Cryptography and Its Applications

Content

- Classical Cryptography
- Symmetric key encryption
  - Stream Cipher
  - Block Cipher
- Message Authentication Code
- JWT
- Public Key Cryptography
- Public Key Infrastructure (PKI)

Goals of Cryptography

- The most fundamental problem cryptography addresses: ensure security of communication over insecure medium
- What does secure communication mean?
  - confidentiality (privacy, secrecy)
    - only the intended recipient can see the communication
  - integrity (authenticity)
    - the communication is generated by the alleged sender
- What does insecure medium mean?
  - Two possibilities:
    - Passive attacker: the adversary can eavesdrop
    - Active attacker: the adversary has full control over the communication channel

Approaches to Secure Communication

- Steganography
  - “covered writing”
  - hides the existence of a message
  - depends on secrecy of method

- Cryptography
  - “hidden writing”
  - hide the meaning of a message
  - depends on secrecy of a short key, not method

Basic Terminology

- Plaintext: origin message
- Ciphertext : transformed message
- Key: secret used in transformation
- Encryption/Decryption: forward/reverse transformation
- Cipher: Algorithm of transformation
- Cryptanalysis: science and practice of breaking cryptographic systems/schemes

Shift Cipher

- The Key Space:
  - [0 .. 25]
- Encryption given a key K:
  - each letter in the plaintext P is replaced with the K'th letter following corresponding number (shift right)
- Decryption given K:
  - shift left

History: K = 3, Caesar's cipher

Shift Cipher: Cryptanalysis

- Can an attacker find K?
  - YES: by a bruteforce attack through exhaustive key search,
  - key space is small (<= 26 possible keys).

- Lessons:
  - Cipher key space needs to be large enough.
  - Exhaustive key search can be effective.

Mono-alphabetic Substitution Cipher

- The key space: all permutations of Σ= {A, B, C, …, Z}
- Encryption given a key π:
  - each letter X in the plaintext P is replaced with π(X)
- Decryption given a key π:
  - each letter Y in the cipherext P is replaced with $\pi^{-1}(Y)$

Example:

Σ= 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

π= B A D C Z H W Y G O Q X S V T R N M L K J I P F E U

BECAUSE => AZDBJSZ

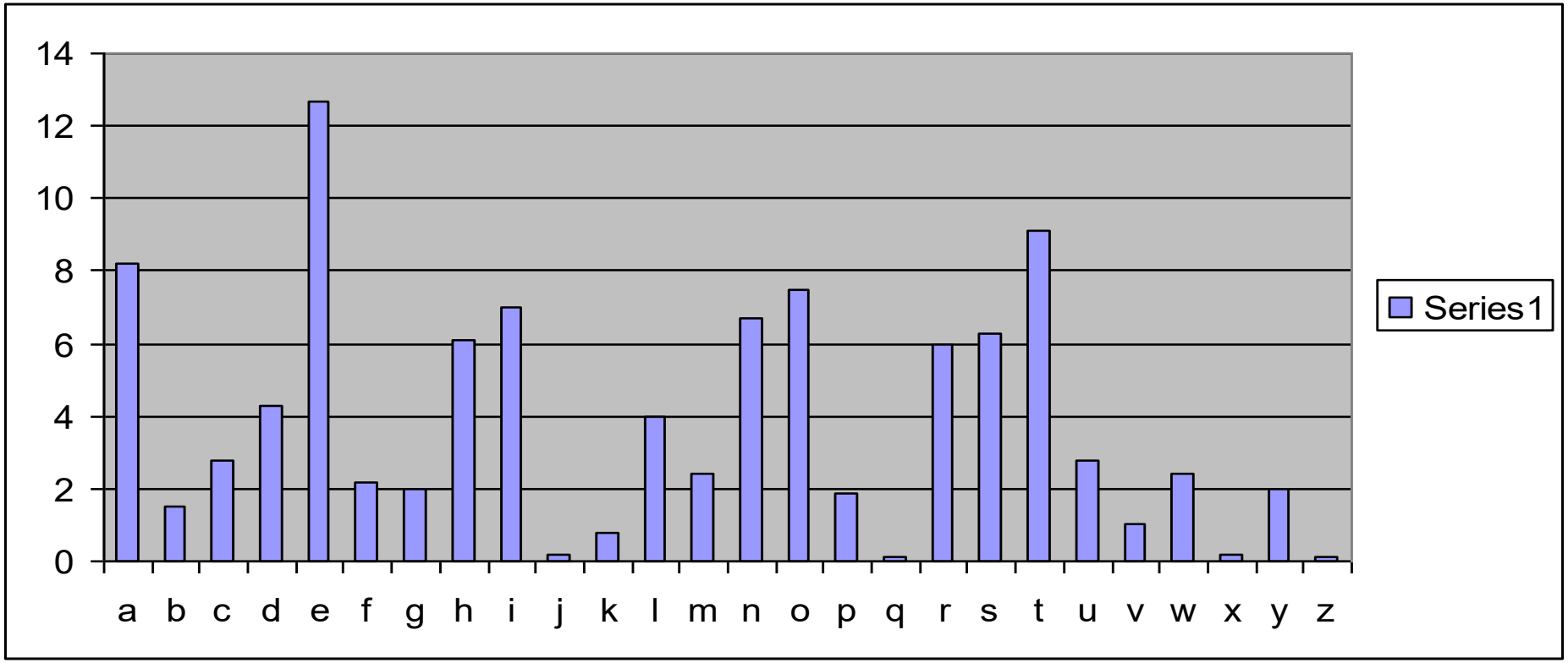
Strength of the Mono-alphabetic Substitution Cipher

