CS2107 | Yap Zher Xiang Jason

Tutorial 1

1. Suppose it takes 512 clock cycles to test whether a 64-bit cryptographic key is correct on input a 64-bit plaintext and the corresponding ciphertext. How long would it take to exhaustively check all the keys using a 4GHz (single-core) processor. How how long would it also take on a cluster of 1024 servers, each with a quad-core 4GHz processor.

   1 clock cycle → used to sync operations in a processor.

$$512\ clock\ cycles \rightarrow 2^9 cycles$$
$$64-bit key \rightarrow 2^{64}\ possible\ combinations$$
$$4GHz = 2^{32} cycles\ per\ second$$
$$Total\ time = \frac{2^{64} \times 2^9}{2^{32}} = 2^{41}\ seconds$$
$$On\ Cluster = 1024\ servers \times 4\ cores = 4096$$
$$Total\ time\ on\ servers = \frac{2^{41}}{4096} = 2^{29}$$

2. For any question if it fulfills the following criteria we can exhaustively search with a precomputation.

   a. know the algorithm of how to obtain the key

   b. exhaustively precompute all possible key

   c. use key to encrypt/decrypt → there must be a pattern to match the plaintext or ciphertext

      → confirms this key is the one that was initially used to decrypt/encrypt.

   e.g. Bob encrypted a music mp3 file using Winzip, which employs the 256-bit key AES. He chose a 6-digit number as password. Winzip generated the 256-bit AES key from the 6-digit password using a (deterministic) function, say SHA13. Alice obtained the ciphertext. Alice also knew that Bob used a 6-digit password and knew how Winzip generated the AES key.

   a. Alice knows the algorithm

   b. Alice knows that there are **256 possible 256-bit AES keys** OR 10^6 possible 6-digit password keys. She can exhaustively precompute the 10^6 possible

passcode keys are run them through the deterministic SHA13 function to obtain a possible 256-bit AES key.

c. Alice has a mechanism to know if the key is correct: if the `mp3` file is playable. Thus for whichever key that decrypts a *playable plaintext*, this is the correct key.

3. Encryption after compression is okay. Why is it meaningless to carry out the operations in the reverse order → encrypting before compression.

   Compressing an encrypted file will yield very little or no compression gain. This is since the encrypted file will resemble a “random” sequence (due to a requirement of a good encryption scheme). A compression algorithm, which takes advantage of repeating patterns, therefore will not work well on an encrypted file.

Tutorial 2

Side Channel Attack

1. A side-channel attack is any attack based on extra information that can be gathered because of the fundamental way a computer protocol or algorithm is implemented, rather than flaws in the design of the protocol or algorithm itself (e.g. flaws found in a cryptanalysis of a cryptographic algorithm) or minor, but potentially devastating, mistakes or oversights in the implementation.

Alice sends instruction to Bob daily using some mobile devices. Each instruction is represented as ASCII string and of this format `action:date`The date is the 8-byte “dd/mm/yy” format, and actions are `buy`, `sell`, `sell everything`, and `hold and see`. Example `buy:02/01/22`, `sell everything:03/01/22`. The message will be padded (with zeros) and encrypted using AES CBC mode and send to Bob. The mobile device, after each re-start, will set the IV to be the string of all zeros, and then increases it by one for every new encryption. The mobile device will re-start after software update. Typically, there would be around 2 or 3 software update per month. An attacker somehow is able to sniff the communication between Alice and Bob and obtained the above ciphertext. What type of information would be leaked?

Side-Channel Attack: Block size, attacker can know if it is a `buy` or `sell` or `buy_and_hold` or `sell_everything` based on the block size. An AES block has 128-bits or $\frac{128}{8} = 16$ bytes.

"buy" message is 3 + 1 + 8 = 12 bytes (1 block)

"sell" message is 4 + 1 + 8 = 13 bytes (1 block)

"sell_everything" message is 15 + 1 + 8 = 24 bytes (2 blocks)

"hold_and_see" message is 12 + 1 + 8 = 21 bytes (2 blocks)

Due to the way IV is generated, the attacker can obtain a few ciphertexts with the same IV. Consider getting a few ciphertexts with the same IV and have 2 blocks. The attacker can compare the first block. All "sell_everything" messages should have the same first block.

Padding Oracle Attack

2. Lecture note assumes that the attacker knows how many bytes are being padded. Now, we consider cases where such knowledge is not available. Let us consider an attacker who has the one-block IV, and a 2-block ciphertext. The attacker does not know how many bytes are being padded. Give a method to determine the number of padded bytes. Use as little calls to the oracle as possible.

b1	b2	b3	b4	b5	b6	b7	b8	b9	00	00	FF	04	04	04	04

Based on the known plaintext (b10 to b16), the plaintext is padded with 4 bytes.

To find the value of b9, we need to try to pad the plaintext with 8 bytes (b9 to b16).

Goal: send the ciphertext that will be decrypted to the following plaintext:

b1	b2	b3	b4	b5	b6	b7	b8	08	08	08	08	08	08	08	08

Idea: change the IV to IV$\oplus$ (0, 0, 0, 0, 0, 0, 0, 0, t, 08, 08, F7, 0C, 0C, 0C, 0C).

Vary t until we get the correct padding.

Finally, get b9 by taking t $\oplus$ 08

b10 $\oplus$ 08
b11 $\oplus$ 08
b12 $\oplus$ 08
…

- Use the padding oracle to determine if the padding has been broken.
- `y1` $\oplus$ `p2` , thus changing the i^{th} bit of `y1` will subsequently affect the i^{th} bit `p2`

-

```
for i in range(1, 17):
  y1[i] = random_byte # y2 would change from this y1 bit change because of AES
  c = IV + y1 + y2
  if (!padding_oracle(c)): # this means that padding has been broken
    break
```

Since padding is modified we now know the padding of the plaintext, and we can use the same padding oracle attack to find the i^{th} block of the plaintext.

Tutorial 3

Entropy

Suppose we want to have 50 bits of randomness using a 1000-word dictionary. How many words are required in the password to have 50 bit randomness.

