Table of Contents

Week 1: Security Basics

Three Types of Threats

Threats on the road
- Eavesdropping, tampering, phishing, rogue routing

Threats at home
- Spyware, ransomware, trojan, worm

Threats at guest house
- Leakage from server

CIA Triad

Confidentiality
- Information not exposed to unauthorized parties

Integrity
- Information not modified by unauthorized parties

Availability
- Information can be accessed by authorized parties at proper time

Authentication
- Ability of system to confirm identity of sender

Authorization
- Ability of system to provide correct level of access that a user should have based on credentials

Accounting
- Ability of system to keep track of what users do while logged into the system

Non-repudiation
- Ability of system to ensure sender cannot convincingly deny having sent something

Threat, Vulnerability & Attack

Threat
- Something bad that could happen

Vulnerability
- Weakness in system that could be exploited

Incident
- When vulnerability is exploited to compromise security of systems

Attack
- Action taken by malicious intruder
- Passive adversary
    - Observes, copies messages…
    - No effect to user's experience
- Active adversary
    - Messages are modified
    - Impacts resources

Principle of Easiest Exploitation

An intruder can exploit any vulnerability to launch a penetration or attack

- Only needs to find one vulnerability
- Defender needs to control all possible vulnerabilities

Achieving Security (Control)

Policy

- What we are trying to protect

Mechanism

- How to enforce security policy

Assurance

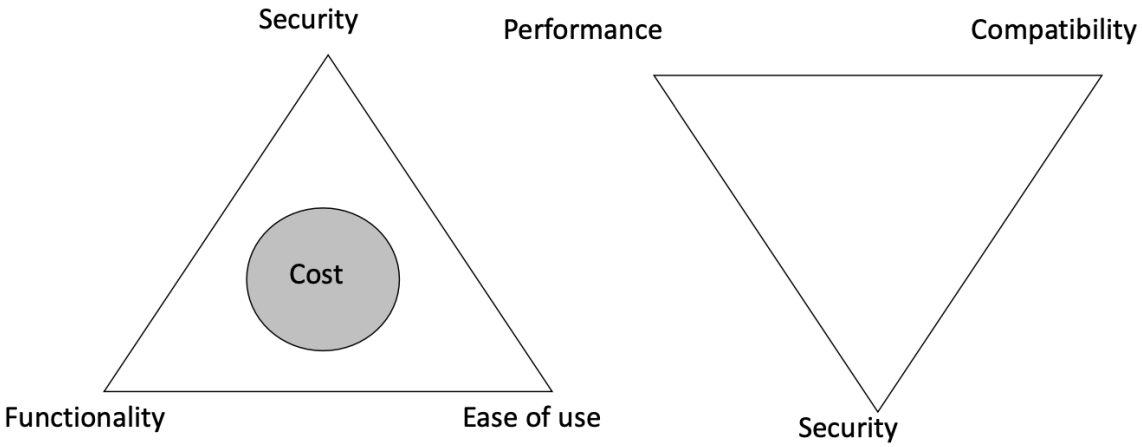
- How well the security mechanism enforces policy

Kerckhoff's Principle (1883)

Only key should be kept secret, while the algorithm itself is publicly known.

Security Tradeoffs

Good-enough security is good enough.



Week 2: Symmetric Key Encryption

Rationale of Encryption

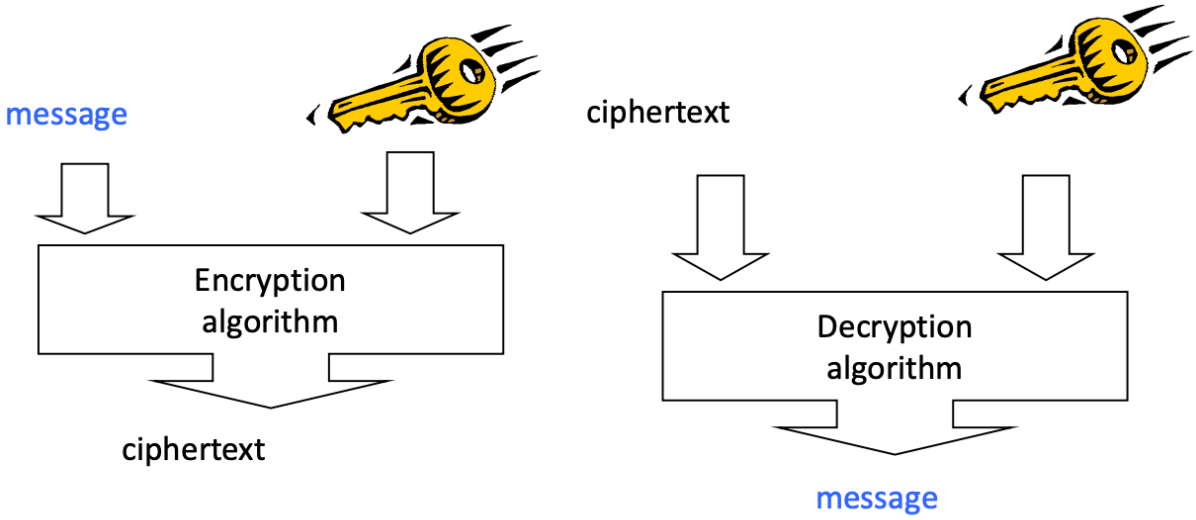
Semantics: meaning
Representation: what it looks like

