(H(m'))$ and mac(k, m') since this is how the mac was designed
- Attacker can forge a message of his choice, so this design does not yield a secure mac

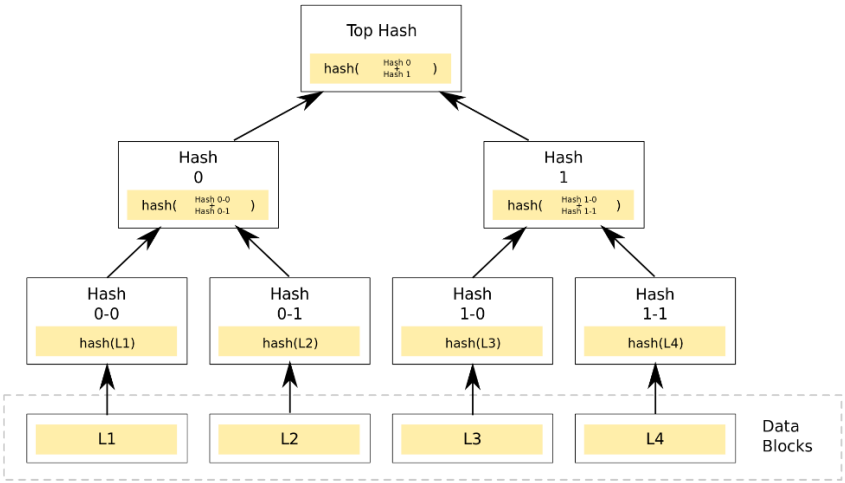
- The primary purpose of a MAC is to provide data integrity and authenticity rather than confidentiality.
  - While encrypting the MAC with the message would provide confidentiality, it would also prevent the recipient from verifying the integrity and authenticity of the message.
  - This is because the recipient would not be able to compute the original MAC value to compare it with the received MAC (which has been encrypted). It would also defeat the purpose of using a MAC to provide integrity and authenticity checks.

To attain both confidentiality and authenticity, we can use authenticated encryption mode. Do not design a new scheme that achieves both! Recently, a number of such modes of operations have been designed for AES. E.g. CCM and GCM. (In this module, students should be aware of such modes.

QNS	Suppose we are given a binary string h of 1,000,000 bits which is randomly chosen. Is it easy to find a message x s.t. SHA3 (x) is a substring in h?
ANS	No… It is difficult There are two problems: • (P1) Original pre-image problem. Given y, find a x s.t. $\texttt{SHA3}(x) = y$ • (P2) This new problem. Given z, find a x s.t. $\texttt{SHA3}(x)$ is a substring of z. Suppose we have an algorithm A2 that solve P2, we want to use it to solve P1. Give a y, we randomly pad it with random bit to get y', so that its length is 10^6 and y is a substring in y'. Now run $A2$. A2 will give a output x. if $\texttt{SHA3}(x) = y$, then halts. Otherwise repeat for, say 10^7 times. Note that each time, the probability of succeed is 10^{-6}. After repeat for 10^7 times, the probability that one of them succeed is more than 0.5.

A hash tree (merkle tree) allows efficient and secure verification of the contents of a large data structure. A hash tree is a generalization of a hash list and a hash chain.

- Merkle trees can help ensure that data blocks received from other peers in a peer-to-peer network are received undamaged and unaltered, and even to check that the other peers do not lie and send fake blocks.



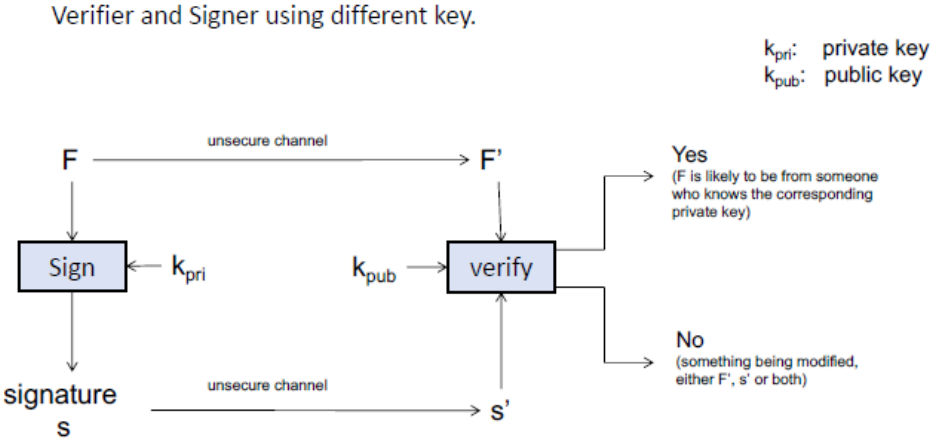
Note:

- Merkle Tree is susceptible to 2nd preimage attack.
- This can be fixed via prepending 0x00 byte to root nodes, and 0x01 byte to internal hash nodes. OR simply limiting the hash tree's size.

Rigorous Explanation:

- The Merkle hash root does not indicate the tree depth, enabling a second-preimage attack in which an attacker creates a document other than the original that has the same Merkle hash root. For the example above, an attacker can create a new document containing two data blocks, where the first is hash 0-0 + hash 0-1, and the second is hash 1-0 + hash 1-1.
- One simple fix is defined in Certificate Transparency: when computing leaf node hashes, a 0x00 byte is prepended to the hash data, while 0x01 is prepended when computing internal node hashes.[13] Limiting the hash tree size is a prerequisite of some formal security proofs, and helps in making some proofs tighter. Some implementations limit the tree depth using hash tree depth prefixes before hashes, so any extracted hash chain is defined to be valid only if the prefix decreases at each step and is still positive when the leaf is reached.

Signature



- The public key version of MAC is called Signature.
- Achieves Non-Repudiation → Assurance that someone cannot deny previous commits/actions
- 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.**
- Security Requirement of Digital Signatures:
  - Without knowing the private key, k_{pri}, it is difficult to forge a signature

- The authenticity of F can be **verified** by **ANYONE** who knows the **public key.**
  - The **valid signature** can only be **computed** by someone who **knows the private key.**
    - Hence, IF the signature is valid → F MUST be authentic.

Using MAC	**Scenario**	Alice sends Bob a message appended with a MAC, which is computed using a shared key between Alice and Bob. Later, when confronted by Bob, Alice denies sending the message and claims that Bob generated the MAC.
	Explanation	• In this case, if Bob can demonstrate that the MAC was generated using a key that only Alice and Bob share, it provides evidence that the message originated from Alice. • This is because the MAC serves as a cryptographic tag that Alice applies to the message, and only someone possessing the shared key (in this case, Alice or Bob) could have generated the MAC. • Therefore, the MAC serves as evidence against Alice's denial.
Using Signature	**Scenario**	Alice sends Bob a message, and she signs it using her private key. Later, Alice wants to deny that she sent the message.
	Explanation	• In this scenario, the digital signature provides strong evidence of the message's origin. • Only Alice possesses the private key necessary to generate the signature. • Bob can verify the signature using Alice's public key. • If the signature is valid, it proves that the message was indeed signed by someone possessing Alice's private key, i.e., Alice herself. • Since only Alice has access to her private key, she cannot deny having sent the message without refuting the cryptographic evidence provided by the signature.

Cryptographic primitive Security Goal	Hash	MAC	Digital signature
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Kind of keys	none	symmetric keys	asymmetric keys

PKI + Channel Security:

Public Key Infrastructure

- In Public Available Directory, its centralization may lead to server overload. As such, we require a distributed way to broadcast the keys securely.
- **Centralized vs. Distributed**:
  - PKI provides a standardized system for securely distributing public keys, addressing the limitations of centralized methods like public directories.
  - It enables the distribution of keys in a distributed manner, avoiding the potential for server overload.
- **Main Objective of PKI**:
  - PKI aims to be deployable on a large scale, accommodating the needs of widespread use cases such as internet domains and email addresses.
- **Secure Distribution**:
  - Users obtain public keys securely through digital certificates signed by trusted Certificate Authorities (CAs).
  - These certificates establish a chain of trust, ensuring the authenticity and integrity of the public keys.
- **Components**:
  - **Certificate**: A digital certificate contains information about the entity it belongs to (such as a website or an individual), along with the associated public key and the digital signature of the issuing CA.
  - **Chain of Trust**: PKI operates based on a chain of trust, where the authenticity of a certificate is verified by tracing it back to a trusted root CA. Intermediate CAs further validate and sign certificates, forming a hierarchical trust structure.

Public vs. Private PKI:

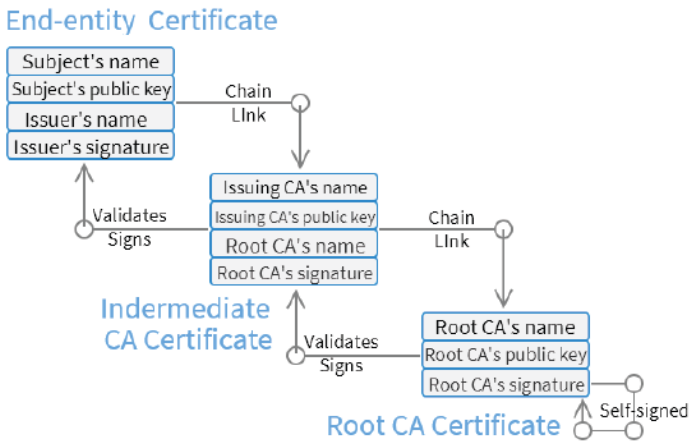
- **Public PKI**: This refers to the PKI adopted in the internet for domain names, email addresses, and other public-facing applications. It's commonly known simply as "PKI."
- **Private PKI**: These are PKI systems used for specific applications or within organizations. Private PKI systems have their own set of CAs, allowing companies to establish their own trust infrastructure for internal use cases.

Certificate

- Certificate: a piece of document that binds a "name" to a "public key" & certified by an authority, 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.)
    - **Chain of Trust**: PKI operates based on a chain of trust, where the authenticity of a certificate is verified by tracing it back to a trusted root CA. Intermediate CAs further validate and sign certificates, forming a hierarchical trust structure.

- 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 CA's.

Chain of Trust



QNS	Risks of accepting expired certificates.
ANS	• Certificate used to be valid prior to the expiry date but information may be outdated • It's sort of like sending a letter to a friend's address which you know was correct 10 years ago, but he may have migrated and someone is residing there now instead • D1 = Name, D2 = Public key, D3 = Valid range, D4 = Signature • Website domain changed hand after the date stated in D3. Previous owner has the old private key and can exploit this to attack Alice • Private key has been compromised but since certificate has already expired, there is no need to revoke. By still accepting this expired certificate, Alice can be attacked • Outdated certificate might still be using SHA1 and signature can be forged (SHA1 is not collision resistant) - We now know that SHA1 is not collision resistant but it was previously (before 2016) accepted as a choice by X509. Hence, a certificate signed using SHA1 before 2016 could be either (1) indeed signed by the CA; or (2) forged by an attacker.

- If the communication is **secured using HTTPS**, a Mallory in the public channel is **unable to compromise confidentiality/integrity**. However, when the **Mallory is a CA accepted by Alice**, even if the communication is secured using HTTPS, the Mallory can **compromise confidentiality/integrity.**

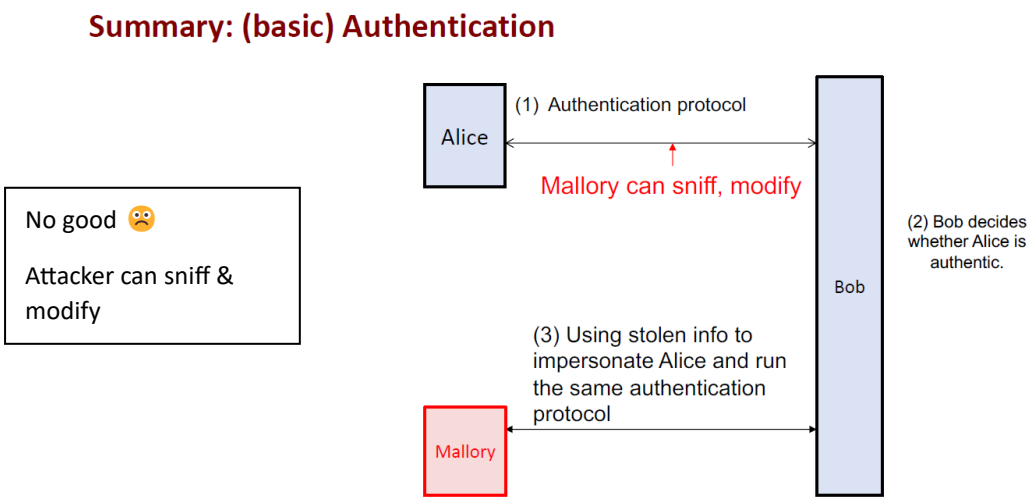
Public Key Distribution vs Symmetric Key Distribution

- Both need secure channel to distribute keys. Nevertheless, it is easier to securely "broadcast" public key compared to "establish" a different symmetric key for every pair.

Public Key	Symmetric Key
• An entity broadcast its public key to some public billboard only once. (constant) • An entity doesn't need to know the existence of the receiver while broadcasting its public key. (e.g Amy could place her name card in anywhere, doesn't need to interact with Bob)	• An entity needs to securely establish a different symmetric key with each of the rest. (linear) • Both entities interact to establish the key.

Limitations/Attacks on PKI:

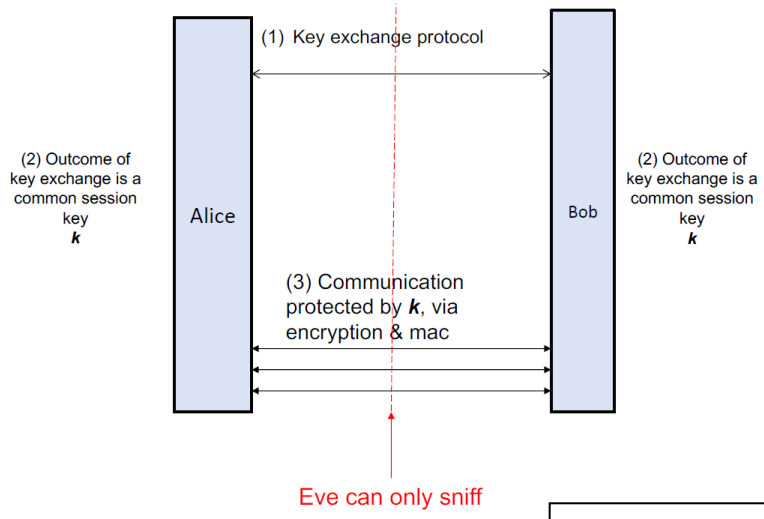
- Implementation Bugs (Browsers ignoring substrings in the name after null)
- Abuse by CA (Malicious CA → MITM → Confidentiality and Integrity compromised. E.g SuperFish)
- Social Engineering: (Generally: DNS spoofing, URL spoofing, Fake URL etc.)
  - Typosquatting
  - Sub-domains
  - Homograph attacks



Summary: Key exchange

Using MAC's property, even if Eve sniffs the communication between A & B, and obtain multiple valid m & t, Eve still cant get the secret key, k, and cannot forge the MAC for the messages Eve has yet to see/sniff.

Eve can't conduct replay attacks



Unilateral Authentication via MAC

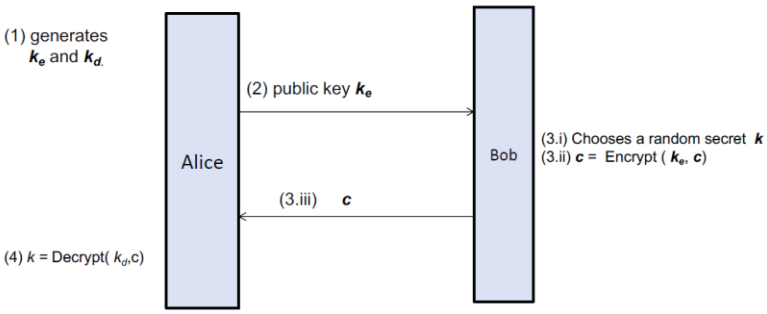
Mutual Authentication via PKC

PKC-based Key-exchange

- Attacker will need to sniff, then steal session key.
- By the property of signatures, Eve can't derive Bob's private key and forge/imitate Bob's responses.
- Can use nonce to ensure freshness

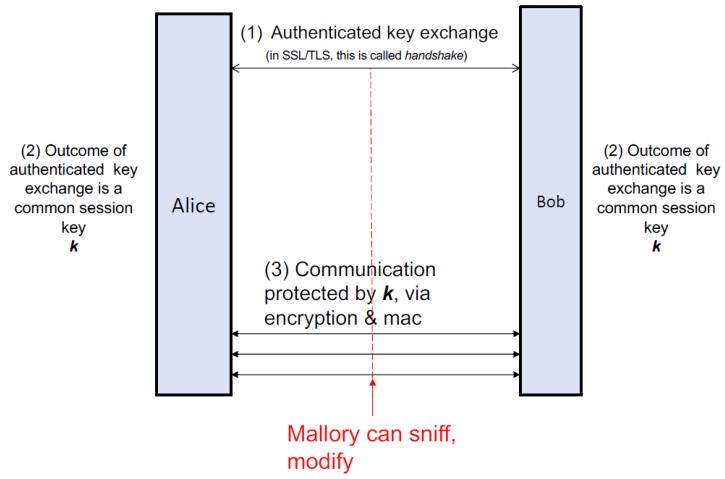
Here is a key-exchange that uses a PKC

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.