- Assume *Kerckhoff's Principle*, the attacker knows the algorithm, this means we have to look at the source of the words (i.e. from the 1000-word dictionary).
- Every word in the 1000-word dictionary has a equal probability of being picked → entropy of each word is $log_2(N) = log_2(1000) \approx 10$ bits of randomness per word
- $\therefore number\ of\ words = \frac{50}{10} = 5\ words$

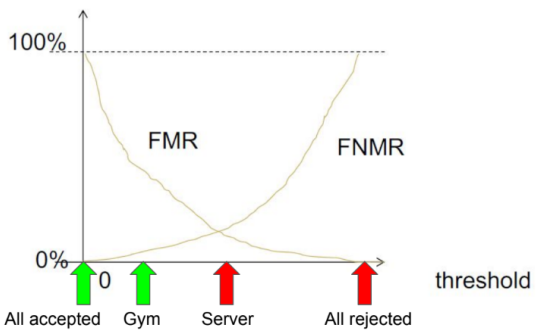
FNMR / FMR

If threshold is 0, everything is accepted, if threshold is 1, everything is rejected.

FMR = number of successful false matches (B) / number of attempted false matches (B+D)

FNMR = number of rejected genuine matches (C) / number of attempted genuine matches (A+C)

	accept	reject
genuine attempt	A	C
false attempt	B	D



Tutorial 4

Cryptographic hashing

`H()` is a hash function, when given a binary string s can generate a pseudorandom sequence. Bob sets s to be 160 zeroes, and says that since `H()` produces a random sequence the sequence produced is random and can therefore be used as a key and IV.

- This is false as the attacker (assuming *Kerckhoffs' principle*) is able to replicate the algorithm to generate the key and IV.

Now we want to use `rand()` in C OR `java.util.Random` why is it still not secure to use these random values to select the s to be put into `H()`.

- `srand()` seeds the pseudorandom number generator for `rand()`. If adversary knows the time, they can determine the seed, and derive the key. If adversary knows approximate time, they can exhaustively search Even if adversary don't know the time, they can still perform exhaustive
  search. int data type in C is either 16-bits or 32-bits.

MAC - HMAC Implementation

Suppose the following verification protocol:

i. $(P \rightarrow V : u)$ — Prover gets userid u and password p from user, send u to verifier

ii. $(P \leftarrow V : c)$ — Verifier looksup password p from password file, and derive 128-bit key from p by $k = SHA3(p)$. Verifier randomly generates a 128-bit string r and sends the ciphertext $c = ENC(k, r)$. $ENC()$ is some secure encryption scheme (e.g. AES in CBC).

iii. $(P \rightarrow V : h)$ — Prover uses p to get k, from c, prover can also get r. They then compute a MAC $h = mac(k, r)$ and send to verifier. $mac()$ is a secure MAC scheme like HMAC.

iv. $(V\ accept/reject)$ — verifier calculates its own mac with its own k and r, checking that the h it receives corresponds to its own mac calculated.

Suppose there is an implementation flaw and r is also fixed with the value 0…0. What is the implication?

Tutorial 5

Birthday Attack: selection from 2 pools

There are 2^7 students in class, and each are assigned a unique 16-bits id known by the student and the lecturer only. You write down 32 different ids. What is the probability that at least one student id is written? How many pieces of paper do we need to submit such that the probability is > 0.5.

> 💡 Let S be a set of k distinct elements where each element is an n bits binary string. Now, let us independently and randomly select a set T of m n-bit binary strings. It can be shown that, the probability that S has non-empty intersection with T is more than
>
> $$1 - 2.7^{-km2^{-n}}$$

Let n = 16, k = 2^7 and m = 2^5 . Using the formula, the probability is at least 0.060190

Let n = 16, k = 2^7, we want to find m such that $1 - 2.7^{-km2^{-n}}$ = 0.5 Solving for m gives us m = 356

Tutorial 6

Certificate Expiry

Suppose a site has a **valid signature**, but expired some time ago. If Alice chooses to accept the certificate, what is the potential risk.

1. Domain name `www.domain.com` might have changed owners, previous owner has a valid but outdated certificate to still use on Alice
2. Suppose website has some disputes with ex-employee, who knows the private key. Since certificate already expired, website does not need to revoke the certificate. However because ex-employee has the private key, they can use it against Alice ⇒ Insider Attack

PKI role IN TLS

Suppose, a communication channel between Alice and Bob is intercepted by a Mallory, with Mallory being the Man-In-The-Middle.

a. Alice carries out TLS handshake, unilateral authenticated key-exchange with Bob. At this point, Mallory *impersonates* Bob instead and carries out the

handshake with Alice.

i. *M* generates certificate with content of name: Bob, public key: M's k_e, signature: signed using *M*'s private key, k_d and sends this to Alice during TLS handshake

ii. Alice accept the certificate generated by *M* because **Alice accepts *M* as root CA.** Authenticated key-exchange will be successfully carried out as *M* knows private key k_d of its own public key k_e.

b. After the successful authenticated key-exchange, *M* and Alice establish the session key, which is the pair (k_a, t_a). Subsequent communication *M* and Alice will be encrypted using k_a and authenticated using t_a .

c. *M* then performs another unilateral authentication with Bob, whereby *M* uses Bob's public key to verify Bob's authenticity. After the successful authentication, M and Bob establish another session key pair k_b, t_b. Subsequent communication between *M* and Bob will be encrypted using k_b and authenticated using t_b.

d. Now, whenever Alice wants to send a message m to Bob, it is to be encrypted using k_a . Since *M* is the man-in-the-middle, *M* can intercept the encrypted m. *M* decrypts it using k_a , inspects it, and then re-encrypts it using k_b, and finally forwards to Bob. Similar steps are carried out for the MAC.

e. Likewise, messages from Bob to Alice are processed in a similar way. Bob's message is to be encrypted using k_b and sent to *M*. *M* decrypts it using k_b, inspects it, and then re-encrypts it using k_a, and finally forwards to Alice. Similar steps are carried out for the MAC.