If adversary cannot find patterns, they have a hard time extracting semantics from representation.

Goal: To make ciphertext without any patterns recognizable to the adversary
** Ideally, ciphertext is random with no pattern.

Encryption / Decryption

Message, Key -> Enc() -> ciphertext
Ciphertext, Key -> Dec() -> plaintext



Vernam Cipher (One-Time Pad)

Plaintext XOR random bitstring (same length as plaintext) -> Ciphertext
Ciphertext XOR random bitstring (same one used to encrypt) -> Plaintext

- Every encryption uses a new freshly chosen key

Is perfectly secure as it is unbreakable even with infinite amount of computational power, but is impractical due to need for synchronization and need for unlimited number of keys.

Block Ciphers

Block = fixed number of consecutive bits in message

- Encryption performed on each block rather than whole message
- Cipher algorithm and key jointly define mapping between plaintext blocks and ciphertext blocks

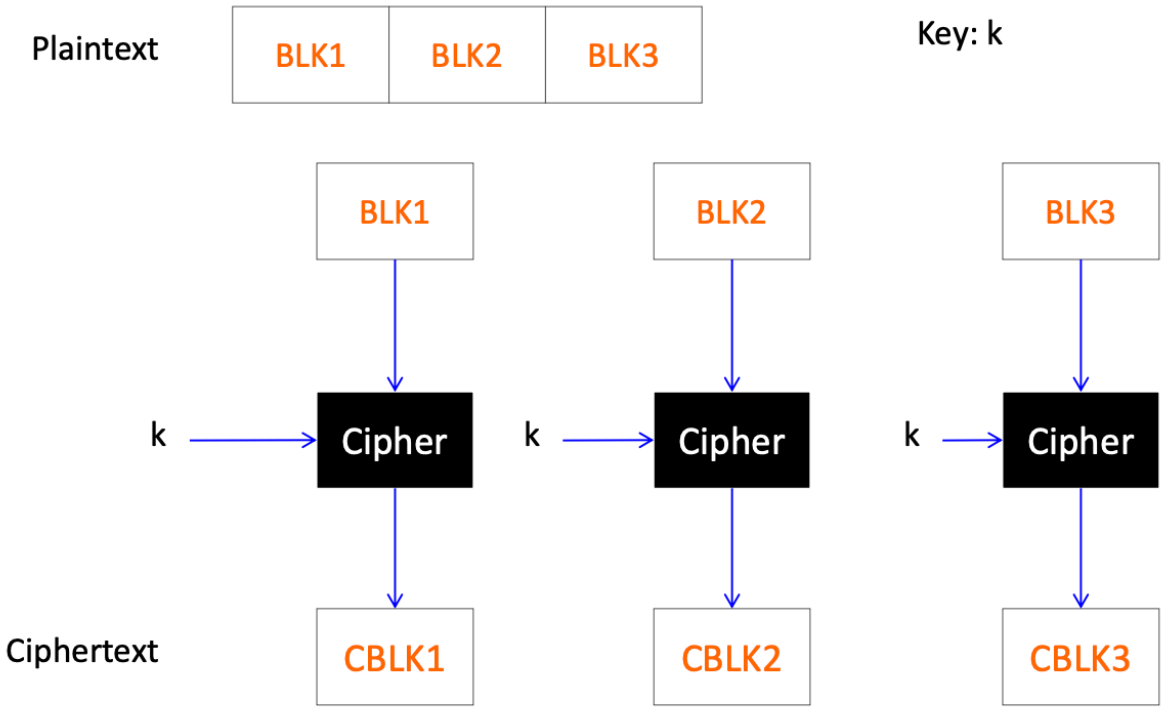
Advanced Encryption Standard (AES)