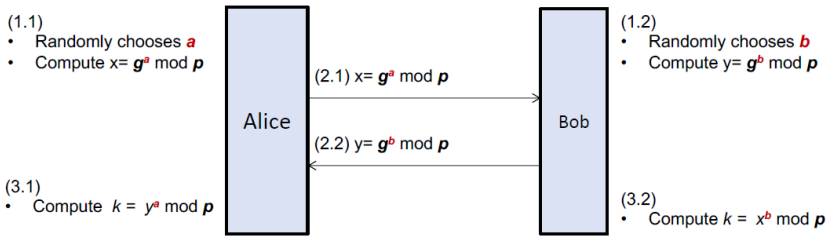
Summary: Authenticated Key exchange

- A & B authenticate via signature (PKC).
- Protection against MITM attacks
- By the security of PKC, from the public key and ciphertext, even if Eve sniffs and obtains the public key, K_e and ciphertext c, Eve can't get any information about the plaintext, which is the secret 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, $x= g^a \bmod p$, $y= g^b \bmod p$, it is computationally infeasible to find $k=g^{ab} \bmod p$.

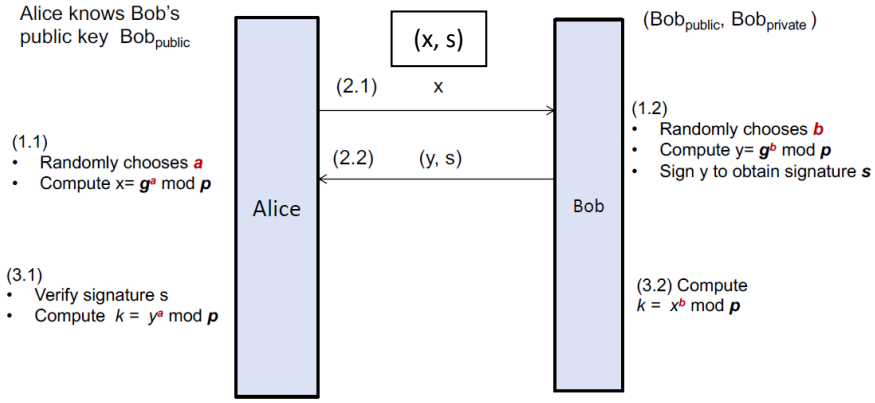
Remarks:
1. Step (1.1)&(1.2), (2.1)&(2.2), (3.1)&(3.2) can be carried out in parallel.
2. The assumption seems self-fulfilling. Nonetheless, there are many evidences that it holds.
3. The operation of "exponentiation" can be applied to any algebraic group, i.e. not necessary integers. CDH doesn't hold 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.

Station-to-station protocol (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. Can be easily extended to mutual authentication.

For mutual authentication, alice can sign at step 2.1



- Given $g^a mod\ p$ and $g^b mod\ p$, it is computationally difficult to get a and b
- Attacker may know the private key of Bob but Bob's private key is merely used to sign $g^b mod\ p$
- What does attacker know?
- $g^a mod\ p, g^b mod\ p, signed\ g^b\ mod\ p$
- Session key that the attacker is trying to get hold of : $g^{ab}\ mod\ p$
- We know that DH is difficult to solve, so is STS
- Hence attacker cannot get $g^{ab}\ mod\ p$, which is the session key
- Forward secrecy is preserved ☺ yay

** Session keys help to contribute to forward secrecy.

- Even if long-term secret keys are compromised (in the future) no one can use it to decrypt past communication sessions. Past communications are protected, and since session keys have been discarded, the attacker can only have access to current/future sessions, not past sessions.

Let's refer to Lecture 4 slide 63 on STS protocol. All the data communicated during STS are (1) x; and (2) (y, s), where s is the signature of y signed using Bob's private key.

Short Answer giving the "intuition". Here is summary of the three problems faced by the attackers:

i. DH: Given x and y, find z where $z = g^{ab}$, $x = g^a$ and $y = g^b$ for some a, b.

ii. STS: Given x and (y, s), public key k_{pub}, find z where (1) $z = g^{ab}$, $x = g^a$ and $y = g^b$ for some a, b; and (2) s is the signature derived from y and the private key of k_{pub}.

iii. STS (FS): This is the attacker who knows the long term private key of STS and wants to get the session key. That is, given x and (y, s), public key k_{pub}, private key k_{priv}, find z where (1) $z = g^{ab}$, $x = g^a$ and $y = g^b$ for some a, b; and (2) s is the signature derived from y and the private key.

We want to show that STS(FS) is not easier than DH. Suppose we are given a DH problem with input (x, y), we can easily construct the input of a corresponding problem for STS (FS) in this way: randomly generate a pair of public key and private key, and then sign the y to get the input to the STS (FS) problem: $(x, (y, s), k_{\text{pub}})$. If STS(FS) can be solved, we solve DH.

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 in 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, ... 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.

Can append sequence number to the message, to ensure that no packets are being dropped/inserted. However, modified packets will not be detected, since order is correct.

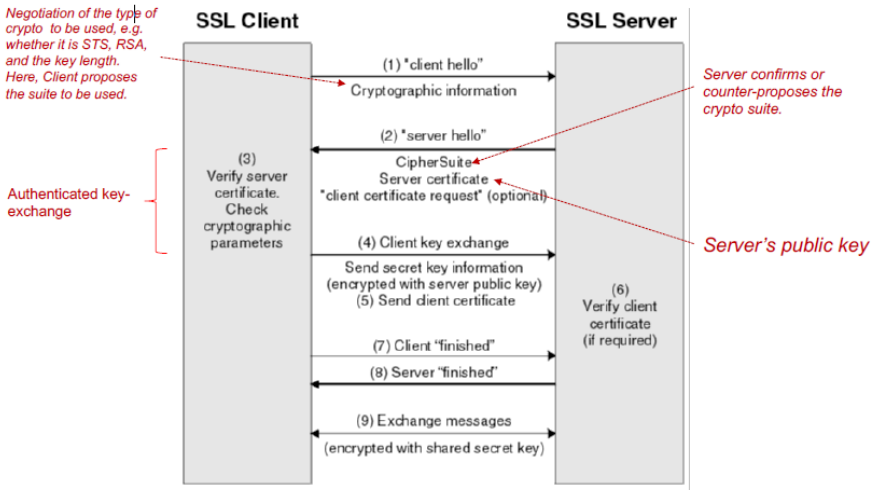
Password Based Encrypted Key Exchange

Asymmetric	• An entity is considered authentic if it can convince the other that it know the private key of the associated public key. - E.g Authenticated Key-Exchange Protocols, Station-to-Station
Symmetric	• 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. - There are secure protocols that, even if the entropy of password is low, it is still secure against offline dictionary attack (hence, to guess whether a password is correct, attacker is forced to interact with the online server). - These are called "Password-Authenticated Key agreement" (PAKE).

Mutual Authenticated Key Exchange

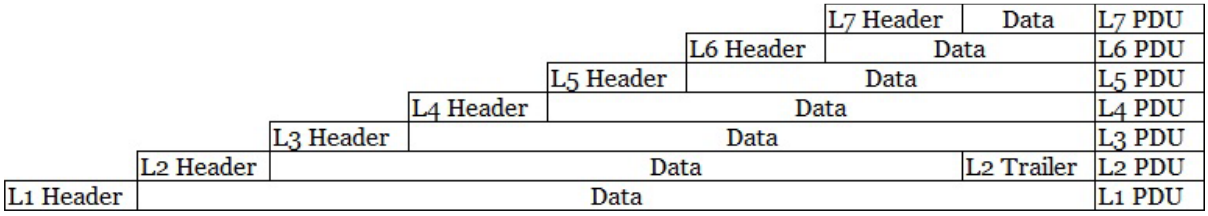
Before Protocol	1. Alice and Bob has a pair of public and private key each. 2. Alice know Bob's public key vice-versa. (Long-term/Master Key) 3. They will carry bout Authenticated Key Exchange protocol. - If an entity if NOT authentic, the other will halt
After Protocol	Both A & B obtain a shared key, k, known as the Session key.
Security Requirements	• **Authenticity**: Both parties are assured that they are communicating with the intended party, who knows their respective private key (with respect to their public key) • **Confidentiality**: The Attacker is unable to access the session key, and all communications are secure.

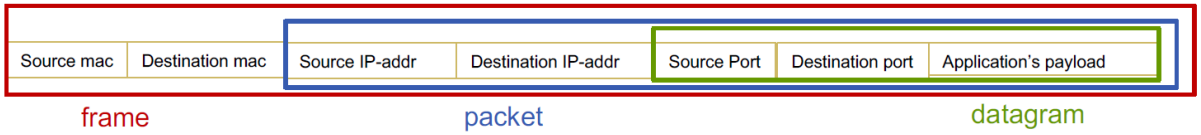
TLS Handshake (Authenticated Key Exchange)



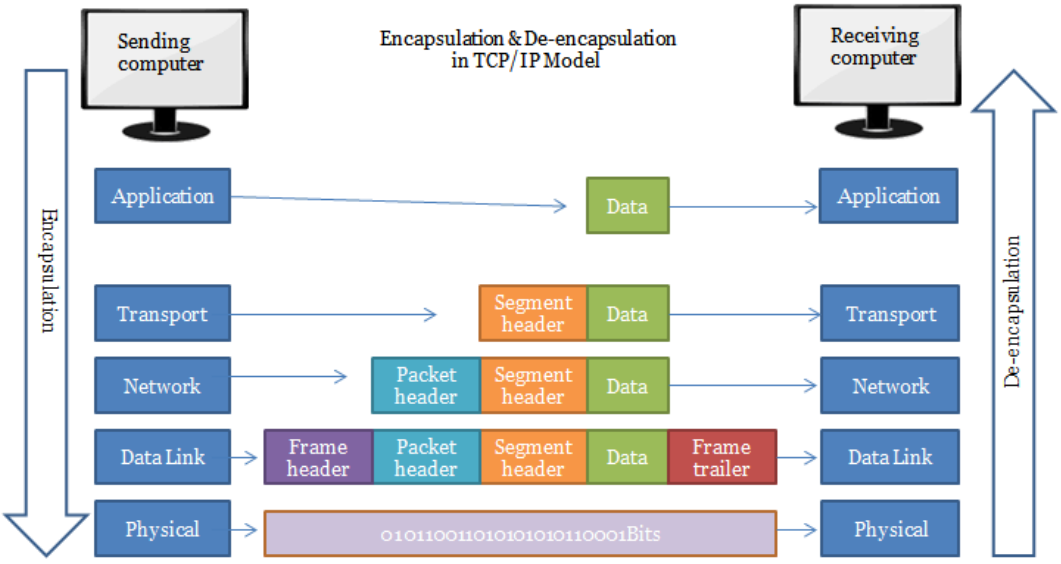
Network Se

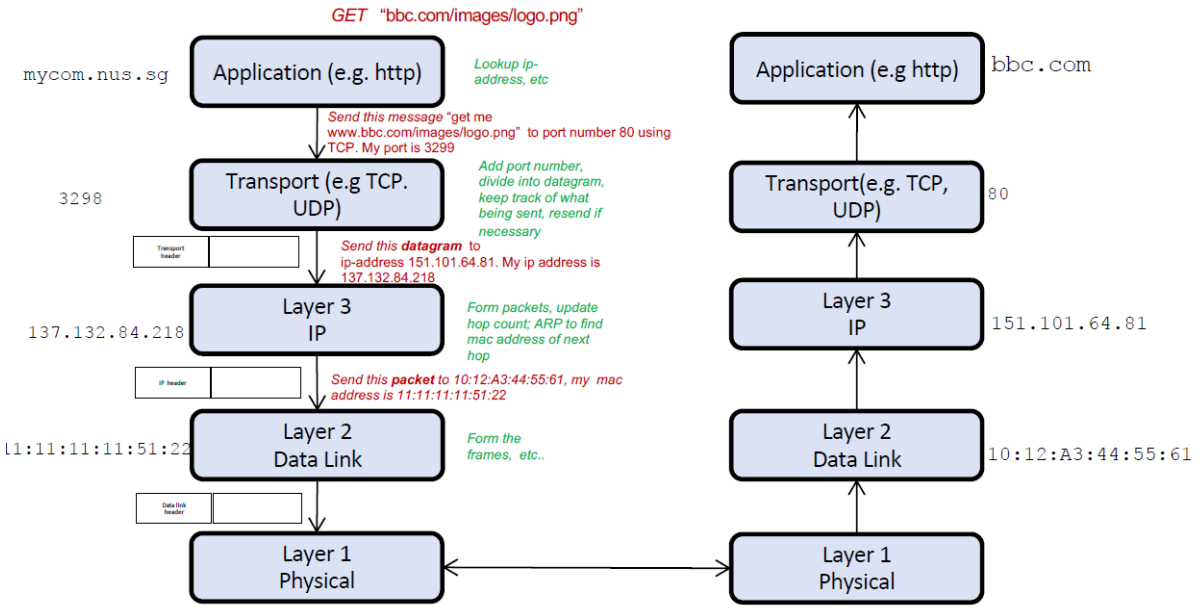
- At each intermediate node, the routing data will be read and modified.
  - Time to Live (TTL) will be updated (Decreases after every hop to prevent congestion)
  - Header information is updated.
  - Source IP, Dest IP, Mac Address, Routing Information, Checksums etc.
- **DNS** (Domain Name System) → Maps Domain Name to IP address (Domain Name ↔ IP address)
- **ARP** (Address Resolution Protocol) → Maps IP address to MAC address (IP address ↔ MAC address)
  - More specifically: IP address ↔ MAC address ↔ Port Number ↔ MAC address ↔ IP address
  - Note: MAC → Media Access Control, not Message Authenticated Code.
  - MAC is unique, 2 devices cannot have the same MAC address.
- **Switch** → Handles MAC address
  - Does not understand IP address, and does not store IP address.
  - The switch connects ports based on their MAC address.
  - Make use of the ARP table to translate MAC address to port number.
  - Conversion of MAC to IP address is done by the node. (e.g Phone, PC)
    - The node has a table that converts between MAC and IP address.
- **Router** → Handles IP address



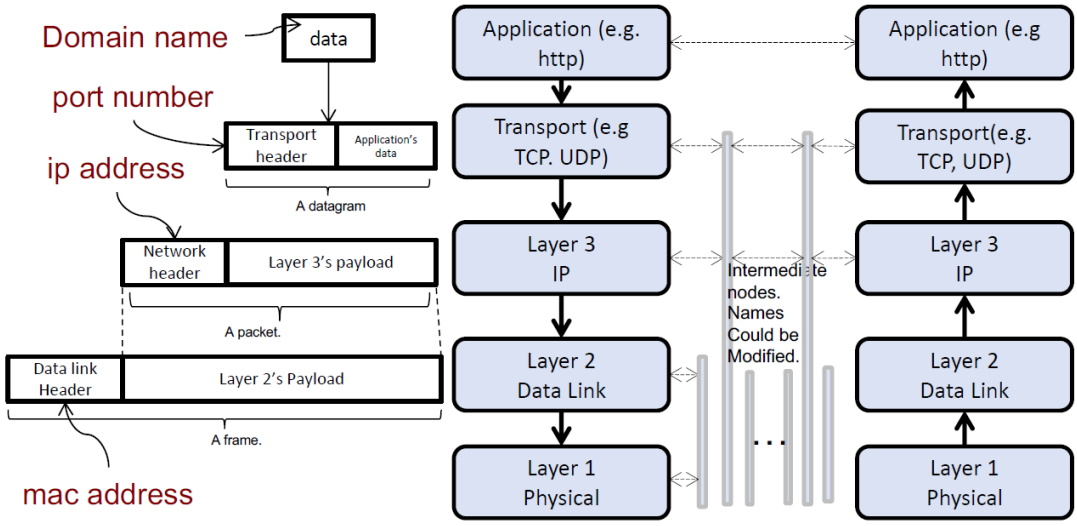


- Each layer will append new header files together with the message chunks/data.
  - In the transport layer (Layer 4), each data unit is called a "**datagram**"
  - In the network layer (Layer 3), each data unit is called a "**packet**" or "**IP packet**"
  - In the datalink layer (Layer 2), each data unit is called a "**frame**"
  - **<u>Data Flow across the different Layers between 2 nodes</u>**





-

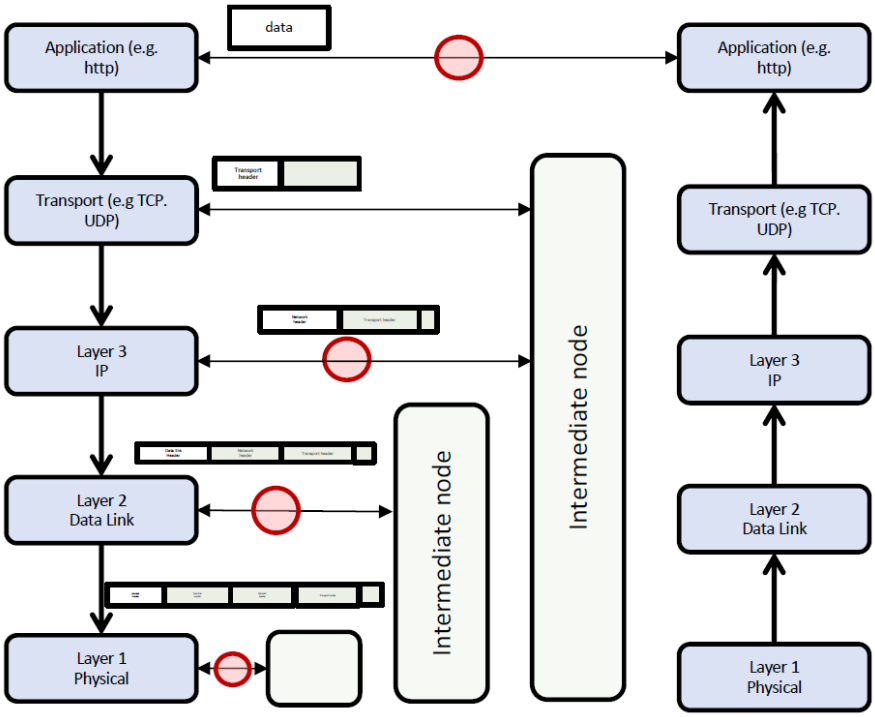


Source mac	Destination mac	Source IP-address	Destination IP-address	Source Port	Destination port	Application data

- Note: Intermediate nodes header information can be changed (mutable)
  - E.g Source and Destination address updates, number of hops update (TTL) etc ..

MITM

- The **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.



