Topic 0

CIA

1. Confidentiality
   - Prevention of unauthorized discolusre of information
2. Integrity
   - Prevention of unauthorized modification of information or processes
   - Non-repudiation
   - Authentication
3. Availability
   - Prevention of unauthorized withholding of information or resources

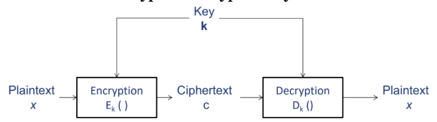
Threat model
- The attacker's goals
- The attacker's capabilities

Trade-off in security
- Ease-of-use
- Performance
- Cost

Threat-Vulnerability-Control
- **Threat**: A set of circumstances that has the potential to cause harm (e.g. an attacker with control of the workstation in the LT could maliciously gather sensitive info like passwords)
- **Vulnerability**: A weakness in the system (e.g. anyone can reboot the workstation from USB or Disk to gain control)
- **Control**: A control, countermeasure, security mechanism is a mean to counter threats (e.g. restrict physical access to the workstation, disable USB booting)
- **A threat is blocked by control of a vulnerability**

Topic 1: Encryption

1.1 Definition: Encryption/decryption/keys



- A symmetric-key encryption scheme consists of encryption and decryption
- A cipher must be correct and secure
  - **Correctness**: For any plaintext x and key k, $D_k(E_k(x)) = x$
  - **Security**: Definition depends on the threat models. Informally, from the ciphertext, the eavesdropper is unable to derive useful information of the key k or the plaintext x, even if the eavesdropper can probe the system.
- Probabilistic encryption: for the same x, there could be different c's. But they all can be decrypted to the same x.

1.2 Security Model and Requirement

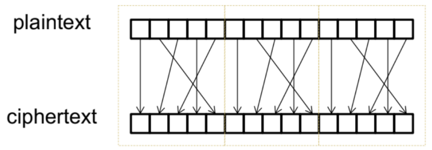
Threat model
- Attacker's goal
  - Total break (most difficult goal)
    * Attacker wants to find the key
  - Partial break
    * Attacker may want to decrypt a ciphertext but not interested in knowing the key
    * Attacker may simply want to extract some info abt the plaintext (e.g. if it is a jpg or excel file)
  - Distinguishability (weakest goal)
    * With some non-negligible probability of $> 1/2$, the attacker acan correctly distinguish the ciphertexts of a given plaintext from the ciphertext of another given plaintext
- Attacker's capability
  - Ciphertext only attack
    * Attacker is given a collection of ciphertext c. The attacker may know some properties of the plaintext (e.g. the plaintext is an English sentence)
  - Known plaintext attack
    * The attacker is given a collection of plaintext m and their corresponding ciphertext c
    * Attacker might get this as they know the header or part of the plaintext
  - Chosen plaintext attack (CPA)
    * The attacker has access to an oracle. The attacker can choose and feed any plaintext m to the oracle and obtain the corresponding ciphertext c (all encrypted with the same key). The attacker can access the oracle many times, as long as within the attacker's compute power. He can see the ciphertext and then choose the next input. This black-box is an **encryption oracle**.
    * e.g. attacker has access to a smartcard
    * e.g. attacker can eavesdrop
  - Chosen ciphertext attack (CCA2)
    * Same as CPA but the attacker chooses the ciphertext and the black-box outputs the plaintext. The black-box is a **decryption oracle**.
    * Padding oracle is a weaker form of a decryption oracle.

From defender's POV, want a cipher that can protect against the attacker with the highest capability. Cipher is secure against CCA2 (decryption oracle) $\implies$ secure against CPA (encryption oracle)

1.3 Classical ciphers + illustration of attacks

1.3.1 Substitution cipher
- Plaintext and ciphertext are both strings over a set of symbols U.
- The key is a 1-1 onto func from U to U
- Key space: set of all possible keys
- Key space size: total number of possible keys
- Key size/length: number of bits required to represent a key
- Attacks
  1. Exhaustive search (examine all possible keys 1 by 1)
     - Running time depends on size of key space
     - If the table size is 27, the key can be represented by a sequence of 27 symbols. The size of key space is 27!. Exhaustive search eneds to carry out 27! loops, which is infeasible using current compute power.
  2. Known plaintext attack
     - Given sufficiently long ciphertext, the full table can be found
     - Substitution cipher is not secure under known plaintext attack.
  3. Ciphertext only attack
     - Given that the attacker knows that the plaintext is an English sentence, he can do frequency analysis attack. The frequency of letters used in English is not uniform. Given a sufficiently long ciphertext, attacker may correctly guess the plaintext by mapping frequent characters in the ciphertext to the frequent character in English.

1.3.2 Permutation cipher



- AKA transposition cipher
- First group the plaintext into blocks of t characters, then apply a secret permutation to each block by shuffling the characters
- The key is the secret permutation, which is a 1-1 onto func e from $\{1, 2, \ldots, t\}$ to $\{1, 2, \ldots, t\}$. t can also be part of the key.
- Attack
  - Fails under known-plaintext attack
  - Easily broken under ciphertext only attack if the plaintext is English text

1.3.3 One Time Pad

Properties of xor:
- Commutative: $A \oplus B = B \oplus A$
- Associative: $A \oplus (B \oplus C) = (A \oplus B) \oplus C$
- Identity element: $A \oplus 0 = A$
- Self-inverse: $A \oplus A = 0$

One Time Pad
- Encryption: plaintext xor key bit by bit
- Decryption: ciphertext xor key bit by bit
- Key is only used once, so 1GB of plaintext would need a 1GB key to encrypt
- Security
  - From a pair of ciphertext and plaintext, attacker can derive the key but useless bc key won't be used anymore

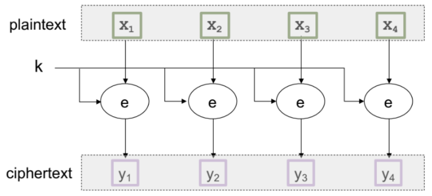
1.4 Modern ciphers + recommended key length

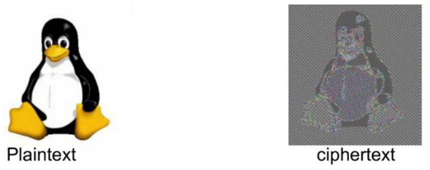
1.4.2 Block cipher & mode of operations

DES/AES are known as block ciphers. Block ciphers have a fixed size of input/output. AES: 128 bits (16 bytes).

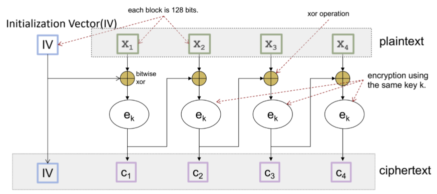
Large plaintext is divided into blocks before applying the block cipher.

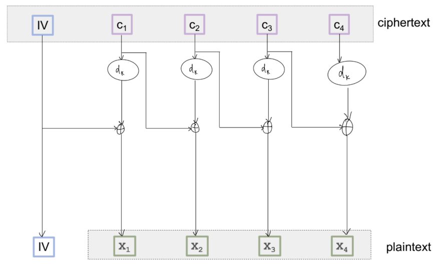
ECB (electronic code book) mode



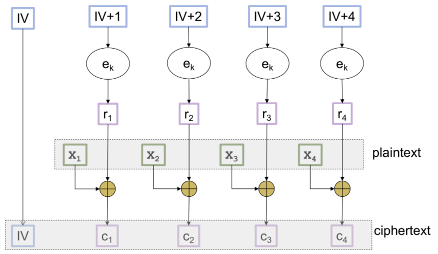


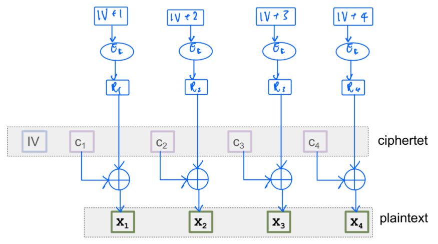
CBC (cipher block chaining) mode on AES
- Initialization vector (IV) is an arbitrary value chosen during encryption, must be different in different encryptions.
- In CBC mode, IV must be unpredictable, else it is susceptable to BEAST attack.
- If IV is randomly chosen, it is unpredictable





CTR (counter) mode



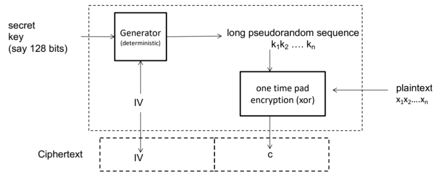


GCM mode (Galois/counter)

Authenticated encryption, ciphertext consists of extra tag for authentication. Secure in the presence of decryption oracle.