Key size: 128, 192, 256 bits
Block size: 128 bits

- Uses block cipher to encrypt message consisting of multiple blocks

Three modes
- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)
- Counter (CTR)

Electronic Code Book
- Each block is passed into Cipher with key k to produce cipher block
- Cons: Blocks with same content will produce same cipher block

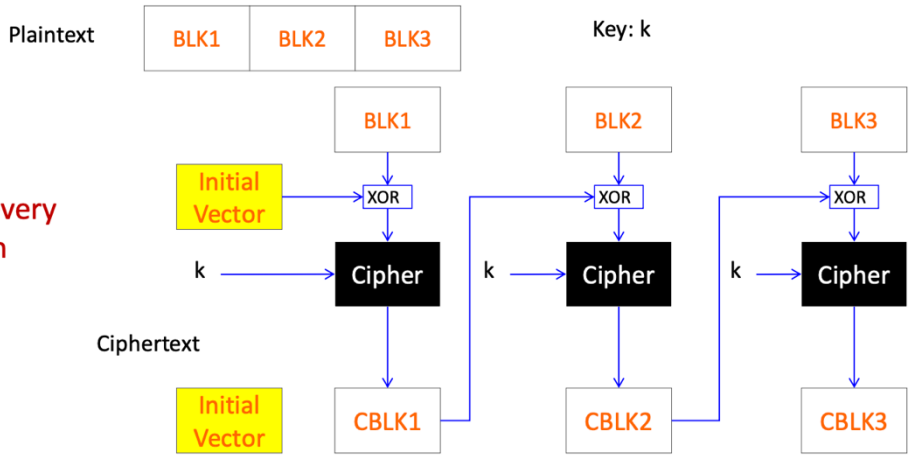


Cipher Block Chaining
- Introduces random block as Initial Vector (IV) of same size as block
- IV is first block in ciphertext
- IV is to introduce randomness into encryption process

Generated cipherblock is used to jumble contents of next block by doing XOR

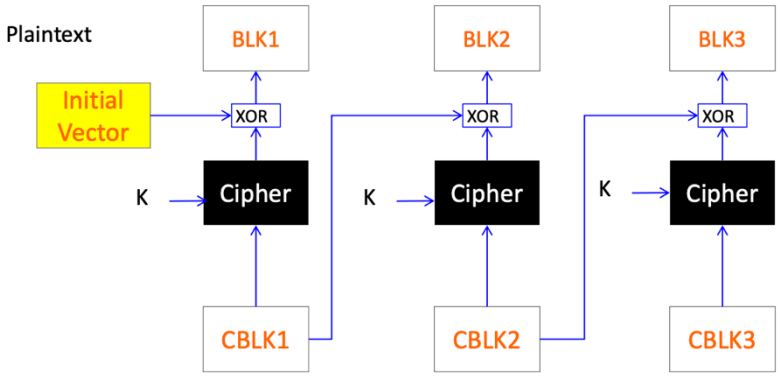
Encryption: $CBLK_i = Enc_k(BLK_i \oplus CBLK_{I-1}), CBLK_0 = IV$