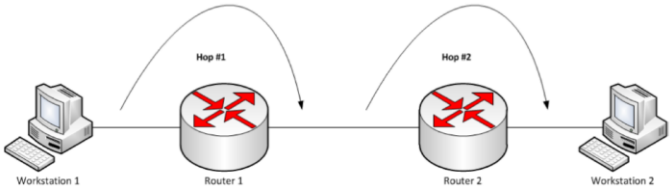
*****Reliability DOES NOT imply Security!!!**

- While TCP/IP is reliable, it is NOT "secure".
- The intermediate nodes along the communication routes can be modified.
  - The Data in the header and payload of the packets can be modified by MITM.
  - Attacker can spoof, modify, reorder, redirect, drop packets.

QNS	As a MITM, in which layer does the VPN operate?
ANS	Just below the IP layer/ In transport layer

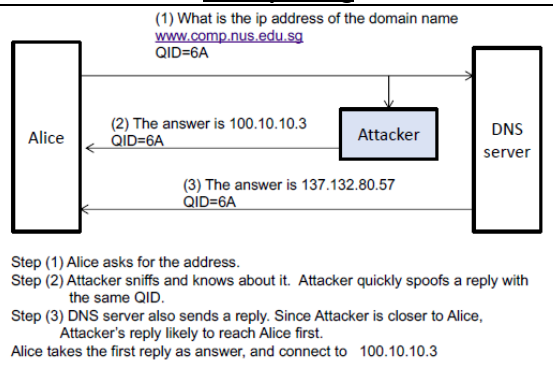
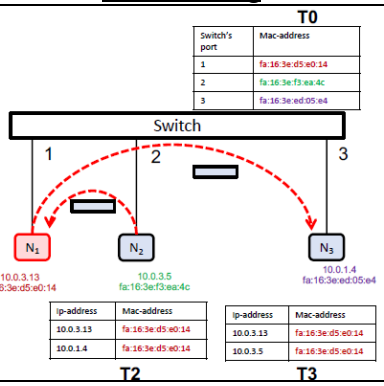
QNS	Can the VPN know that Alice is watching YouTube?
ANS	Yes. VPN can see the destination IP address

QNS	Can the VPN figure out which video Alice is watching?
ANS	• No. YouTube uses TLS. Videos are encrypted before going down the layers. • Possibly. When data is broken down into pieces, based on the timing and size of pieces, VPN could find out which particular video Alice is watching. - Traffic Analysis, Machine learning.

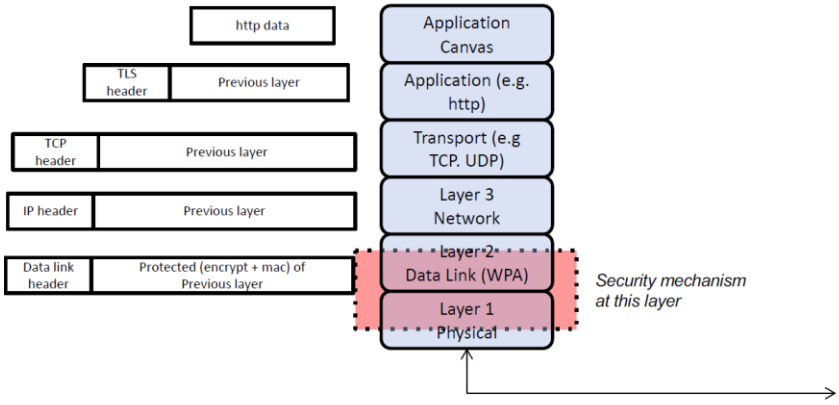
QNS	Can the VPN figure out Alice's mac address?
ANS	Unlikely. Mac address likely modified before reaching VPN.  Unless the number of hops <2, the script will not learn of Alice's MAC address. (i.e not a direct hop from sender to receiver). Recall: MAC (src & dest) address updates every time it hops.

Port Listening/Scanning

- Total of $2^{16} - 1 = 65535$ ports.
  - Port 0 reserved for TCP/IP
  - Port 80 reserved for HTTP/S (Webserver)
- If the port is NOT listening, then it is a "closed port"
- Data sent to a closed port will be dropped.

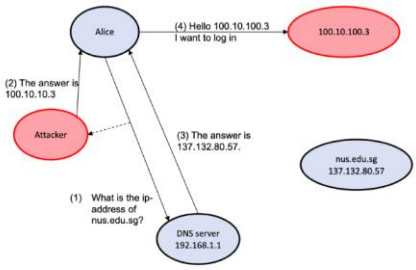
DNS Spoofing:	**ARP Poisoning**
	
Since the Attacker sits closer to Alice than the DNS server, the Attacker's reply will always reach Alice first before the DNS server. Although both Attacker and DNS server have the same QID, Alice will only accept the 1st response in the network, and ignore the other one.	MITM in layer 2. Attacker poison ARP table by broadcasting its macaddress to all possible IP address in the ARP table. After which, all frames in layer 2 will be funneled to the attacker's port, which is N1 in this case.

Alice uploading a report

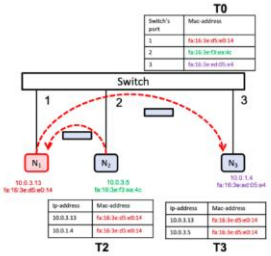


(S1) *Alice was in a cafe with free wifi. Mallory sat at another corner. The cafe owner, Darth, is the system admin of the computer system and wifi router/switch.*

- DNS spoofing possible
- Able to sniff communication and thus able to perform DNS spoofing attack

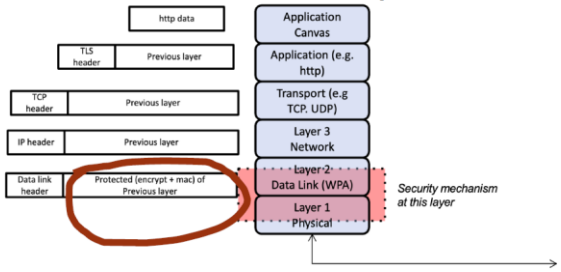


- ARP poisoning possible
- No protection in layer 1 & 2



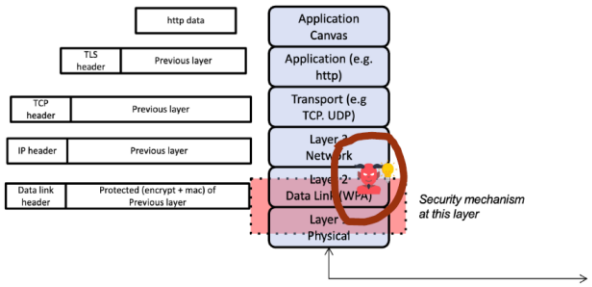
(S2) *Alice was in a cafe with WPA2 personal wifi. Alice knew the password. Mallory sat at the corner. Mallory didn't know the password. Darth was the cafe owner.*

- DNS spoofing NOT possible
- Physical layer encrypted
- Cannot see packet content
- ARP poisoning NOT possible
- Protection in layer 1 & 2
- WPA detects any alteration done by Mallory



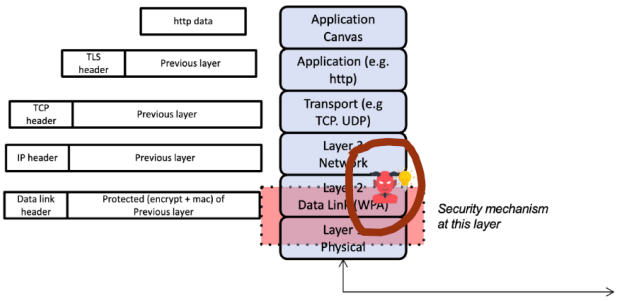
(S3) *Alice was in a cafe with WPA2 personal wifi. Alice knew the password. Mallory sat at the corner. Mallory knew the password. Darth was the cafe owner*

- DNS spoofing is possible
- In WPA2 personal, authenticated key exchange is weak
- No forward secrecy
- Master key here is password
- Attacker could get the session key
- ARP poisoning is possible
- Protection in layer 1 & 2 but Mallory who knows the password is now at data link layer, before the protection



(S4) *Alice was in a clubhouse cafe with WPA2 enterprise wifi Alice had an account to access the wifi. Mallory sat at the corner accessing the same wifi network. Mallory also had an account (but different from Alice). Darth was the cafe owner*

- ARP poisoning is possible, Mallory is at data link layer
- DNS spoofing is possible after Mallory become MITM in the link layer
- Such ambiguous question will NOT appear in exam – Prof Chang ☺



3 possible ways of how a layer 3/4 VPN client handles DNS Query:

Method 1: The VPN client treats the DNS request as a typical UDP packet. The VPN client tunnels the UDP packet to the VPN server.

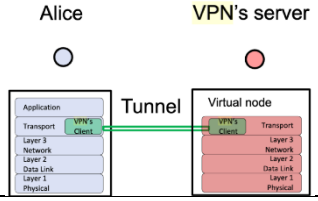
- This might not work as:
  - IP address of the DNS server in the client local network might be different from the IP address of the DNS server in the VPN network.
  - So, the DNS query won't be answer.

Method 2: Once the VPN client has detected that the payload is a DNS query, it does not tunnel it to VPN server. Instead, it let the DNS query go through the usual non-VPN connection.

- For this method, it is susceptible to DNS attack.
- DNS spoofing is possible as the query go through the usual connection

Method 3: Once the VPN client has detected that the payload is a DNS query, it update the address of the "destination ip" and tunnels the UDP packet to the VPN server. This leads to using a DNS server specified by the VPN.

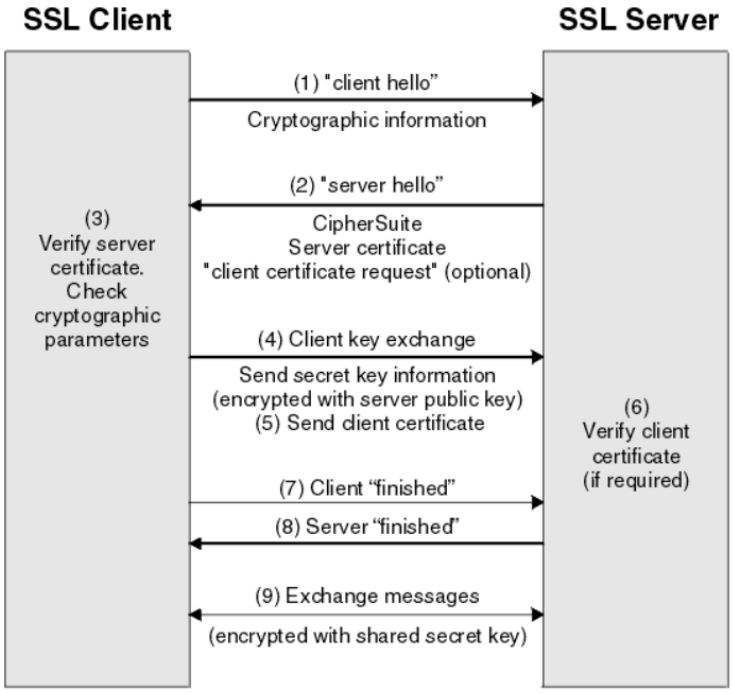
- For this method, it is not susceptible to DNS attacks.
- Mallory is unable to sniff the VPN network.
- The DNS query was tunnelled to the VPN server.
- The tunnel was protected (encrypt and mac).
- Mallory, at best, can only see the encrypted tunnel.



DDOS: ** Using botnet makes it covert.

- **Reflection**: Sends request to intermediate node, which echos/broadcast request and overwhelm the network (victim). Chain effect.
- **Amplification**: Trigger multiple responses from the intermediate nodes. E.g loop the echo multiple times.

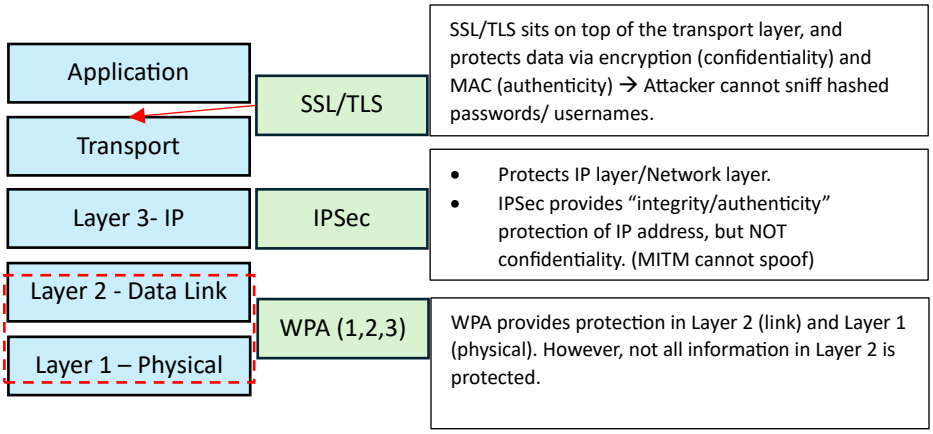
TLS Handshake:



Securing Communication Channels using Cryptography

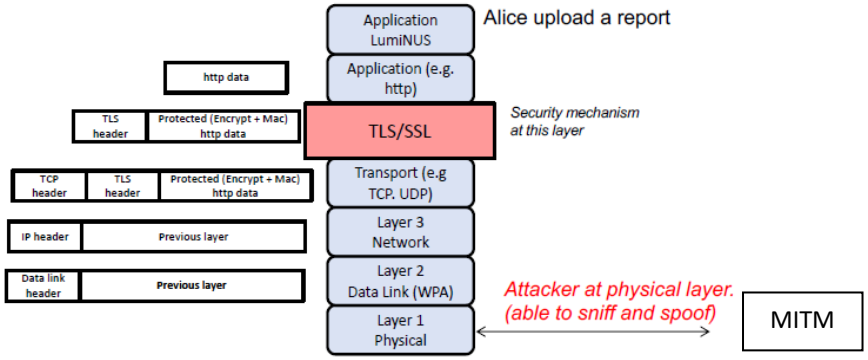
- TLS/SSL, WPA, IPSEC.
- Some protections span across multiple layers/do not provide full protection of the targeted layer.
- A security protocol that protects layer N, will also protect information above it.

Application (e.g HTTP/s)	Protecting Layer 3 also protects Application and Transport layer, but it does not protect the datalink and physical layer.
Transport (e.g TCP/UDP)	
Layer 3 – IP/ Internet Protocol	
Layer 2 – Data Link	
Layer 1 - Physical	



Possible Attacks with SSL/TLS: **Replay attacks as well

Attack 1: Attacker is at Physical Layer



- Suppose that there is an attacker in the physical layer who can sniff and spoof message at that layer (i.e. MITM in the physical layer).
- For example, Alice uploading her report in a cafe using free wifi. Hence anyone in the café who has access to the physical can sniff messages in that layer. (assuming that TLS is the only protection employed)

QNS	Can the attacker on the physical layer know what site Alice is on?
ANS	Yes. The attacker can see the destination IP of what Alice is searching. In this case, the attacker can see the GET request from Alice to Canvas's IP. From there on, the attacker can do a mapping to the IP address to see that Alice is on Canvas.

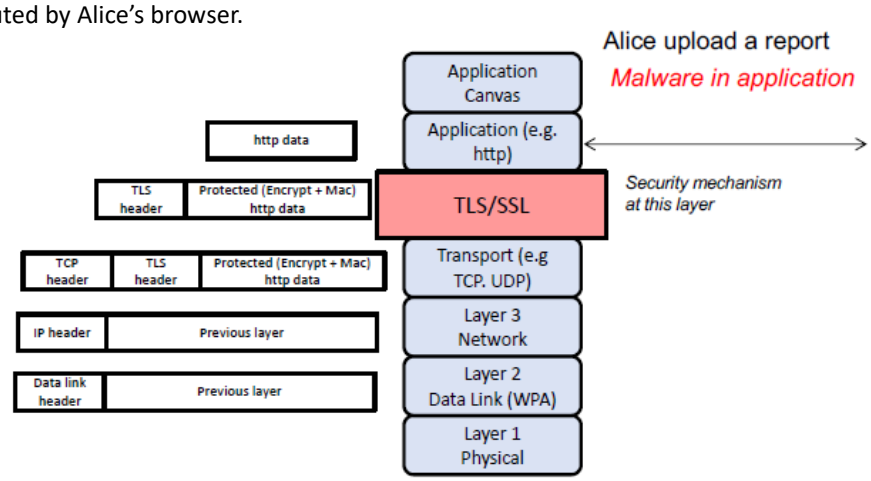
QNS	Can the attacker on the physical layer know what Alice is doing on Canvas?

ANS	No. Canvas uses TLS, and all the data are encrypted. Add on: If Alice is watching Youtube, will the attacker know what Alice is watching? • NO. Youtube also used TLS. The video packets are encrypted. • Possibly. When data is broken down into pieces, based on the timing and size of pieces. - The attacker could find out which particular video Alice is watching via traffic analysis and machine learning. - Its harder to discern what Alice was watching, but it is possible in theory.

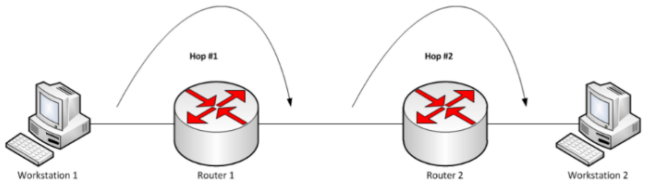
** Note that this case plays out similarly when using a **VPN**. A VPN at the physical layer can see what website we are surfing, but not the details of the transaction.

Attack 2: Attacker is at Application Layer (i.e Malware in Browser)

- Suppose that there is an adversary in the application layer. For example, a malicious java script injected and executed by Alice's browser.



QNS	Can the script know what Alice is searching on Canvas?
ANS	Recall: A security protocol that protects layer k, would protect information in that layer and above. • No. Since this malware sits ontop of the TLS/SSL layer, the TLS/SSL mechanism also protect the application layer and encrypts the data. • As such, the script will not know what Alice is doing on Canvas. Note: • The specific capabilities of the malicious script would determine what information it can potentially steal. Some scripts might try to **capture keystrokes** or form data even if they can't see the encrypted content. • In some cases, the script might gather indirect information about Alice's activity on Canvas by tracking accessed URLs or user interface changes.