Remarks:

- This questions illustrates that **EVEN WITH SECURE COMMUNICATION CHANNEL**, a Mallory in the public channel is unable to compromise confidentiality/integrity.
- However, when Mallory is a CA accepted by Alice, no matter the secure communication channel, Mallory can compromise confidentiality/integerity.

Tutorial 7

Forward Secrecy

In cryptography, *forward secrecy*, also known as *perfect forward secrecy*, is a feature of specific key agreement protocols that gives assurances that session keys will not be compromised even if long-term secrets used in the session key exchange are compromised.

1. RSA based key-exchange
    i. Bob send Alice certificate
    ii. Alice verify certificate, extracts Bob's public key
    iii. Alice *randomly picks* session key k, encrypting k using Bob's public key and send to Bob. Alice also send *randomly chosen* r to Bob
    iv. Bob decrypts to obtain k, Bob computes MAC of r using k and sends to Alice.
    v. Alice verifies if MAC is correct, if so carry on. Else halt.

- This does not work because if an attacker Eve manages to get the master key: i.e. Bob's private key, then Eve can decrypt cipher text in step iii, and thus obtain the session key. Subsequently, further messages encrypted using k can also be obtained ⇒ NO FORWARD SECRECY

2. Station-to-Station protocol — Why does this work?

    Even if Eve knows the master key (Bob's private key), she is not able to find the session key that are agreed by both Alice and Bob because she does not know either the values of a or b (the exponents chosen by Alice and Bob).

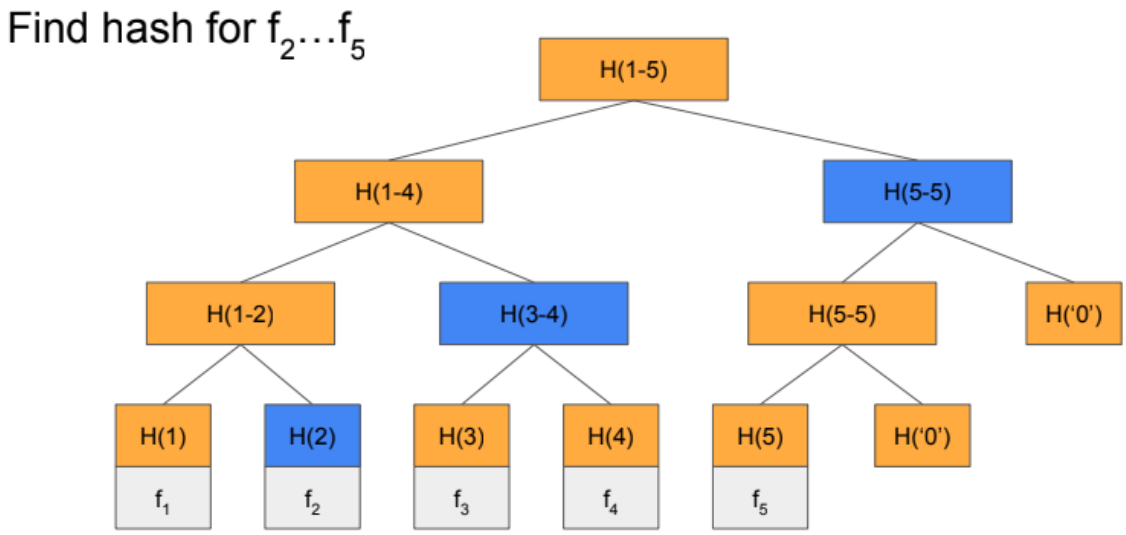
    i. If we show that a Diffe Helman problem can be reduced to a STS problem, we can show it is hard to solve STS, hence hard for eavesdropper to obtain master key ⇒ establishing forward secrecy.
    ii. Given x, y as inputs, generate random RSA key (k_e, k_d) where k_e is public and k_d are private keys respectively.
    iii. Sign y using k_d to get s
    iv. We can solve STS with the following (x, y, s, k_d) to get output k, thus this means the DH problem can be reduced to STS.

Merkle Trees

Suppose I need to chain hashes, and querying i, j needs to give me the output: $< f_i, f_i + 1, ..., f_k >$ and its digest. Given $f_1, f_2, ..., f_n$. Consider a tree with all the

$SHA3(f_i)$'s as leaf. For each intermediate node, its value is $SHA3(h_{left}||h_{right})$ where h_{left}, $h_{right}$are its children. If the intermediate node has only one child, the right child is taken as a special string (say, “0”) which indicate that it is empty.

- The value of all the roots of subtrees containing f_i to f_j in the leaf nodes.
- Start from merkle root
  - If all the values covered in the root are in the range $[i, j]$
    - Return the root
  - Else recurse on left and right subtrees if their values overlap with [i,j]