Initial Vector (IV) is freshly chosen for every plaintext encryption



Decryption: $BLK_i = Dec_k(CBLK_i) \oplus CBLK_{I-1}, CBLK_0 = IV$

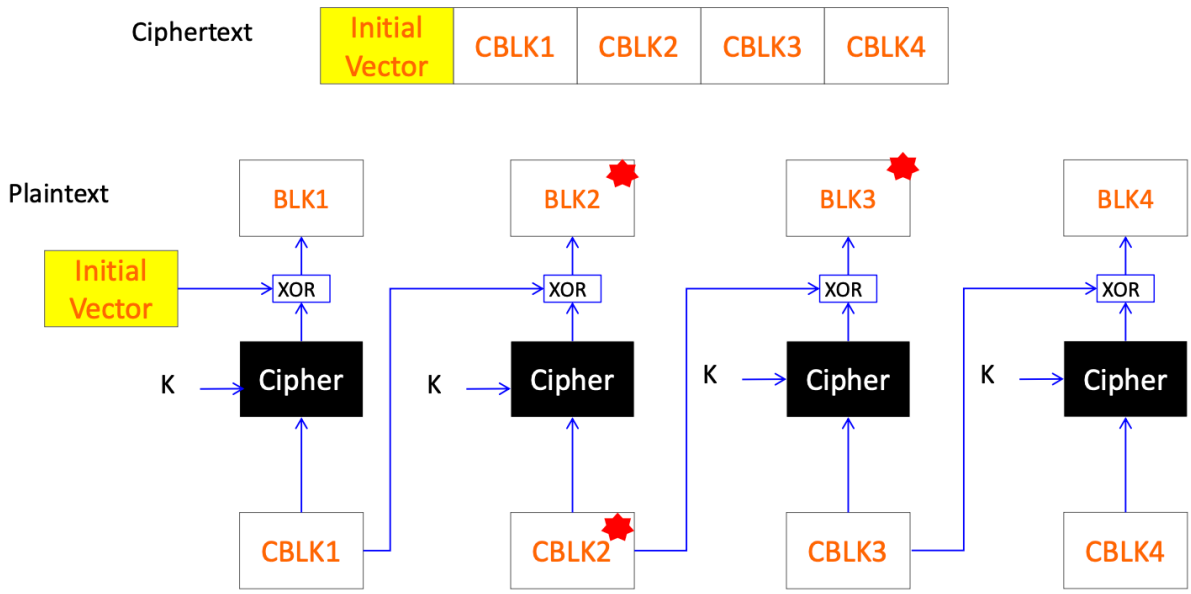
CBC ciphertext includes the IV used in encryption



** Encryption must be done sequentially. Decryption can be parallel.

Error Propagation

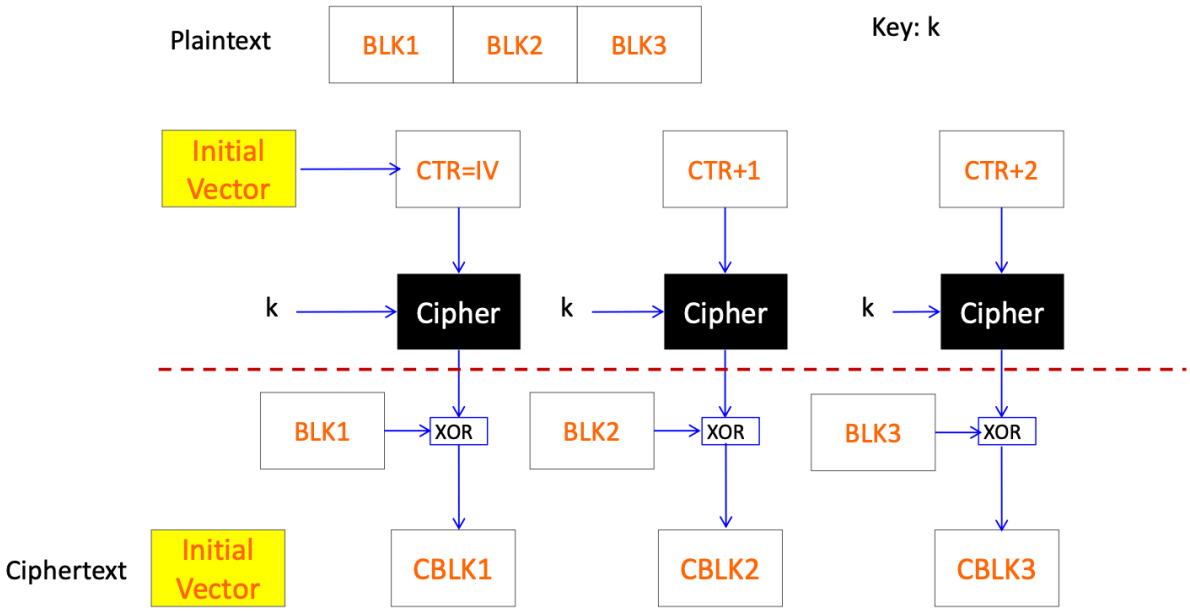
- If the ciphertext has been modified, then only the block affected and the corresponding block will be affected.