QNS	Can the script know Alice's MAC address?
ANS	Unlikely. Mac address likely modified before reaching VPN.  Unless the number of hops <2, the script will not learn of Alice's MAC address. (i.e not a direct hop from sender to receiver). Recall: MAC (src & dest) address updates every time it hops.

WPA2 (Wifi Protect Access II)

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

KRACK Attacks (Key Reinstallation Attacks)

Key Reinstallation Attacks, also known as KRACK Attacks, exploit a vulnerability in the WPA2 protocol to potentially decrypt Wi-Fi traffic.

1. **Reused Nonces**: The attack tricks a device into reinstalling an already used encryption key. This resets a critical security value called the "nonce" used for encryption.
2. **Exploiting Reset**: With the nonce reset, the attacker can potentially manipulate messages and force the device to reuse the same nonce for multiple encryptions.
3. **Decrypting Traffic**: By reusing the nonce, the attacker can potentially decrypt some Wi-Fi data packets if they have enough captured traffic.

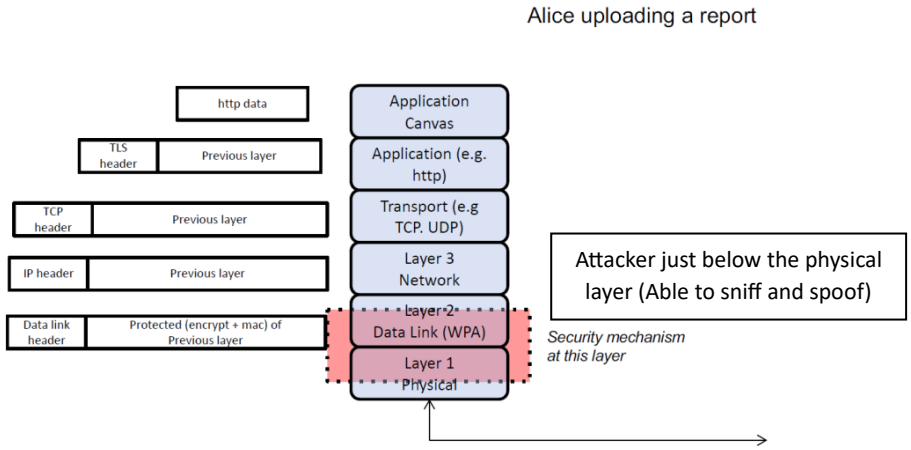
Impact	Protection
This attack can potentially decrypt data on vulnerable Wi-Fi networks, compromising privacy.	• Software updates patching the WPA2 protocol vulnerability are crucial. • Using strong encryption algorithms on websites and services can further mitigate risks.

Note:

- While KRACK Attacks are a serious concern, applying security patches and using reputable Wi-Fi networks significantly reduces the risk.
- There is WPA3 as well. However, it may not be compatible with some devices.
  - WPA3, the next generation of Wi-Fi security, addresses the KRACK vulnerability.
  - While not all devices are compatible yet, upgrading to WPA3 offers stronger protection if your router and devices support it.

Possible Attacks with WPA

Attacker at Physical Layer



Assuming WPA used.

QNS	Can the attacker know that Alice is visiting Canvas?
ANS	Yes. WPA only encrypts data packets, not the destination address. The attacker can see the destination IP address in the packets, revealing the website (Canvas in this case) Alice is visiting.

QNS	Can the attacker know Alice's Mac Address (Link Layer)?
ANS	Yes. MAC addresses are included in the packet header and are not encrypted with WPA. The attacker can see Alice's MAC address in the sniffed packets.

QNS	Can the attacker know what Alice is doing on Canvas?

ANS	No. While the attacker knows the destination and Alice's MAC address, WPA encrypts the actual content of the data packets. This means the attacker cannot see what Alice is typing, searching, or doing on Canvas itself.

Assuming WPA2 used.

QNS	Can the attacker know that Alice is visiting Canvas?
ANS	No. WPA2 encrypts the destination address as well, so the attacker cannot see the specific website Alice is visiting. They would only see encrypted data going to a certain IP address.

QNS	Can the attacker know Alice's Mac Address (Link Layer)?
ANS	Yes. MAC addresses are still included in the packet header and remain unencrypted. The attacker can identify Alice's device by its MAC address.

QNS	Can the attacker know what Alice is doing on Canvas?
ANS	No. WPA2 uses strong encryption algorithms (AES) to scramble the data content within the packets. The attacker cannot decrypt the information and see what Alice is typing, searching, or doing on Canvas.

Difference between WPA and WPA2:

	WPA	**WPA2**
Source /Destination Address	Not encrypted. The attacker can see the destination IP address in the packets, revealing the website (Canvas) Alice is visiting.	Encrypted. The attacker cannot see the specific website Alice is visiting. They can only see encrypted data going to a certain IP address.
Content	Encrypted. The attacker cannot see what Alice is doing on Canvas itself (typing, searching, etc.).	Encrypted (stronger algorithms than WPA). Similar to WPA, the attacker cannot see what Alice is doing on Canvas.
Mac Address	Not encrypted. The attacker can see Alice's MAC address in the sniffed packets.	Not encrypted. The attacker can still identify Alice's device by its MAC address.

Different WPA versions:

	WEP	**WPA**	**WPA2**	**WPA3**
Encryption Method	RC4	Temporal Key Integrity Protocol (TKIP) with RC4	CCMP + AES	AES
Encryption Strength	Weak	Weak	Stronger	Stronger
Session Key Size	40 bit	128 bit	128 bit	128 bit (WPA-Personal) 192bit (WPA3-Enterprise
Cipher	Stream	Stream	Block	Block
Data Integrity	CRC-32	Message Integrity Code	CDC-MAC	Secure Hash Algorithms
Forward Secrecy	No	No	Yes	Yes
Vulnerability to KRACK	Yes	Yes	Yes	No
Address	Not Encrypted	Not Encrypted	Encrypted	Encrypted

IPSec (Internet Protocol Security)

- MITM are unable to "spoof" the source ip-source but can learn the source and destination ip-address of the sniffed packets.
- "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."

Layered Defense

- Typical implementation use TLS/SSL + WPA2.
  - TLS/SSL encrypts the application data itself, protecting the content from eavesdroppers.
  - WPA2 encrypts the data packets at the network layer, securing transmission and preventing unauthorized access to the data stream.
- IPSEC turns out to be expensive to implement and difficult to deploy, since it "touches" the OS).

VPN (Virtual Private Network)

- A Virtual Private Network (VPN) creates a secure tunnel over a public network like the internet.
- It encrypts your data traffic and hides your original IP address, protecting your online privacy and security.

How VPN works:

1. **Initiating the Connection**: You connect your device (computer, phone, etc.) to a VPN client application.
2. **Tunnel Establishment**: The VPN client contacts a VPN server operated by the VPN service provider. They establish a secure connection using a VPN protocol (more on that later).
3. **Data Encryption**: All your internet traffic gets encrypted using the chosen VPN protocol before being sent through the internet tunnel.
4. **Data Routing**: The encrypted data travels through the internet tunnel to the VPN server.
5. **Decryption and Forwarding**: The VPN server decrypts the data and forwards it to the website or service you're trying to access. It appears to originate from the VPN server's IP address.
6. **Response Processing**: The response from the website/service travels back through the tunnel, gets encrypted again by the server, and is sent back to your device.
7. **Decryption and Delivery**: The VPN client on your device decrypts the final data and delivers it to you as usual.

Benefits of using VPN:

- **Enhanced Privacy**: Encrypted traffic and hidden IP addresses make it harder for anyone to track your online activity or identify your location.
- **Security on Public Wi-Fi**: VPNs protect your data from snooping on unsecured public Wi-Fi networks.
- **Geo-restriction Bypassing**: Some VPNs allow you to access websites or services restricted in your region by appearing to connect from a different location.

Different types of VPN (Different Layers)

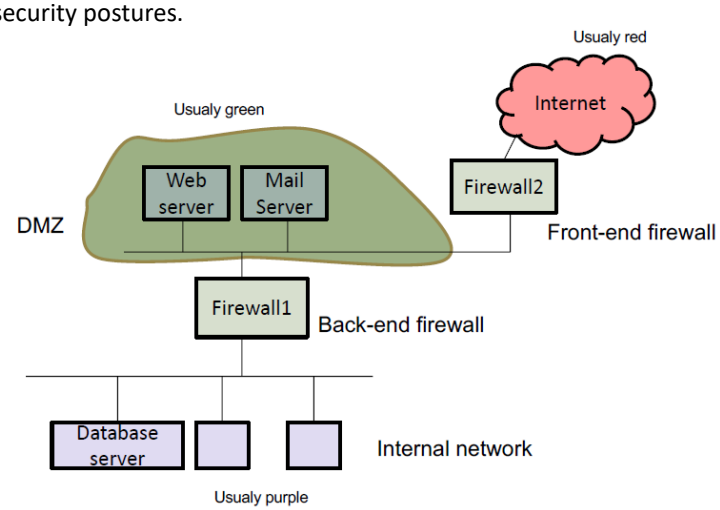
Layer 2 (Data Link Layer)	• Tunnelling Protocols (L2TP) • Operates at the data link layer (Layer 2) of the OSI model. • L2TP itself doesn't provide encryption, but it's often used in conjunction with other protocols like IPSec for a secure tunnel. • Less popular
Layer 3 (Network Layer)	• OpenVPN. • A popular open-source Layer 3 (network/IP layer) protocol known for its flexibility, security, and customization options.
Application Layer	• Secure Sockets Layer/ Transport Layer Security (SSL/TLS) • These operate at the application layer. • While not technically a VPN protocol on its own, SSL/TLS provides encryption for specific applications like web browsing (HTTPS). • Some VPN services might utilize SSL/TLS tunnels for specific purposes.

Layer 3 and Layer 2 VPN Comparison

Layer 3 VPNs	Layer 2 VPNs
• Provider devices forward customer packets based on Layer 3 information (for example, IP).	• Provider devices forward customer packets based on Layer 2 information.
• **SP Involvement in Routing**	• **Tunnels, Circuits, LSPs, MAC Address**
• MPLS/BGP VPNs (RFC 2547), GRE, Virtual Router Approaches	• Pseudo-Wire Concept

Firewalls

- A Firewall controls what traffic is allowed to enter the network (ingress filtering) or leave the network (egress filtering).
  - Firewall, Intrusion detection system (IDS) are tools to control access to the network.
  - Whitelist → Drop all packets except those in white list
  - Blacklist → Accept all packets except those in black list.
- Firewall are devices or programs that control the flow of network traffic between networks or hosts that employ differing security postures.



- Principle of Least Privilege, Compartmentalization.

Firewall Design

- A firewall enforces a set of rules provided by the network administrator.
- Example on the 2-firewall setting:
  - Rules for Firewall1.
    - Block: HTTP.
    - Allow internal to Mail Server: SMTP, POP3
  - Rules for Firewall2
  - Allow from anywhere to Mail Server: SMTP only
- How the rules are to be specified differ on different devices and software.

Rule	Type	Direction	Src Addr	Dest Addr	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

- 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".

Types of Firewall

- NIST's document (NIST 800-41) categorises firewalls into 3 types:

SN	Type	Description
1	Packet Filters	Inspect only the header (mainly IP packet's header)
2	Stateful Inspection	Keep a state on previously received packets. E.g counting the number of connections a particular IP address has made in the past hour.
3	Proxy	Modifies packets.

Intrusion Detection System (IDS)

- 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. 3 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).

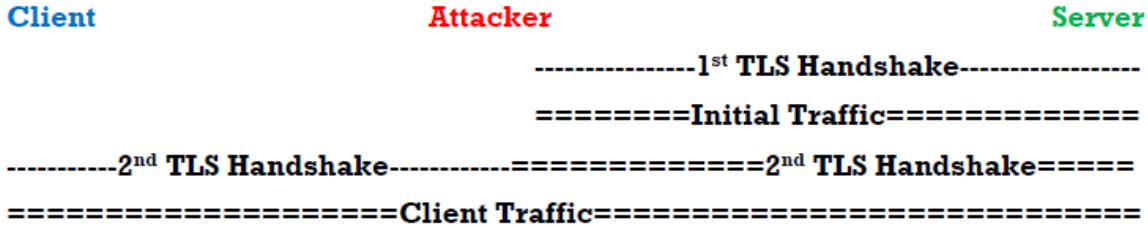
Management in Network Security

- Management needed to monitor and adjust network.

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)

Re-negotiation Attack:

- A renegotiation attack is a vulnerability in the process used to establish a secure connection with encryption protocols like SSL/TLS.
- Renegotiation attack compromises the integrity of the system.
- The vulnerability lies in how the renegotiation process is handled. The attacker might be able to manipulate or inject data during this handshake.
  - **Downgrade Attack**: The attacker might downgrade the connection to a weaker encryption algorithm or even an unencrypted connection.
  - **Man-in-the-Middle Attack**: The attacker might inject themselves into the renegotiation process, becoming a "man-in-the-middle" and intercepting encrypted traffic.
  - **Denial-of-Service Attack**: In some cases, the attacker might exploit the renegotiation process to crash the server or client, causing a denial-of-service attack.



QNS	The TLS carries out an unilateral authentication. Which entity Server or Client, carries out the verification?
ANS	• Client verifies the server. • Client checks the validity of the server's certificate.

QNS	While TLS performs unilateral authentication, the application still has to authenticate the client. How does the application verify the client?
ANS	• The application can verify the client via an authentication token stored in the cookie.

QNS	How does the Client and the Attacker obtain the Server's public key?
ANS	• The server's public key can be obtained from the server's certificate. • The server sends its certificate in the TLS handshake step.

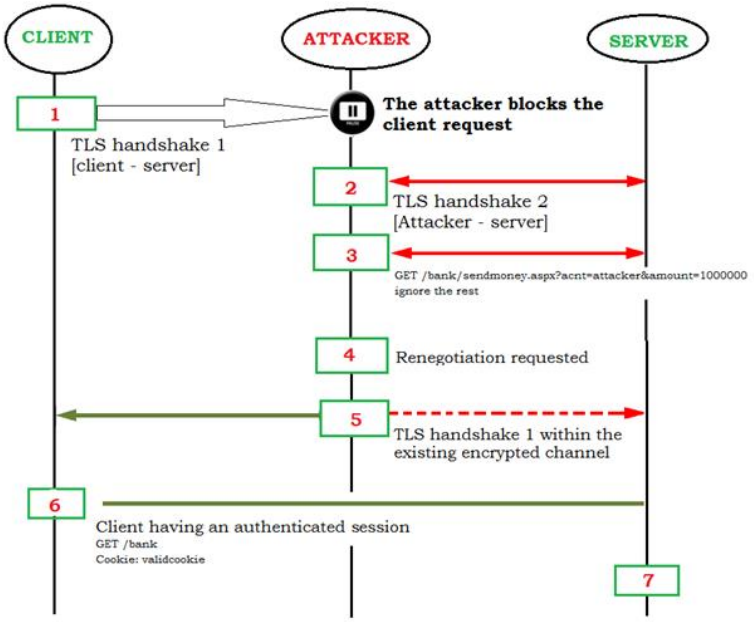
QNS	From the Server's POV, the Server has carried out 2 handshakes. Let K_1 and K_2 be the session keys established during the 1st and 2nd handshakes. Does the attacker know K_2?
ANS	• No. The attacker does NOT know K_2, • K_2 was exchanged between the client and the server in the 2nd TLS handshake. • Ther is no way the attacker can obtain K_2. • Property of Diffie-Hellman Key exchange.

Note:

- K_1 is used to protect the "Initial Traffic", which is obtained from the 1st TLS handshake.\
- The 2nd TLS handshake carried out between a) Client-Attacker, b) Attacker-Server.
  - a) The communications between Client and Attacker are NOT protected by the session key, K_2
  - b) The communications between Attacker and Server are protected by the session key, K_2
- Key, K_2 is used to protect the "Client Traffic", which is derived in the 2nd TLS handshake done before the Client Traffic is communicated.
- Renegotiation attack on TLS DOES NOT compromise confidentiality.
  - Since the Client Traffic is encrypted by K_2.
  - Attacker cannot steal anything from m, confidentiality is not compromised. However, since the attacker can prepend a string payload, p to the message m, where m' = p ||m, the integrity of the system is compromised.
- We can use MAC using the secret key previously established with the Client to prevent renegotiation attacks.
  - E.g using the secret key, MAC the finished message with handshake number.
- We cant just disable renegotiation in TLS, as it will affect availability since some applications may need the renegotiation feature of the TLS protocol.

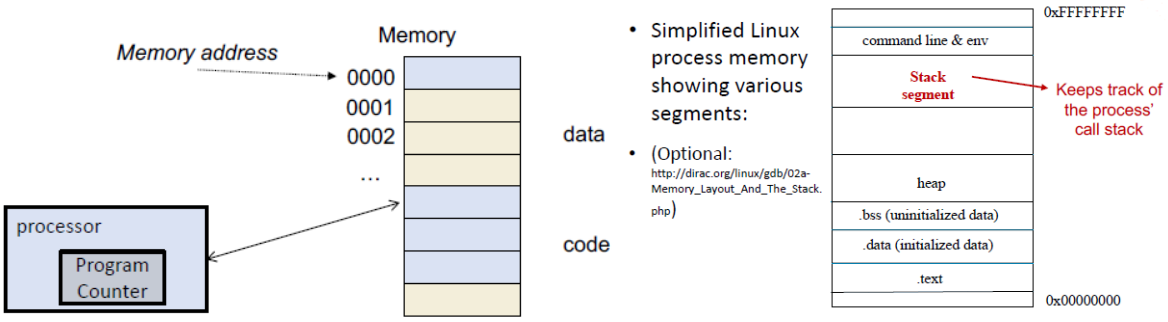
Variants of Renegotiation Attacks:

Attacker blocks client's request, and spoofs TLS handshake.



