Made by LI MINGYANG

Tutorial Definitons

Cryptology: The broader study of techniques for secure communication, including both cryptography and cryptanalysis.

Cryptanalysis: The art of breaking cryptographic systems, analyzing security weaknesses, and deciphering encrypted messages without access to the decryption key.

Cryptography: The practice of secure communication through encryption and decryption methods to protect data from unauthorized access.

NSA (National Security Agency): A U.S. government agency responsible for signals intelligence and cybersecurity, often involved in cryptographic research and surveillance.

NIST (National Institute of Standards and Technology): A U.S. government agency that develops standards for encryption, including AES (Advanced Encryption Standard) and other cryptographic guidelines.

Cryptography Backdoor: A hidden method for bypassing encryption, often inserted by governments or malicious actors to access encrypted communications.

Key Escrow: A system where a third party (usually a government or regulatory body) holds encryption keys to enable authorized decryption under specific circumstances.

Decryption Order: A legal directive requiring individuals or organizations to provide access to encrypted data, often in law enforcement investigations.

Graphical Passwords: Authentication methods where users select images, patterns, or gestures instead of alphanumeric passwords.

Covert Channel: A method of transmitting information in an unauthorized or hidden way within a system.

Side Channel Attack: A cryptanalysis technique that exploits information leaked from a system (e.g., power consumption, electromagnetic emissions, ciphertext size to deduce plaintext) rather than breaking encryption directly.

End-to-End Encryption: A security measure ensuring that data is encrypted on the sender's side and decrypted only by the intended recipient, preventing third-party access.

Single Sign-On (SSO): A user authentication process allowing access to multiple applications with a single set of login credentials.

Retinal Scan vs. Iris Scan:

- Retinal Scan: Uses the unique pattern of blood vessels in the retina for biometric identification.

- Iris Scan: Captures the unique patterns of the iris (colored part of the eye) for authentication, considered less invasive than a retinal scan.

Skimming: Skimming refers to the act of stealing sensitive data from payment cards (such as credit or debit cards) without the cardholder's knowledge. It is commonly used by cybercriminals to clone cards and commit fraud.

Practical examples of padding oracle attack on AES CBC: A notable example of a vulnerability based on the AES-CBC padding oracle attack is CVE-2016-2107. This vulnerability affected the AES-NI implementation in OpenSSL versions prior to 1.0.1t and 1.0.2h. Due to improper memory allocation during padding checks, remote attackers could exploit this flaw to perform a padding oracle attack, allowing them to decrypt sensitive information from AES CBC sessions. Interestingly, this issue arose from an incorrect fix for a previous vulnerability, CVE-2015-0293.

Kerckhoffs's principle: A cryptosystem should be secure even if everything about the system, except the key, is public knowledge. Don't rely on security by obscurity.

Typosquatting: Also known as URL hijacking, is a form of cybersquatting (sitting on sites under someone else's brand or copyright) that targets internet users who incorrectly type a website address into their web browser (e.g., "Ggoogle.com" instead of "Google.com")

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

Eve: Eavesdropper, who is usually a passive attacker. While she can listen in on messages between Alice and Bob, she cannot modify them.

Mallory: A malicious attacker who is active and who can modify messages, substitute messages, or replay old messages.

Trent: A trusted arbitrator, who acts as a neutral third party.

Graphical Passwords: an authentication system that works by having the user select from images, in a specific order, presented in a graphical user interface (GUI).

Cryptosystem: A system for encryption and decryption

Introduction Lecture

A system can fail due to various reasons: 1. Operator mistakes, 2. Hardware failures3. Poor implementation, 4. Deliberate human actions designed to cause failure.

Cyber security is concerned with such intentional failures. We are concerned with the following:

Assets: 1. Hardware, 2. Software 3. Data and information 4. Reputation, which is intangible

Threat

• A set of circumstances that has the potential to cause loss or harm

Vulnerability

• A weakness in the system, for example, in procedures, design, or implementation,

that might be exploited to cause loss or harm

Control

• A control, countermeasure, security mechanism is a mean to counter threats

• It is an action, device, procedure, or technique that removes or reduces a **vulnerability**

A **threat** is blocked by **control** of a **vulnerability**

There are a few other terminologies, but it sums up to:

There is a **threat agent** that gives rise to a **threat** that exploits a **vulnerability** that leads to

a **risk** that can damage an **asset** and cause an **exposure**, all of which can be counter

measured by a **safeguard** that directly affects the **threat agent.**

CIA triad and Others

Confidentiality

• The ability to ensure that an asset is viewed only by authorized parties

• Prevention of unauthorized disclosure of information

Privacy

• Assures that individuals control or influence what information related to them may be collected or stored and by whom and to whom that information may be disclosed.

Integrity

• The ability to ensure that an asset is modified only by authorized parties

• Prevention of unauthorized modification of information or processes

Availability

• The ability to ensure that an asset can be used by any authorized parties at anytime

• Prevention of unauthorized withholding of information or resources

Authenticity/Authentication

• The ability of a system to confirm the identity of a sender

Non-repudiation/Accountability

• The ability of a system to confirm that a sender cannot convincingly deny having sent something

The US Department of Defense adds:

Auditability

• The ability of a system to trace all actions related to a given asset.

We can also view CIA in terms of the nature of harm caused to assets, characterized by three acts:

• **Interception:** Confidentiality suffers if someone intercepts the data

• **Interruption:** Availability is lost is someone or something interrupts a flow of data or access

to a computer

• **Modification and Fabrication:** Integrity can fail

It can be difficult to achieve security due to:

• Not considering security during early design stage of a system

• Difficult to formulate security requirements

• Various design constraints

• Difficult to verify that a design achieves the intended security requirements

• Even if the design is secure, the implementation may be wrong

• There will always be a weakest point

• The humans involved in operating the system can be exploited: Configuration errors, Mismanagement of credentials/patches/etc.

• Open Web Application Security Project(OWASP) has reported outlining concerns for web applications, focusing on the 10 most critical risks.

• Crowdstrike report examines how attackers are adapting to evade detection

There is always a trade-off between security and:

• Ease-of-use: Security mechanisms interfere with working patterns users are originally familiar with.

• Performance: Security mechanisms consume more computing resources

• Cost: Security mechanisms are expensive and difficult to develop

Attack models

-we need a more precise way to describe security requirement compared to CIA triad.

Eg. we can compare 2 systems: if some attacks are successful on s1 but s2 can prevent all possible attacks, then s2 is more secure than s1, with respect to that attack model

- We describe a class of attacks by giving

•The attacker's goals

•The attacker's capabilities

This description is also known as attack model.

Attacker goals

• **Total break:** (VERY HARD) attacker wants to find the key

• **Partial break:** attacker wants to decrypt a cipher text but not interested in the secret key, or simply extract some information about the plaintext.

• **Distinguishability:** (WEAKEST) A very modest goal. With some "non-negligible probability" more than half, the attacker can correctly **distinguish** the ciphertexts of a given plaintext (say "Y") from the ciphertext of another given plaintext (say "N"). If the attacker is unable to do so, we call this indistinguishability or the scheme is semantically secure.

- If an attacker can achieve total break, the attacker can achieve partial break and indistinguishability.

- We want to design a secure system to prevent attacker to even achieve the weakest goal.

Attacker capabilities

Ciphertext-only attack (CTO): The attacker is given a collection of ciphertext c (attacker cannot choose) and may know some properties of the plaintext. The attacker will analyse the ciphertext itself and apply various statistical tests to it. Eg. attacker knows plaintext is English language hence can use a letter frequency table to map a ciphertext to a letter of the English alphabet. Most frequent ciphertext likely to be letter e or a. Eg. Exhaustively attempt all possible keys to decrypt the ciphertext and obtain the plaintext.