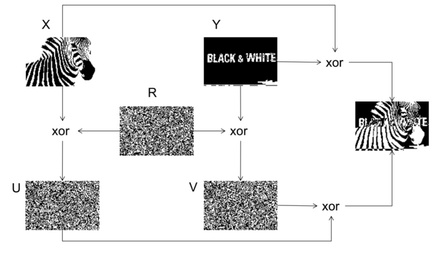
1.4.3 Stream cipher and IVs

Stream cipher is bit by bit. CTR mode is a "stream cipher" but it is not bit by bit.

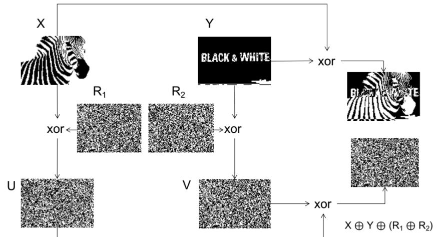

secret key (say 128 bits)
Generator (deterministic)
long pseudorandom sequence k₁k₂ .... kₙ
one time pad encryption (xor)
plaintext x₁x₂....xₙ
IV
Ciphertext
IV
c

- Need IV and no two IVs can be the same

Stream cipher without IV



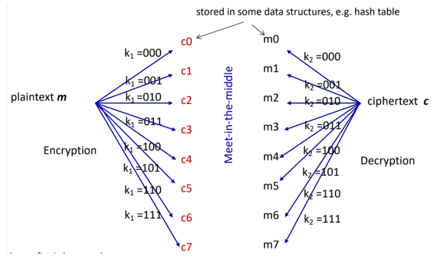
Stream cipher with IV



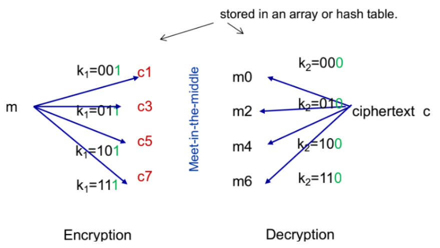
- IV makes an encryption probabilistic

1.5 Examples of attacks on crypto

1.5.1 Meet-in-the-middle
- DES is not secure $\rightarrow$ improve by encrypting multiple times using different keys
- Consider double encryption under known plaintext attack. Attacker has m and c and wants to know k_1, k_2.
- Using exhaustive search, amount of DES encryption/decryption would be 2^{56+56}
- Hence use meet-in-the-middle attack.
- for k-bit keys, this reduces the number of crypto operations to 2^{k+1}



Tradeoff with time and space



- Last bit of k_1 fixed to 1, last bit of k_2 fixed to 0
- Perform meet-in-the-middle on the first 2 bits of k_1 and k_2

1.5.2 Padding Oracle

Plaintext needs to be padded to split into blocks
- PKCS#7 is a padding standard

01

02 02

03 03 03

04 04 04 04

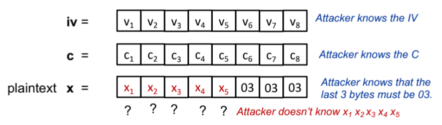
....

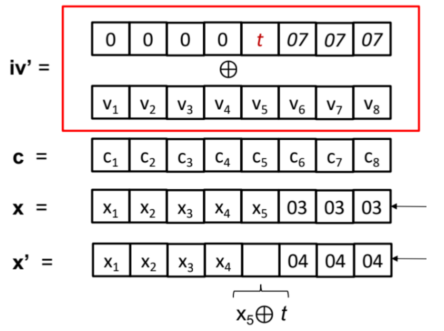
08 08 08 08 08 08 08 08

Padding oracle attack

Padding oracle attack on AES CBC mode

Attacker knows:





$$iv \oplus d(c) = 03$$

$$iv' \oplus d(c) = 04$$

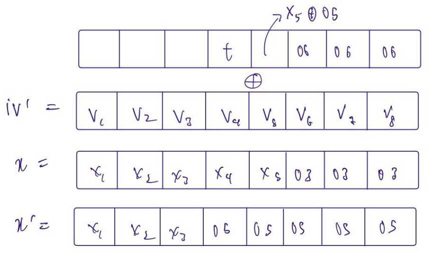
xor the 2 tgt to get $iv' = 07 \oplus iv$

$$iv' = iv \oplus 00\ 00 \ldots t\ 07\ 07\ 07$$

$$d(C_5) \oplus t \oplus V_5 = 04$$
$$d(C_5) \oplus V_5 \oplus t = 04$$
$$d(C_5) \oplus V_5 = x_5$$
$$x_5 \oplus t = 04$$
$$x_5 = 04 \oplus t$$

Keep guessing t until padding oracle outputs YES, then we know x_5

To get next byte:



1.6 Pitfalls in usages and implementations
1. Wrong choice of IV / reusing one-time pad
2. Randomness is predictable
3. Modify existing or design your own encryption scheme
4. Reliance on obscurity: Kerkchoff's principle
   - Kerkchoff's principle: A system should be secure ven if everything about the system, except the secret key, is public knowledge

Topic 2: Authentication Credential

Authentication

Authentication is the process of assuring that the communicating identity, or origin of a piece of information, is the one that it claims to be.
- Authentication implies integrity.

Data-origin authentication: is a piece of data generated by an authentic entity?
- Signature or MAC (message authentication code)

Commnication authentication: is the enitity interacting with the verifier an authentic entity?
- Authentication protocol

2.2 Password

Password vs key

Passwords are generated by human and can be remembered by human. Keys are binary sequences that are infeasible to be remembered by humans.

Password system
1. Bootstrapping
   - User and server establish a common password, server keeps a password file keeping the identity and the corresponding password
   - Password established during boostrapping either by a default password or by the server/user choosing a password and sending it to the user/server through another communication channel
2. Authentication
   - Server authenticates an entity. An entity who can convince the server that it knows the password is deemed to be authentic.
3. Password reset
   - Need to authenticate the entity before allowing the entity to change password.
   - Need another credential (other than the old password) to authenticate bc ppl might want to reset password when they forget
   - Can be done through OTP, security question (not secure as entropy of answers is lower than the password)

Attack on passwords

2.2.1 Attack on Bootstrapping
- Attacker intercepts the password during bootstrapping, e.g. if password is sent through postal mail, an attacker steals the mail to get the password
- Attacker uses the "default" passwords
  - Mitigation: require the user to change password after first login

- Example: zoom flaw allowed account hijacking

2.2.2 Attack on Password Reset

- Mechanism of security questions weakens the password system, but it is less common now
- Social engineering + password reset

2.2.3 Searching for the password

Dictionary attacks

- Test passwords using a dictionary that could contain words from English dict, known compromised passwords, etc.
- Also test combinations of words in the dictionary. e.g. combinations of 2 words, all possible capitalizations of letters in each word, substituting 'a' with '@', etc.
- **Online dictionary attack**
  - To test a password, attacker must interact with the authentication system
- **Offline dictionary attack**
  - Attacker first obtains some information D about the password, possibly by sniffing the login session of an authentic user, or by interacting with the server. (e.g. attacker obtains the hashed password)
  - Next, the user carries out dictionary attack using D without interacting with the system (e.g. attacker compares the hashed password with the hashed words in dictionary)
- Guessing password from social information

2.2.4 Stealing the password

1. Sniffing
   - Shoulder surfing: look-over-the-shoulder attack
   - Sniffing the communication: Some systems simply send the password over the public network in clear (i.e. not encrypted), e.g. FTP, Telnet, HTTP
   - Sniff wireless keyboards that employ insecure encryption method
   - Using sound made by keyboard
   - Viruses, key-logger
     - Key-logger captures keystrokes and sends the info back to the attacker.
     - Can be in the form of software (viruses) or hardware.
2. Phishing
   - Victim is tricked into voluntarily sending the password to the attacker
   - Asks for password under false pretense
3. Spear Phishing
   - Phishing that is targeted to a particular small group of users, e.g. NUS staff

Phishing Prevention

- User training
- Blacklisting, e.g. phishtank.com
- Visually spot by ensuring that there is a padlock in the address bar and that the domain name in the url is correct

2.2.5 Password strength

- We quantify the key-strength by the size of the key if best-known attack is exhaustive search.
- If best-known attack is faster, then we quanitfy it by its equivalent in exhaustive search.

Using strong password

- Truly random password: password is chosen randomly among all possible keys using an automated passsword generator. High entropy but difficult to remember.
- User selection:
  - Mnemonic method
  - Altered passphrases
  - Combining and altering word
- Usability:
  - Strong passwords are difficult to remember
  - It is difficult to enter alphanumeric passwords into mobile devices. There are alternatives, e.g. graphical or gesture-based

Password entropy

Suppose a set P contains N unique passwords. A password is chosen by randomly picking a password from P. Entropy of password is

$$-\sum_{i=1}^{N} p_i \log_2 p_i$$

where p_i is the probability that the i-th password is picked.