Counter

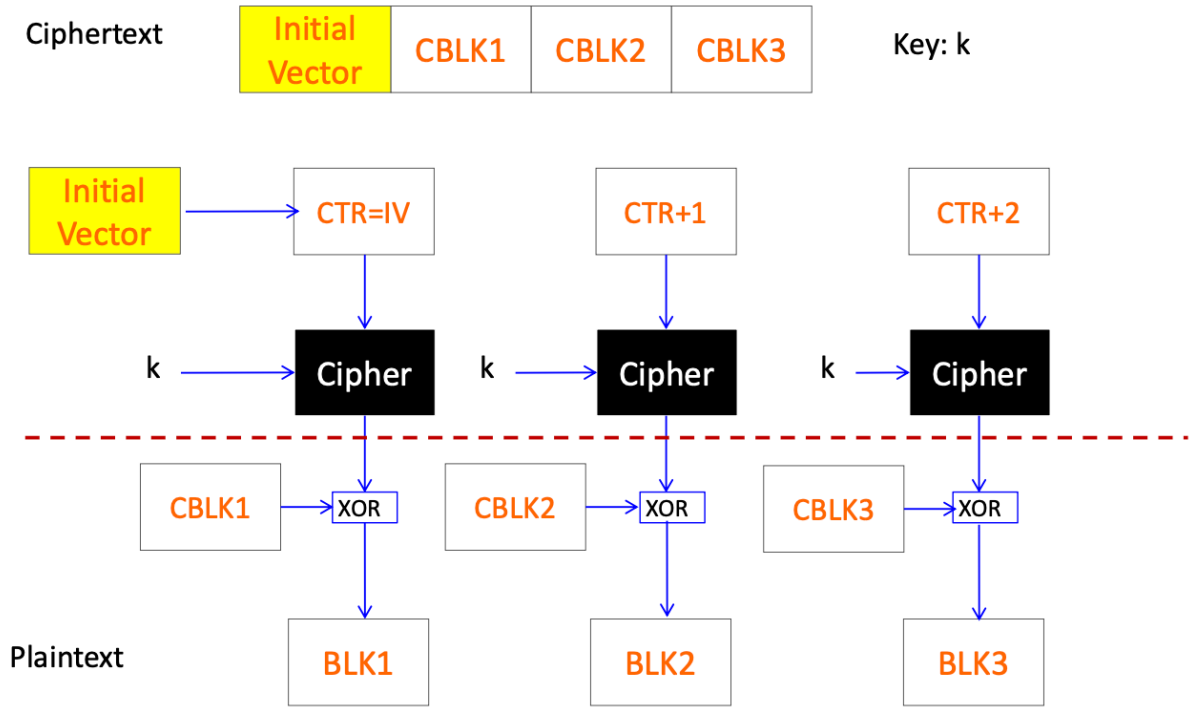
- Key and IV are passed into cipher instead of block and key
- Output from cipher is used to XOR against block
- Initial vector is incremented for each block

Encryption



Decryption

- Same as encryption except cipherblock is used to XOR against output of the cipher to get block.



CTR mode is able to utilise stream ciphers instead.

- Stream cipher is pseudo-random bit sequence produced by pseudo-random bit generator
- A key stream is generated using IV and key, before knowing message
- Avalanche effect can guarantee generated key stream is pseudo-random

** Avalanche effect: Changing a single bit in either key or plaintext results in about half of output bits being different

Comparison of three modes

Identical plaintext blocks result in…

ECB: Identical ciphertext blocks
CBC: Different ciphertext blocks
CTR: Different ciphertext blocks

Chain dependency

ECB: Blocks are enciphered independently
CBC: Proper encryption / decryption requires correct preceding block / cipherblock
CTR: Blocks are enciphered independently (with incrementing IV)

Error Propagation

ECB: None
CBC: Cipherblock's error affects decryption of itself and next block
CTR: None

Cryptanalysis

Attacker…

- Knows encryption and decryption algorithm
- Given a ciphertext, objective is to find plaintext

Ciphertext only	Cryptanalyst only knows ciphertext
Known plaintext	Cryptanalyst happens to know some plaintext-ciphertext pairs (He cannot choose what to know.)
Chosen plaintext	Cryptanalyst knows some plaintext-ciphertext pairs for plaintext chosen by himself
Chosen ciphertext	Cryptanalyst knows some plaintext-ciphertext pairs for selected ciphertext chosen by himself