This is the **WEAKEST** attack as attacker has least amount of information and the attack time is not conclusive.

Known plaintext attack (KPA): The attacker is given a collection of plaintext m and their corresponding ciphertext c. The attacker may be able to capture one or more (plaintext, ciphertext) pairs. From the captured pairs, if a ciphertext that matches one of the pairs known apear, he knows that it will map to a certain plaintext. The attacker may be able to find the key based on the way the known plaintext is transformed.

Chosen plaintext attack (CPA): The attacker can choose arbitrary plaintext's to be encoded and obtain corresponding ciphertexts. (Attacker has encryption oracle). The attacker can feed any plaintext to the oracle and obtain the corresponding ciphertext, all encrypted with the same key. He can compute multiple ciphertexts and analyse how different plaintext inputs affect ciphertext. Attacker can access the oracle as many time as possible. Eg. Smart card, protocol

Chosen ciphertext attack (CCA2): Same as CPA but here, the attacker chooses the ciphertexts and the decryption oracle outputs the plaintexts. (Padding oracle attack is a weak form of decryption oracle) If a cipher can defend against an oracle(an attacker with the highest capability), then the cipher can defend against all other weaker forms. Many systems employ cipher that is only Secure against CPA but not CCA2.

Encryption

Encryption is designed for confidentiality(not necessarily provides integrity although some do)

Encryption is the process of converting a message (plaintext) into a meaningless message (ciphertext). Decryption is the reverse process.

Properties

Correctness: For any plaintext c and key k, dec(k, enc(k, c))=c

Security: Challenging to define. Informally, from the ciphertext, the eavesdropper is unable to derive any useful information of the key k or plaintext c even if the eavesdropper can properly the system. The ciphertext is indistinguishable from a random steam.

Probabilistic: Encryption could be probabilistic. For the same plaintext, there could be a different ciphertext, However, all the ciphertexts are decrypted to the same plaintext.

Lecture 2: Symmetric key encryption

Classical ciphers: substitution and permutation cipher

Substitution Cipher

• **Plaintext and ciphertext**: a string over a ***set of symbols U***.

E.g., Let U = {"a", "b", "c", ..., "z", "_"}

• **Key**: a ***substitution table S***, representing an ***1-1 onto function*** from U to U (each letter in the plaintext is mapped to a ***unique letter*** in the ciphertext).

• E.g., S(a) = g, S(b) = v, ...

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

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

Substitution Cipher: Encryption

• Given a plaintext: X = x1 x2 x3 ... xn and key S,

• Ciphertext: $E_s(X) = S(x1)\ S(x2)\ S(x3)\ \ldots\ S(xn)$

• **Plaintext**: h e l l o _ w o r l d

• **Ciphertext**: h n l l q p o q i l b

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

Substitution Cipher: Decryption

• Given a ciphertext: C = c1 c2 ... cn and the key S,

• Plaintext: $D_s(C) = S^{-1}(c1)\ S^{-1}(c2)\ \ldots\ S^{-1}(cn)$

• **Ciphertext**: h n l l q p o q i l b

• **Plaintext**: h e l l o _ w o r l d

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

Permutation Cipher/Transposition Cipher

• The encryption **first groups** the plaintext into ***blocks of t characters***, and then apply a ***secret "permutation"*** to each block by ***shuffling the characters***.

• The key is the ***secret "permutation"***, which is an 1-1 onto function, e

from {1, 2, .., t} to {1, 2, .., t}

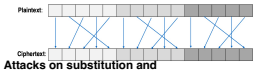
• The size t could be part of the key, that is, t is also kept secret.

• Permutation p as a sequence $p = (p_1, p_2, p_3, \ldots, p_t)$

○ shifts the character at position i to the position p_i

Permutation Cipher: Example

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



Attacks on substitution and permutation ciphers

Exhaustive search

Attacker's goal: Find the key or to obtain information of the plaintext

Attack step: Exhaustively search the keys, examining the possibles keys one by one. Eventually a correct key is found. (If exam question is on brute forcing, always ask yourself how do I know if a certain key is correct? What can I compare with to know this is the correct or wrong key?

- For a cipher to be secure, exhaustive search must be computationally infeasible

- Sophisticated attacks exploit weakness of the encryption scheme so that that it can break faster than exhaustive search.

Terminology

Key space: The set of all possible keys.

Key space size: The size of key space is the total number of possible keys. For 3 bit key there is 2^3=8 possible keys

Key size: Number of bits required to represent a key. Eg. 3 bit key has size 3.

Confusion: An attacker should not be able to predict what will happen to the ciphertext when one character in the plaintext changes. In other words, the input (plaintext and key pair) undergoes complex transformations during encryption. A cipher with good confusion has a complex functional relationship between the plaintext/key pair and the ciphertext.

Diffusion: A change in the plaintext will affect many parts of the ciphertext. In other words, information from the plaintext is spread over the entire ciphertext. The transformations depend equally on all bits of the input. A cipher with good diffusion requires an attack to access much of the plaintext in order to infer the encryption algorithm.

One time pad

Given a n-bit plaintext (x1x2…xn) and a n-bit key (k1k2…kn), we can output the ciphertext, C:

C = (x1⊕k1)(x2⊕k2)(x3⊕k3)…(xn⊕kn)

The condition is that the key and the plaintext must be of the same length.

We can decrypt the ciphertext by XORing the ciphertext with the key once more to get the plaintext, X:

X = (c1⊕k1)(c2⊕k2)(c3⊕k3)…(cn⊕kn)

For this to work, we need to be able to transfer the key securely. The OTP must also contain truly random characters, only two copies of the OTP should exist and it should be only used once, being destroyed right after usage. However, if we can securely transfer the key, we might as well securely transfer the plaintext!

Some interesting properties of XOR:

• **Commutative:** A ⊕ B = B ⊕ A

• Associative: A ⊕ (B ⊕ C) = (A ⊕ B) ⊕ C

• **Identity element:** A ⊕ 0 = A, where 0 is the identity element

• **Self-inverse:** A ⊕ A = 0

For One-Time-Pad:

- Key Space

- One-Time-Pad: Set of all possible keys of the same length as the plaintext

- Key Space Size

- Total number of possible keys

- One-Time-Pad: 2^n

- Key Size or Key Length

- Number of bits to represent a particular key

Security of One-Time-Pad:

It is easy to derive the key from knowing the ciphertext and plaintext, but this key will be useless, as the key will not be used again. It is also very difficult to perform exhaustive search on it. Furthermore, even if you know part of the key, you cannot figure out the rest i.e. it leaks no information of the plaintext, even if the attacker has an arbitrary running time. Hence, the one-time-pad is also called "unbreakable", provided that a good "random" key is used.

Stream cipher

Stream cipher is inspired by the one-time-pad. Suppose the plaintext is 220 bits long, but the secret key is only 256 bits. Stream cipher will generate a 220-bit sequence from the short key, and take the generated sequence as the "secret key" in one-time-pad. This generator will need to be carefully designed so as to give a cryptographically-secure pseudorandom sequence. The stream cipher is thus a symmetric cipher, where the same key (or rather, same secret key) is used for both encryption and decryption. Stream cipher ensures confusion but not diffusion.

Stream cipher without an IV is vulnerable to **key stream-reuse attack**. Without the IV, if an attacker can obtain two ciphertexts, they can simply XOR the two ciphertexts together.