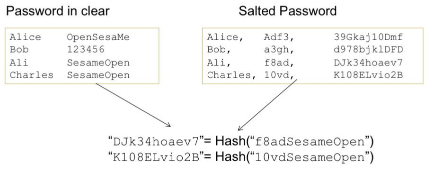
If the password is chosen uniformly, each password in P has probability of $1/N$ of being chosen. The entropy of the password is

$$log_2 N \text{ bits}$$

For the entropy to be highest for a set of N items, p_i must be $1/N$

Additional protection to password files

Password file should be hashed and salted



Password in clear

```
Alice     OpenSesaMe
Bob       123456
Ali       SesameOpen
Charles   SesameOpen
```

Salted Password

```
Alice,    Adf3,    39Gkaj10Dmf
Bob,      a3gh,    d978bjk1DFD
Ali,      f8ad,    DJk34hoaev7
Charles,  10vd,    K108ELvio2B
```

"DJk34hoaev7"= Hash("f8adSesameOpen")
"K108ELvio2B"= Hash("10vdSesameOpen")

Make it harder for rainbow table attack

2.3 Biometric

Biometric data is the password

FMR (false positive)

$$= \frac{\text{no. of successful false matches (B)}}{\text{no. of attempted false matches (B + D)}}$$

FNMR (false negative)

$$= \frac{\text{no. of rejected genuine matches (C)}}{\text{no. of attempted genuine matches (A + C)}}$$

	accept	reject
genuine attempt	A	C
false attempt	B	D

Threshold: FNMR/FMR. Lower threshold more relax, higher threshold more stringent

Attack on biometric system

Biometric data can be spoofed, use liveness detection e.g. temperature sensor in fingerprint scanner

2.4 n-factor Authentication and Multi-Step Verification

n-factor Authentication

Requires at least 2 different authentication "factors"

1. Something you know: password, PIN
2. Something you have: Security token, smart card, phone, ATM card
3. Who you are: Biometric

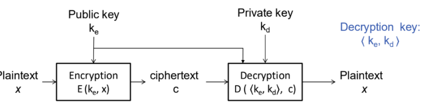
Multi-Step Verification

If both are the same category of factors (2 passwords, both are something you know) then it is 2-step verification

Topic 3: Authenticity (data origin)

3.1 PKC

- With multiple identitites, many pairs of symmetric keys are required.
- Symmetric key requires both entities to know each other before the actual communication session. Hence use public key encryption.

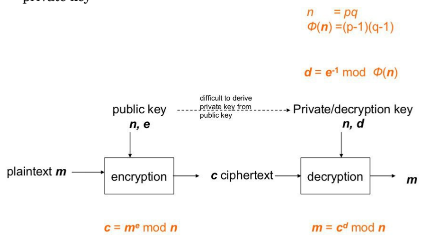


Popular PKC schemes

- RSA
- ElGamal
- Paillier
- Post-quantum cryptography

3.1.1 RSA

1. Owner randomly chooses 2 large primes p, q and computes $n = pq$
2. Owner randomly chooses an encryption exponent e s.t. $\gcd(e, (p-1)(q-1)) = 1$
3. Owner finds decryption exponent d where $de \mod (p-1)(q-1) = 1$
4. Owner publishes $\langle n, e \rangle$ as public key, and safe-keeps d as the private key



Got algo to find d given e, p, q. For faster speed, choose small e. Common value is 65537. e is not a secret in such cases

3.1.2 Security of RSA

Getting RSA private key from public key is as difficult as factorizing n.

Padding of RSA

- Some forms of IV is required so that encryption of the same plaintext at different times would give different ciphertexts. Additional padding required for security.

3.1.3 Efficiency

RSA encryption/decryption is significantly slower than AES. Can use PKC to encrypt a symmetric key then use AES for encryption

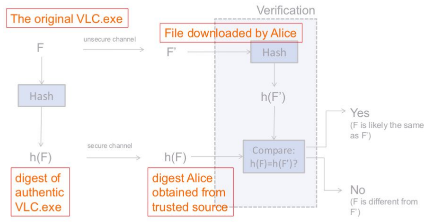
3.2 Data Authenticity

Security requirement of hash

- Collision-resistant
  - Collision: Find 2 different messages m_1, m_2 s.t. $h(m_1) = h(m_2)$
- 2nd pre-image resistant
  - 2nd pre-image: Given m_1, find m_2 s.t. $h(m_1) = h(m_2)$
- One-way
  - Pre-image: Given y, find m s.t. $h(m) = y$

Application of unkeyed hash

When downloading something from a website, match the hash of the file with the checksum displayed in the browser.



If not 2nd pre-image resistant, can be attacked

3.3 Data Origin Authenticity (mac), keyed

Keyed-hash is a function whose input is an arbitrary large message and a secret key, output is a fixed-size mac (message authentication code)

- Security requirement (forgery): Even if attacker sees 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
- CBC-MAC: based on AES operated under CBC mode
- HMAC: Hashed-based MAC based on SHA

Application of mac

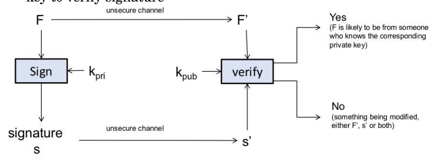
Same situation as before but dh secure channel to deliver digest. Protect the digest with the help of some secrets.

- In symmetric key setting, called mac
- In public key setting, called digital signature
- mac typically appended to file, also called authentication tag or authentication code

3.4 Data Origin Authenticity (Signature)

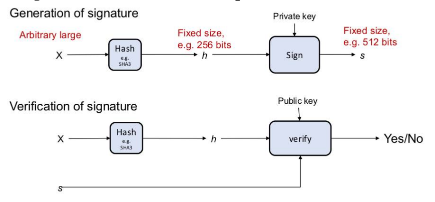
Public key version of MAC is called signature

- Owner uses private key to generate signature, public uses public key to verify signature


F
unsecure channel
F'
Sign
k_{pri}
k_{pub}
verify
Yes
(F is likely to be from someone who knows the corresponding private key)
No
(something being modified, either F', s' or both)
signature s
unsecure channel
s'

Signature is appended to the file F

- Signature scheme achieves **non-repudiation**



hash and sign / hash and encrypt

3.5.1 Birthday Attacks

Birthday attack is used to find collision.

Suppose we have M messages, and each message is tagged with a value randomly chosen from $\{1, 2, 3, \ldots, T\}$. Then the probability that there is a pair of messages tagged with the same value is approx

$$1 - e^{\left(-\frac{M^2}{2T}\right)}$$

Let S be a set of k distinct elements where each element is an n-bits binary string. Let us independently and randomly select m n-bit binary strings and put them in the set T. The probability that S has non-empty intersection with T is more than

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

4. PKI + Channel Security

4.1 Distribution of public keys

- PKC requires a secure broadcast channel to distribute public keys: PKI
- If no secure way to distribute public key, attacker can impersonate by giving his own public key
- Certificate: A piece of document that binds a "name" to a "public key" & certified by a CA
- Certificate contains:
  - Name, public key, expiry date
  - Meta info: usages, type of crypto, name of CA, etc
  - CA's signature

4.2 PKI

4.2.1 Certificate & CA

- Certificates are used to distribute public keys. A CA issues certificates.
- CA: trusted authority that manages a directory of public keys
- CA has its own public-private key, some CA's public keys have been distributed securely through other means.
- OSes and browsers have pre-loaded CA's public keys, these CAs are known as root CAs.
- Other CA's public keys added through chain-of-trust
- A **certificate** is a digital document containing at least the following:
  - Name (e.g. alice@yahoo.com / bbc.com / *.bbc.com)
  - Public key of the owner
  - Time window that this cert is valid
  - Signature of CA

4.2.2 CA's chain-of-trust

Responsibility of CA

- Issue certificate
- Verify that information is correct (by checking that the applicant indeed owns the domain name)

Certificate chain-of-trust

- If Alice's cert is issued by CA#1, but Bob doesn't have the public key of CA#1, then Alice can send her email, her cert, and CA#1's cert (issued by root CA) to Bob
- Bob can now:
  - Verify CA#1's certificate using root CA's public key
  - Verify Alice's certificate using CA#1's public key
  - Verify Alice's email using ALice's public key
- Bob can also obtain CA#1's certificate from other sources

4.2.3 Revocation

Reasons for revoking non-expired certs:

- Private key compromised (breaches, insider attack, vulnerability in choosing private keys)
- System admin left an organization
- Business entity closed
- Issuing CA was compromised

Verifier needs to check whether certificate is still valid, even if it has not expired. 2 approaches:

- Certificate Revocation List (CRL)
  - CA periodically signs and publishes a revocation list
- Online Certificate Status Protocol (OCSP)
  - OCSP Responder validates a cert in question. Need online OCSP responder