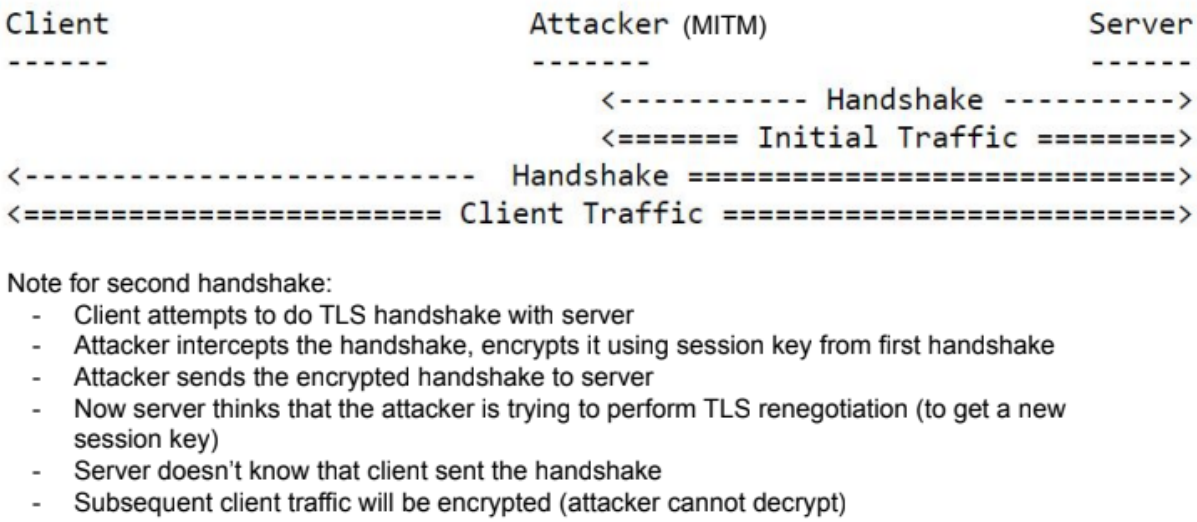
Tutorial 8

VPN

- Using IPSEC ⇒ sits in layer 3 if VPN uses IPSEC. VPN can just prepend the IPSEC-Header to the payload, thus it operates just below the IP layer.
- Using TLS ⇒ user first establishes TLS with VPN server, and send the instruction
  - forward this packet to IP-address X, port Y for me, pretending the packet is from your IP address + port number.

    - VPN obtains data from TLS and then construct datagram (src port decided by VPN) and IP packet (VPN's IP address).
    - For this implementation, it sits in layer 4 (transport) as it manages the packets routing.
- Remember that the VPN is unable to determine the user's MAC address, because for MAC addresses, they are modified on each hop, thus even if the Link Layer is not encrypted, there is no way for the VPN to know where the frame came from.

Renegotiation Attack



- Server's POV — it performed 2 handshakes. Attacker is unable to know the second key because the second handshake is directly performed between Client and Server.
- Second handshake carried out between (1) client and attacker & (2) attacker and server
    1. As seen above, client and attacker's handshake is not encrypted
    2. Attacker and server's handshake is however encrypted by the initial session keys
    3. Client Traffic is however, protected by the key from the second handshake (between client and server)

Tutorial 9

In a shell, issue command `more a.txt` . In another shell execute `ps` command.

a. Real UID and effective UID of a `more a.txt` → Both are **Alice** (person who started running the command)

b. Read UID and effective UID of `ps` → Real UID is Alice, Effective UID is `root`

```
Hey>ls -l /bin/ps
-rwsr-xr-x 1 root wheel 203504 7 Feb 05:22 /bin/ps
All can read and execute. SetUID is on. Owner is root.
```

1. **Real UID**: For a process, Real UID is simply the UID of the *user that has started* it. It defines which files that this process has access to.
2. **Effective UID:** It is normally the same as Real UID, but sometimes it is changed to enable a non-privileged user to access files that can only be accessed by a privileged user like root. e.g. `rwsr-xr-x 1 root root 59640 Mar 23 2019 /usr/bin/passwd` → So if a non-root user runs this file, the EUID of the process will be “0” i.e. root and UID remains the same as of original user.
3.

Cryptography Objects:

Block Cipher

DES / AES , designed for fixed size input/output. Larger plaintexts divided into blocks, block cipher applied. A block cipher operates on a fixed-sized block of input and can provide high diffusion and confusion properties.

> 💡 Diffusion means that **if we change a single bit of the plaintext, then about half of the bits in the ciphertext should change**, and similarly, if we change one bit of the ciphertext, then about half of the plaintext bits should change.

Stream Cipher

Inspired by one-time-pad, if plaintext is 2^{20} bits and secret key is 256 bits, stream cipher generates a 2^{20} bit sequence from the key and treats the generated sequence as secret key in one-time-pad and encrypts the plaintext. A stream cipher operates on a binary bit or byte in a data stream, one at a time. It combines plaintext

bits or bytes with a pseudorandom sequence. Its operation is fast, yet has a low diffusion property.

Initial Value

Arbitrary value chosen during encryption. So, it is different in different encryptions of the same plaintext. IV can be randomly chosen, obtain from a counter, or from other info.

Pseudo random sequence

As a security requirement: without knowing the secret key, it is computationally difficult to distinguish the sequence from a truly random sequence. This implies that it is computationally difficult to get the short secret key from the sequence. It also implies that it is computationally difficult to predict part of the sequence after seen another part of the sequence.)

One-time-pad

Given n-bit plaintext `x1 x2 x3 ... xn` and n-bit key: `k1 k2 k3 ... kn`, ciphertext = `x1 XOR k1` `x2 XOR k2` … `xn XOR kn`. Same for decryption.

- Note that even exhaustive search can't work on one-time-pad.
- It can be shown that one-time-pad leaks no information of the plaintext, even if the attacker has arbitrary running time. Hence, it is sometime called "unbreakable". (perfect secrecy)

Symmetric Key

Both parties hold the same symmetric key, both entities interact to establish the key. An entity needs to securely establish a different symmetric key with each of the rest (this means that is has a quadratic number of keys to establish).

Public Key

Usually comes in a pair with the private key. An entity broadcast its public key to some public billboard only once. An entity doesn't need to know the existence of the receiver while broadcasting its public key.

Private Key

Usually comes in a pair with the private key. The private key is only known to the owner, and not to anyone else.

Signature

Public key version of MAC. Here owner uses the ***private key*** to generate the signature. The public can use the ***public key*** to verify the signature. — Anyone can verify authenticity but only the person who knows the private key can generate the signature.

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

Certificate

Useful in distributing public keys. It is a piece of document that binds a `name` to a `public key` &
certified by an authority, called CA (signed by the CA). A Certificate contains:

- name, public key, expiry date, usages,
- meta info (type of crypto, name of CA, etc),
- CA's signature

Self-signed Certificate

A self-signed certificate is signed by its owner (i.e the "name" in the certificate). It is to be verified using the "public key" listed in the certificate. Self-fulfilling! — A self-signed certificate by a CA is also called "root-certificate"

Certification Authority

CA is a trusted authority that manages a directory of public keys. An entity can request adding its public key to the public directory. Anyone can send queries to search the directory. The CA also has its own public-private key. We first assume that the CA's public key has been securely distributed to all entities involved. (still need secure channel to distribute the CA's public key).