Secure Programming

- The code are treated as data and both are stored in the same memory.

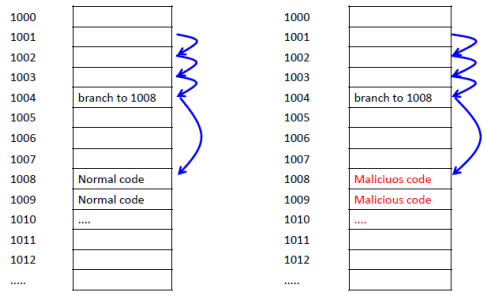


- Execution of code is done sequentially in the memory, with PC guiding the process.
  - Direct Branch: Replace by a constant value specified in the instruction
  - Indirect Branch: Replace by a value fetched from the memory.
- We can make use of such memory structures to compromise the memory integrity of the code, by compromising the control flow integrity.
- Suppose an attacker can modify some memory, the attacker could compromise the execution integrity by either:
  - directly modifies code in the memory; or
  - modifying the address in indirect branch.

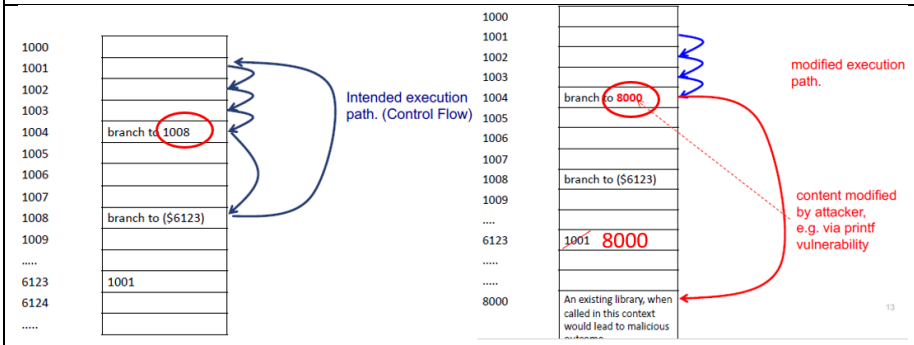
Possible Attacks

1. Overwrite existing code portion with malicious code
2. Overwrite a piece of control-flow mechanism
   a) Attack 1: Replace a **memory location** storing a code address that is used by a **direct jump**
   b) Attack 2: Replace a **memory location** storing a code address that is used by an **indirect jump**
      - This is known as Stack Smashing (Buffer Overflow)
      - Stack Smash has 2 effects:
        1. If return address is modified, the control flow is compromised.
        2. If local variables are modified, the computed result will be wrong.

Attack 1: Replace the Code



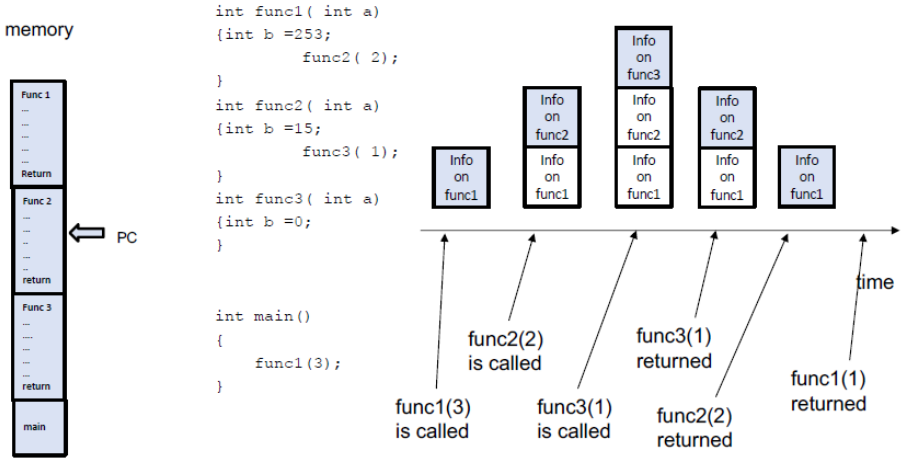
Attack 2a,b: Modify Memory Location that stores code



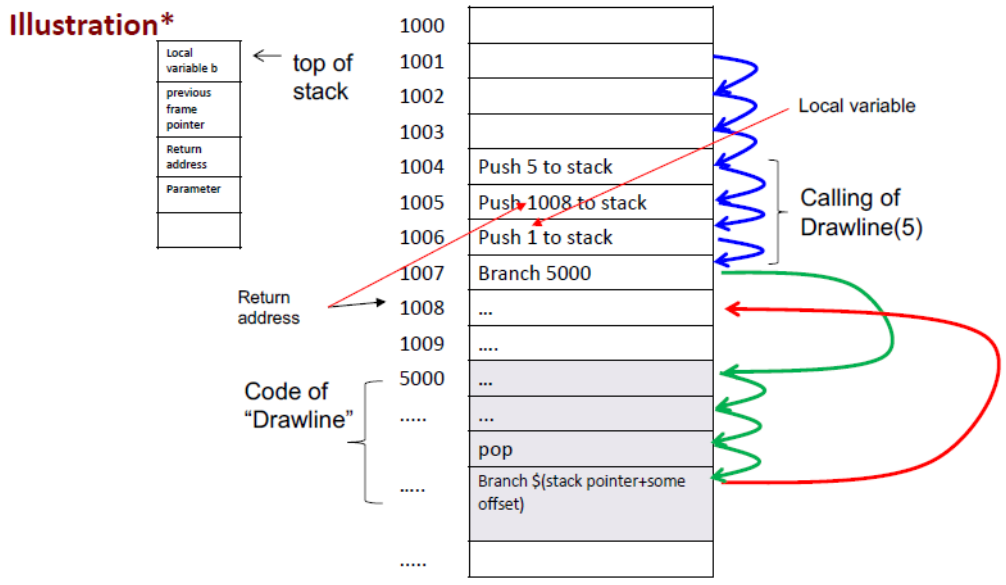
Attacker modifies the execution path, and modify memory values via buffer overflow

Stack: (Execution Stack, Call Stack)

- A call-stack is a stack that store important information of the function calls.
- Each element of the stack is called a stack frame.
- Stack pointer is a variable that stores location* of the first element.
  - There are two operations in stack: Push and Pop. (Last-In-First-Out).
  - The stack can either grow upwards or downwards
  - A value that is the location of memory is often referred as a “pointer”. It ”points” to a memory
- During execution, a stack is maintained to keep control flow information and different runtime environment. This stack is known as Call stack.
- Call Stack keeps track of:
  - The parameters to be passed
  - Control Flow Information: Return address, loops etc, PC
  - Local variables and functions.
- Each call of a function adds a stack frame to the memory.



- Call Stack



Unsafe Function, printf()

How to prevent such attacks(only in compilers):

- Most compilers can statically check format strings and show warnings for suspicious or dangerous formats.
- We can append these compiler flags to detect bad format strings known ONLY AT COMPILE TIME.
  - -Wall, -Wformat, -Wno-format-extra-args, -Wformat-security, -Wformat-nonliteral
- If the format string comes from the user at RUNTIME → Woops, compiler cant help.
  - E.g scanf, then printf

Functions and Parameters for Format String Attack

Format Functions:

Format Function	Description
fprint	Writes the printf to a file
printf	Output a formatted string
sprintf	Prints into a string
snprintf	Prints into a string checking the length
vfprintf	Prints the a va_arg structure to a file
vprintf	Prints the va_arg structure to stdout
vsprintf	Prints the va_arg to a string
vsnprintf	Prints the va_arg to a string checking the length

Common Parameters used in a Format String Attack:

Formatters	Output	Passed as:
%%	% character (literal)	Reference
%p	External representation of a pointer to void	Reference
%d	Decimal	Value
%c	Character	
%u	Unsigned decimal	Value
%x	Hexadecimal	Value
%s	String	Reference
%n	Writes the number of characters into a pointer	Reference

- "%x" reads data from the stack
- "%s" reads char strings from the process' memory
- "%n" writes an integer to locations in the process' memory.
- If an application acceps the format and parses the format as shown in the table above, high chance it is vulnerable to format string attacks.

Example:

```
#include  <stdio.h>
void main(int argc, char **argv)
{
   // This line is safe
   printf("%s\n", argv[1]);

   // This line is vulnerable
   printf(argv[1]);
}
```

If we compile and run: ./example "Hi %s%s%s%s%s%s"

- The printf in the second line will interpret the %s%s%s%s%s%s in the input string as a reference to string pointers.
- It will try to interpret every %s as a pointer to a string, starting from the location of the buffer (probably on the Stack).
- At some point, it will get to an invalid address, and attempting to access it will cause the program to crash. (Segmentation Fault)

- Apart from crashing the application/software by accessing forbidden memory, an attacker can also use this method to get information (different payloads)
  - ./example "Hello World %p %p %p %p %p %p"
  - The following lines will be printed (or similar):

```
 SAFE: printf("%s\n", argv[1]);Hi %p %p %p %p %p %p
UNSAFE: printf(argv[1]);Hi 000E133E 000E133E 0057F000 CCCCCCCC CCCCCCCC CCCCCCCC
```

  - reading and writing to any memory location is possible in some conditions, and even code execution.

- If no parameters are supplied to the format string, the compiler will "compute" the value for that variable from the stack. It doesn't check 😟

DON'T DO THIS	DO THIS
printf (t) printf (t, a1, a2)	printf ("hello"); printf ("The value of %s is %d", a1, a2)

How such printf() vulnerability can be exploited:

- If a program is vulnerable, the attacker might be able to:
  1. obtain more information (confidentiality)
  2. cause the program to crash, e.g. using %s. (execution integrity)
  3. modifying the memory content using "%n". (memory integrity which might lead to execution integrity)
- If the program that invokes printf has elevated privilege (set UID enabled), a user (the attacker) might be able to obtain information that was previously inaccessible. E.g
  - Suppose the program for a web-server has the unsafe printf.
  - Under normal usage, the server would obtain a request from the client, and then display (via the printf) it for confirmation.
  - Now, a client (the attacker) might able to submit a web request to obtain sensitive information (e.g. the secret key), or to cause the web-server to crash.

Data Representation

Different parts of a program (or system) adopts different data representations. Such in-consistencies could lead to vulnerability. → URL spoofing, hostname check bypassing.

UTF-8 Encoding Data Representation

- UTF-8 is a variable-width Unicode encoding that encodes each valid Unicode code point **using one to four 8-bit bytes.** UTF-8 is a standard for representing Unicode numbers in computer files.
- UTF-8 is backwards compatible with ASCII, (1st 128 characters)
- Data Representation of UTF-8:
  - U000000-U00007F: 0xxxxxxx 7 bits
  - U000080-U0007FF: 110xxxxx 10xxxxxx 11 bits
  - U000800-U00FFFF: 1110xxxx 10xxxxxx 10xxxxxx 16 bits
  - U010800-U10FFFF: 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx 21 bits
    - ➢ the xxx bits are the least significant bits of the respective Unicode character.
    - ➢ By the above rules, byte representation of an UTF-8 character is unique.
    - ➢ However, many implementations **accepts multiple and longer "variants" of a character**.
- Different representation of the same UTF-8 Character:
  - E.g the character '/' → **ASCII: 0b0010 1111, 0x2F**
  - Under UTF-8 definition, the 1 byte representation for '/' is 2F (unique)
  - However, in many implementations, the following longer variants may also be treated to be the same as '/' → Not unique mapping already

    (2 bytes version) 11000000 10101111
    (3 bytes version) 11100000 10000000 10101111
    (4 bytes version) 11110000 10000000 10000000 10101111
  - All the versions above will all be translated (decoded) to '/'
  - This may result in inconsistencies between the verification process and actual usages.
- An attacker will just need to substitute any usage of '/' in their injection with the 1,2,3 or 4 bytes version of '/' in UTF-8. Using HEX values + % formatter
  - E.g To access password.txt: /home/…/password.txt →/home/…%2Fpassword.txt

IP Address Data Representation

- An ip address has 4 bytes. There are many ways to represent ip address in a program.
- An ip address can be represented as
  1. A string, e.g "132.127.8.16". This is human readable.
  2. 4 integers, and each is a 32-bit integer.
     - Formula: IP = $a \times 2^{24} + b \times 2^{16} + c \times 2^{8} + d$
     - Where AAA.BBB.CCC.DDD is the int in the IP address segment.
  3. A single 32-bit integer. Note that 32 bits = 4 bytes. Etc…
- IP address range: 0.0.0.0 to 255.255.255.255
  - To access the IP address 11.12.**1.0** (blacklisted IP, e.g due to censorship)
  - We can trick the BL checkers + IP int checker by using overflows.
  - We just input the IP as 11.12.**0.256** !
    - The 256 in block d will overflow into the c block → 11.12.**1.0**
  - Both IP int will be the same:
    - 11.12.1.0: 185336064
    - 11.12.0.256 : 185336064
- To fix this issue → Just use Canonical representation. DON'T trust the input from the user.
- Always convert it to a standard (i.e canonical) representation

Buffer Overflows

- Recall: C & C++ does not employ "bound checks" during runtime.
- Hence, while efficient, they are buggy!
- In general, buffer overflow refers to a situation where data are written beyond the (buffer's) boundary.
- Example of an unsafe function → strcpy(s1, s2). DOES NOT CHECK FOR BUFFER SIZE
- Use strncpy(s1, s2, n) instead. → At most n char copied

Special Cases:

- Stack smashing → Buffer overflow that targets stack (aka stack overflow, stack overrun)
  - If the stack buffer is filled with data supplied from an untrusted user then that user can corrupt the stack in such a way as to inject executable code into the running program and take control of the process.
- Usage of **Canary** can detect stack overflows, but it is not a foolproof mechanism.
  - Canaries are secrets inserted at carefully selected memory locations during runtime.
  - Checks are carried out during runtime to make sure that the values are not being modified.
  - IF these values are touched/ modified → Program halts.
  - Need to keep the value as "secret". If the attacker happens to know the value, it may able to write the secret value to the canary while over-running it.

QNS	What is the disadvantage of having the canary protection?
ANS	Takes up more space.

QNS	Consider a C program t.c compiled using gcc and to be run in a particular OS, say ubuntu. Who should implement the canary protection?
ANS	MY ans: Gcc compiler. Why? We cant have the user write the canary, since the code space will also take up the same memory space, and it is susceptible to stack overflow. Its ownself check ownself → ineffective. • In C, by default, canary is on. To off it: *** DEPENDS ON GCC version, some is off - gcc programname.c -fno-stack-protector

Integer Overflow

- The integer arithmetic in many programming language are actually "modulo arithmetic".
- Suppose a is a single byte (i.e. 8-bit) unsigned integer, in the following C and java statements what would be the final value of a?
  - a = 254;
  - a = a + 2;
    - ➢ Its value is 0.
    - ➢ The addition is with respect to module 256.
    - ➢ Hence, the following predicate is not necessary always true.
    - ➢ (a < (a+1))

Code (Script) Injection

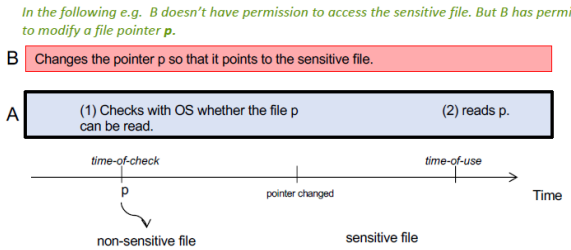
- Tool: sqlmap
  - sqlmap is an open source software that is used to detect and exploit database vulnerabilities and provides options for injecting malicious codes into them.
  - It is a penetration testing tool that automates the process of detecting and exploiting SQL injection flaws providing its user interface in the terminal.
- **Scripting Language**
  - In the context of security, mixing "code" and "data" is potentially unsafe. Many attacks inject malicious codes as data.
  - However, in most computer architecture, code is sometimes treated as data.
  - Scripting languages are vulnerable to injections.
  - To remedy this, layer the query, e.g: SQL ↔ webserverchecker ↔ User
    - ➢ This prevents users from injecting into SQL, as there is an intermediate server that checks and filters the data from the scripts.

Undocumented Access Points (Easter Eggs)

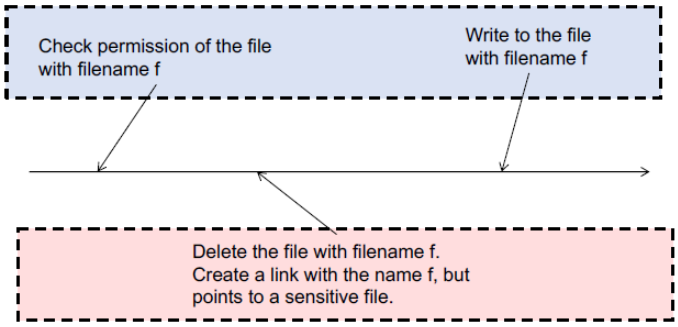
- Sometimes, programmers may insert "undocumented access points" (backdoors) to inspect the states of a program during debugging.
- These access points, if not removed and mistakenly left in the final product leaves a backdoor for attackers to exploit (aka easter eggs)

Race Conditions (TOCTOU)

- Generally, race condition occurs when multiple processes (running in parallel) access a piece of shared data in such a way that the outcome depends on the sequence of accesses.
- In the context of security, the "multiple processes" typically refer to
  1. a process A that checks the permission to access the data, follow by accessing the data, and
  2. another process B (the malicious one) that "swaps" the data.
- There is a "race" among A and B. If B manages to be completed in between the **time-of-check and time-of-use** in A, the attack succeed. Thus, this type of race condition is also known as **time-of-check-time-of-use** (TOCTOU).