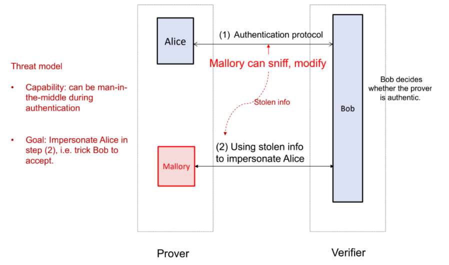
Recommendation: User periodically updates its local cache of revocation list, user does not need to check online to verify a cert

4.3 Limitations / attacks on PKI

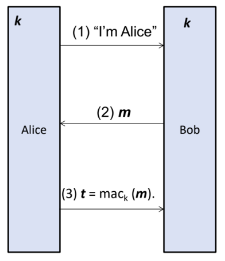
1. Implementation bugs
   - Some browsers ignore substrings in the "name" field after the null characters when displaying it in the address bar but include them when verifying the cert.
     - Name in cert: `www.comp.nus.edu.sg\0.hacker.com`
     - Displayed in browser as: `www.comp.nus.edu.sg`
2. Users think they are connected to `www.comp.nus.edu.sg` but are actually connected to `www.comp.nus.edu.sg\0.hacker.com`
3. Abuse by CA
   - Rogue CA can forge any certificate
4. Social engineering
   (a) Typosquatting
   - An attacker registers for a domain name that looks similar to another website

   (b) Sub-domain
   - Attacker is the rightful owner of `134566.com`
   - Attcker creates a sub domain `luminus.nus.edu.sg.134566.com`
   - Attacker can get a valid certificate

4.4 Protocol 1: Authentication

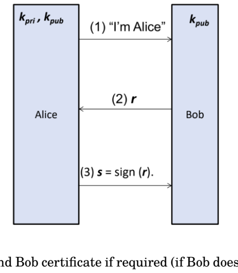

Threat model
Capability: can be man-in-the-middle during authentication
Goal: Impersonate Alice in step (2), i.e. trick Bob to accept.
Alice
(1) Authentication protocol
Mallory can sniff, modify
Stolen info
Mallory
(2) Using stolen info to impersonate Alice
Bob
Bob decides whether the prover is authentic.
Prover
Verifier

To prevent replay, use challenge-response



- m is picked at random
- k is shared secret between Alice and Bob
- Protocol only authenticates Alice. Unilateral authentication.

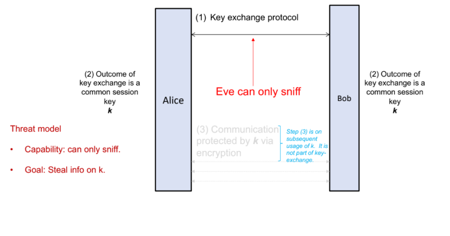
Can do unilateral authentication using PKC as well



- Alice may send Bob certificate if required (if Bob doesn't have Alice's public key)

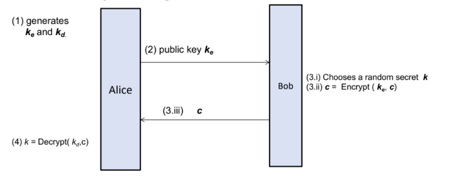
If only authentication is done, Mallory can interrupt and take over the channel after Bob is convinced that he is communicating with Alice. Use authenticated key-exchange to get a new shared secret k known as session key. Then all subsequent communication will be encrypted + mac using k

4.5 Protocol 2: Key Exchange

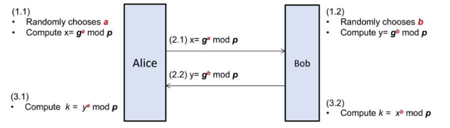

(1) Key exchange protocol
(2) Outcome of key exchange is a common session key k
Alice
Eve can only sniff
Bob
(2) Outcome of key exchange is a common session key k
Threat model
Capability: can only sniff.
Goal: Steal info on k.
(3) Communication protected by k via encryption
Step (3) is on subsequent usage of k. It is not part of key-exchange.

It is about confidentiality. Authenticity not considered.

PKC-based key exchange

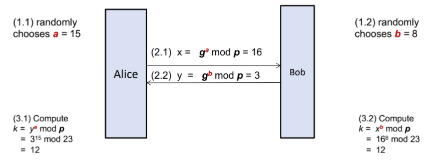

(1) generates k_e and k_d.
(2) public key k_e
Alice
Bob
(3.i) Chooses a random secret k
(3.ii) c = Encrypt (k_e, c)
(3.iii) c
(4) k = Decrypt(k_d,c)

Diffie-Hellman key exchange



- Alice and Bob have agreed on g and p. They are not secret and known to the public.
- Security relies on the CDH (computational diffie-hellman) assumption: given g, p, $x = g^a \% p$, it is computationally infeasible to find $k = g^{ab} \% p$

g = 2, p = 23

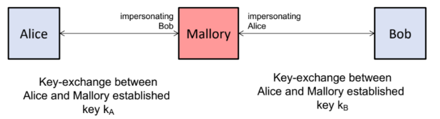

(1.1) randomly chooses a = 15
Alice
(2.1) x = g^a mod p = 16
(2.2) y = g^b mod p = 3
Bob
(1.2) randomly chooses b = 8
(3.1) Compute $k = y^a$ mod p = 3^{15} mod 23 = 12
(3.2) Compute $k = x^b$ mod p = 16^8 mod 23 = 12

From 16, and 3, it is very difficult to get 12.
(here, we are referring to very large, say 2000-bit integer)

Diffie-hellman meets forward secrecy requireent but PKC based method doesn't. TLS 1.3 mandates forward secrecy

4.6 Protocol 3: Authenticated Key Exchange

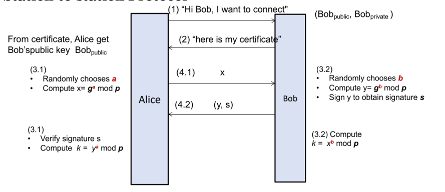
Key exchange alone cannot guard against Mallory:


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

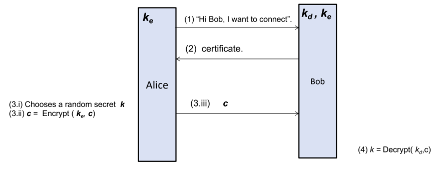
Need authenticated key-exchange, which can be easily obtained from existing key exchange.

- Diffie Hellman based:
  - Sign all communication using private key. Known as station-to-station protocol.
- PKC based:
  - Omit setep (1) (generating public/private keys) and use Alice's existing public/private keys. Since only Alice has her private key, the entity that can correctly decrypt it must be Alice

Station-to-station Protocol


(1) "Hi Bob, I want to connect"
(Bob$_{public}$, Bob$_{private}$)
(2) "here is my certificate"
From certificate, Alice get Bob'spublic key Bob$_{public}$
(3.1)
Randomly chooses a
Compute x= g^a mod p
(4.1) x
Alice
(4.2) (y, s)
Bob
(3.2)
Randomly chooses b
Compute y= g^b mod p
Sign y to obtain signature s
(3.1)
Verify signature s
Compute $k = y^a$ mod p
(3.2) Compute $k = x^b$ mod p

PKC-based Authenticated Key-exchange (AKA RSA-based)

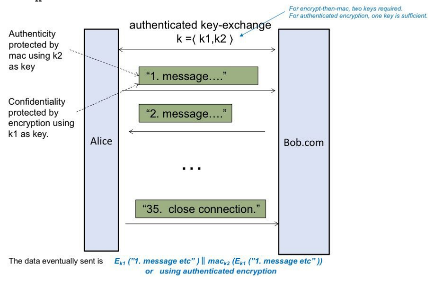


k_e
(1) "Hi Bob, I want to connect".
k_d, k_e
(2) certificate.
Alice
Bob
(3.i) Chooses a random secret k
(3.ii) c = Encrypt (k_e, c)
(3.iii) c
(4) k = Decrypt(k_d,c)

Mutual Authenticated Key exchange
- Before:
  - Alice has a pair of public, private keys (A_{public}, $A_{private}$)
  - Bob has a pair of public, private keys (B_{public}, $B_{private}$)
  - Alice knows Bob's public key and vice versa. The two sets of keys are known as the long-term key or master key
- Carry out authenticated key exchange protocol (e.g. STS). If an entity is not authentic, the other will halt.
- After:
  - Both A and B obtain shared key k, known as session key
- Security requirement:
  - (Authenticity) Alice is assured that she is communicating with an entity who knows $B_{private}$
  - (Authenticity) Bob is assured that he is communicating with an entity who knows $A_{private}$
  - (Confidentiality) Attacker is unable to get the session key