Week 3: Public Key Encryption

Challenges of Secret Key Distribution

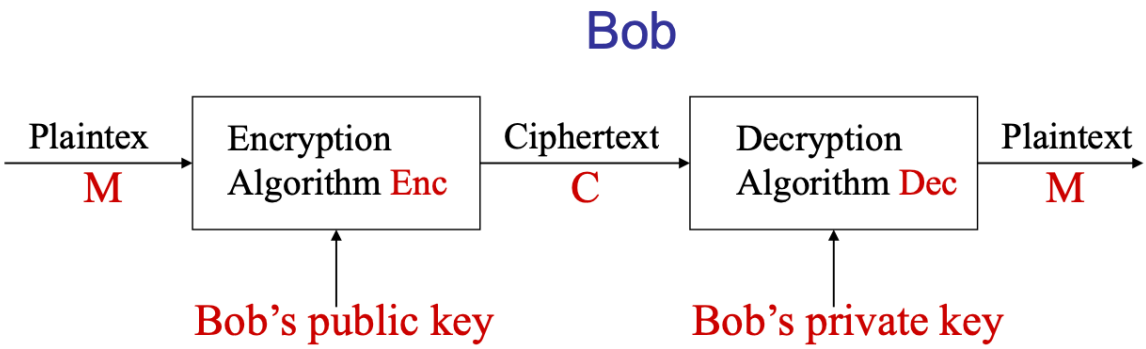
How do I send the key to my recipients securely?
To achieve pair-wise secure communication among N users, you need…
- N(N-1) / 2 secret keys in total
- N-1 secret keys stored by each user

Problem:
- Requires secure key distribution channel
- Is not scalable

Asymmetric Cryptosystem

Uses public key and private key.

Recipient's public key is accessible to everyone.
Recipient's private key is only known to himself.
Requirement: Computationally infeasible to derive private key from public key.



Rivest-Shamir-Adleman (RSA)

RSA Setting
3 parameters
- n: composite integer where n = pq, where p and q are large primes
- e: integer coprime to (p-1)(q-1)
- d: integer satisfying e * d ≡ 1 mod (p-1)(q-1)

- Public key: (n, e)
- Private key: d

How to use RSA
Encryption is only performed on integers smaller than RSA modulus (n)
- Message is converted to number 0 ≤ M < n
- Encryption: $M^e \ mod \ n$ as C
- Decryption: $C^d \ mod \ n$ as M

Computational Security of RSA
Depends on whether attacker can factor n…
- Commonly agreed that when n is large enough and n is in the form of pq, then it is infeasible to factor n in practice.

Key size: Common to choose n to be 2048 bits
- Security depends on difficulty of solving factorization problem to get p and q from n

Limitations of RSA
Trouble encrypting large files
- File to integer may be bigger than RSA modulus and cannot be encrypted directly

Is insecure
- Idk. Textbook say one

Symmetric vs Asymmetric Encryption

Symmetric Encryption	Asymmetric Encryption
Secret key must be distributed over secure channel	Public key can be sent over public channel
Not scalable for multi-party communication	Scalable for multi-party communication
Relatively short keys (128 bits)	Long keys (2048 bits)
Fast encryption speed	Slow encryption speed

Hybrid Approach (Envelope Encryption)
- Uses asymmetric key encryption to deliver symmetric key to other parties in communication

Encryption
- Select random symmetric key K
- Use K to encrypt message M to get ciphertext C1
- Use recipient's public key to encrypt K to get ciphertext C2
- Send (C1, C2) to recipient

Decryption
- Use recipient's private key to decrypt C2 to get symmetric key K
- Use K to decrypt C1 to get M

Significance of RSA
- Solves challenging problem of secure key distribution in symmetric key setting
- RSA cipher rarely used for data encryption
- Mainly used to establish secure channel without a shared secret
    - Can be used to distribute secrets

Week 4: Integrity

Data Integrity

How do I determine whether the data has been altered by adversary?

Is independent of data confidentiality.

Important conditions to meet:
- Only sender can generate tag
- Tag is generated based on message. If message is changed, then binding is broken.