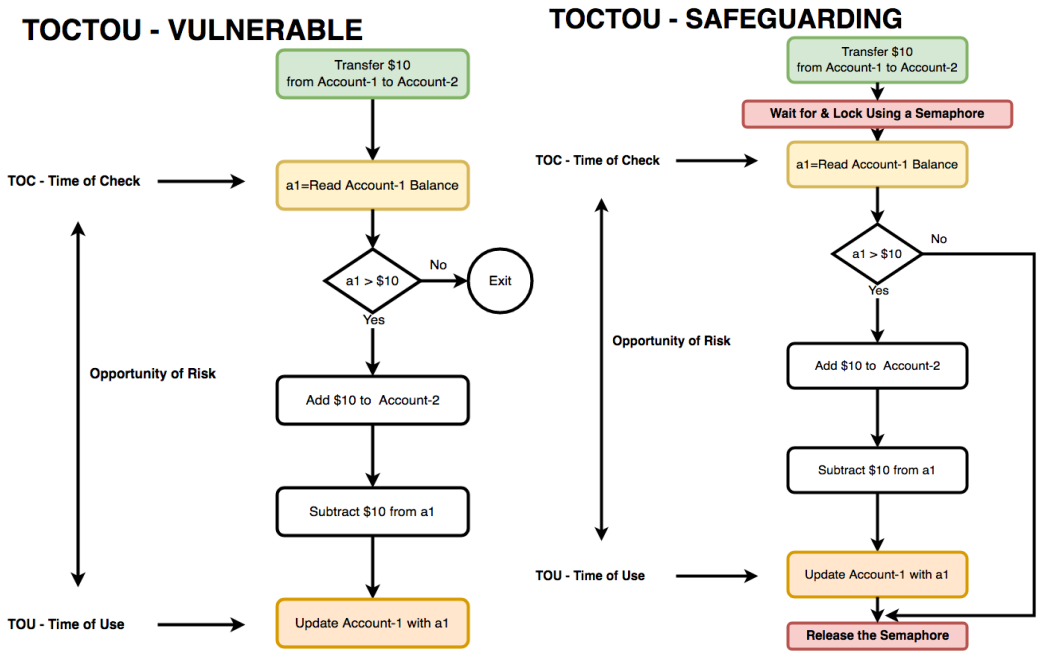


- Spam attack until you "lose race"
  - More specifically, we just want to update the pointer between TOC and TOU.
  - Recall race condition → slowest executed code wins the final output



/usr/year1/bob/list.txt

/usr/course/cz2017/grade.txt



Protection & Good Practices:

- Input validation, filtering , parameterised queries → can prevent most attacks.
- Use safe functions. → weakest link... if 1 unsafe function → whole code is vulnerable.
- Bounds Check & Type Safety → Prevents memory leaks + prevents overflow.
- Canary and Memory Projection → prevents stack overflow.
- Memory Randomization → Store data and code at separate locations → Segmentation scheme (CS2106)
- Code inspection/Testings → Find issues before they become an issue.
  - White-Box Testing → The tester has access to the source code.
  - Black-Box Testing → The tester does not has access to the source code.
  - Grey-Box Testing → A combination of the above.
- Principle of Least Privilege → do not give the users more access rights than necessary, and do not activate options unnecessary.
- Patching:
  - vulnerability is discovered → affected code is fixed → the revised version is tested → a patch is made public a patch is applied.
  - It is crucial to apply the patch timely. Although seems easy, applying patches is not that straightforward. For critical system, it is not wise to apply the patch immediately before rigorous testing. Patches might affect the applications, and thus affect an organization operation.

** number of successful attacks goes up after the vulnerability/patch is announced, since more attackers would be aware of the exploit.

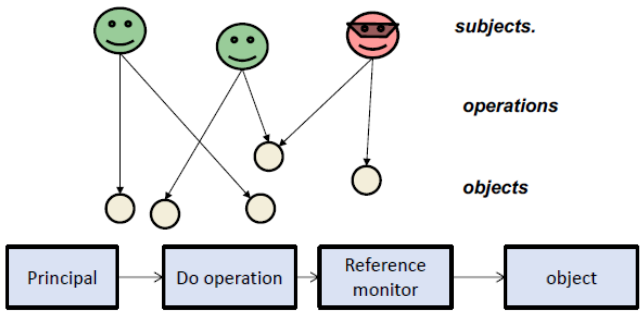
Access Control

- The Access Control setup security perimeter/boundary.
- The boundary will protect the resources within the perimeter. Any malicious activities (e.g malware) outside the boundary will NOT affect the resources inside the boundary.
- Note: Any malicious activities WITHIN the boundary stays WITHIN the boundary.
- Design of the Boundary is guided by 4 principles:

1. **Principle of Least Privilege**
   - Give only the minimum level of access or permissions necessary to perform their tasks. (i.e give users the least amount of privilege required for them to do their job effectively), minimizing the potential damage if their account is compromised.
   - It is better to grant apps with the least privilege, to preserve CIA triad
2. **Compartmentalization**
   - Dividing the systems/networks into distinct compartments or segments.
   - Each compartment has its own set of controls and boundaries, which helps to contain breaches and limit the spread of damage if one compartment is compromised.
3. **Defense in Depth/ Swiss Cheese Model**
   - Layering multiple security measures throughout a system or network.
   - Idea: Although each layer of security may have its weaknesses, when stacked together, they create a robust defense. (More hoops to go through).
4. **Segregation of Duties.**
   - Separating tasks and responsibilities among different individuals or systems to prevent any single entity from having complete control or authority.
   - This helps to reduce the risk of errors, fraud, or abuse.

Principal/Subject, Operation, Object

- In an Access Control Model, we want to restrict **operations** on **objects** by **subjects/principal**.



- A **principal (or subject)** wants to access an **object** with some **operation**.
- The reference monitor either grants or denies the access.

Principals vs Subjects	
Principal	The human users
Subjects	The entities in the system that operate on behalf of the principals
The accesses to objects can be classified to the following: (r,w,e)	
Observe	E.g Reading a Fille
Alter	E.g Writing, Deleting, Changing properties
Action	E.g Executing a program

Ownership

- Every object has an "owner"
- There are 2 options to decide the access rights of an object:
  1. The owner of the object decides the rights (aka DAC, Discretionary Access Control)
  2. A system-wide policy decides. (aka MAC, Mandatory Access Control).
     - Note: MAC are strict rules that everyone MUST follow.

Access Control Matrix

- An Access Control Matrix is a security model used in computer systems to define and enforce access rights to resources such as files, directories, and applications.
- It's a table that specifies which users or groups have permissions to access specific resources.

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

r → read
w → write
x → execute
s → execute as owner
o → owner

- Although the above access control matrix can specify the access right for all pairs of principals and objects, the table would be very large, and thus different to manage.
- Hence, it is seldom explicitly stored. (Large space needed)
- Access Control Matrix are often represented in 2 different ways: ACL or Capabilities.

Access Control List (ACL) and Capabilities

ACL (Per Object)

An ACL stores the 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})

- Straightforward to manage the accesses to resources by associating permissions with objects.
- However, they can become complex to manage, and inefficient. (Per Object)
- It is also difficult to obtain the list of objects a particular subject has access to.

Capability (Per Subject)

- A subject is given a list of capabilities, where each capability is the access rights to an object.
- "a capability is an unforgeable token that gives the possessor certain 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})

- Provide flexibility in access control by associating permissions with subjects.
- However, they can be complex to track (Per Subject)
- It is also difficult to get the list of subjects who have access to a particular object.

- ACL and Capability is a transpose of each other.
- Most of the time, such detailed management of access control is not practical, as they are very specific and is not viable in large datasets (not efficient). Moreover, it is very tedious for 1 person to manage all the access for everyone.
- As such, we will make use of intermediate controls to remedy this overhead.

Intermediate Control (Often achieved via grouping & Protection Rings)

Groupings

- The grouping can be determined by the "role" of the subject.
- A role associates with a collection of procedures. To carry out these procedures, access rights to certain objects are required.
- Privilege → describes access rights
- Privilege can also be viewed as an intermediate control, which can be represented by a number.
  - The higher the number, the higher the privilege.
  - If a subject can access an object, another subject with a higher privilege can also access that object.

Protection Rings

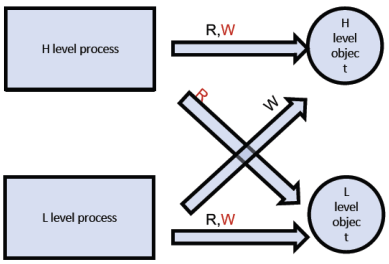
- Protection Rings is the layer of privileges in an OS.
  - There are 2 rings in Unix, Superuser and User.
- In a protection ring, each object (data) and subject (process) is assigned a number.
  - Object with smaller number are more important (higher privilege)
  - This is opposite to Grouping's convention.
- 2 well known models for Protection Rings:

Bell-LaPadula	• Focusses on Data Confidentiality, and there is controlled access to classified data. • WURD → Write Up, Read Down. - Subject at a given security level may not read an object at a higher security level - Subject at a given security level may not write to any object at a lower sec level. • A subject can append to objects at higher security level. It is also possible that, by appending to an object, one could distort its original content. (Renegotiation Attacks)
Biba	• Focusses on Process Integrity, and subjects may not corrupt data in a level higher than itself/ be corrupted by data from a lower level than itself. • RUWD → Read Up, Write Down. - No Write Up (I.e only Write Down): ➢ 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 (I.e only Read Up) ➢ A subject does not has read access to objects lower level. This prevents a subject from reading data poisoned by lower level subjects. • 3 goals of data integrity preservation: 1. Prevent data modification by unauthorized parties 2. Prevent unauthorized data modification by authorized parties 3. Maintain internal and external consistency (i.e. data reflects the real world)

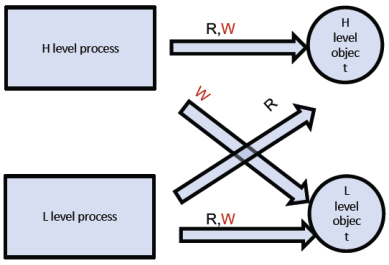
Summary of Bell-LaPadula and Biba Models:

Model	Read?	Write?	For?
Bell-LaPadula	Read Down	Write Up	Confidentiality
Biba	Read Up	Write Down	Integrity

- If a model imposes both Biba and Bell-LaPadula, subjects can only read/write to objects in the SAME LEVEL (which is not practical)!
- Direction of Information flow in the 2 models:



Bell-LaPadula (confidentiality)
No "sensitive" information leaking down



Biba (Integrity)
No "malicious" information going up.

Unix File System

- In Unix, objects includes files, directories, memory devices and I/O devices.
- All these resources are treated as files.

Permission Types

- **read** → grants the ability to read a file.
  - File objects can read the names, but not to find out any further information about them such as contents, file type, size, ownership, permissions.
- **write** → grants the ability to modify a file.
  - grants the ability to modify entries in the directory, which includes:
    - creating files,
    - deleting files, and
    - renaming files.
  - This requires that execute is also set; without it, the write permission is meaningless for directories.
- **execute** → grants the ability to execute a file.
  - This permission must be set for executable programs, for the OS to run them.
  - When set for a directory, the execute permission is interpreted as the **search permission**:
    - it grants the ability to access file contents and meta-information if its name is known, but not list files inside the directory, unless read is set also.

Symbolic Notations

Three Permission Triads		Each Triad	
1st Triad	What Owner can do	1st Character	r: readable
2nd Triad	What Grp member can do	2nd Character	w: writable
3rd Triad	What Other (World) can do	3rd Character	x: executable s or t: setuid/setgid or sticky (also exec) S or T: setuid/setgid or sticky (not exec)

The following are some examples of symbolic notation:

- -rwxr-xr-x: a regular file whose user class has full permissions and whose group and others classes have only the read and execute permissions.
- crw-rw-r--: a character special file whose user and group classes have the read and write permissions and whose others class has only the read permission.
- dr-x------: a directory whose user class has read and execute permissions and whose group and others classes have no permissions.

In some permission systems additional symbols in the ls -l display represent additional permission features:

- + (plus) suffix indicates an access control list that can control additional permissions.
- . (dot) suffix indicates an SELinux context is present. Details may be listed with the command ls -Z.
- @ suffix indicates extended file attributes are present.

https://en.wikipedia.org/wiki/File-system_permissions

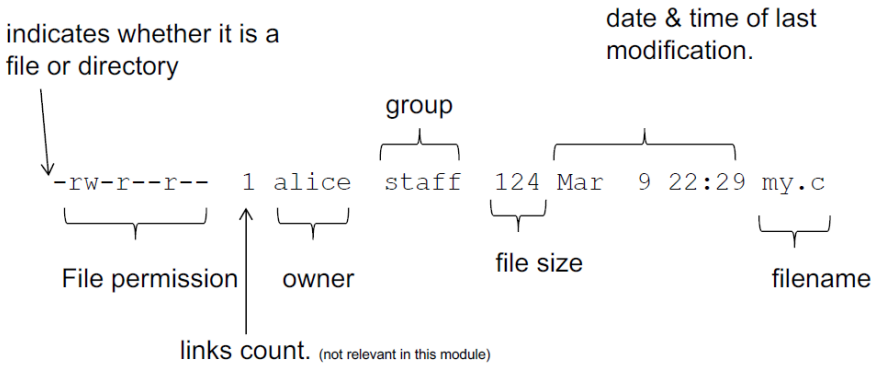
Numeric Notation

- Each of these digits is the sum of its component bits in the binary numeral system. As a result, specific bits add to the sum as it is represented by a numeral:
  - The read bit adds 4 to its total (in binary 100),
  - The write bit adds 2 to its total (in binary 010), and
  - The execute bit adds 1 to its total (in binary 001).

Examples:

Symbolic Notation	Numeric Notation	Meaning
- - - - - - - - - -	0000	NO Permission
- r w x- - - - - -	0700	Can read, write, & execute, only for owner
- r w x r w x- - -	0770	Can read, write, & execute, for owner and group
- r w x r w x r w x	0777	Can read, write, & execute, for owner, group & others
- - - x - - x - - x	0111	Can Execute (owner, group, others)
- - w - - w - - w -	0222	Can Write (owner, group, others)
- - w x - w x - w x	0333	Can Write and Execute (owner, group, others)
- r - - r - - r - -	0444	Can Read (owner, group, others)
- r - x r - x r – x	0555	Can Read and Execute (owner, group, others)
- r w - r w - w r -	0666	Can Read and Write (owner, group, others)
- r w x r - - - - -	0740	Only Owner can Read, Write, Execute Group can only read, Others have NO permissions.

Summary of File System Permission (Not tested)



- The file permission are grouped into 3 triples, that define the read, write and execute access for owner, group, other (aka world).
- A "-" indicates access not granted.
- r : read
- w: write (including delete)
- x: execute (s: allow user to execute with the permission of the owner)

Unix: Checking rules for file access

- Recall that each objects are files, with each file associated with a 9-bit permission.
- Each file is owned by a user, and a group.
- When a user (subject) wants to access a file (object), the following are checked in order:
  1. If the user is the owner, the permission bits for **owner** decide the access rights.
  2. If the user is NOT the owner, but the user's group (GID) owns the file, the permission bits for **group** decides the access rights.
  3. If the user is NOT the owner, nor the member of the group that own the file, then the permission bits for **other** decides.
- The owner of a file, or the superuser CAN CHANGE the permission bits. (Chmod)

Controlled Invocation & Privilege Elevation

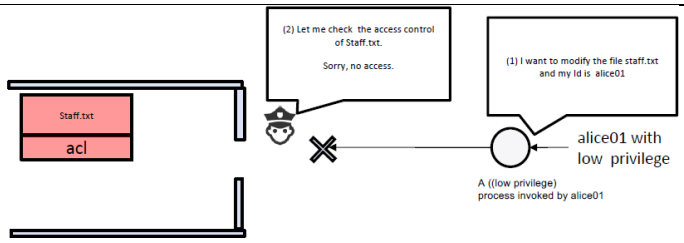
Controlled Invocation

- Eg in Unix: Some sensitive resources (such as network port 0 to 1023, printer) should be accessible only by the superuser. However, users sometime need those resources.
- Eg: Consider a file F that contains home addresses of all staffs.
  - Clearly, we cannot grant any user to read F. However, we must allow a user to read/modify his/her address, and thus need to make it readable/writeable to that user.
  - The polarized setting where either a process can read/cannot read a file would get stuck!

Solution: Controlled Invocation

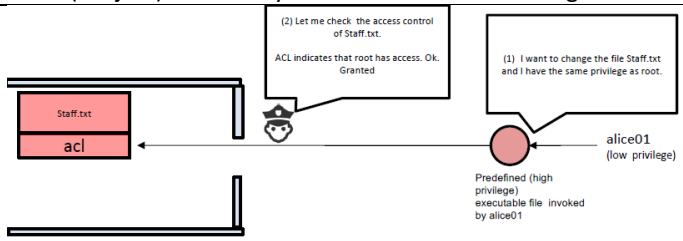
- The system provides a predefined set of applications that have access to F.
- These application is granted "elevated privilege" so that they can freely access the file, and any user can invoke the application. Now, any user can access F via the application.
- The programmer who write the application 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.

Without Controlled Escalation



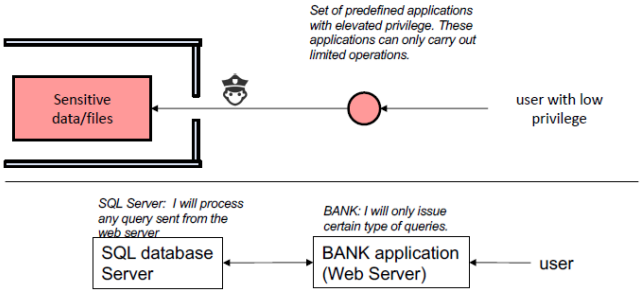
- alice01 doesn't has access right to Staff.txt.
- Any processes (subject) invoked by alice01 inherit alice01's right

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.