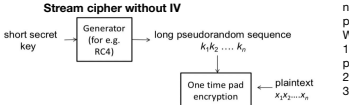
Let's say we have ciphertext U and V, which are obtained from plaintexts X and Y respectively without using IV.

U ⊕ V = (X ⊕ K) ⊕ (Y ⊕ K) = (X ⊕ Y) ⊕ (K ⊕ K) = X ⊕ Y

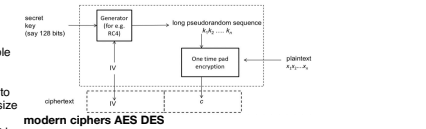
The above is obtained after doing some manipulation as a result of the associative and commutative properties of XOR. We can get X ⊕ Y.Although X ⊕ Y does not seem like a big deal but it is. Recall the zebra example in the I slides.

With a random IV value, even if same key is used for multiple encryptions, the generated key stream will be different.

Stream cipher without IV



Stream cipher with IV



modern ciphers AES DES

They are "supposed" to be secure so that any successful attack does not perform noticeably better than exhaustive search.

DES: symmetric-key block cipher developed by IBM in the early 1970s. Block size: 64 bits; Key size 56 bits. Over time, DES's 56 -bit key length has proven vulnerable to brute-force attacks. **AES:** In 2000, a new standard introduced, AES (Advanced Encryption Standard), was proposed by NIST. NIST called for proposal in 1997 and in 2000, Rijndael was selected as AES. AES block length is 128, and key length can be 128, 192 or 256 bits. Currently, no known attacks on AES (there are some attacks on the mode-of-operation)

Mode-of-operations

DES and AES are also known as "Block Cipher". A block cipher has fixed size input/output. Hence for a large plaintext, we need to first divide into blocks, then we can apply cipher. The method of extending encryption from a single block to multiple blocks to encrypt is called mode-of-operation.

ECB:

Mode-of-Operation: Electronic Code Book (ECB)

• Divides plaintext into blocks and then ***applies block cipher to each block***, all with the same key.

• $C_1 = E_K(P_1)$

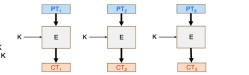
• $P_1 = E^{-1}{}_K(C_1)$

E_K: encryption function using key K

$E^{-1}{}_K$: decryption function using key K

• **Disadvantage**?

○ ***Same plaintext*** always corresponds to ***same ciphertext***



CBC:

Mode-of-Operation: Cipher Block Chaining (CBC)

• $C_1 = E_K(P_1 \oplus CT_{j-1})$

• $P_1 = E^{-1}{}_K(C_1) \oplus CT_{j-1}$

• $C_0 = IV$ called initialization vector IV

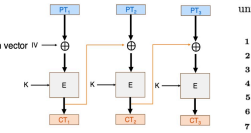
• **Advantages**

○ Semantic security

• **Disadvantages**

○ Cannot be parallelized

CTR:



Mode-of-Operation: Counter Mode (CTR)

• $X_1 = IV$ called initialization vector

• $X_j = X_1 + j - 1$

• $CT_j = E_K(X_j) \oplus PT_j$

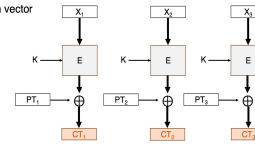
• $PT_j = E_K(X_j) \oplus CT_j$

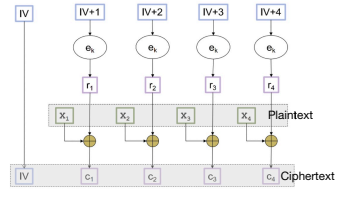
• **Advantages**

○ Semantic security

○ Can be parallelized

• Similar to stream cipher





Attacks

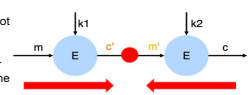
Similar to stream cipher, CTR more is vulnerable to key stream reuse attack

Meet in the middle attack on DES

DES is not secure: improve it by multiple encryptions;

c = DES(k2,DES(k1,m)).

Using exhaustive search, amount of DES encryption required is **not 2^(56+56) but rather 2^(56+1). In general for K-bit keys, number of crypto operations is 2^(k+1), using approximately 2^(k+1) storage space.**



c'=DESenc(k1,m) c=DESenc(k2, c')

m'=DESdec(k2,c) m=DESdec(k1,m)

FIND c' == m'

3-DES with 2 keys are believed to have to same level of security of 2-DES. 4-DES with 4 keys requires 2^113 operations.

Note: All attacker needs to do is to brute force the keys k1 and k2 such that c' == m'. Brute forcing k1 requires 2^56 operations, storing the result of each operation c' inside a table. Brute forcing k2 requires 2^56 operations, storing the result of each operation m' inside a table. Then iterate through the 2 tables to find c' == m'

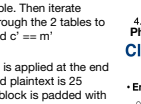
Padding oracle attack

In general, for modes of operations, padding is applied at the end of a plaintext. Eg. If block size is 16 bytes and plaintext is 25 bytes long, the remaining 7 bytes in second block is padded with seven 07s. If the last block is full, ie plaintext is now 32 bytes just nice, an extra block is added where each block has value of 16 as padding.

What Padding oracle does:

1. Decrypt the ciphertext using IV and the key only it knows to get plaintext
2. Check the padding of the plaintext
3. If correct padding, return true else return false

Attackers Steps



• For t = 00 to FF (hexadecimal representation, i.e. 0 to 255 in decimal)

○ Let IV = IV ⊕ | 0 | 0 | 0 | 0 | t | 07 | 07 | 07 |

○ Sends the two-block query (IV' ‖ c) to **Oracle**.

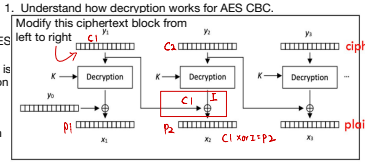
○ If Oracle gives YES, then outputs (04 ⊕ t) as the X_4 FORCED BRUTEFORCED

• End-for-loop

• **Idea**: Attacker uses the padding oracle by ***submitting modified CTs*** and observing the system's ***responses (yes/no)*** to infer the correct padding bytes, eventually to learning about the plaintext.

Alternative attack to determine number of padding in plaintext for A CBC

1. Understand how decryption works for AES CBC.



2. Understand that if ciphertext has 2 blocks, how decryptions works is that the second ciphertext is first decrypted by AES. Then, the intermediate I, is xored with the precious ciphertext block instead of the IV. Hence, we can do linear probing on the previous ciphertext block c1. Modify byte by byte from left to right for ciphertext block c1, querying the oracle for each modification. if oracle says correct padding, then the byte you modified is NOT the padding and you need to modify the next byte to the right until the oracle replies wrong padding. The idea behind this is that the padding has a fixed value hence cannot be modified. However, other bytes that are not the padding can be anything. Minor note: instead of linear probing, binary probing can be done too.

Method 1: Linear Probing

Assume that the length is i, and test it. Test one by one starting from 1 until 16.

Input: c, the ciphertext consisting of 3 blocks (including IV)

```
for i = 1..16 do
    c' ← c with the i^th byte of the 2nd block changed
    reply ← Query(c')
    if reply is "invalid" then
        return 17 − i;
end
```

This would take at most 16 queries to the oracle, with a simple optimization to reduce it to a maximum of 15 queries.