- Most OSes and browsers have a few pre-loaded CAs' public keys: they are known as the "root" CAs. Not all CAs' public keys are preloaded. Other CA's public keys can be added through the chain-of-trust to be described later.

Chain-of-trust

There are many CAs, most browsers come preloaded with some.

Chain-of-trust is a mechanism used in certification authorities to verify the authenticity of a public key. It involves a hierarchy of trusted entities, where each

entity in the chain verifies and vouches for the authenticity of the entity immediately below it. The root of the chain is usually a trusted entity, such as a root CA, whose public key is preloaded in most operating systems and browsers. When a public key is presented, the chain is traced back to the root CA to verify the authenticity of the public key.

Certification path

A certification path is a sequence of certificates, starting from a trusted root CA (usually pre-loaded in operating systems and browsers) and ending with the certificate of the entity whose public key is being verified. The chain-of-trust is traced back to the root CA to verify the authenticity of the public key.

Hash

A (cryptographic) hash is a function that takes an arbitrary large message as input, and outputs a fixed size (say 160 bits) digest.

- Security requirement: It is difficult for an attacker to find two different messages m_1, m_2 that
  "hash" to the same digest. That is,

$$h(m_1) = h(m_2)$$

- This is known as collision-resistant.
  - A hash that is collision-resistant is also one-way, that is, given a digest d, it is difficult to find a message m s.t. h (m)=d.

Message Authentication Code (MAC)

Also known as a keyed-hash. 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). A MAC is a symmetric-key based technique that can provide data-origin authenticity but not non-repudiation.

- Security requirement: (***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.

Authenticated Encryption

Authenticated Encryption (AE) are forms of encryption which simultaneously assure the confidentiality and authenticity of data by outputting both ciphertext and

authentication tag during its encryption process. This ensures that the data has not been tampered with during transmission, while also providing confidentiality. AES GCM (Galois/Counter Mode) is a widely used implementation of AE. It combines the use of a block cipher (AES) with a stream cipher (GCM) to provide both encryption and authentication. AES GCM uses a 128-bit block size and a 128-bit authentication tag. It is known for its high speed and strong security, making it a popular choice for secure communication protocols such as TLS.

Nonce

The challenge m ensures freshness of the authentication process. It is also known as the cryptographic nonce (or simply nonce).

A nonce is an arbitrary number used only once in a cryptographic communication, in the spirit of a nonce word. They are often random or pseudo-random numbers. — helps to prevent replay attacks

Mode-of-operation

The method of extending encryption from a single block to multiple blocks is not straightforward. It is called mode-of-operation.

- To encrypt a plaintext longer than its blocksize, a block cipher needs to employ a good mode-of-operation such CBC and not a weak one like ECB.

Cryptography Notions:

Symmetric key Cryptography

When a secret key is leveraged for both encryption and decryption functions

Public Key Cryptography

Involves a pair of keys known as a public key and a private key (a public key pair), which are associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data.

RSA Scheme

Relies on the fact that integer factorisation is computationally much more difficult than its (inverse) multiplication operation

Public Key Infrastructure

Standardized system to distribute public keys

Kerckhoff’s principle

The concept that a Cryptographic system should be designed to be secure, even if all its details, except for the key, are publicly known

Kerckhoffs's principle says that "no inconvenience should occur if the cryptosystem falls into the hands of the enemy".

One Way

A property that ensures that a function is hard to invert. In other words, given an element in the range of a hash function, it should be computationally infeasible to find an input that maps to that element.

Attacks

Distributed denial of service (DDoS) attack

A distributed denial-of-service (DDoS) attack is a malicious attempt to disrupt the normal traffic of a targeted server, service or network by overwhelming the target or its surrounding infrastructure with a flood of Internet traffic.

DDoS attacks achieve effectiveness by utilizing multiple compromised computer systems as sources of attack traffic. Exploited machines can include computers and other networked resources such as IoT devices.

Denial of Service (DoS) attack

A denial-of-service (DoS) attack is a type of cyber attack in which a malicious actor aims to render a computer or other device unavailable to its intended users by interrupting the device's normal functioning. DoS attacks typically function by overwhelming or flooding a targeted machine with requests until normal traffic is unable to be processed, resulting in denial-of-service to addition users. A DoS attack is characterized by using a single computer to launch the attack.

A distributed denial-of-service (DDoS) attack is a type of DoS attack that comes from many distributed sources, such as a botnet DDoS attack.

Man-in-the-middle attack

An attack in which an attacker is positioned between two communicating parties in order to intercept and/or alter data traveling between them. In the context of authentication, the attacker would be positioned between claimant and verifier, between registrant and CSP during enrollment, or between subscriber and CSP during authenticator binding

Chosen-plaintext attack

The adversary has access to a blackbox (i.e. the oracle). He can choose and feed any plaintext m
to the blackbox and obtain the corresponding ciphertext c (all encrypt with the same key). He
can access the black box for a reasonable large number of times. We call this black-box an
encryption oracle.

Known-plaintext attack

The adversary is given a collection of plaintext m and their corresponding ciphertext c.

Frequency analysis attack

In cryptanalysis, **frequency analysis** (also known as **counting letters**) is the study of the frequency of letters or groups of letters in a ciphertext. The method is used as an aid to breaking classical ciphers.

Brute-force attack

A brute force attack is a hacking method that uses trial and error to crack passwords, login credentials, and encryption keys. It is a simple yet reliable tactic for gaining unauthorized access to individual accounts and organizations' systems and networks. The hacker tries multiple usernames and passwords, often using a computer to test a wide range of combinations, until they find the correct login information.

Side-channel attack

An attack enabled by leakage of information from a physical cryptosystem. Characteristics that could be exploited in a side-channel attack include timing, power consumption, and electromagnetic and acoustic emissions.

One type of side-channel (attack) is timing attack, which measures how much time various computations (e.g. comparing an attacker's given password with the victim's unknown one) take to perform, without knowing the performed computations.