- Exhaustive search is difficult
  - key space size is $26! \sim 4 \times 10^{26} \sim 2^{88}$
- Dominates the art of secret writing throughout the first millennium A.D.
- Thought to be unbreakable by many back then
- How to break it?

Cryptanalysis of Substitution Ciphers: Frequency Analysis

- Basic ideas:
  - Each language has certain features: frequency of letters, or of groups of two or more letters.
  - Substitution ciphers preserve the language features.
  - Substitution ciphers are vulnerable to frequency analysis attacks.

Frequency of Letters in English



Adversarial Models for Ciphers

- The language of the plaintext and the nature of the cipher are assumed to be known to the adversary.
- **Ciphertext-only attack**: The adversary knows only a number of ciphertexts.
- **Known-plaintext attack**: The adversary knows some pairs of ciphertext and corresponding plaintext.
- **Chosen-plaintext attack**: The adversary can choose a number of messages and obtain the ciphertexts
- **Chosen-ciphertext attack**: The adversary can choose a number of ciphertexts and obtain the plaintexts.

Security Principles

- **Kerckhoffs's Principle:**
  - A cryptosystem should be secure even if everything about the system, except the key, is public knowledge.
- **Shannon's maxim**:
  - "The enemy knows the system."
- Security by obscurity doesn’t work
- Should assume that the adversary knows the algorithm; the only secret the adversary is assumed to not know is the key
- What is the difference between the algorithm and the key?

Notation for Symmetric-key Encryption

- A symmetric-key encryption scheme is comprised of three algorithms
  - **Gen** the key generation algorithm
    - The algorithm must be probabilistic/randomized
    - Output: a key k
  - **Enc** the encryption algorithm
    - Input: key k, plaintext m
    - Output: ciphertext c := $\mathbf{Enc}_k(m)$
  - **Dec** the decryption algorithm
    - Input: key k, ciphertext c
    - Output: plaintext m := $\mathbf{Dec}_k(m)$

Requirement: $\forall k \, \forall m \;\; [\, \mathbf{Dec}_k(\mathbf{Enc}_k(m)) = \mathrm{m} \,]$

Stream Ciphers

- Plaintext P
- Ciphertext C
- Keystream Ks:
  - Len(Ks) = Len(P)
  - Each P has its own Ks
- Encryption & Decryption:
  - C = P XOR Ks (XOR bit by bit)
  - P = C XOR Ks (XOR bit by bit)
- Why does it work?:
  - Sender and receiver use the same way to generate K for each message.