4.7 Securing Communication Channel
TLS

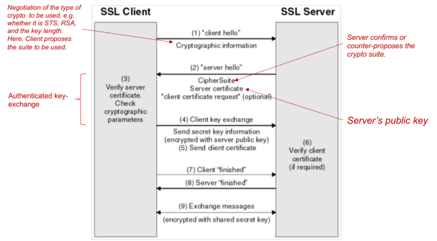
Alice wants to visit `bob.com`.
1. Bob sends his certificate to Alice.
2. Alice and `bob.com` carry out unilateral authenticated key exchange protocol with Bob's public/private key. After the protocol, both Bob and Alice obtain k, which could come in the form of $k = \langle k_1, k_2 \rangle$ whwere k_1 is the secret key of the MAC, and k_2 is the secret key of the symmetric-key encryption, or a single key k when authenticated encryption (e.g. GCM) is in use. These keys are called the session keys. By property of the protocol, Alice is convinced that only Bob and herself know the session key.
3. 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
   $\mathtt{E}_{\mathtt{k1}}\mathtt{(i || m) || mac}_{\mathtt{k2}}\mathtt{(E}_{\mathtt{k1}}\mathtt{(i || m))}$ (still in use but not recommended)
   For GCM mode or other authenticated encryptions, the message to be sent is simply
   $\mathtt{E}_{\mathtt{k}}\mathtt{(i || m)}$



For encrypt-then-mac, two keys required.
For authenticated encryption, one key is sufficient.
authenticated key-exchange
k =⟨ k1,k2 ⟩
Authenticity protected by mac using k2 as key
Confidentiality protected by encryption using k1 as key.
"1. message...."
"2. message...."
...
"35. close connection."
Alice
Bob.com
The data eventually sent is E_{k1} ("1. message etc") ‖ mac_{k2} (E_{k1} ("1. message etc"))
or using authenticated encryption

- SSL and TLS are protocols that secure communication using cryptographic means
- SSL is the predecessor of TLS
- HTTPS is built on top of TLS

TLS handshake

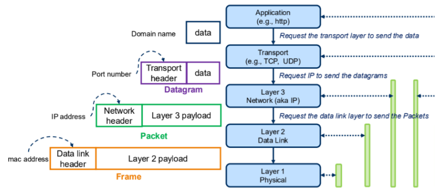


Negotiation of the type of crypto to be used, e.g. whether it is STS, RSA, and the key length. Here, Client proposes the suite to be used.
SSL Client
SSL Server
(1) "client hello"
Cryptographic information
Server confirms or counter-proposes the crypto suite.
(2) "server hello"
CipherSuite
Server certificate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

4.8 Forward Secrecy
If Eve cannot recover plaintext even though she knows the master key, then the authenticated key exchange achieves forward secrecy
- PKC-based authenticated key exchange does not achiever forward secrecy
- Station-to-station key exchange achieves forward secrecy
  - If attacker can solve CDH, then forward secrecy of station-to-station is compromised

5. Network Security
5.2 Background: networking
Computer network: a collection of interconnected devices that can communicate with one another

Network layers
1. Physical (wifi, ethernet)
2. Data link
   - MAC address
3. Network (IP)
   - IP address
4. Transport (TCP, UDP)
   - Port number
5. Application
   - Domain name

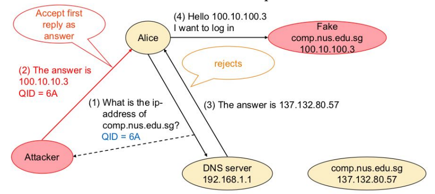


Transport + IP layer
- Often treated as one single layer
- Address of a communicating entity is an ip addr and a Port
- Each node in the network has 65535 ports
- Communication channel between 2 nodes is established by connecting 2 ports
- Src ip, src port to dest ip, dest port

5.3 Network attacker
Unless otherwise stated, MITM can sniff, spoof, modify, drop the header and payload
MITM in layer x means MITM along the layer x virtual connection (MITM can see and modify data unit of that layer)
1. MITM in the physical layer
   - Tap into the internet cable, sniff the wireless communication in the cafe
2. MITM in the link layer
   - Malicious cafe owner who offers the wifi
3. MITM in the IP / Transport layer
   - ISP (e.g. Singtel)
4. MITM in the application
   - Malware in the browser

5.4 DNS attacks
DNS query and answer
- Client sends a query to DNS server using UDP
- DNS sends the answer back using UDP
- The query contains a 16-bit number known as QID (query ID)
- The response from the server must also contain a QID
- If the QID in the response does not match the QID in the query, the client rejects the answer

DNS spoofing
- Alice is using free cafe wifi and wants to visit and log in to `comp.nus.edu.sg`
- Consider an attacker who is also in the cafe. Since the wifi is not protected, the attacker can
  - Sniff data from the communication channel
  - Inject spoofed data into the communication channel
- The attacker cannot remove/modify data sent by Alice
- Attacker owns a webserver which is a spoofed NUS website



Accept first reply as answer
(4) Hello 100.10.100.3
I want to log in
Alice
Fake comp.nus.edu.sg 100.10.100.3
rejects
(2) The answer is 100.10.10.3
QID = 6A
(1) What is the ip-address of comp.nus.edu.sg?
QID = 6A
(3) The answer is 137.132.80.57
Attacker
DNS server 192.168.1.1
comp.nus.edu.sg 137.132.80.57

Cannot verify if response is coming from correct source or has not been modified - only matches QID
If cached into local DNS server (should have expiry), Alice will go to the fake website all the time

Denial of Service on DNS
- A DNS server an be a "single-point-of-failure" of the network
- DoS attacks can be launched on a web service instead of directly attacking the web server
- When DNS server is down, the web service is no longer reachable
- Countermeasures: redundant servers, rate limiting, etc.

5.5 Poisoning ARP table
Address resolution protocol (ARP)
- Resolution of IP address to mac address
- Data link layer
- When a device knows the IP address of the next hop router on the same network but needs the corresponding MAC address
- ARP resolves the router's IP address ot its MAC address to allow packet forwarding at the data link layer

Switch
- Directs data packets between devices or nodes on teh same local network using MAC addresses
- The switch keeps a table that associates the port to the mac addresses
- Switch does not understand and does not store IP addresses.

Address resolution Protocol (ARP) Table
- An ARP table is a database maintained by each device or nodes on a subnet
  - Stores mappings between IP addresses and MAC addresses

Eg. if N2 wants to send to IP addr 10.0.1.4,
- N2 looks up ARP table to find MAC addr
- N2 sends frame to swtich, specifying destination MAC addr
- Switch looks up table to find port
- Switch redirects frame to correct port

If ARP table does not have info of a particular IP addr, N2 will broadcast an ARP request packet to all devices on the local network "Who has IP address X.X.X.X? Tell me your MAC addr"
The device with the requested IP address recieves the ARP request and replies with an ARP reply packet. He sends the IP address and MAC address over. Then the requesting node will update its ARP table with the new IP-to-MAC address mapping
Any node can also broadcast the info or to a specific node even if there isn't a request

Attack: N1 wants to be MITM
- N1 informs N2 that N3's MAC address is N1's
- N1 informs N3 that N2's MAC address is N1's
- ARP tables of N2 and N3 are now poisoned
- All the frames will be sent to N1, and N1 can relay the frames, or modify the frames before relaying
- If N1 does not forward, then it is DoS.

DoS Attacks
- Attempt to disrupt the normal functioning of a targeted server, service or network $\Rightarrow$ affect availability
- Many successful DoS attacks simply fllood the victims with overwhelming requests/data
- DoS carried out by a large number of attackers is called DDoS (distributed denial of service)

Reflection Attack
- A type of DoS in which the attackers send requests to intermediate nodes, which in turn send overwhelming traffic to the victim.
- Attacker spoofs the victim's IP address as the source
- Intermediate nodes then send responses back to the spoofed IP (the victim)
- Indirect $\Rightarrow$ more difficult to trace
- Preventive measures:
  - Most routers are configured to not broadcast by default
  - Configure firewalls to block incoming ICMP echo requests packets directed at broadcast addresses

Amplification Attack
- Variation of reflection attacks where the intermediate nodes response is significantly larger than the attacker's request
- A single request could trigger multiple responses from the intermediate nodes

5.7 Securing different layers
SSL / TLS
- Between layers 4 and 5
- On top of transport layer
- When an application wants to send data to the other end point, it first passes the data and the address to SSL / TLS
- SSL / TLS then protects the data using encryption and mac

IPSec
- Layer 3
- Mechanism that protects the IP layer and secures all IP traffic between endpoints
- Securing network connections between host-to-host, network-to-network (gateways), or network-to-host (gateway and host)

WPA2 (wifi protected access 2)
- Layers 1 and 2
- Protect data transmitted over wifi networks
- Offers protection in layer 2 and layer 2
- Data travelling between a wireless device and the access point is confidential and protected from eavesdropping
- Not all information in layer 2 are protected