Data Authentication

- Symmetric
    - Message Authentication Code (MAC)
    - HMAC
- Asymmetric
    - Digital Signatures
    - RSA Signatures

Hash Function and HMAC

Hash Function

Maps data of arbitrary size to a fixed-length binary string

Requirements for secure hash function
- One-way computation
    - Easy to compute, hard to inverse
    - Pre-image attack: Find M given h
- Collision resistant
    - Difficult to find different messages with same hash output
    - Collision attack: Find different messages with same hash output
- Fixed-length output
    - Variable-length M -> fixed-length h

Hash functions are not secure by themselves
- Hash functions are publicly known
- If adversary modifies message, then they can simply pass new message to hash function to regenerate a tag that is valid
- Requires secret component to be secure (HMAC)

HMAC (Keyed hash function)
- Shared secret key K
- Sender generates tag H(K || M)
- Sender sends message (M, H(K || M))
- Recipient receives (M, H(K || M))
- Recipient verifies integrity by generating own version of HMAC tag using H(K || M)
- Verification successful if tags match

** Adversary without secret key cannot manipulate HMAC

- MAC provides both integrity and authentication service
    - Integrity because HMAC can only be generated by parties with secret key, cannot be replicated adversary without secret key
    - Authentication because can ensure data is indeed from someone with secret key
- Does not provide confidentiality and non-repudiation
    - Confidentiality because does not involve encrypting contents of message
    - Non-repudiation because anybody with secret key can generate the tag, thus cannot prove who amongst those with the secret key sent the message

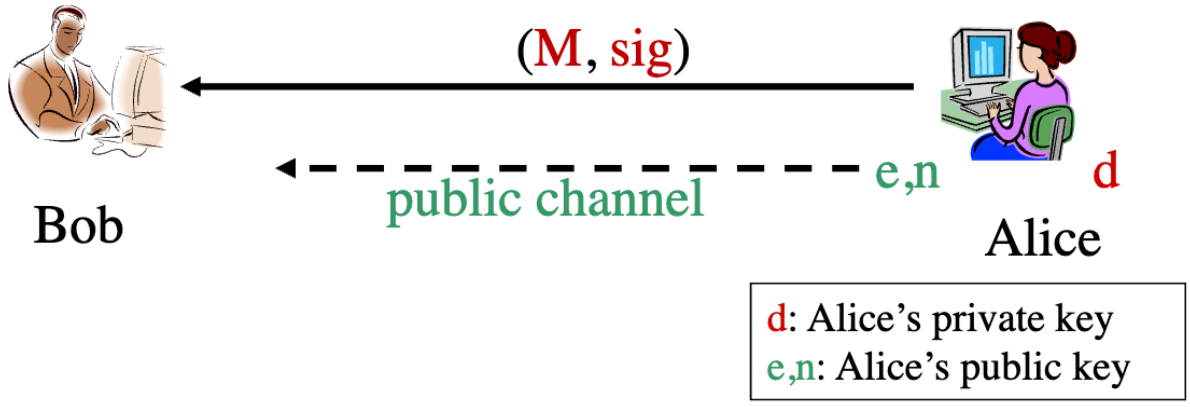
Digital Signature

- Uses asymmetric key cryptography
- Provides authentication, integrity and non-repudiation service
    - Authentication and integrity for the same reason as HMAC
    - Non-repudiation because signature can only be generated by signer with their own private key (only they have it)

Secretly Sign: Only private key holder can sign messages on their behalf
Publicly Verifiable: Anyone can verify signature using signer's public key

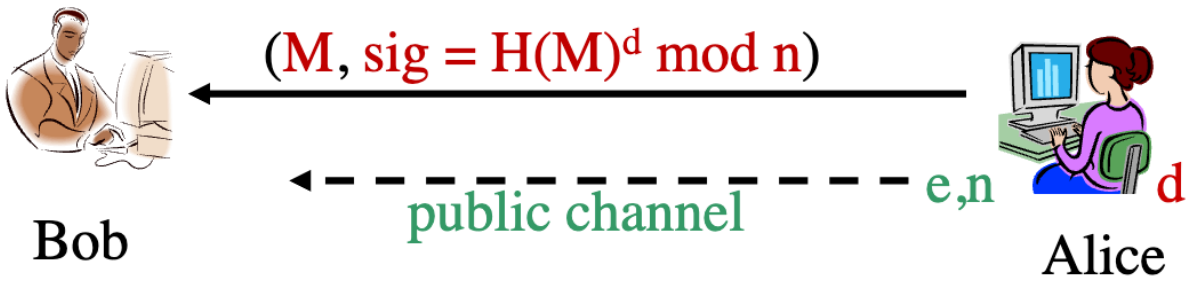
Signing

- Sender computes digital signature SIG on her message M
- $SIG = H(M)^d \ mod \ n$
  **SIG for M can only be generated by sender with private key
- Sender sends (M, SIG) through public channel to recipient