Stream Ciphers characteristics

- Encrypt/decrypt bit by bit
  - Prevent frequency analysis
- Very fast
- Keystream generation algorithm based on Pseudo Random Number Generation
- Become vulnerable if a keystream is used twice.

Stream Cipher applications and standards

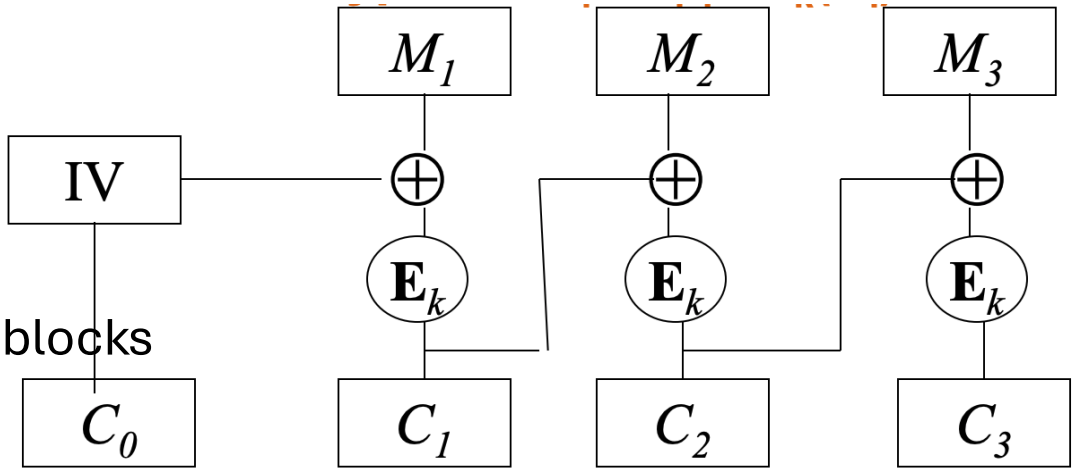
- Online content broadcasting
  - Content scramble
  - DRM (Digital Right Management)
- Typical standards
  - RC4
  - DVD – CCA (DVD Copy Control Association)

Block Ciphers

- Encrypt block-by-block
  - n-bit Plaintext --> n-bit ciphertext
- Common notations:
  - $P : \{0,1\}^n$
  - $C : \{0,1\}^n$
  - $K : \{0,1\}^s$
  - n: block size
  - s: key size
- Prevent frequency analysis

Block Cipher Encryption Modes:

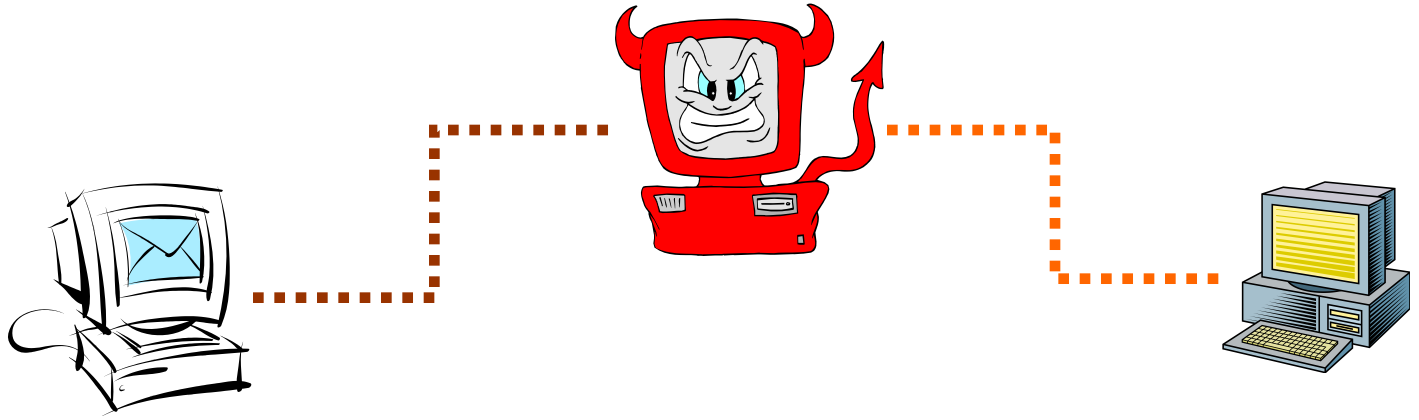
- ECB : Electronic Code Book
  - Each block is encrypted/decrypted separately
  - Ci = E(k, Xi)
  - Xi = D(k, Ci)
  - Not recommended for encrypting more than one block
- CBC: Cipher Block Chaining
  - Random Initial Vector (IV)
  - Next input depends on previous output
  - Randomized:
    - Same input block can have different output blocks