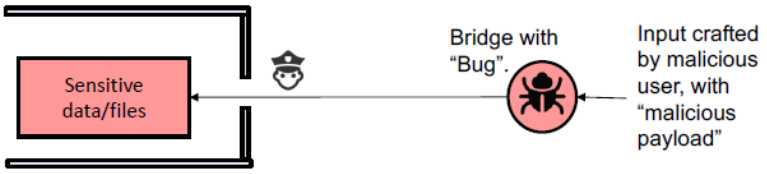
Bridges with Elevated Privilege

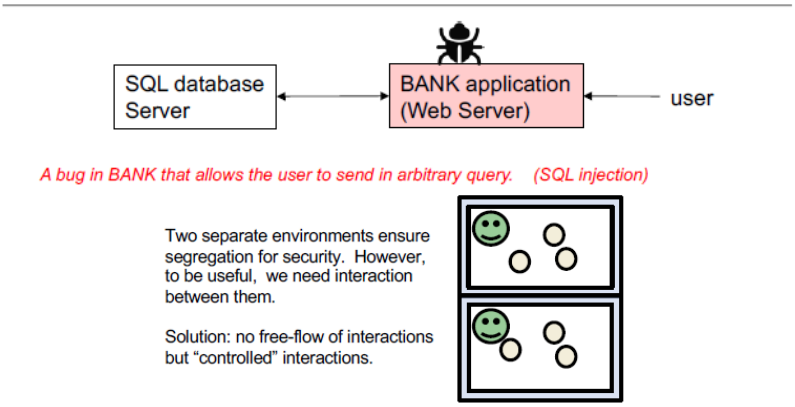


- Firewall ensure that users can't directly send query to the SQL database server.
- Users can indirectly send query to SQL server via website. In this way, only certain types of queries can be sent by the users.
- Suppose a "bridge" is not implemented correctly and contains exploitable vulnerabilities.
  - In some vulnerabilities, an attacker can trick the bridge to perform "illegal" operations not expected by the programmer/designer.
  - This would have serious implication, as the process is now running with "elevated privilege".
- Attacks of such form is also known as "privilege escalation".
- "Privilege escalation is the act of exploiting a bug, design flaw or configuration oversight in an operating system or software application to gain elevated access to resources that are normally protected from an application or user.

- The result is that an application with more privileges than intended by the application developer or system administrator can perform unauthorized actions."

Bugs in brigade leads to Privilege Escalation





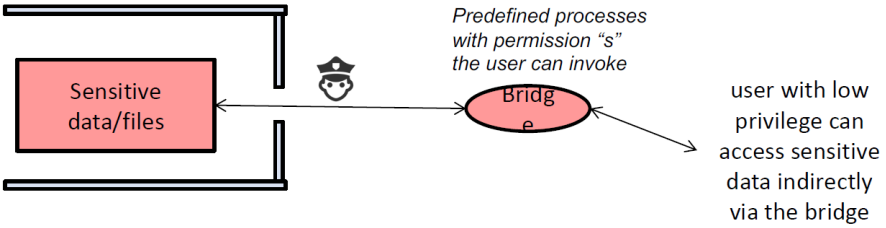
Summary:

When SUID is Disabled	• If the user alice invokes the executable, the process will have its effective ID as alice. • When this process wants to read the file employee.txt, the OS (reference monitor) will deny the access Process info: name (editprofile) real ID (alice) effective ID (alice) ACCESS DENIED -rw------- 1 root staff 6 Mar 18 08:00 employee.txt -r-xr-xr-x 1 root staff 6 Mar 18 08:00 editprofile
When SUID is Enabled	• If the permission of the executable is "s" instead of "x", then the invoked process will has root as its effective ID. • Hence the OS grants the process to read the file • Now, the process invoked by alice can access employee.txt Process info: name (editprofile) real ID (alice) effective ID (root) ACCESS GRANTED -rw------- 1 root staff 6 Mar 18 08:00 employee.txt -r-sr-xr-x 1 root staff 6 Mar 18 08:00 editprofile

Elevated Privilege

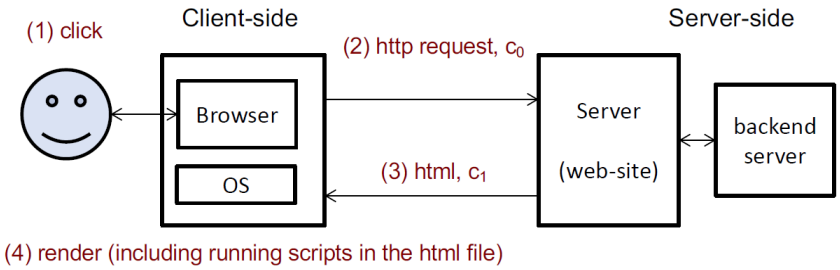
- In this example, the process "editprofile" is temporary elevated to superuser (i.e. root), so that it can access sensitive data.

- 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 the 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.



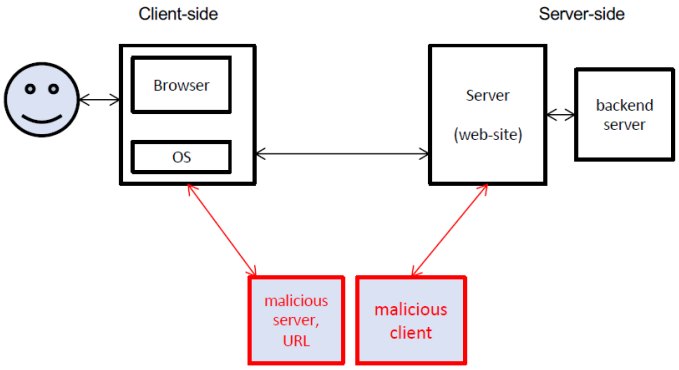
- If the bridge is not built securely, there could be privilege escalation attack.

Web Security



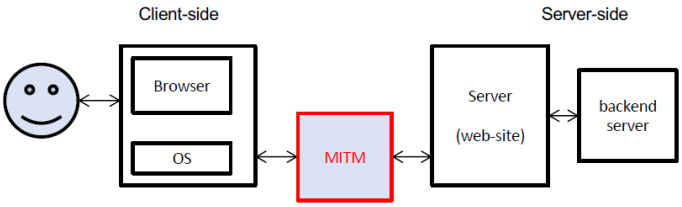
1. User clicks on the “link”
2. A HTTP request is sent to the server (with cookie c_0 if any)
3. The server then constructs and sends a “HTML” file to the browser, possibly with a cookie c_1.
   - The cookie can be viewed as some information the server wants the browser to keep.
   - This cookie can be passed back to the server during the next visit by the browser.
   - The cookie can be some secrets (e.g login session cookie etc).
4. The browser renders (including running the scripts) the HTML file.
   - The HTML file describes the layout to be rendered and presented to the user.
   - The HTML file also instructs the browser to construct and store cookies.
     - This cookie may be different from c_1.

Threat model 1: Attackers as another end systems.



Here, the attacker is just another end system. For e.g. a malicious web-server that the victim is lurked to visit, or attackers who has access to the targeted server.

Threat model 2: Attackers as MITM.



Here, the attacker is a Man-in-the-middle in the IP or lower layer. The attacker can gain MITM in a few ways, for e.g.,

- As the café-owner who provides the free wifi;
- Via DNS spoofing attack or ARP attack;
- As owner of the VPN server, last hop in the TOR network (not covered in this module).

Because the MITM is in the IP layer, the MITM can attempt to impersonate a server (e.g. phishing attack). So, this threat model also covers some settings of threat model 1 where the attacker is the server.

Cookie

- A http cookie is a piece of data sent by the server in the response of a HTTP request.
  - It is under the "Set-Cookie" header field.
  - The browser permanently stores the cookie.
- Whenever a client revisits the website, the browser automatically sends the cookie back to the server.

Remarks:

- Most websites get users' to consent tracking cookies. (EU's GDPR, General Data Protection Regulation)
- The cookie are sent back only to the "same origin", in the sense that, it is sent to the server which is the "origin" of the cookie.
  - Viewing from access control perspective, the "same origin" set the boundary among different web sites.
  - A script from a web site can only access cookies within its boundary. (unfortunately, this is not done clearly, leading to many problems…)
- There are also a few types of cookie, eg. session cookies (deleted after the session ends), secure cookies (can only be transmitted over https), etc.
  - See https://en.wikipedia.org/wiki/HTTP_cookie#Session_cookie

Same-Origin Policy

- A "script" which runs in the browser could access cookies.
  - Which scripts can access the cookie?
  - Due to security concern, browser employs this access control policy:
    - The script in a web page A (identified by URL) can access cookies stored by another web page B (identified by URL), only if both A and B have the same origin.
- Origin is defined as the combination of protocol, hostname, and port number.
- The above access policy is simple and thus seeming safe. However, there are several complications.

Compared URL	Outcome	Reason
http://www.example.com/dir/page2.html	Success	Same protocol, host and port
http://www.example.com/dir2/other.html	Success	Same protocol, host and port
http://username:password@**www.example.com** /dir2/other.html	Success	Same protocol, host and port
http://www.example.com:**81**/dir/other.html	Failure	Same protocol and host but different port
https://www.example.com/dir/other.html	Failure	Different protocol
http://**en.example.com**/dir/other.html	Failure	Different host
http://**example.com**/dir/other.html	Failure	Different host (exact match required)
http://**v2.www.example.com**/dir/other.html	Failure	Different host (exact match required)
http://www.example.com:**80**/dir/other.html	Depends	Port explicit. Depends on implementation in browser.

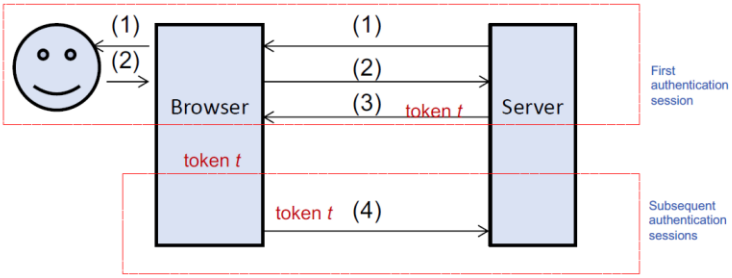
- URL's with the same origin as: http://www.example.com
- Limitations:
  - Confusing definition: There are many exceptions, and exceptions of exceptions.
  - Very confusing and thus prone to errors.
  - Different browsers may adopt different definitions.

Choices of Token

- Suppose t is a randomly chosen number, then the server needs to keep a table storing all the tokens it has issued. To avoid storing the table, one could use
  - (**secure version**) A message authentication code (MAC).
    - The token t consists of two parts: a randomly chosen value, or meaningful information like the expiry date, concatenated with the mac computed using the server secret key.
    - e.g t ="alicetan:16/04/2015:adc8213kjd891067ad9993a"
  - (**insecure version**) The cookie is some meaningful information concatenated with a sequence number that can be predicted.
    - e.g t = "alicetan:16/04/2015:128829"
- For both methods, when the server finds out that the token is not in the correct format (or not the correct mac), the server rejects.
  - The first method relies on the security of mac, while the second method relies on obscurity – attackers don't know the format.
  - The second method is insecure because an attacker, who knows how the token is generated (for e.g. by observing its own token), can forge it. This further illustrates the weakness of security by obscurity.

Remarks:

- Token typically has an expiry data.
- An identification of the session can be used as the token, so the token is also called SID.
- In web applications, the token is often stored in the cookie.



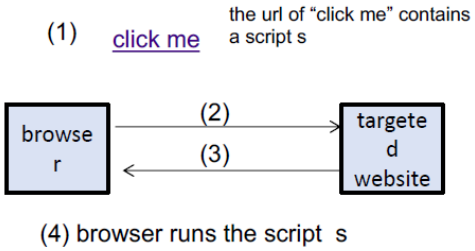
(1) Authentication challenge (e.g. asking for password).
(2) Authentication Response that involves the user.
(3) Server sends a token *t*. Browser keeps the token
(4) Browser presents the token *t*. Server verifies the token.

Cross Site Scripting (XSS) Attacks

- In some websites, if the browser sends a text that contains a substring s, the replying HTML sent by the server would also contain the same substring s.
- Example:
  - Enter a wrong address: http://www.comp.nus.edu.sg/nonsense_test
  - Search for a book in library: http://nus.preview.summon.serialssolutions.com/#!/search?q=heeheeheee
- In these instances, if the string s contains a script, this will be considered XSS.
  - Example: http://www.comp.nus.edu.sg/<script>alert("heehee!");</script>
  - Note: The attack above wont work as the server replaces the special characters "<" by &lt
  - Filters are used to prevent XSS.

How XSS works?

1. Attacker tricks a user to click on an url, which contains the target website, and a malicious script s. e.g the link could be sent via email with "click me", or a link in a malicious website.
2. The request is sent to the server.
3. The server constructs a response html. The response contains the script s.
4. The browser renders the html page and runs the script s



Why is this an attack?

The malicious script could

1. deface the original webpage;
2. steal cookie.

- The control that protects access to the cookie is the same origin policy.
- Now, since the malicious script is coming from the same origin (the victim clicked on it..), it can now access cookie previously sent by the website.
- This is yet another example of privilege escalation.
- A malicious script has elevated privilege to read the cookie.
- This attack exploits the client's trusts of the server.

Types of XSS

- Reflection (aka non-persistent)
  - Similar to the example above
- Stored XSS (aka persistent)
  - The script s is stored in the targeted website (e.g in the forum page)

Defense against XSS

- Most defense rely on input-validation carried out by the server.
- That is, the server filters and removes malicious script in the request while constructing the response page.
- However, this is not a fool proof method.

Cross Site Request Forgery (XSRF)

- Aka "sea surf", cross-site reference forgery, session riding.
- This is the reverse of XSS. It exploits the server's trust of the client. Previous attack of XSS exploits the client's trust of the server

Example:

- Suppose a client A is already authenticated by a targeted website S, say www.bank.com and the site keeps a cookie as "token".
- The attacker B tricks A to click on a url of S. The url maliciously requests for a service, say transferring $1000 to the attacker Bob's

  www.bank.com/transfer?account=Alice&amount=1000&to=Bob
- The cookie will also be automatically sent to S which is sufficient to convince S that A is already being authenticated. Hence the transaction will be carried out.

See https://en.wikipedia.org/wiki/Cross-site_request_forgery

Defense against XSRF

- Relatively easier to prevent compared to XSS.
- Include authentication information in the request. For e.g.

  www.bank.com/transfer?account=Alice&amount=1000&to=Bob&Token=xxk34n890ad7casdf897e32433

Other Attacks (Often caused by Misconfiguration, searching for filenames…)

- Drive-by-download
- Web Bug (aka web beacon, tacking bug, tag, page tag)
- Clickjacking (User Interface redress attack, https://www.owasp.org/index.php/Clickjacking)
- CAPTCHA
- Click Fraud

Simple Implementation Mistakes

- Authentication/filtering at the client side.
- Security credential embedded in the public web pages.
- Server's secrets stored in cookies.
- Configuration errors.
- URL as secrets, e.g. in password reset link, or zoom link.
  - This is acceptable and commonly used. However, some implementations do not have adequate protection, or unknowingly leak info of the url.

Chapter 0: Security Introduction

Confidentiality	Prevention of unauthorized disclosure of information. • Only sender & intended receiver should "**understand**" message contents
Integrity	Prevention of unauthorized modification of information or processes. • Only admin/whitelisted personnel can edit/modify content
Availability	Prevention of unauthorized withholding of information or resources. • Only allowed personnel can view info • Public file/servers should be up for users to use (cannot down bro)

- Need precise formulation of "Security" for analysis.
- Aware of:
  - Security Trade-off (usability, cost)
    - Attackers go for the weakest point
    - Implementation flaw
    - Legacy System, don't care
    - Designers not aware of the attack scenarios (info attacker can access, attacker's goal)
    - Human error
  - Need to be managed
  - Adversarial thinking in analysis (think like the attacker when analyzing a system)

Chapter 1: Encryption

- Encryption are designed for **confidentiality** (Not necessarily providing integrity, although some encryption schemes achieve some aspects of integrity)
- Formulate attack scenario (aka threat model) by defining attacks that it can prevent
  - Attacker's goal: in(distinguishable)
  - Info available to attacker: Ciphertext, Plaintext, etc
    - A general form: Oracle
- Notions of "Oracle"
  - Encryption Oracle, Decryption Oracle, Padding Oracle
- Key Strength: Quantifying security by equivalence of best-known attack to exhaustive search (2048-bit RSA key only has key strength less than 128 bits)
- No known efficient attacks on modern schemes under the "original" threat models, but there are pitfalls.
  - Implementation error(s): Wrong mode, Wrong random sources, Mishandling of IV.
  - Side-Channel Attack
  - Implicitly require Integrity. (Padding Oracle Attack)
- Designs of various symmetric key encryption schemes
  - One-time pad: "Unbreakable" even if attacker has sufficient time to exhaustively search
  - Stream Cipher: xor'ing with a "pseudo-random" string
  - Block Cipher: Modes of operation
    - CBC: Provides some form of integrity
      - ➔ Secure against CPA (Chosen plaintext Attack), vulnerable to padding oracle attack, BEAST attack, might achieve some forms of integrity.
      - ➔ To secure against BEAST, IV needed to be random)
    - ECB: Negative demonstration. Deterministic (no IV involved)
    - CTR: Stream cipher
      - ➔ Secure against CPA: vulnerable to padding oracle attack if padded.
      - ➔ No "integrity" at all and easily changed (aka malleable)
      - ➔ If IV of 2 ciphertext is the same, leak significant information (in contrast, if IV of 2 ciphertext under CBC is the same, there are some leakages but not as bad)
    - GCM: Authenticated-Encryption (AE).
      - ➔ Achieves both integrity and confidentiality
      - ➔ Secure against "CCA2"
      - ➔ Only standardized quite recently (around 2007) and not in some legacy system
- Crucial role of IV (need randomness to have indistinguishability)
  - Make the encryption probabilistic.
  - How is it employed? Whi is it important?

Chapter 2: Authentication Credentials

- Authentication Credentials
  - Something (data, device, etc) hold by entity for authenticity verification e.g Password
- Password Strength
  - Online vs Offline
- Attacks on Password
  - Phishing, Bootstrap. Default password.
  - Phishing is very effective
- 2-Factor & 2-Step Verification
  - 2-factor/2-Steps is better than 1. (e.g Online Banking)
  - Compare different Combinations.