Lecture 3

Public key(asymmetric key) encryption scheme uses two different keys for encryption and decryption. In truth, there is no need to clearly indicate which key is the encryption key and which is the decryption key, as both keys are able to perform either decryption or encryption, just not both. As such, the following two statements are both valid:

c = Dec(Kpriv, Enc(Kpub, c))

c = Dec(Kpub, Enc(Kpriv, c)

Typically, the public key is the one shared with everyone, while the private key is kept secret by the user. It is thus possible for a person sending a message to me to encrypt it with my public key and send it to me for me to decrypt with my private key. **Note, also when I say I use private key to decrypt, I am actually using both private and public key to decrypt.**

Security requirement: As usual, Given the public key and the ciphertext (but not the private key), it is difficult to determine the plaintext(plaintext are indistinguishable). The requirement implies that it must be difficult to get the private key from the public key. **Note:** with PKS, encryption oracle is given since anyone can encrypt. Hence **chosen plaintext attack(CPA)** must be considered. **Why public key scheme(PKS)?** Without PKS, every individual would need to keep a symmetric secret key pair with everyone else. If the number of entities is large, the number of keys to keep track would be very huge as well. Any new parties joining in would result in a lot of secret key establishing as well, which is high difficult. The total number of keys needed without using PKS and using symmetric-key setting is **n(n-1)/2 = O(n^2).** With public-key setting, every entity just needs its own public and private key. The total number of keys is **2n = O(n). Entities don't need to know each other before broadcasting the keys.**

RSA

The RSA Algorithm was proposed in 1977, after Diffie and Hellman published the concept of the public-private key cryptosystem in 1976. It is based on the mathematical properties of large prime numbers and modular arithmetic. The basic principle of RSA encryption is based on the mathematical difficulty of factoring large numbers into their prime components.

Three phases

1. Key Generation: using two large prime numbers

2. Encryption: modular exponentiation using public key

3. Decryption: modular exponentiation using private key

Phase 1:

Classroom RSA: Key Generation

1. Alice randomly chooses 2 large primes p, q and computes

   $n = pq$, $\Phi(n) = (p-1)(q-1)$ [$\Phi(n)$ is the Euler's totient function]

2. Alice randomly chooses an *encryption exponent* e such that

   $gcd(e,(p-1)(q-1)) = 1$

   i.e., e does not have common factor with (p-1) or (q-1)

3. Alice finds the *decryption exponent* d *(multiplicative inverse of e)* such that

   $de \bmod \Phi(n) = 1$

4. Owner publishes (n, e) as public key, and safe-keeps d as the private key.

Phase 2 and 3

Classroom RSA: Encryption and Decryption

• **Encryption**

○ **Public key**: <n, e>

○ Given m, the ciphertext $c = m^e \bmod n$

• **Decryption**

○ Decryption key: <n, d>

○ Given c, the plaintext $m = c^d \bmod n$

Note: Difficult to derive private key, d from public key <n, e> unless you can factor n to find p and q

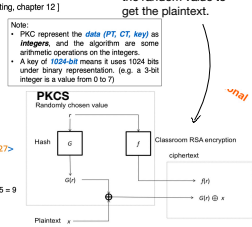
• **Correctness**: For any positive $m < n$, and any pair of public/private keys,

○ $D(E(m)) = m$

○ That is, $(m^e)^d \bmod n = m$ [Optional Proof: Security in Computing, chapter 12]

RSA has the same problem that symmetric key encryption faces, which is that the encryption of the same plaintext at different times will introduce the same ciphertexts. Thus, some form of IV or padding is required to introduce an element of randomness.

PKCS adds optimal paddings to achieve the diagram below. Sender only needs to encrypt the random value using RSA. Only the right person can decrypt the random value to get the plaintext.



Example

• p = 5, q = 11, n = 55

• $\Phi(n) = (p - 1)(q - 1) = 40$

• Suppose e = 3, then d = 27

○ Choose a d such $de \bmod \Phi(n) = 1$

○ For, d =27; 3x27 mod 40 = 81 mod 40 = 1

• Public key : <55, 3> and decryption key = <55, 27>

• Suppose m = 9

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

○ **Decrypt** : $c^d \bmod n = 14^{27} \bmod 55 = 14^{8x3+3} \bmod 55 = 9$

Properties

Security of RSA

The security of RSA is based Integer factorization problem: While multiplying two large primes to generate n is easy, factoring n back into its prime components (i.e., p and q) is computationally infeasible for large values of n. We know that given the public key <n,e>, it is difficult to get the private key d due to the factorization problem. However, given the plaintext m and public key <n,e>, it is not known whether the problem of getting the plaintext is as difficult as factorization. The difficulty is associated with the integer factorisation problem but not definitively proven. RSA remains secure when using sufficiently large key sizes (e.g., 2048 bits or more). Note that "classroom" RSA still have to be modified with an IV to ensure indistinguishability between same plaintext.

Homorphic property

Multiplication of ciphertexts corresponds to the multiplication of the plaintext in modulo n i.e, c1 = m1^e mod n; c2 = m2^2 mod n; c1 x c1 = (m1 x m2)^e mod n

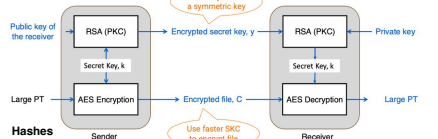
Malleable property

RSA is malleable so an attacker can exploit this property to manipulate ciphertexts in unintended ways. Hence we need padding scheme as described before in the cheatsheet.

Efficiency

Following NIST recommendation, 128-bit AES and 3072-bit RSA has the equivalent key strength. Larger key might need to difficulty in key management. Next, Encryption in RSA is done by exponentiation compared to substitution and permutation in symmetric scheme. RSA encryption/decryption is significantly slower than AES. To encrypt a large file F, it would be too slow to directly RSA on F.

To improve this, **we combine RSA with symmetric encryption.**



Hashes

A **cryptographic hash** is a function that takes in an arbitrarily long message as input and outputs a **fixed size digest.** Hashes are. Good way to ensure **Integrity**. If even one bit of the message is altered, the hash be very different and will not match. Note: no secrets involved as the hash algorithm is known to the attacker.

Properties of a good hash

Collision-resistant: It is difficult to find ANY two different messages m1 ≠ m2 that "hashes" into the same digest, i.e. h(m1) = h(m2)

Preimage Resistant or One-way: Given a digest d, it is difficult to find a m such that h(m) = d. In other words, it is difficult to reverse-engineer a hash.

Second-preimage resistant: Given a specific m1, it is difficult to find a specific second preimage m2, where m1 ≠ m2, but h(m1) = h(m2).

NOTE: hash digests should follow a uniform distribution, not normal distribution.

Collision-resistance implies second preimage resistance but not the other way around.

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

SHA-0: Published by NIST in 1993. Produces a 160-bits digest. In 1998, an attack that finds collision in 2 ^61 operations was discovered. In 2005, Wang Xiaoyun gave an attack to find collision in 2^39 operations.

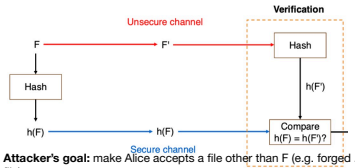
SHA-1: Revised version. Produces a 160 bit message digest. In 2005, Wang Xiaoyun gave a method to find collisions using 2^69 operations which was later improved to 2^53 operations. A collision was found in 2017 which took 110 GPU years, and completed 2^63 SHA-1 operations.

SHA-2: In 2001, NIST published SHA-224, SHA-256, SHA-384, SHA-512, collectively known as SHA-2. The number in the name indicates the digest length. No known attack on full SHA-2 but there are known attacks on "partial" SHA-2, for e.g. attack on a 41-rounds SHA-256 (the full SHA-256 takes 64 rounds).