Block Cipher Standards and applications

- DES: Data Encryption Standard
  - Obsolete: 1977 – 2001
  - Block size: 64 bits
  - Key size: 56 bits
  - Vulnerable for brute-force attack (short key size)
  - Designed for hardware
  - Obsolete
- AES: Advanced Encryption Standard
  - Replace DES
  - Block size: 128 bits
  - Key size: 128, 192, 256 bits
  - Applications: Data/Disk encryption, WiFi, VPN, HTTPs

Data Integrity and Source Authentication



- Encryption does not protect data from modification by another party.
  - Why?
- Need a way to ensure that data arrives at destination in its original form as sent by the sender and it is coming from an authenticated source.

Hash Functions

- A hash function maps a message of an arbitrary length to a m-bit output
  - output known as the fingerprint or the message digest

- What is an example of hash functions?
  - Give a hash function that maps Strings to integers in [0,2^{32}-1]

- Cryptographic hash functions are hash functions with additional security requirements

Common Hash Function

- MD5
  - 128-bit output
- SHA1
  - 160-bit output
- SHA2 (SHA-224, SHA-256, SHA-384, SHA-512)
  - 224-,256-,384-, 512-bit outputs

Usages of Cryptographic Hash Functions

- Software integrity
- File integrity
- Message integrity:
  - MAC (HMAC): Message Authentication Code

Message Authentication Code

- Original Message M
- MAC is a code calculated from M (using a secret K)
  - MAC(K, M) = $H_K(M)$
- Data to send : D = M || MAC(K, M)
- Upon received:
  - Verify received MAC(K, M) against M and K
- ==> Message integrity verification
- MAC can be calculated using hash function
  - MAC(K, M) = H(K || M)

HMAC

- HMAC(K, M) = **H(** K || **H(**K || M**))**
  - Two hashes: Inner hash and outer hash
- Why single hash is not enough?
  - With simple MAC, the original message can be added and a new digest can be recalculated without knowing the secret key.
- HMAC Application :
  - Message Authentication
  - JWT

JSON Web Token

- JWT = secure, compact, open standard for information exchange between client and server
- Applications:
  - Authentication: verify "who you are"
  - and Authorization: verify "what you are allowed to do"
- Structure:
  - HEADER . PAYLOAD . SIGNATURE

JSON Web Token (cont.)

- HEADER:
  - Base64 encoded JSON object
  - Contain metadata including "algorithm" and "type"
  - Example:
    - {"alg":"HS256","typ":"JWT"} <-- base64 --> eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
- PAYLOAD:
  - Base64 encoded JSON object
  - Contain "claims" (application data)
  - Registered claims (optional): iat, sub, iss, aud, exp
  - User claims: what ever you want

JSON Web Token (cont.)