VPN (virtual private network)
- Tunnel at layer 3
- Enable remote user to securely connect to private network
- VPN client and VPN server establish a connection, called a tunnel. After authentication and verification, establish session keys
- When Alice communicates with Bob, VPN client encrypts the entire payload and adds a new IP header
- From Bob's POV, Alice is communicating with the virtual node with IP address in NUS and does not know Alice's actual IP address

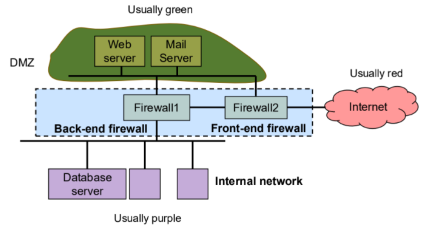
Which layer to protect
- By protecting lowest layer, can protect info in all layer but not feasible.
- Intermediate node need to access some higher layer info and sit in higher layer. Hence malicious intermediate node could be a MITM in higher layer.
- TLS / SSL + WPA2
- IPSec is expensive to implement and difficult to deploy

5.6 Firewalls and IDS
In a computer network in an organization, some nodes contain more sensitive information than other. Some nodes are more "secure" (certain nodes operate in a more hostile environment). We need to divide the network into segments and deny unnecessary access.
- Principle of least privilege: control access to the network
- Compartmentalization: keep things separated to limit the impact of any single failure or attack

Tools to control access to the network: firewall , IDS (intrusion detection system)
Firewall: Gatekeeper, prevents unauthorized access
IDS: Watcher that raises alerts by monitoring and analysing

Firewall and DMZ
- A firewall controls what traffic is allowed to enter the network (ingress filtering) or leave the network (egress filtering)
  - Firewall are devices or programs that control the flow of network traffic between networks or hosts that employ differing security postures.
- DMZ: Demilitarized zone
  - A sub-network that exposes the organization's external service to the (untrusted) internet.
  - Separates an internal local area network (LAN) from untrusted external networks, usually the internet.
  - Adds an extra layer of security to an organization's internal network.
- DMZs are created using firewalls

2-firewall setting



If the web / mail server in DMZ is compromised, the attacker still has to bypass the back-end firewall to reach sensitive data.

One firewall restricts external access, the other restricts access to the internal network

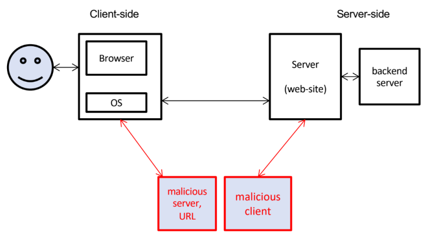
Packet filtering / screening
- Firewall's controls are achieved by "packets filtering"
- Packet filtering inspects every packet, typically only on the TCP / IP packet's header information (network & transport layer).
- If the payload is inspected, we call it deep packet inspection (DPI).
- Actions: Allow to pass, drop, reject, log, notify admin, modify

Types of Firewall
1. Packet filters
   - Inspect only header (mainly IP packet's header)
   - Use: blocking traffic from certain IP or port
2. Stateful inspection
   - Keep a state on previously received packets
   - E.g. counting number of connection a particular IP address has made in the past one hour
   - Use: blocking abnormal connection pattern or unauthorized session attempts
3. Proxy
   - Act as intermediaries that fully receive inspect, and forward (possibly modify) packets between client and server
   - Use: block certain URLs or scan for malware in HTTP traffic

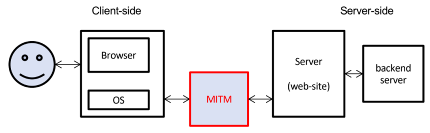
Intrusion Detection System (IDS)
- An IDS is a security tool or software that monitors computer systems and networks for signs of malicious activity, policy violations, or security breaches.
- An IDS system consists of a set of "sensors" that gather data such as logs, network packets, etc. Sensors can be deployed on hosts, or netowrk router.
- Data are analyzed for intrusion either in real-time or after collection to detect suspicious patterns, attacks, or abnormal behaviour.

Three types of IDS
- Attack Signature Detection
  - Looks for specific, well-defined patterns or "signatures" of known attacks in the data collected by sensors (e.g. using certain port number / source ip addr)
- Abnormaly Detection
  - IDS attempts to detect abnormal patterns that deviate from the established "normal" behaviour of the network (e.g. sudden surge of packets with certain port number)
- Behavior-based IDS
  - Anomaly detection that focuses on human behaviour (e.g. system keeps the profile of each user and detects any user who deviates from the profile)

6. Web security
6.1 Background
Threat model 1: attackers as another end system



Threat model 2: attackers as MITM

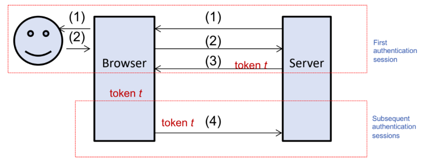


- Attacker is MITM in the IP or lower layer
- Can gain MITM in a few ways
  - Cafe owner providing free wifi
  - DNS spoofing attack or ARP attack
  - Owner of the VPN server, last hop in TOR network

6.3 Misleading user on domain name
- Hostname could contain character that resembles "/"
- Address bar spoofing
  - Address bar is the only indicator of which url the page is rendering
  - If the address bar can be "modified", attacker can trick the user to visit a malicious url X, but user thinks that the url is Y.
  - Poorly-designed browser can allow attacker to achieve that (e.g. Last time malicious page could overlay spoofed address bar on top of the actual bar. Now cannot anymore)

6.4 Cookie and same-origin policy
The script in a web page A can access cookies stored by another web page B only if both A and B have the same origin (protocol, hostname, port number)

Application of cookie: token-based authentication.



(1)
(2)
Browser
(1)
(2)
(3)
token t
Server
token t
token t
(4)
First authentication session
Subsequent authentication sessions

(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.

Choice of token

If token t is a random number, server needs a db to store all issued tokens. Instead, can use
- (secure) MAC. Use random value or some meaningful information like expiry date, concat w mac computed using the server secret key
  - Relies on security of mac
- (insecure) Some meaningful info concat w a sequence number that can be predicted
  - Relies on obscurity. If attacker knows how it is generated, he can forge it

6.5 XSS (cross site scripting) attacks
The one where u put the `<script></script>`
In some websites, if the browser sends a text containing a substring s, then replying html sent by server would contain s. Can add a script in s.

Attack
1. Attacker tricks a user to cick on a url, which contains the target website, and a malicious script s.
2. The request is sent to the server.
3. The server constructs a response html. The response contains the script s.
4. Browser renders the html page and runs s.
5. Malicious script could deface the original webpage or steal cookie.

The attack exploits the client's trust of the server

Types of XSS
- Reflection (non-persistent)
- Stored XSS (persistent). Script s is stored in the targeted website (e.g. in forum page)

Defense

Input validation carried out by server

XSRF (cross site request forgery / sea surf)

Reverse of XSS. Exploits server's trust of client.
Example:
1. Suppose client A is already authenticated by a targeted website `www.bank.com` and the site keeps a cookie as "token".
2. Attacker B tricks A to click on a url of S. The url maliciously requests for a service, e.g. transferring $1000 to Bob's account
3. Since cookie sent to S, A is already authenticated and transaction will be carried out.

Defense

Include authentication information in request as param in url

6.7 Other attacks
Web tracking, drive-by-download, pixel stealing, clickjacking, user interface redress attack, CAPTCHA, click fraud

Common simple implementation mistakes

Client side authentication / filtering, security credential embedded in public web pages, sever's secrets stored in cookies, configuration errors, URL as secrets (e.g. in password reset link or zoom link)

8. Secure programing
Program must be correct, efficient, secure

8.1 Unsafe function `printf()`
- `printf(format, s)`
- e.g. `printf("the value in temp is %d\n", temp)`

Example 1

`printf("hello world %d")` $\Rightarrow$ `printf()` will still fetch value of 2nd param from the supposing location and display it $\Rightarrow$ unknowingly revealed extra info to the person observing the screen. Confidentiality compromised.

Example 2
```
        int main() {
            char t[100];

            scanf("%s", t);
            printf(t);
        }
```
User can set `t` to be "`hello world %d`", then can get info

Preventive measure

Avoid:
- `printf(t)`
- `printf(t, a1, a2)`

Use:
- `printf("hello");`
- `printf("The value of %s is %d", a1, a2);`

How it can be exploited
1. Obtain more information (confidentiality)
2. Cause the program to crash (execution integrity)
3. Modify the memory content using `"%n"` (memory integrity which might lead to execution integrity)

If the program that invokes `printf()` has elevated privilege, a user might be able to obtain information that was previously inaccessible.

8.2 Data Representation
Example 1: string representations
- `printf()` uses null termination. Length is not stored.
- Some systems use non-null termination.
- Systems that use both null and non-null definitions to verify the certificate may get confused
  - E.g. refer to 4.3 for when browser verfies cert based on non-null but displays name based on null.