SHA-3: NIST standardized it in 2015. SHA-3 includes several hash function variants: SHA3-224, SHA3-256, SHA3-384, and SHA3-512.

MD5: Designed by RIvest. MD, MD2, MD3, MD4, MD5, MD6. MD5 produces a 128-bit digest. In 1997, Dobbertin announced a collision of the compress function of MD5. In 2004, collision was announced by Wang Xiaoyun, which was reported to take one hour. In 2006, Klima give an algorithm that can find collision within one minute on a single notebook. In 2008, MD5 was officially declared “cryptographically broken”.

Application of using hashes without to secret key to ensure integrity



Attacker's goal: make Alice accepts a file other than F (e.g. forged .exe file)

To trick the user, the attacker need to have a F' such that H(F')=H(F) and F'!=F. The above is an example of 2nd pre-image problem.

Limitations of unkeyed hash:

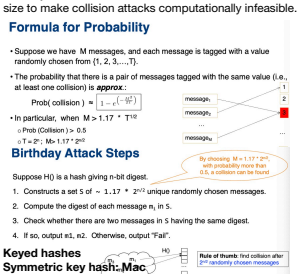
No Authentication: Only integrity verification but not authentication is provided, since identity of sender is not verified. It can be argued integrity is not provided if MITM can edit original message and rehash it using the same hash function.

The hash h(f) might not have been transmitted securely as well.

If a hash function is weak, it is vulnerable to Collin attacks or 2nd preimage attack.

Hence, we need **keyed hash.**

Birthday Attack: Based on the **pigeonhole principle**. Given the hash output is fixed at 256 bits for SHA-256 while the input is the entire universe, if more than 2^256 messages are hashed, at least 2 distinct messages must have the same hash output. However, due to the birthday paradox we don’t actually need 2^256 operations to find A collision. (**TAKE NOTE: we are talking about any collision not a specific collision for a message**) Hence, for SHA-256: an attacker would need to perform about 2^128 operations to find a collision(currently infeasible). We need to design a hash so that known attacks can’t do better than birthday attack i.e, the hash function must have sufficient large output size to make collision attacks computationally infeasible.



Keyed hashes

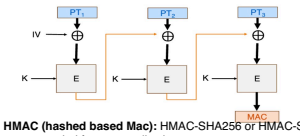
Symmetric key hash: Mac

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).

NOTE: MAC is a general symmetric key hash but **HMAC** is a specific implementation of **MAC**.

Authenticity is achieved as secret key is shared between sender and receiver.

MAC using AES under CBC

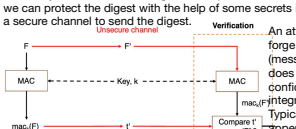


HMAC (hashed based Mac): HMAC-SHA256 or HMAC-SHA512 recommended for new applications

HMAC(k,x) = SHA-1(k⊕opad II SHA-1(k⊕ipad) II x), where II is string concatenation, opad is 3636….36, ipad is 5c5c…5c.

Application scenario with MAC

For unkeyed hash, the hash digest is not protected. However for MAC, we can protect the digest with the help of some secrets if we do not have a secure channel to send the digest.



An attacker will want to forge a valid pair of (message, mac). MAC does not ensures confidentiality, but only integrity and authenticity. Typically the Mac is appended to F, hence it is also called authentication tag/code. MAC with **symmetric key stream cipher(tutorial)** does not guarantee secure MAC.

Digital signatures (asymmetric key)

Digital Signature is the asymmetric-key version of MAC. If we used RSA to generate the digital signature, we now use the **private key to encrypt and public key to decrypt.**

(swap role of d and e so anyone can decrypt but only owner can encrypt). This property is only special about RSA. It usually does not hold for other public key schemes. **The steps are as follows for general digital signature: 1.**The original file, F, is first hashed using a hash function. **2.** The hash value is then passed through the signature function together with the private or signature key kPRIV to produce signature s, the signature of the authentic file. **3.** The signature is then appended to the original file F and sent to the recipient via an insecure channel. **4.**The recipient then passes the received signature and the public or verification key kPUB into the verification function. **5.** The verification function will produce a hash value. **6.**The file F’ is then pass through a hash function to return a hash value. **7.**The hash values are compared to see if the file has been modified, or if the digital signature is valid. **8.** Since digital signature is created by ‘private’ key of signer and no one else can have this key; the signer **cannot repudiate** signing the data in future.

The computed signature is typically appended to F and stored as a single file. Subsequently, the authenticity of F can be verified by anyone who knows the signer’s public key. This is because the valid signature can be computed only be someone knowing the private key. In addition to **authenticity**, signature scheme also achieves **non-repudiation**: assurance that someone cannot deny his/her previous commitments or actions. (Integrity is provided by default) **Minor note:** RSA-based signature schemes encrypt the hash in the signing algorithm(hash and encrypt rather than hash and sign for other signatures), but there are many other signature schemes that do not involve encryption. The method of encryption works with RSA encryption but not necessarily with others.

Why hash and sign? This is due to the greater efficiency. As the hash value is a unique representation of the data, it is sufficient to sign the hash in place of data. **(Integrity is preserved while size of data signed is smaller)** Let us asume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/ signing process using RSA involves modular exponentiation. Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence signing a hash is more efficient than signing the entire data.

For RSA digital signature steps, the signature function used in the previous list of steps described is just substituted to be the RSA algorithm.

Non repudiation:

As discussed earlier, **MAC does not provide non repudiation** as there are two parties with the symmetric key who can generate the MAC. Hence one party can deny sending a MAC, saying the other party that sent it as he also has knowledge of the key. For **digital signature**, as there is only one party with the private key to generate the signature, that party cannot deny having signed the message. **The signature is proof that the party with the private key generated the message.**

Lecture 4

Password Authentication: mainly for entity authentication.

Authentication is the process of assuring that the communicating entity or the origin of a piece of information is what it claims to be. Two types of authentication:

Communicating Entity Authentication: This is for connection-oriented communication, where the communicating entity is an entity involved in a connection. Some common mechanisms for this type of authentication are thus password, challenge and response.

This may be required to ensure authenticity over various communication channels, Eg.Logging in to LumiNUS via my account: Is the server that I’m interacting with authentic? Why would the LumiNUS’ server be convinced that the entity logged in isthe authentic me?

Data-origin Authentication: This is for connectionless communication, where the communicating entity is the origin of a piece of information. Data-origin authenticity implies data integrity.

This may be required to ensure authenticity of digital data and physical documents Eg. Submitted a MC for a test: Is this MC authentic?

Authenticity vs Integrity

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

To attain communicating entity authentication, we need some information bound to the owner.

Credentials: pieces of information that verify the identity of an individual. A user provides credentials to prove their identity. If a user can provide the credentials, it is proof that the user is indeed who they claim to be.

Types of credentials: 1. Something the user knows eg. Passwords, PIN numbers. 2. **something the user is (biometric)** eg. fingerprint, iris, Face ID. **3. something the user has** eg. Identity badge/physical key.

SOMETHING THE USER KNOWS: Password

Password is a way to achieve communicating entity authentication. - Username: you announce who you are (Identification) - Password: you prove that you are who you claim to be (Authentication).

Used for login system, wifi security.

Note: identity information is not considered to be secret, although it is not advisable to voluntarily make it public.

Password vs secret key

Similarity:

- Both can be used for entity authentication
- Both are sensitive information, if compromised can lead to breaches

Difference:

- Passwords are generated by human and can be remembered by human
- Secret keys are long binary sequence that are infeasible to be remembered by human
- Secret keys can be randomly generated or derived from password

A typical password system

Bootstrapping: A user and the server establish a common password through a secure channel. The server keeps the password file: user_id, (username) and the corresponding password. Sometimes a default password can be chosen.