- SIGNATURE:
  - HMACSHA256(HEADER . PAYLOAD, SECRET)
  - Can use other signing algorithm (RSA256)
  - It is used for verification

- DEMO:
  - base64 command
  - xxd command
  - openssl command to generate HMAC

Symmetric key cryptography review

- Sender and receiver must share a single secret key
- The key use for:
  - Encryption/Decription : AES, Stream Cipher
  - Message Authentication: HMAC
- Because of the share keys:
  - Secure channels for key distribution
  - Many keys required for multiple parties ($O(n^2)$ keys for n parties)
  - Two parties should have prior relationship (key exchange) --> not convenient

Solution: Public Key Cryptography

- Assume multiple party communications:
- One party (party A) has:
  - Public key (Pub_A): known by others
  - Private key ($Priv_A$): kept in secret (only A knows the key)
- Encryption application: B --- send M ---> A
  - At B: C = encrypt(Pub_A, M)
  - At A: M = decrypt($Priv_A$, C)
- Digital Signature application: A --- send (M || signature) ---> B
  - At A: signature = sign($Priv_A$, M)
  - At B: verify(signature, Pub_A)

Real world application of Public Key Cryptography

- Public key cryptographic algorithms are slow --> Do not apply it on the data
- Combine Public Key encryption and symmetric keys
  - Before communication session: senders send symmetric keys (encryption keys) to receivers using Public Key encryption
  - Use the encryption keys for secured data exchange
- Practical digital signature:
  - Messages are hashed --> hash value
  - Hash value is signed with signing key (public key)

HMAC vs Digital Signature

- HMAC guarantees integrity and authenticity
- Digital Signature guarantees integrity, authenticity and non-repudiation
  - The signer cannot deny that he/she didn't do this.
- Digital Signature is slower than HMAC

Public Key Application : SSH Authentication

- At SSH server :
  - ~/.ssh/authorized_keys: contains public keys of authorized clients/hosts
- SSH client :
  - Use its private for data encryption to communicate with the server

Public Key Infrastructure

- A framework for applying Public Key Cryptography
- It helps exchanging public keys in a trusted way.
  - Publishing "certificates" to a centralized repository
- Digital Certificate: A document containing:
  - Subject identifier
  - Public Key
  - Digital Signature to protect the document.

Certificate related operations

- Issuing a certificate:
  - Done by a CA
  - To bind a public key with an identity
- Storing a certificate:
  - At your own devices (just like persons keeping their ID cards)
- Sharing and verifying certificate:
  - Others can download certificates and verify it with VA to make sure its validity (Trust Model)

OpenSSL

- Implement core cryptographic components of PKI
  - Keypair generation
  - Certificate Signing Requests (CSR)
  - Certificate issuance
  - …

Certificate Issuance procedure

- [Client] Client generate key pair (private-public keys)
  - OpenSSL , Certbot
- [Client] Client generate CSR (Code Signing Request), a text file
  - CSR contains: client information and client's public key
  - OpenSSL , Certbot
- [Client] client uploads CSR to [CA]
  - Manual via a web form / Automatic via ACME client
- [CA] CA validates CSR and issues an SSL Certificate
- [CA] CA sends SSL certificate to client

ACME - SSL certificate issuance protocol

- Certificates issuance procedure can be automatic via ACME protocol
  - Automated Certificate Management Environment
- Most CAs provide ACME interface
  - Let's Encrypt
  - Google Trust Services
  - ZeroSSL
  - GlobalSign
  - Sectigo

Self-signed SSL certificate

- Generate keypair
  - Expected input: private key file location. Sent via command line switch
  - Output:
    - private key file created
    - Private key file contains : both privatekey and publickey
  - E.g: openssl genrsa -out private.key 2048
- Generate CSR:
  - Expected Inputs: client information (CN: your domain name or ip address)
  - E.g: openssl req -new -key private.key -out certificate.csr
- Upload certificate.csr to CA : skipped
- Generate certificate:
  - E.g.: openssl x509 -req -days 365 -in certificate.csr -signkey private.key -out certificate.crt -notes

Using the self-signed certificate

- [Server] Using the certificate to config nginx
  - you need: crt file and private.key file
- [Client] Import the certificate to client's certificate store
  - Make the client trust the certificate without CA assistance.
- Import certificates at client