Example 2: IP address

IP address can be represented as
- String e.g. "132.127.8.16"
- 4 integers, and each is a 32-bit integer
- A single 32-bit integer

- etc.

E.g. A blacklist is stored in 4 arrays of integers `A, B, C, D`. Function `BL` takes in 4 integers `a, b, c, d` and check if the IP address represented is in the blacklist. It searches for the index `i` s.t. `A[i] == a, B[i] == b, C[i] == c, D[i] == d`
Suppose another program that uses BL is written using the following flow:

1. Get a string `s` from user.
2. Extracts four integers in this way:
3. `a, b, c, d = int(s.split("."))` where int converts to 32-bit int
4. Invokes `BL(a, b, c, d)`. If TRUE, quits
5. Else, let `ip` $= a \times 2^{24} + b \times 2^{16} + c \times 2^{8} + d$ where `ip` is a 32-bit int
6. Continue the rest of the processing with address `ip`

The above program is vulnerable because if ip address of 11.12.1.0 is blacklisted, user can change input string to “11.12.0.256”, then a, b, c, d = 11, 12, 0, 256 and not detected by `BL` but ip becomes 11.12.1.0.
To prevent this, **use canonical representation** by converting to a standard representation immediately. Do not trust input from user.

8.3 Buffer overflow
C and C++ do not employ “bound check” during runtime. Efficient but prone to bugs.

```
int a[5]; // Size 5, up to index 4
int b;
int main() {
    b = 0;
    printf("value of b is %d\n", b);
    a[5] = 3;
    printf("value of b is %d\n", b);
}
```

Value 3 is written to `a[5]`, which is also the location of `b`
`strcpy` is also prone to buffer overflow. Use `strncpy(s1, s2, n)` instead so that at most `n` chars are copied

Heartbleed
Heartbleed is a protocol for 2 connecting entities to check whether the connection has broken.
A to B: If u r alive, repeat after me this `x`-character string `s`.
B to A: `s`.
OpenSSL library didn't implement it securely and didn't verify that the length of the string `s` is `x`. E.g. if `x = 500` but `s = "POTATO"`, B will output 500 characters starting from the location of `s` in B's memory

Stack smashing
Special case of buffer overflow that targets stack. Called stack overflow, stack overrun, stack smashing.
In call stack, if return address is modified, execution control flow will be changed. Can inject attacker's shell-code into the memory, then run the shell-code.

8.4 Integer Overflow
`a = 254; a += 2` will give `a = 0`
The predicate that `a < a + 1` is not always true.

8.5 Code (script) injection
SQL injection attack
`SELECT * FROM client WHERE name = '$userinput'`
User can set input to `anything' OR 1=1 --`

Prompt injection
If teacher use LLM to mark scripts, students can add prompts in pdf file with text same color as background. LLM is confused between data and instruction.

8.6 Undocumented access point (easter eggs)
Programmers insert undocumented access points to inspect states for debugging purposes. E.g. by pressing certain combi of keys , value of certain variables would be displayed, or for certain input str the program branches to debuggingmode
If these access points mistakenly remain in the final production system, it provides a “back door” for attackers.

8.7 Race Condition (TOCTOU)
TOCTOU (time-of-check-time-of-use) is a race condition in the context of security.

1. Process A checks the permission to access the data, followed by accessing the data
2. Process B (malicious) swaps the data

If B manages to swap the data between time-of-check and time-of-use in A, then the attack succeeds.

8.8 Defense and preventive measures
Input validatation, filtering, parameterized queries (SQL)
Perform input validation whenever input is obtained from user. If not in expected format then reject. Can be white list or black list. For both white and black list, no assurance that all malicious input will be blocked.
Difficult to design a filter that is complete (doesn't miss out any malicious string) and accepts all legitimate inputs

Parameterized queries
- Mechanisms introduced in some SQL servers to protect against SQL injection. Queries sent are divided into queries and parameters.
- SQL parser will know that the parameters are “data” and not “script”.
- SQL parser is designed to never execute any script in the parameters.
- Still cannot stop XSS.

```
sqlQuery = 'SELECT * FROM custTable WHERE
            User=? AND Pass=?'

parameters.add("User", username)
parameters.add("Pass", password)
```

Use “safe” function
Use safe versions of functions that are known to create problems, e.g. `strncpy` instead of `strcpy`
But still can be vulnerable e.g. one uses a combination of `strlen()` and `strncpy()`

Bound checks
Some programming languages perform bound checking during runtime by storing upper and lower bounds during array instantiation so for `a[i] = 5;`, check if i < lower bound or i > upper bound then stop else assign 5 to location

Type safety
Some programming languages carry out type checking to ensure that the arguments an operation gets during execution are always correct. Can be done during runtime (dynamic type check) or when it is being compiled (static type check).

Canaries
- Canaries are secrets inserted at carefully selected memory locations during runtime
- Program halts if values are modified
- Helps to detect overflow esp. stakc overflow bc if attacker wants to write to a particular memory location via overflow, canaries would be modified.
- But value needs to be kept secret else attacker can write secret value to canary while over-running it

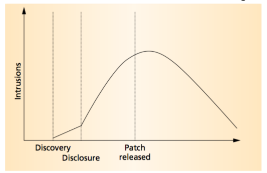
Memory randomization
Attacker is at an advantage when data and codes are always stored in the same locations. Address space layout randomization (ASLR) can help to decrease the attacker's chance of success.

Code inspection
- Manual checking (tedious)
- Automated checking
  - Taint analysis: variables that contain input from the (potentially malicious) users are labeled as source. Critical functions are labeled as sink. Taint analysis checks whether the sink's arguments could potentially be affected (i.e. tainted) by teh source. If so, special check (for e.g. manual inspection) would be carried out. The taint analysis can be static (i.e. checking the code without “tracing it”) or dynamic (i.e. run the code with some input).
  - E.g. **Sources:** user input, **Sink:** opening of system files, function that evaluates a SQL command, etc.

Testing
- Types:
  - White-box testing: tester has access to source code
  - Black-box testing: tester does not have access to source code
  - Grey-box testing: Combination of the above
- Security testing attempts to discover intentional attack, so need to test for inputs that will rarely occur under normal circumstances.
- Fuzzing is a technique that sends malformed inputs to discover vulnerability. More effective than sending in random input. Fuzzing can be automated or semi-automated.

Principle of Least Privilege
The principle of least privilege (PoLP), also known as the principle of minimal privilege (PoMP) or the principle of least authority (PoLA), requires that in a particular abstraction layer of a computing environment, every module (such as a process, a user, or a program, depending on the subject) must be able to access only the information and resources that are necessary for its legitimate purpose.
- E.g. When deploying a software system, do not grant users more access rights than necessary, and avoid enabling unnecessary options.
  - For instance, a webcam application might offer various functions that allow users to control the device remotely. Typically, users can choose which features to enable or disable. As the software developer, you should consider whether all features should be turned on by default when the product is delivered to clients. If every feature is enabled by default, it becomes the client's responsibility to disable those that are unnecessary. However, clients may not fully understand the security implications, which can increase their risk exposure.
- E.g. in Canvas, consider the appropriate level of access to grant a student TA. If the TA's role does not require editing quizzes, they should not be given permission to modify them.

Patching
Life cycle of vulnerability: Vulnerability is discovered → affected code is fixed → revised version is tested → patch is made public → patch is applied
- Patch can be useful to attackers bc attackers can inspect the patch and derive the vulnerability
- Number of successsful attacks goes up after vulnerability / patch is announced as more attackers are aware of the exploit



- Need to apply patch timely
- For critical system, not wise to apply before rigorous testing

9. Access Control
9.1 Access Control model
Access control: controlling operations on objects by subjects

Security perimeter
- Malicious activities outside of the boundary would not affect resources within the perimeter.
- Malicious activities withinn the boundary stays within the boundary.
- Design of the boundary is guided by
  - Principle of least privilege
  - Compartmentalization
  - Defense in depth / swiss cheese model
  - Segregation of duties
- Examples:
  - Calculator app should not have access to contacts so that even if the app is malicious / vulnerable , the confidentiality / integrity of contacts are still preserved (Principle of Least Privilege)
  - School website hosts two services: (1) course's fee payment and (2) exam result. Perimeter between them so that result system remains intact even if got SQL injection attack on (1). (Compartmentalization)
  - A company deploys a firewall separating their server from DMZ. An IDS (intrusion Detection System) reside in the firewall to detect malicious traffic based on known attack signature. In addition, processes in the server are monitored for abnormal behavior. Attack that evade the firewall might be caught by the process monitor, and vice versa. (Defence in depth) (Swiss Cheese Model)
  - A company keeps backup of is business-critical data. The company implements a policy: a single person must not have access to both the production copy and the backup copy. Assigning different components to different person is aka Segregation of Duties. The goal is to eliminate single-point-of-failure. With that, a single rogue system admin (insider) is unable to corrupt all. (Segregation of duties).