Authentication: The server authenticates an entity. An entity who can convince the server that it knows the password, is deemed to be authentic.

Password reset: Only the authentic entity can reset password (need to verify using old password or another credential like biometric before reset). There are many reasons to reset password. E.g., User forgets the password. E.g., A password policy could require regular changes of password.

Having human (i.e., admin or IT support) involved in resetting password is expensive. Hence Many systems provide “self-help password reset” (initiate the password reset process themselves) like using **Recovery Email Account** (1. Alice requests a password reset from the server. The server generates a one-time password (OTP) link and sends it to Alice’s recovery email. 3. Alice receives the password reset URL (https://aaa.com/reset?QTP=J3ar92DadySEga5) via email. 4. Alice clicks the link, proving ownership of the email, and enters a new password. 5. The server verifies the OTP and allows Alice to reset her password) **OR** using **Security Questions** (users should know answers to their personal even if they don’t know password).

There also needs to be Secure Password File Storage.

- Password file should never be stored in clear text
- Use of hashed passwords and a fixed-length random salt value (virtually all UNIX variants)
- Password + salt is hashed and stored in the password file
- Pre-image resistance property: cannot get back the password from the hash value

Attack on bootstrapping

It is possible for an attacker to attack the bootstrapping. They can target the use of **default passwords** (real example: IP Security Cameras).

They can also aim to intercept the password in the process of being sent via an alternative channel (e.g. stealing the postal mail containing the NUSNET password).

Why do manufacturers use default passwords?

For the device manufacturer – cost will increase since there is a need to:

- print the password on each equipment/ manual/case
- record the passwords in a password database
- have a customer service line or Web site to help users retrieve their device’s default password.

For device users – usability will decrease if a user is unable to find or retrieve the default password. This may lead to a bad product review, and also affect manufacturer’s sales.

Mitigation: Require the user to change password after first login.

Mirai Botnent is an attack on default username and password, so as to use for DDOS attack later on. Known attack on default username and password: Mirai Botnet, which is later used for DDOS.

Attack on password reset

Attacker goal: get the OTP included in reset link then change victim password to take control of their account.

Vulnerability: some application constructs password reset URL link by taking the domain name from the Host header of HTTP request without validation eg. In a HTTP POST request for password reset, the application could use the HOST header to generate the reset URL link.

By using HTTPS, the attacker will not be able to perform step 1 as HTTPS is encrypted. Thus if the attacker sniffs Alice’s traffic he would not know that Alice is trying to perform a password reset and cannot modify the HTTP request fields. However attack can still occur if attacker initiates password reset using alice’s username and manipulate the host header in the request from there.

Password vs poisoning steps:

1. The attacker sniffs Alice’s HTTP traffic and notices that she is making a POST request to reset her password 2. The attacker intercepts the request, and changes the Host field in the HTTP request to evil.com
3. The server aaa.com does not verify that the entry in the Host field is evil.com and simply creates a new URL http://evil.com/reset?QTP=J3ar92DadySEga5 to send to Alice 4. Alice clicks on the URL and thus connects to evil.com which is controlled by the attacker, sending the OTP to the attacker 5. The attacker now knows the OTP and can reset the password on Alice’s behalf.

Mitigation:

- Input Validation: Ensure robust validation and sanitization of HTTP headers and user inputs.
- Secure URL Generation: Use server-side configuration for generating URLs
- User Awareness: Educate users to verify password reset emails.
- HTTPS
- 2FA

Attack on password reset security questions:

Although Personal security questions facilitate self-help password reset and improve “usability”, it weakens the password system.

- A victim’s resume/webpages/ biographic descriptions would reveal the answer to “with which company did you hold your first job?”
- Social media accounts exposes structured information about users

Mitigation using CAPTCHAs: ask a question about an image or audio file that had been previously specified by the user.

Password guessing

Exhaustive search: try all possible combinations of valid symbols, up to a certain length

Dictionary attack: search through a restricted name space by maintaining a dictionary Typically, exhaustive search on passwords not very feasible given the large key space. There is also no reason to try some combinations, given the low probability of the common user using certain characters.

Dictionary attack

This is a form of exhaustive search where the attacker can restrict the search space to a pre-compiled large collection of probable passwords.

These words tend to be words from the English dictionary, known compromised passwords, dictionaries from other languages etc. It is thus possible to carry out a hybrid attack, i.e. a mix of exhaustive search and dictionary attack. For example, we could try all combinations of 2 words from the dictionary, and exhaustively try all possible capitalizations of each word, substituting ‘a’ with ‘@’, etc.

Online dictionary attack

Attacker interact with the authentication system to test the password:

- Attacker obtained a list of 1000 valid nusnet id.
- Goal: The attacker wants to find the password for some of them.
- Step: writes an automated script that attempt to login to Canvas using guessed passwords for each nusnet id.
- Guessing password: using some social information about the user.
- Mitigation: rate limiting and account lockout.
- Tool: Hydra (open-source network login cracker)

Offline dictionary attack

The attacker obtains some information, D about the password, e.g., the “hashed” password is sent over, and the attacker manage to obtain the hash. Next, the attacker carries out dictionary attack using D without interacting with the system. The attacker generates a large dictionary of possible passwords and its hash values. Compare with the hashed words with the table

- Mitigation: adding salt to password and thenhash lead to increased attack complexity
- Compare with the hashed words with the table.
- Mitigation: adding salt to password and then hash lead to increased attack complexity
- Tool: John the Ripper

Other mitigation

- Limited login attempts
- Password checker or metering for weak passwords when user register
- Password usage policy: Users must use strong passwords
- Password ageing: Users must regularly change passwords
- 2FA

Password stealing

Sniffing:

- Shoulder surfing aka look- over-the-shoulder attack.
- Side-channel attack: Using sound made by keyboard.

Virus and Keylogger:

-Captures the keystrokes and sendsthe information to the attacker. • (software) Computer viruses are designed as a key-logger. • (hardware) Hardware key-logger

Cache and Lost password file:

- When using a shared workstation information keyed in could be cached. • Password file can be stolen

Phishing:

- Tricks the user to visit a website,which is a spoofed login web.
- Social engineering attack

Phishing mitigations

•User training: Embedded Phishing Exercise, similar to fire drill, authorized entities send out “phishing” emails to test employees.

•Blacklisting phishing site and collate on https://phishtank.com/.

Password strength

Guidelines for Strong Password:

•NIST password guidelines: At least 8 char long, 15 char preferable, at least one each of uppercase, lowercase, numeric digits, and special characters. 2. •Proactive password checker: At the time of selection, the system checks to see if the password is allowable and, if not, rejects it. we need to strike a balance between user acceptability and strength

Key Strength vs Password Strength

In encryption, we “quantify” the key-strength by the size of the key if best known attack is exhaustive search. Eg. Strength of 128-bit AES key is: 128 bits. (exhaustive search goes through 2^128 keys) E.g. Strength of 256-bit SHA is: 128 bits

For password, we use an imprecise/hard to estimate measure: **Entropy**

Entropy is a measurement of randomness (amount of variation). **Entropy is the number of bits required to represent the total number of possible paswords.**

Given a password of length L from a set of N possible symbol, the total number of possible passwords is N^L. the entropy is just log2(N^L) since we are converting N^L to be represented in bits. Entropy=Password Length x log2(Symbol Set Size). Note increasing N or L increases entropy. eg calculation: using upper, lower alphanumeric characters for 8 char password, symbol set size=62. entropy = log2(62^8).