Chapter 3: Authenticity (Data Origin)

- Public Key Encryption.
  - RSA (based on integer factorization. Only integer). ElGamal (based on discrete log of "Algebraic group". Many choices: ECC).
  - Differences with symmetric key. Implications:
    - Symmetric key requires a secure channel to distribute key.
    - Public key requires a secure broadcast channel to distribute key.
  - Post-Quantum Crypto (case studies)
  - Pitfall: using RSA in symmetric key setting.
- Authentication primitives: digest, mac, signature.
  - Security models and application scenarios (Summary Slide 72,73,74)
  - No key or other secret information in hash/digest.
  - Advantages of signature: public key setting (broadcast channel); non-repudiation.
  - Disadvantages of signature: efficiency.
  - Hash requirement: Collision resistant. Collision resistant vs 2nd pre-image attack.
  - All hash subjected to Birthday attacks.
- Achieve Confidentiality != Achieve Authenticity (if a scheme preserves confidentiality, it might not preserve authenticity)
- Various construction:
  - hash: SHA
  - Mac: CBC-mac, HMAC
  - Signature: DSA, hash-and-sign. Special property of RSA for signature (hash-and-encrypt).
- Time-memory-tradeoff in inverting hash

Chapter 4: PKI and Channel Security

Public Key Infrastructure (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, 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 CA's.

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.

Chapter 5: Network Security

Summary & Takeaways:

- Crypto + PKI can establish end-2-end security (w.r.t. Confidentiality & authenticity) even in the presence of MITM. However, there are other issues in Networking:
  - **Availability** not addressed.
  - Routing need protection.
  - Want to mitigate MITM as much as possible (e.g. concerns 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.
- Some 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)
  - Which layer to employ e-2-e encryption.
- 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.

Chapter 6: Call Stack & Secure Programming

- Demonstrate how process integrity can be compromised by modifying values in memory.
- Call stack facilitates function calls. It keeps track of different instances of function call. Without that, we won't have modular software design.
- An object in the stack is a "stack frame" and each frame contains info of the environment an instance of the function operates. Info include control flow info (return address), local variables, and parameters. Malicious modification of those info would have significant consequence (next lecture on stack smash + buffer overflow).

Chapter 7: Access Control

- How access control is specified:
  - Operation, Objects, Subjects (Principle).
  - Access Control Matrix, Intermediate control to simplify representation.
    - Examples of intermediate control
    - Representation: ACL vs Capabilitu
- Guideline:
  - Security perimeter, security boundary, principle of least privilege, segregation, compartment, segmentation, firewall.
- How to share information across security perimeter: "Bridge" and "Privilege Elevation" (aka "privilege escalation" in the context of attack)
- Example in Unix:
  - File System: ACL
    - Intermediate Control (User, Group, World(others))
  - Objects: Program, Resources
    - E.g Keyboard, Display, Network, treated as files
  - Subjects: Processes. Each process has:
    1) **Real UID**, specifying its owner (principle)
    2) **Effective UID** that is used by the reference monitor to check permissions.
  - Operations: Read, Write, Execute
  - Ring: Root(0) vs User(1)
  - Example of "Bridge"
- Example: Apps in Android vs Users in Linux/Window

QUIZES:

QNS	A hacker installed a hardware keylogger on the desktop in LT11 and successfully stole a few passwords. This incident illustrated which of the security requirement mentioned in this class?
ANS	Confidentiality

QNS	An attacker was given a pair (x, c) where x was a plaintext and c was the corresponding ciphertext encrypted with some secret key k. Using (x, c), the attacker carried out an attack. This was a
ANS	Known Plaintext Attack

QNS	Suppose a plaintext x is being encrypted twice, using a same 40-bit secret key k. In other words, the ciphertext is: Enc (k, Enc (k, x)) An exhaustive attack would carry out checking of _____ keys.
ANS	2^{40} keys.

QNS	Suppose security is the main concern and as a programmer, Alice must choose one AES mode. Which of followings is the most appropriate?
ANS	GCM. Why not CBC and CTR?

QNS	Among CTR and CBC mode, Alice suggested using CTR for a system during development discussion. Alice gave a list of reasons to justify that CTR is not worse than CBC. Which of the following reasons is not valid?
ANS	• "CTR is secure against padding" is wrong. If padded, CTR is still vulnerable to padding oracle. • Many documents indicate CTR is secure against to padding oracle. - This is true only if there is no padding at all. - If no padding required, there is no padding oracle and CBC of course would also be secure.

QNS	Most documents indicate that the IVs for CTR mode are to be chosen in a way so that they are different for different encryptions, but they do not give specific steps on how to choose them. Which of the following should not be the way to choose the IV for CTR mode?
ANS	Derive from meta data

QNS	Alice knows the names of 1024 Pokémons. She chooses her password in this way: • Randomly & uniformly picks one Pokémon, say Evee. • With probability 0.5, the password is all uppercase letters, e.g. EVEE, and with probability 0.5, the password is all lowercase, e.g. evee. What is the entropy of Alice's password?
ANS	Total number 1024 * 2 log (1024 x 2) = log(1024) + log(2) = 11 Hence, entropy = 11

QNS	Alice has another way to choose password. It is a randomly chosen Pokémon (all lowercase), following by several randomly chosen numeric digits, e.g. evee1234. Recap that the entropy of one numeric digit is approximately 3.3. Alice is going to choose a password for her Gmail account where she logins to get email. Following a guideline mentioned in lecture, among the followings, which is the least number of digits that is suffice?
ANS	7. (6 is min required but not in the answer list). It is an online attack and thus 29 bit suffice. Total number = 1024 x 106 Log (1024 x 106) = log(1024) + 6 log (10) > 29

QNS	Which of the following method of door access should not be considered as two-factor authentication?
ANS	Pin + secret images are both "what you know" and 1-factor Voice + pin = "who you are" + "what you know", so it is 2-factor

QNS	Suppose Bob found two sequences y1, y2 whose digests of the hash H are the same. Which of the followings is the most appropriate description?
ANS	• Bob had demonstrated pigeonhole principle by giving a concrete example. • Bob had successfully carried out 2nd pre-image attack. • Bob had successfully carried out collision attack. • Bob had successfully carried out birthday attack. • Bob had successfully carried out pre-image attack.

QNS	Suppose we want to choose a hash function. We should choose one that is…
ANS	• One way, i.e pre-image resistant • Collision Resistant • 2nd pre-image resistant • Secure under decryption oracle • Birthday attack resistant

QNS	A lecturer needed to assign a 16-byte sequence to each student. The lecturer chose the sequence to be the 16 leading bytes of SHA3(x) where x was the 7-digit student number. The lecturer next posted the assigned sequence in Canvas. The lecturer didn't use student ID directly due to privacy concerns. It was against the school policy to reveal IDs of the student in the class. What do you think?
ANS	• The method would not work. This is because there is a good chance that 2 students have the same assigned sequence • Lecturer violated school policy. Lecturer should use salted hash, i.e the assigned number should be (r|| SHA3 (r||x)). • Lecturer violated school policy. One could exhaustively search for the IDs that match the digests. • Lecturer didn't violate school policy. Although not truly random, SHA3 is cryptographically pseudo random. We are unable to distinguish the generated number from the randomly chosen number. • Lecturer didn't violate school policy. SHA3 digests are random.

QNS	There are many advantages of public key cryptography over symmetric key. Which of the following is being highlighted the most by the lecturer? (some the followings statement are wrong, but some are correct. Choose the correct one that the lecturer keeps mentioning)
ANS	• PKC is probably secure • Symmetric key needs secure channel among each pair to establish the key, while PKC only need a secure broadcast channel to publish the public key. • PKC has the homomorphic property which is useful in applications. E.g Blind Signatures. • PKC can provide non-repudiation but not for symmetric key. • PKC doesn't need mode-of-operation while block cipher is only designed for small block.

QNS	Which of the following most appropriately describe key-exchange's attack model?
ANS	• A common key among Alice and Bob will be established after key exchange. - This statement describes the objective of key-exchange rather than an attack mode. It does NOT describe an attack. • Attacker can sniff all communications between Alice & Bob, and wants to guess some info of the established key. - This describes the scenario where an attacker intercepts communications to try and gain information about the established key. This is a type of attack known as a passive attack. • Attacker's goal is to be a MITM between Alice and Bob. - This describes a scenario where the attacker inserts themselves as a MITM between Alice and Bob to intercept and manipulate their communications. This is an implied goal. • With access to an oracle, the attacker wants to impersonate the prover. - Oracle is irrelevant here. • There is no pre-shared secret key and all parameters are publicly known.

	- This describes a scenario where there's no pre-shared secret, and all parameters of the key exchange algorithm are publicly known. This doesn't directly describe an attack model.

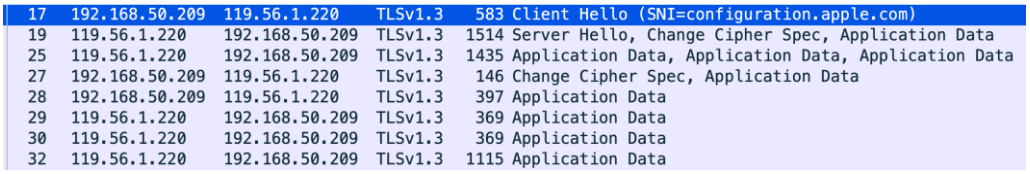
QNS	Alice believes that the "Big brother" who has control of the network is watching her and has been logging all her TLS sessions. While TLS can protect confidentiality of the logged sessions, she is worried that when quantum computer become a reality, her RSA key would be broken and hence the logged encrypted sessions. Which of the following best describe her concern?
ANS	• Authenticity • Forward Secrecy • Non-repudiation • Authenticated Key-Exchange • Confidentiality

QNS	Alice Tan Fei Yong is a Singaporean with NRIC number **1234567A**. She has a free emails account from Google **atfy123@gmail.com**, and an account from NUS **u1234567@nus.edu.sg**. She owns a domain name, **www.alicesg.com**. She recently obtained a certificate (the certificate format follows the standard in lecture) from some authorities. Among the 4 underlined strings, which could be in the name field of the certificate?
ANS	• All except the Gmail address. • Only the domain name • Both email address and the domain. • All. • NUS's email address and the domain name.

QNS	There are many ways to describe a certificate. Here are some obtained from public domains and ChatGPT. Which one is most precise, close to the version given by the lecturer, and illustrates the essence discussed in this class?
ANS	• Digital certificates are the credentials that facilitate the verification of identities between users in a transaction • These are electronic documents, which, via the underlying PKI that binds the private key to its entity, verify the authenticity of the entity. • A document that binds a public key to a name, and this document is certified by an authority. • Is a digitally signed document that is comparable with a physical identity card or a passport in the analogue world. • It is issued by a CA and used to verify the authenticity of a user.

QNS	Alice suspected that her laptop is a bot controlled by some C2 (Command and Control) server. Which of the following tools would best help Alice to investigate?
ANS	• Wireshark • nslookup • Nmap • Python3 • gcc

QNS	Here is an explanation from public domain that compares TLSv1.2 and v1.3. It doesn't describe forward secrecy properly. Which part is wrong or most confusing? *"By changing the encryption keys for every session, Diffie-Hellman in TLS v1.3 achieves forward secrecy. TLS v1.2 allows RSA algorithm which does not achieve forward secrecy because it uses a static key."*
ANS	• Diffie-Hellman indeed achieves forward secrecy. The explanation does not point out the argument that, the security of Diffie-Hellman arrives from the computational hardness in getting the established key from the intercepted communication. - This is a correct fact about DH, but it has NOTHING to do with the statement in red. It is not answering the question LOL. • The explanation is NOT clear on the role of the statis key. It "generates" the static key, not "uses" the static key • The explanation confuses the role of RSA. RSA is an asymmetric key algorithm for encryption. By itself, cannot be directly employed for Authenticated Key-Exchange. • While RSA private key is static, its public key changes for different sessions. So, the fact that RSA private key is statis does not imply that it does not achieve forward secrecy. - Wrong, public key is always the same. • The explanation is wrong about RSA. RSA based algorithm indeed has a changing session key.

QNS	Below was a snapshot of Wireshark on my laptop (192.168.50.209), while I was connecting to a webserver.  17 192.168.50.209 119.56.1.220 TLSv1.3 583 Client Hello (SNI=configuration.apple.com) 19 119.56.1.220 192.168.50.209 TLSv1.3 1514 Server Hello, Change Cipher Spec, Application Data 25 119.56.1.220 192.168.50.209 TLSv1.3 1435 Application Data, Application Data, Application Data 27 192.168.50.209 119.56.1.220 TLSv1.3 146 Change Cipher Spec, Application Data 28 192.168.50.209 119.56.1.220 TLSv1.3 397 Application Data 29 119.56.1.220 192.168.50.209 TLSv1.3 369 Application Data 30 119.56.1.220 192.168.50.209 TLSv1.3 369 Application Data 32 119.56.1.220 192.168.50.209 TLSv1.3 1115 Application Data a) Which packets carried out authenticated key-exchange? b) Which packet contains the server's certificate?
ANS	a) 1st 4 packets are used for authenticated key-exchange. - 17, 19, 25, 27 - 17 & 19 carry out authenticated key exchange, 25 and 27 confirms it. b) 2nd packet (Hint: Should be coming from the server) - 19

QNS	M was a MITM between A and a web server S. M successfully conducted re-negotiation attack. Suppose originally (without attack), A intended to send a message "INS:123:A" to S. Among the choices, which was the possible message received by S after the attack.
ANS	Recap: With renegotiation attack, the attacker can pre-pend messages of the attacker's choice to the original victim's message • att > INS:123:A:att • att > INA:123:A • INS:123 • att > INS:123 • INS:att:A

QNS	Continuing from the previous question. Before the attack, M knew that the message A intended to send was of the form "INS:???:A" where "?" indicated the 3 characters the M didn't know. After the attack, what had M learned about the message?
ANS	• M learned nothing about the 3 characters, because TLS was designed to be secure against MITM. • M leaned nothing about the 3 characters, because M was unable to get the 2nd session key. • This depends on which crypto primitive was used. If the handshake employed DH-based authenticated key exchange, then due to forward secrecy, M was unable to learn anything about the 3 characters. However, if RSA based authenticated key exchange was employed, M could uncover the session key and then the 3 characters.

	• M could obtain the 3 characters. Due to the flaw in the re-negotiation protocol, M could decide the session key between M and A, and another session key between M and the server, i.e, taking the role of MITM in the application layer. Thus, M could obtain the 3 characters.

QNS	Alice uploaded her report to Canvas while at home. Her laptop was connected to her home's wifi protected by WPA2 personal, and the wifi password was 8 randomly chosen characters. Later, Alice found out that her neighbor Eve was a "cybersecurity euthanistic" and was trying Aircrack to break wifi. Would Eve able to steal Alice's report?
ANS	• No. WPA2 personal employ some password authentication scheme that is secure against dictionary attacks. • No. Because TLS is above the link layer. So, as long as the TLS functioned as it supposed to be, whatever malicious activities in the link layer would not matter. • It depends on the capability off Eve. If Eve was resourceful, Eve was able to get the report. We know that no system is secure. • Yes. WPA2 personal is known to be vulnerable to offline dictionary attacks. Eve was able to find the wifi password. Note: HTTPS (TLS) provides end-to-end encryption, ensuring that data transmitted between Alice's laptop and the website is secure, regardless of potential vulnerabilities in the lower layers of the network stack. Therefore, even if Eve managed to crack the Wifi password, the wound not be able to steal Alice's report because it is "protected/encrypted" by HTTPS.

QNS	A few days ago, lecturer mentioned in Canvas a newly discovered backdoor in a software. What was that about?
ANS	• Ransomware • Stack smacking via buffer overflow • Mirai Attack • Padding Oracle Attack • XZ Utils ** jia-tan-xz-backdoor. - Don't assume Jia tan is Chinese. "Jia Tan" was stealthy and patient, which were indicators of state sponsored attack. - This happened on open-source project. Now, this incident can be treated as a limitation or strength of open-source project: more secure as any backdoor can be discovered by the community, less secure as attack can insert backdoor.

QNS	Alice wanted to allocate memory to an array to store a 5 characters long null-terminated string, e.g. "hello". Alice wanted to know how many bytes were to be allocated (i.e. size of the array). Among the choices below, which is the smallest possible to avoid security vulnerability?
ANS	• 5 • 6 • 7 • 8 • 10 H + e + l + l + o + \0 → total 6 minimum

QNS	Consider the following two consecutive lines in a C program. char x[10]="hello"; printf (x); The first line simply initializes the array x as a nullterminate string "hello". When compile using gcc, the complier likely would give some warning messages. Among the followings, which is the most sensible explanation?
ANS	• Compiler would give a warning that "printf(x)" is not safe. Nonetheless, provided there are not other errors, it still can be compiled. But running the executable likely would cause segmentation fault. • Compiler would warn that "printf(x)" is not safe, as printf with one parameter of variable is not safe. However, these 2 lines will not lead to vulnerability because the variable x is always the fixed string "hello". • Compiler would warn that "printf(x)" is not safe. This would cause syntax error and compilation will fail. • Compiler would warn that "printf(x)" is not safe because "printf" has to be avoided. Note: Not so easy for printf() to cause buffer overflow. printf () mostly compromise "confidentiality". Overflow is on "integrity". There are special ways of using printf() to modify memory but not easy to carry out.

QNS	A frame in a call stack maintains the running environment of a particular call to a routine. Which of the followings likely is not included in the frame?
ANS	• Return address • Parameters • Canary - Note: Canary is not always present in every frame. • Local variables • Compiler version

QNS	Security of hash function against quantum computer is quite complicated, as there are debates on how to formulate the requirements. One argument claims that quantum computer can find collisions in $2^{n/3}$ operations, where n is the digest length. Let's accept this claim. Suppose an agency set a requirement of 256 bits for hash digest by considering birthday attack. Now, in view of the threat from quantum computer, what should be the new requirement?
ANS	• 128 • 256 • 384 • 512 • 768 Original 256-bit digest length to withstand birthday attacks. Then for quantum computers solve for n: $2^{n/3} = 2^{256/3}$ Solving for n, we get n = 384.a Hence, 384 is the answer.