Security perimeter on Android
Android apps must request permission to access sensitive user data such as contacts and SMS, as well as certain system features (such as camera and internet).
A central design point of th Android security architecture is that no app, by default, has permission to perform any operations that would adversely impact other apps, the OS, or the user. This includes reading or writing the user's private data (such as contacts or emails), reading or writing another app's files, performing network access, keeping the device awake, and so on.
Each app has a “manifest” file which lists down permissions the app wishes to have. During runtime, OS will grant the request based on default setting or prompt the user.
This is different from a typical multi-user system which has no boundary between two apps run by the same user.

Implications
- A game G and an image editing tool T is implemented for Windows and Android. Alice installed both G and T in a Windows desktop and Android device.
- T can read / write files generated by G in Windows but not Android
- When G is executing T cannot access the memory space of G for both Android and Windows
- T cannot modify the installation of G in Android but it can in early versions of Windows
- In android / ios, information is passed from one app to another only when the user explicitly gives permission by indicating in “share”.

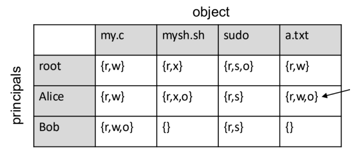
Newer verisions of OS like Windows and Mac have started to impose boundaries between apps

Principal / subject, operation, object
A principal (or subject) wants to access an object with some operation. The reference monitor either grants or denies the access.
E.g. In Canvas, a **Student** wants to **submit** a **forum post**
- Principals: human users
- Subjects: Entities in the system that operate on behalf of the principals

Accesses to objects:
- Observe: e.g. reading a file
- Alter: e.g. writing a file, deleting a file, changing properties
- Action: e.g. executing a program

Definition: ownership
Every object has an owner. Access rights to an object are decided by:
1. Owner of the object decides the rights (discretionary acces control)
2. System-wide policy decides (mandatory access control)

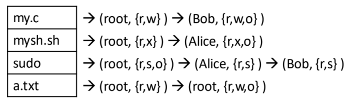
9.2 Access Control Matrix



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

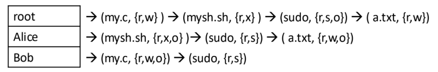
r: read, w: write, x: execute, s: execute as owner, o: owner
Seldom explicitly stored bc the table is very large and difficult to manage

Access Control List (ACL) and Capabilities
Access control matrix can be represented as ACL or capabilities
ACL



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})

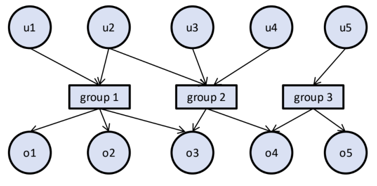
Capability



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})

Most Unix file systems use ACL.

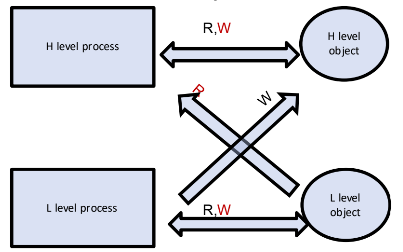
9.3 Intermediate Control
We want an intermediate control that is fine grain (e.g. in Facebook, allow user to specify which friend can view a particular photo) and yet easy to manage.
Not practical for owner to specify each single entry in the access control matrix, so we “group” subjects / objects and define the access rights n the group. This is called intermediate control.



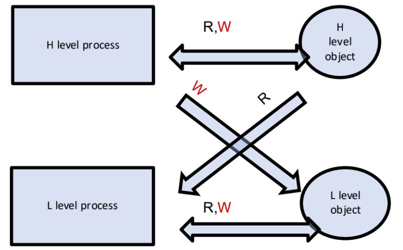
Role-based access control

Protection rings
In OS, “privilege” is often called protection rings. Each object (data) and subject (process) is asssigned a number. Objects with smaller numbers are more important. We call processes with lower ring number as having “higher privilege”. A subject cannot access (both read/write) an object with smaller ring number.
Unix only has 2 rings: superuser and user

Bell-LaPadula vs Biba
Bell-LaPadula
- No read up: Prevents lower level from getting info in higher level
- No write down: Prevents malicious insider from passing information to lower levels
- Confidentiality
- No “sensitive” information leaking down



Biba
- No write up: Prevents a malicious subject from poisonng upper level data
- No read down: Prevents a subject from reading data poisoned by lower level subjects
- Integrity
- No “malicious” information going up



9.4 Unix file system
Unix file system access control
Objects in Unix include files, directories, memory devices, and I/O devices

```
ls -al
-rw-r--r-- 1 alice staff 124 Mar 9 22:29 my.c
```

- First - indicates whether it is a file or directory
- Remaining file permissions are grouped into 3 triples that define **read, write, execute** access for **owner, group, other**
- `1`: links count (not relevant)
- `alice`: owner
- `staff`: group
- `124`: file size
- `Mar 9 22:29`: date & time of last modification
- `my.c`: filename

Principals, subjects
- Principals are user-identities (UIDs) and group-identities (GIDs)
- Information of user accounts are stored in the “password” file `/etc/passwd`
- Subjects are processes and each process has a process ID (PID)

Now the password file does not actually store the password, last time it did and everyone could see the hashed passwords of others, thus it was vulnerable to offline dictionary attack. Now it is replaced with * and actual hashed passwords are stored somewhere else.

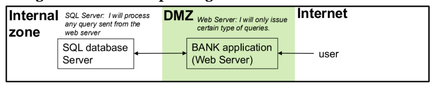
Superuser (root)
UID 0, username root, all security checks are turned off for root

Checking rules for file access
- The objects are files. Each file is associated with a 9-bit permission.
- Each file is owned by a user and a group
- Whena user wants to access a file (object), the following are checked in the 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 decide the access rights.
  3. If the user is not the owner, nor member of the group that own the file, then the permission bits for other decide.

Owner of a file, or superuser can change permission bits

9.5 Controlled invocation & privilege elevation
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 or cannot read a file would get stuck!
Solution: **controlled invocation**.
- The system provides a predefined set of applications that have access to F.
- These applications are 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 wrote the application bears the responsibility to make sure that the application only performs the intended limited operation.

Bridges with elevated privilege



Internal zone
SQL Server: I will process any query sent from the web server
SQL database Server
DMZ
Web Server: I will only issue certain type of queries.
BANK application (Web Server)
Internet
user

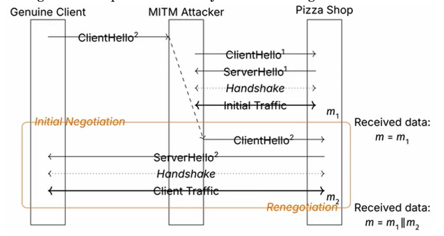
If the bridge is not implemented correctly and contains exploitable vulnerabilities, an attacker can trick the bridge to perform “illegal” operations not expected by the programmer / designer. This would have serious implication, since the process is now running with “elevated privilege”.
Attacks of such form is also known as “privilege escalation”.

9.6 Controlled invocation in UNIX
- Process got PID, real UID, and effective UID.
- Real UID is inherited from user who invokes the process.
- If Set UID is disabled (permission will be `'x'`), process' effective UID = real UID
- Else (SUID enabled, permission will be `'s'`), effective UID is inherited from file's owner

E.g. got file containing personal information of users with SUID disabled, users cannot change their own data, so have another program to edit the file with SUID enabled and owner = root, then anyone can run the program to edit their own data.

Tutorials
What happens when Alice accesses a website with expired cert?
- Previous owner
  - Website may no longer belong to original owner
  - Previous owner had valid but outdated certificate and can use this against Alice
  - Previous owner can impersonate the current website
- Compromised key
  - Private key was stolen after expiry
  - Certificate was already expired, so website did not revoke certificate
- Legacy issue
  - SHA1 is broken
  - Some expired certificates are signed by CA using SHA1
  - Signature may be forged
  - Public key may also be short, and thus broken

Renegotiation attack
Renegotiation: Update session key without closing TLS session



In initial traffic, attacker sends:
`GET /pizza?toppings=pepperoni;address=attackersaddress HTTP/1.1 X-Ignore-This:`
In client traffic, genuine client sends:
`GET /pizza?toppings=sausage;address=victimssaddress HTTP/1.1 Cookie: victimscookie`
As such, server receives the following request:
`GET /pizza?toppings=pepperoni;address=attackersaddress HTTP/1.1 X-Ignore-This: GET /pizza?toppings=sausage;address=victimssaddress HTTP/1.1 Cookie: victimscookie`
Client starts an initial negotiation, attacker holds it, starts TLS session with server, sends some prefix data to server, then continues negotiation (which from the server's pov is the renegotiation)
Result: the attacker is able to inject some data into the server buffer before the client traffic comes in.