RFC 4089 recommends at least **29 bits** of entropy to be secure against **online attacks**. It recommends at least 36- 59 bits for “higher security”. For **offline attacks**, since NIST recommend 128 bits for crypto keys, in this course, let’s take **128 bits** entropy asthe requirement

SOMETHING THE USER IS: Biometrics

Biometrics use unique physical characteristics of a person for authentication.

During enrollment: a reference template of a user’s biometric data is constructed and stored, similar to bootstrapping in the password system.

During verification: the biometric sample data of the person-in-question is captured and compared with the template using a matching algorithm. The algorithm decides whether to accept or reject the authentication.

Noise:

Authentication by password gives a clear reject or accept at each authentication attempt.

- In contrast, with biometrics will hardly ever match precisely the templatederived from the current measurements.
- A matching algorithm measures the similarity between reference template and current template.
- The user is accepted if the similarity is above a predefined threshold

Matching Algorithm

The matching algorithm typically makes decision based on some adjustable threshold, depending on the application.**Lower threshold** => more relax in accepting (**higher FMR**), opposite for higher threshold.

Attacks on biometrics

Presentation attacks: using some artefact to impersonate (Fake fingerprint)

Replay attacks: capturing and reusing a biometric data sample

Mitigation: 1. liveness detection(temperature sensor for scanners), 2. Limit attempts, 3. 2FA

SOMETHING YOU HAVE: OTP token, phone

Since something you have can be lost, we need 2FA. Eg. bank cards come with a pin.

Types of authentication for what you have:

1. Time-based: Both the user device and the server share a secret key. Using this secret key and the current time as inputs, an algorithm generates the same OTP on both the user’s device and the server. (Eg. Google Authenticator)
2. Sequence-based: Both the server and the user’s device maintain a counter along with the secret key. Upon each event (e.g., button press), the counter increments, and an algorithm uses it with the shared secret to generate the OTP (hardware tokens). N-factor authentication Require at least two different authentication “factors” / credentials. It is called a 2-factor authentication if 2 factors are employed (strict definition). eg. password(what u know)+mobile phone sms(what u have)

2-factor authenticaton ≠ 2 step

authentication: Many online platforms use login password + link that is sent to the email account(this is not 2FA as both are “what you know”). This is 2-password authentication method although platforms still call it 2FA.

$$False\ Match\ Rate = \frac{Number\ of\ successful\ false\ matches\ (B)}{Number\ of\ attempted\ false\ matches\ (B+D)}$$

$$False\ Non-Match\ Rate = \frac{Number\ of\ rejected\ genuine\ matches\ (C)}{Number\ of\ attempted\ genuine\ matches\ (A+C)}$$

	Accepted / Match	Rejected / Non-Match
Genuine Attempt	**A**	**C**
False Attempt	**B**	**D**

Lecture 5

Public Key Distribution

After we have decided on using public and private keys, one issue remains – there needs to be a way to transfer the public key securely. A secure channel is needed.

Methods of distribution:

Public Announcement /Hardcoded:

The owner broadcasts her public key via blogs/web pages/physical name card.

Hard coded: Some apps/software have the public key hardcoded in the programs. The public key is to be used to verify data sent by the developer. **Limitations:** 1. Not standardized 2. No systematic way to search/verify the public key. 3. difficult to update keys if fixed in hardware

Publicly Available Directory:

Keys are stored in a directory accessible to the public, similar to a phone directory. If Bob wants to find Alice’s “public key”, he can search the public directory by querying the directory server with Alice’s email.

Limitations: 1. Anyone can post their public keys in the server so not clear how to verify info. 2. Not everyone trusts the server. 3. The server could be overwhelmed and can’t handle the load.

Public Key Infrastructure (PKI):

It PKI is a framework that encompasses the software, policies, procedures, and standards to required for secure key distribution to address limitation of the previous methods. A main objective of PKI is to be deployable on a large scale. “Public PKI” refer to the PKI we adopted in Internet for domain name, emails address etc. “Private PKI” are systems for private applications, and it has its own set of “CA”.

Key features: 1. Key pair generation: public/private key pair **2. Digital certificate:** bind public key to identity; signed by trusted third party (TTP) called Certificate Authority (CA) **Certificate revocation:** invalidating a certificate before its expiration.

Digital Certificate

Binds an entity to some public key. Correctness of this binding is asserted via a signature by a certificate authority (CA). A CA is a trusted authority: Trusted Third Party (TTP). CA verifies and issues certificate to the entity requesting for it using its its own public-private key.

1.Alice requests for a certificate for {alice@gg.com, abcdef} from a trusted CA. 2.The CA verifies depending on the type of certificate request. 3.CA places the entry {alice@gg.com, abcdef} into its trusted directory and issues a signed certifcate. The certificate has these a.Identity of the owner, b.Public key of the owner, c.Valid time window of this certificate, d.Identity AND Digital Signature of the Certificate Authority (entity that issued the certificate) computed from the CA’s private key. e. Optional other information based on the intended purpose of the certificate. 2. Alice can now send to Bob her certificate (which has Alice’s public key) via a message signed using her private key (to ensure authenticity, and non-repudiation). 3. ASSUMING bob already trusts that CA (very important), he then verifies that the signature in the certificate is indeed signed by the CA. 4. Since No one except the CA can produce the valid signature, the authenticity of the certificate is as good as coming directly from the CA and there is no need to contact the CA. 5. Bob trusts Alice’s public key

NOTE: People don't need to query the CA for public keys. Certificate is a “smart” work around to get the public key in an offline manner, hence the CA’s server will not be overloaded. **NOTE:** Bob verifies the certificate using the CA’s own trusted public key. We assume the CA’s public keys are already securely distributed. Most browsers have a few pre-installed CAs’ public keys: they are known as the “root” CAs. However, not all CAs’ public keys are preloaded. Other CA’s public keys can be added through the chain-of-trust. **NOTE:** There is a need to check if the certificate has been revoked which is done with the online CRL Distribution point or OCSP Responder.

PKI models

Single CA: One CA responsible for issuing and managing certificate so everyone trusts it. Advantages: 1.Easy to build and maintain, 2.Consistent approach to certificate issuance, 3.Users trust only a single CA. **Disadvantages:** 1.Single point of failure, 2.Does not scale particularly well

Multiple CAs: commonly used in a browser. macOs has about 100 root CAs. Advantages: 1.Will not completely fail even if one CA fails, 2. Geographically distributed. **Disadvantages:** 1.Complexity

Multiple CAs are organized into a hierarchy of trust. Root CAs (trusted) at the top, Intermediate CAs(can be verified) are in the middle. At the bottom of the hierarchy are the entities that are verified by the intermediate CAs. Hierarchy helps to delegate trust and manage the issuance of certificates in a scalable and controlled manner. example hierarchy: root CA -> inter CA -> Entity Alice. Each arrow -> indicates the direction of trust. Root CA signs and trusts int CA and inter CA signs and trusts entity Alice.

Note: root CA are trust anchors and are pre-installed aka self signed. **Note:** usually at certificate Extensons >> Certificate Basic Constraints, you can view whether the entity being trusted is a CA i.e, root and intermediate CA should say it is a CA.

Note: certificates are formated in X.509 format. Current version is verion 3.