Phishing attack

The victim is tricked to voluntarily sends the password to the attacker.

Phishing attacks ask for password under some false pretense. Typically, it tricks the user to visit a website, which is a spoofed login web.

Phishing attack is a social engineering attack.
"Social engineering, in the context of information security, refers to psychological manipulation of people into performing actions or divulging confidential information."

Birthday attack

This attack is similar to "exhaustive search" in encryption. Birthday attack can be applied to all hash functions, similar to exhaustive search on all encryption schemes.

Birthday attack can be applied to all hash functions, similar to exhaustive search on all encryption schemes

We want to design a hash so that known attacks can't do better than birthday attack

Replay attack

Information sniffed from the communicated channel can be replayed to impersonate the user.

A challenge–response authentication technique improves a basic username-and-password authentication scheme because, by using a challenge value that is time-varying, the former offers protection against a/an replay attack

Browser exploit against SSL/TLS attack (BEAST) Attack

This attack was revealed at the Ekoparty Security Conference in 2011. BEAST is based on a type of cryptographic attack called the "chosen plain text attack." Before I jump into explaining the details of this attack, let us take a look at some of the basic concepts to be understood.

- **How is the attack accomplished?**

> It was noticed that TLS 1.0, when dealing with multiple packets, allows the following packets to use an IV that is the

> last cipher text block of the previous packet. In other words, an attacker who can see the encrypted traffic can note the IV used for session cookie (Why? Because the cookie's location is predictable). Simply put, an active attacker will be able to gather the IVs for each record just by sniffing the network. So if the attacker can "guess" a plaintext message, he can make a guess at the session cookie and see if the cipher text matches. [Note that, since this is a MITM attack, the attacker can mix his traffic with the victim traffic to see the results].

Privilege Escalation

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, since the process is now running with "elevated privilege". ⇒ attacks of such form are known as ***privilege escalation***.

Access Control

Mandatory Access Control

A system-wide policy that decides (strict rules that everyone must follow)

Discretionary Access Control

Owner of the object decides the rights

Intermediate Access 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**.

(IAC) Role-based Access Control

The grouping can be determined by the "role" of the subject. A role associates with a collection of procedures. In order to carry out these procedures, access rights to certain objects are required. ⇒ follow principle of ***least privilege***.

(IAC) Protection Rings

We call processes with lower ring number as having “higher privilege”. A subject cannot access (both read/write) an object with smaller ring number. Unix has only 2 rings, superuser and user.

(IAC) Bell-LaPadula Model (for data confidentiality)

- **no read up**: A subject does not have read access to object in higher level. This prevent a lower level from getting info in the higher level.
- **no write down:** A subject does not have append-right to object in lower level. This prevents a malicious insider from passing information to lower levels. (e.g. a clerk working in the highly classified department is forbidden to gossip with other staff).

(IAC) Biba Model (for process integrity)

- **no write up:** A subject does not has “write” access to objects in higher level. This prevent a malicious subject from poisoning upper level data, and thus ensure that a process will not get compromised by lower level subjects.
- **no read down:** A subject does not has read access to objects lower level. This prevents a subject from reading data poisoned by lower level subjects.

Miscellaneous:

Covert channel

An unintended or unauthorized intra-system channel that enables two cooperating entities to transfer information in a way that violates the system's security policy but does not exceed the entities' access authorizations.

2FA

Require at least two different authentication “factors.”
Example of factors:

1. Something you know: Password, Pin.
2. Something you have: Security token, smart card, mobile phone, ATM card.
3. Who you are: Biometric.

It is called a 2-factor authentication if 2 factors are employed

Bring-your-own-device

Bring your own device (BYOD) means that **employees use personal devices to connect to an organization's network, accessing work-related systems and possibly, sensitive data.**

Botnet

A botnet is a group of Internet-connected devices, each of which runs one or more bots. Botnets can be used to perform Distributed Denial-of-Service attacks, steal data, send spam, and allow the attacker to access the device and its connection. The owner can control the botnet using command and control software.

Worm

A computer worm is **a subset of the Trojan horse malware that can propagate or self-replicate from one computer to another without human activation after breaching a system**. Typically, a worm spreads across a network through your Internet or LAN (Local Area Network) connection.

Virus

A computer program that can copy itself and infect a computer without permission or knowledge of the user. A virus might corrupt or delete data on a computer, use e-mail programs to spread itself to other computers, or even erase everything on a hard disk. See malicious code.

The primary difference between a virus and a worm is that viruses must be triggered by the activation of their host; whereas worms are stand-alone malicious programs that can self-replicate and propagate independently as soon as they have breached the system.

Key-logger

Keyloggers, or keystroke loggers, are **tools that record what a person types on a device**.

Zero-Day Vulnerabilities

Vulnerabilities that are discovered but not yet published, if an attack deploy attacks on zero-day
vulnerabilities, the victims will have “zero-day” to react.

Common Vulnerabilities and Exposures (CVE)

Repository containing discovered vulnerabilities, it is public with a list of entries with identification numbers, description, and public references.

Common Weakness Enumeration (CWE)

Repository on vulnerability forms and concepts and not actual instances

Key Escrow

The system responsible for storing and providing a mechanism for obtaining copies of private keys associated with encryption certificates, which are necessary for the recovery of encrypted data.

Typo squatting

1. An attacker registered for a domain name: luminus.nus.edv.sg and obtained a valid certificate of the above name. No one has registered edv.sg and thus the attacker is about to get it.
2. The attacker employed “phishing attack”, tricking the victim to click on the above link, which was a spoofed site of luminus.nus.edv.sg
3. The address bar of the victim’s browser correctly displayed https://luminus.nus.edv.sg but the victim didn’t notice that and logged in using the victim’s password.

Fuzzing

Fuzz testing or fuzzing is an automated software testing method that injects invalid, malformed, or unexpected inputs into a system to reveal software defects and vulnerabilities. A fuzzing tool injects these inputs into the system and then monitors for exceptions such as crashes or information leakage.

People in cryptography

Alice and Bob

The original, generic characters. Generally, Alice and Bob want to exchange a message or cryptographic key.