The structure of an X.509 v3 digital certificate decided by **Public-Key Infrastructure Working Group (PKIX)** is

- Certificate
Version Number
Serial Number
Signature Algorithm ID (Note Signature Algorithm below too)
Issuer Name
Validity period: Not Before, Not After
Subject Name
Subject Public Key Info: Public Key Algorithm, Subject Public Key
Issuer Unique Identifier (optional)
Subject Unique Identifier (optional)
Extensions (optional)
- Certificate Signature Algorithm
- Certificate Signature

Extensions are to increase flexibility If Extensions marked as critical cannot be processed by an implementation, the certificate must be rejected. Non-critical extensions may be ignored.

Types of certificate:

1. Self-signed certificate: self-generated, free, signed with organisations's own private key. Might not be trusted by another user so he/she must manually accept certificate. Advantage: 1. Fast and easy to issue, 2. Not dependent on others for certificate issuance. Disadvantage 1. Not via trusted CA, if compromised, security risk, 2. Not recommended for public-facing websites. Eg. openssl -generate self-signed certificates

2.Domain Validated (DV) Certificate: least stringent validation. Only verification check: if applicant owns the domain (web site address) associated with certificate. Advantage and disadvantage same as self signed.

3. Extended Validation (EV) Certificate: stringent identity validation and takes time to issue. Additional verification: organization details, registration number and jurisdiction, etc. Advantages: Advantages: 1. Highest level of trust due to rigour identify verification. Disadvantages: expensive and time consuming.

DV vs EV

In certificate subject, if you can see small minor details about the subject, it's a EV else its DV.

Security issues of PKI

CA Single Point of Failure: 1. Security breach of a single CA will compromise security of sites protected by any other CA! 2. Any website can be impersonated using a rogue certificate issued by a rogue CA

Compelled certificates: Government agencies compel (via court order) a CA to issue false certificates for website to be spoofed. Hence govt become MITM and covertly intercept individuals' secure communications.

Typo-squatting: since domain name is similar, it gets issued a certificate.

Subdomain Takeover: An attacker who owns 134566.com can create a deceptive subdomain (nus.edu.sg.134566.com) and obtain a valid SSL certificate for it. This can trick users into believing they are visiting a legitimate site.

Well known incident: TrustWave issues “subordinate root certificate” used for MITM attack.

MITM Attack by Rogue CA

A rogue CA can perform a man-in-the-middle attack by doing proxy re-encryption. The requirement is that the rogue CA’s self-signed certificate is already accepted by the victim.

These are the steps that will happen:

a) First, Mallory intercepts Alice’s ClientHello to Bob.

b) Mallory pretends to be Bob. As a CA, Mallory is able to issue a certificate with Bob’s
details but with Mallory’s own public key, and the digital signature is signed with Mallory’s private key.

c) Alice will accept the information listed in the certificate issued by Mallory, because
Alice has already accepted Mallory’s certificate.

d) Alice and Mallory will establish communication, coming up with session keys k1 to
encrypt communications and t1 for authentication e.g. MAC.

e) Mallory then sends ClientHello to Bob. Bob will send over his certificate, which Mallory will accept, thus coming up with session keys k2 and t2.

f) Mallory now can see all traffic and also modify all data.

a. Alice – ke1 – kd1 – Mallory – ke2 – kd2 → Bob

b. Bob – ke2 – kd2 – Mallory – ke1 – kd1 → Alice

Certificate Revocation: mechanism to invalidate certificates

Certificate Revocation List: CA periodically publishes Certificate Revocation List (CRL) that contains a complete list of all the unexpired, revoked certificates.

Disadvantage: large size, high overhead. **BETTER Delta CRLs** only contain changes from the last complete CRL. **Advantage:** 1.short and easy to publish, 2.Issuing delta CRL periodically removes need to issue full CRL

Short-lived certificate, no need to revoke: Servers update certificates from CAs daily. If unable to update → previous certificate is still valid for next few days. **Advantage:** Risk associated with a compromised certificate is minimized. **Disadvantage:** Too much overhead to CAs and domain owners.

On-line certificate verification: Client query in real timeto get revocation status. 1. client connect to site. 2. site send certificate. 3. client send OCSP to check certificate status. 4. OCSP returns status(good/revoked/valid). Advantages: 1.Timely information regarding the revocation status, 2.Reduced latency since OCSP responses is smaller in size than CRLs. Disadvantages: 1.Traffic overhead to CAs for querying, 2.CA learns user’s activity so privacy is lost.

OCSP stapling is better: 1. OCSP is periodically refreshed within the web server. 2. client conect to web server 3. Web server sends certificate + OCSP to client.

Certificate transparency: complements PKI

Issues with PKI

Forged Certificates: CA might incorrectly issue forged certificates

Delayed Detection: rogue CA could issue fraudulent certificates cannot be immediately detected

Reporting Breaches: Many CA won't report a breach because of losing trust and credibility

Idea of certificate transparency: transparent certificates reveals misbehaviour -> reduce misbehaviours

Certificate log (CL): Public, verifiable, and append-only log of certificates. Server can monitor for mis-issued certificates. Browsers reject if certificate is NOT in one of the CLs: a Client decline to connect to sites whose certificates have not been logged, b.Trust the certificate only if logged

Server can monitor the log for mis-issued certificate.

How it works

1. Domain requests a certificate from the Certificate Authority (CA). 2. CA issues the certificate and submits it to a public Certificate Log. 3. Certificate Log provides proof of inclusion to ensure transparency. 4. CA returns the certificate along with proof of inclusion to the domain. 5. Browser requests a secure connection (e.g., HTTPS). 6. Domain presents the certificate and proof of inclusion to the browser, allowing verification.

How do we build a public, verifiable, and append-only log? One obvious approach is one global log maintained as a list. Each client continuously downloads the entire log and verify. Issue: Huge waste of bandwidth and storage. Better approach: Merkle Tree.

Merkle Tree (Picture below)

Leaves → Hashes of individual certificates in the log. **Nodes** → Hashes of paired child nodes. **Tree Head** (TH) → Root hash summarizing all certificates. **Signed Tree Head (STH)** → A cryptographic signature of the root, publicly distributed for verification.

To show that a particular certificate is in the log, 1. need the hash of that leaf, 2. a list of neighbouring hashes working up the tree.

Proof of inclusion: Eg to verify d3, server shares node hashes of c,i,n with the client. Client can recalculate the root by working up the tree and comparing it with public STH.

Issues

Availability: we need multiple logs in distributed regions to ensure availability of the log. To make these logs consistent, we need to allow them to synchronise.

Latency: time to register a new certificate and get the proof of inclusion

To solve the issues: The certificate log can issue signed certificate timestamp (SCT). When CA sends the certificate to the log server, log server immediately issues a SCT as a promise that it will insert this certificate to its Merkle tree. SCT promises that the actual insertion will take place within a maximum merge delay (MMD) (e.g., 24 hours) CA can use the certificate + SCT from that moment. SCT list can be embedded directly within the X.509 certificate itself as an extension.

1. When a domain requests a certificate, the CA submits it to a Certificate Log.
2. The certificate log issues a Signed Certificate Timestamp (SCT), promising to publish the certificate within 24 hours. 3. The SCT is returned with the certificate and can be embedded inside the X.509 certificate. 4. Browsers verify the SCT to ensure the certificate was transparently logged, preventing unauthorized certificates.

A good Ecosystem

Anyone can monitor the logs to ensure: 1.CAs do not register improper certificates 2. Logs do not misbehave; for example, violating append-only property, presenting different logs to different clients.

Community effort needed: Monitors: continuously observe CT logs for any suspicious or unauthorized certificates, Auditors: verify the integrity and correctness of CT logs.