Criag

A password cracker, often encountered in situations with stored passwords.

Eve

An eavesdropper, who is usually a passive attacker. While they can listen in on messages between Alice and Bob, they cannot modify them.

Faythe

A trusted advisor, courier or intermediary. Faythe is used infrequently, and is associated with faith and faithfulness. Faythe may be a repository of key service or courier of shared secrets.

Mallory

A malicious attacker. Associated with Trudy, an intruder. Unlike the passive Eve, Mallory is an active attacker (often used in man-in-the-middle attacks), who can modify messages, substitute messages, or replay old messages. The difficulty of securing a system against a Mallory is much greater than against an Eve.

Trudy

An intruder.

Organisations & Hacks

NIST

National Institute of Standards and Technology is an agency of the United States Department of Commerce whose mission is to promote American innovation and industrial competitiveness.

OWASP

The Open Worldwide Application Security Project is an online community that produces freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security.

HangZhou XiongMai Technology

Hangzhou Xiongmai Technology, a vendor behind DVRs and internet-connected cameras, said on Sunday that security vulnerabilities involving weak default passwords in its products were partly to blame.

According to security researchers, malware known as Mirai has been taking advantage of these vulnerabilities by infecting the devices and using them to launch huge distributed denial-of service attacks, including Friday’s outage.

“Mirai is a huge disaster for the Internet of things,” Xiongmai said in an email to IDG News Service. “We have to admit that our products also suffered from hacker's break-in and illegal use.”

Mirai works by enslaving IoT devices to form a massive connected network. The devices are then used to deluge websites with requests, overloading the sites and effectively taking them offline.

Because these devices have weak default passwords and are easy to infect, Mirai has been found spreading to at least 500,000 devices, according to internet backbone provider Level 3 Communications.

Microsoft

- Wrong choice of IV

One of the most important rules of stream ciphers is to never use the same keystream to encrypt two different documents. If someone does, you can break the encryption by XORing the two ciphertext streams together. The keystream drops out, and you end up with plaintext XORed with plaintext—and you can easily recover the two plaintexts using letter frequency analysis and other basic techniques.

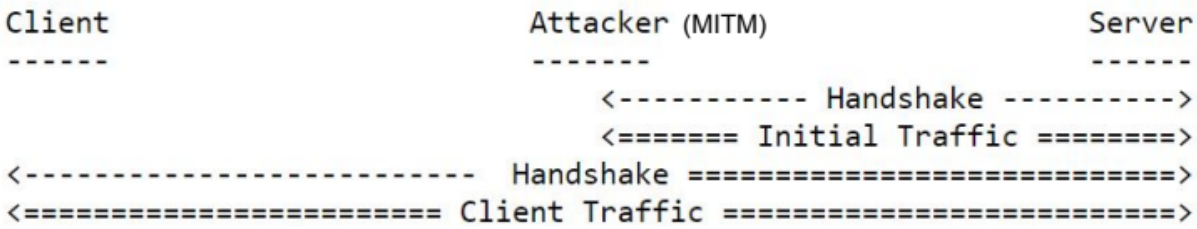
It's an amateur crypto mistake. The easy way to prevent this attack is to use a unique initialization vector (IV) in addition to the key whenever you encrypt a document.

Microsoft uses the RC4 stream cipher in both Word and Excel. And they make this mistake. Hongjun Wu has details (link is a PDF).

> In this report, we point out a serious security flaw in Microsoft Word and Excel. The stream cipher RC4 [9] with key length up to 128 bits is used in Microsoft Word and Excel to protect the documents. But when an encrypted document gets modified and saved, the initialization vector remains the same and thus the same keystream generated from RC4 is applied to encrypt the different versions of that document. The consequence is disastrous since a lot of information of the document could be recovered easily.

MIFARE

MIFARE Classic is a contactless smartcard widely used in Europe. It uses a set of proprietary protocols/algorithms. However, they are reverse-engineered in 2007. It turns out that the encryption algorithms are already known to be weak (with 48-bit keys) and breakable.



Note for second handshake:

- Client attempts to do TLS handshake with server
- Attacker intercepts the handshake, encrypts it using session key from first handshake
- Attacker sends the encrypted handshake to server
- Now server thinks that the attacker is trying to perform TLS renegotiation (to get a new session key)
- Server doesn't know that client sent the handshake
- Subsequent client traffic will be encrypted (attacker cannot decrypt)

Common Terms

End-to-End Encryption (E2EE)

method of secure communication that prevents third parties from accessing data while it's transferred from one end system or device to another. In E2EE, the data is encrypted on the sender's system or device, and only the intended recipient can decrypt it.

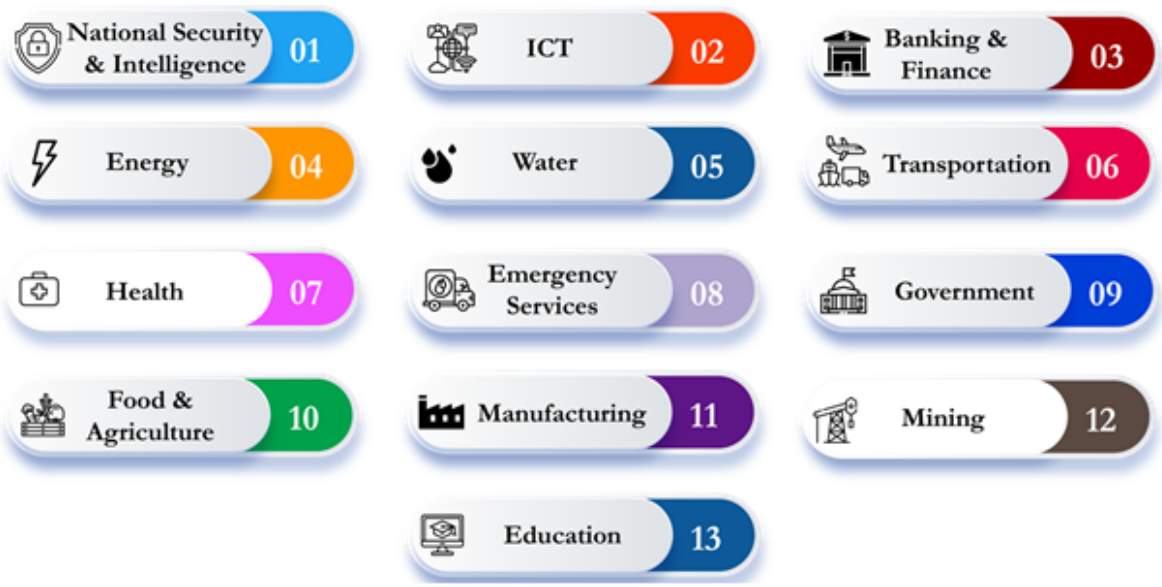
- Public key, or asymmetric, encryption uses a public key that can be shared with others and a private key. Once shared, others can use the public key to encrypt a message and send it to the owner of the public key. The message can only be decrypted using the corresponding private key, also called the decryption key.
- The key used in single-key encryption can be a password, code or string of randomly generated numbers and is sent to the message recipient, enabling them to unencrypt the message. It may be complex and make the message look like gibberish to intermediaries passing it from sender to receiver. However, the message can be intercepted, decrypted and read, no matter how drastically the one key changes it if an intermediary gets ahold of the key. E2EE, with its two keys, keeps intermediaries from accessing the key and decrypting the message.

CII - Critical Information Infrastructure

Energy, Water, Banking and Finance, Healthcare, Transport, Infocomm, Media, Security and Emergency Services, and Government. I Availability is key. Operational technology (OT) cybersecurity references the software, hardware, practices,

personnel, and services deployed to protect operational technology infrastructure, people, and data.

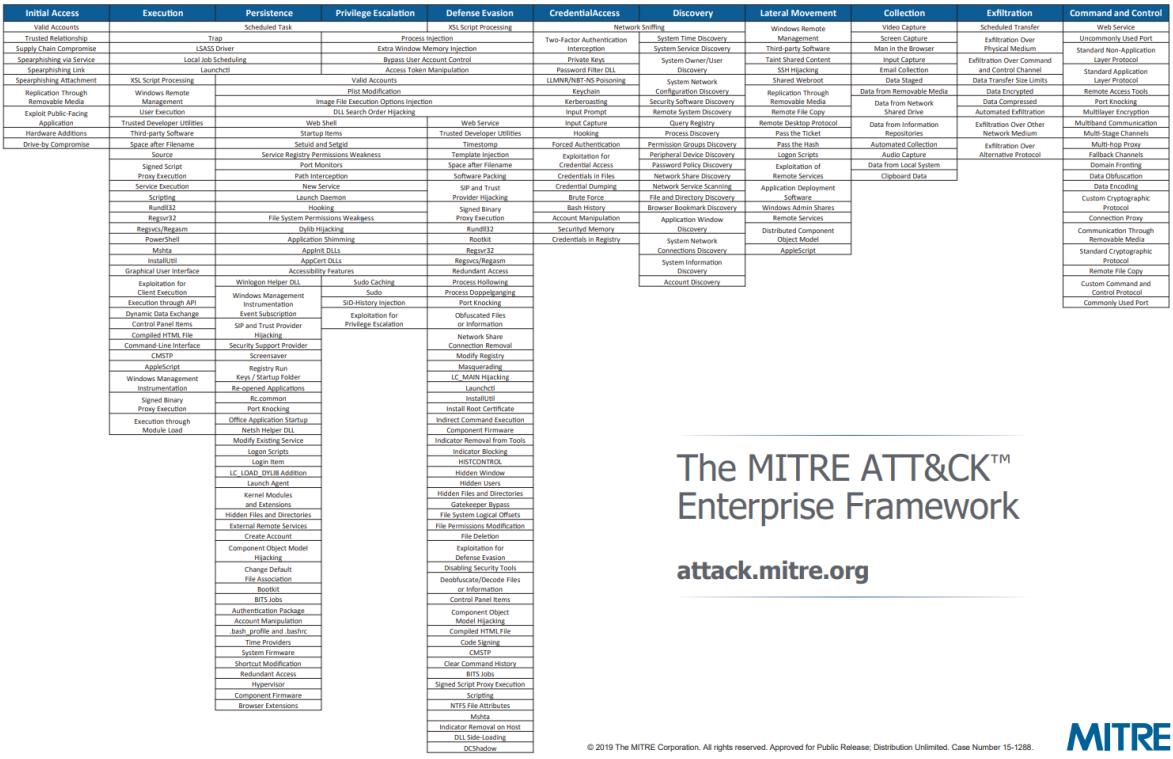
Critical Information Infrastructure (CII) constitutes assets (real/virtual), networks, systems, processes, information, and functions that are vital to the nation such that their incapacity or destruction would have a devastating impact on national security, the economic and social well-being of citizens. CII may comprise a number of different infrastructures with essential interdependencies and critical information flows between them.



MITRE ATT&CK

ATT&CK was first created by a MITRE internal research program using our own data and operations. Now based on published, open source threat information, MITRE provides the framework as a resource to the cyber community. Anyone is free to leverage it, and everyone is free to use and contribute to ATT&CK.

globally-accessible knowledge base of adversary tactics and techniques based on real-world observations. The ATT&CK knowledge base is used as a foundation for the development of specific threat models and methodologies in the private sector, in government, and in the cybersecurity product and service community



`setuid`

`setuid`: a bit that makes an executable run with the privileges of the owner of the file.

The Linux shell will check in this order:

1. Is Bob the owner? No
2. Is Bob in Alice's group? No
3. Check the 'Others' permission bits. Execute bit is not set. Access denied.

So the answer is (e) - Linux shell will reject Bob's execution invocation. The role of the SetUID bit is that once the file is executed and the corresponding process is created, it will have an effective UID of the owner. But Set-UID does NOT just allow any arbitrary user to just run the program. The user still has to have executable rights in the first place.