Verification

- Recipient receives (M, SIG)
- Recipient computes h = H(M)
- Recipient computes $h' = sig^e \bmod n$
- Verifies that $h = h'$, pass if true, else fails



RSA Signatures vs Encryption

	RSA Signatures	RSA Encryption
Objective	Data integrity	Data confidentiality
Requirement of private key for the sender	Yes	No
Requirement of private key for the recipient	No	Yes
Limitation of data size	No limitation	Must be smaller than RSA modulus

RSA Signatures vs HMAC

RSA Signatures	HMAC
Data integrity	Data integrity
Non-repudiation: Sender cannot deny	Sender can deny
Publicly verifiable: Anyone can verify	Only sender and intended recipient can verify because they share secret key
Slow due to big number computation	Fast

Week 5: Certificates

Challenges

Availability Problem

- Hybrid encryption: what if attacker replaces sender's public key with their own?
    - Recipient cannot decrypt message because message was not encrypted with recipient's public key, but with attacker's public key

Authenticity Problem

- Digital signatures: what if attacker replaces sender's public key with their own?
    - Public unavailable to correctly verify sender's signature because signature was generated using sender's private key but attempted to be verified with attacker's public key instead of sender's public key.

Public Key Certificate and Infrastructure

Public Key Certificate

- Digital token issued by trusted authority to certify owner of public key

Public Key Infrastructure

- Set of policies, procedures, and products used to implement public key cryptosystem in large setting
- Core Idea: Certificate Authority (CA) issues certificates to public key holders
- X.509 Standard: Widely used standard for defining and managing digital certificates

Public Key Certificate Contents

Mandatory components

- Subject identifier
    - Public key owner's identity
- Issuer's identifier
    - CA's identity
- Public key information
    - Certified public key (n, e)
- Validity period
    - How long it lasts for
- CA's signature
    - Show genuineness of certificate
    - Only computed upon certificate data is ready

Certificate Request

By public key owner and CA

- Let M be the message consisting of n and e
- Sender signs M to get signature SIG
- Sender sends (M, SIG) to CA as certificate request
- CA extracts (n, e) from M
- CA verifies (M, SIG) using (n, e)
- If (M, SIG) is valid, then request is approved
    - CA is convinced sender knows d

Certificate Issuance

By CA

- CA checks sender's identity and verifies sender's request using public key being signed
- If both are correct, CA issues certificate for sender's public key in the following steps
    - Construct certificate body consisting of mandatory components
    - Sign certificate body with CA's private key
    - Sender's certificate = certificate body + CA's signature

Certificate Verification

By verifier

- User obtains web server's certificate either from server or retrieving from some directory
- To verify someone's certificate, verifier…
    - Check if certificate is expired. If expired, certificate is invalid. Else,
    - Verify signature of the certificate with certificate body using CA's public key. If not verified, certificate is invalid. Else,
    - Verify whether CA's public key in use is trusted. If trusted, certificate is valid. Else, certificate is invalid.

Certification Chain

Subject's certificate is signed by Subject's CA…
Subject's CA is signed by Subject's CA's CA…
Subject's CA's CA is signed by Subject's CA's CA's CA…
Eventually will reach root CA.

Root CA will self-certify itself by signing its own public key using its private key. Root CA's certificate is called self-signed certificate.

To verify original subject's certificate, verifier must go through everybody in the chain. When they reach the root CA, verifier decides whether root CA is trustworthy.

How to establish trust?

- Two approaches for true establishment
- **Preloaded root CA certificates** which are widely considered as trusted, such as Microsoft, Google's root CA certificates.
- **Manually instruct browser** to trust root CA's certificate

Certificate Revocation

By CA

- What if private key is compromised?
    - Ideally, corresponding private key should not be in used anymore
- Authority periodically publishes Certification Revocation List (CRL)
    - CRL contains revoked certificates' serial numbers
- User's responsibility to download latest CRL

Certificate Verification with CRL

- Check if expired. If not,

- Verify signature on certificate using CA's public key. If verified,
- Verify entire certification chain. If OK,
- Check whether subject's certicate is in CRL

** Not widely used in practice because of implementation issues.

Week 6: Authentication

Authentication in Cyberspace

- Difficult / costly to manage credentials
- User authentication based on shared secret

Weak authentication: Client proves identity to server by revealing knowledge of shared secret

Strong authentication: Client does not reveal shared secret

Three forms of shared secret
- What you know
    - Password, PIN…
- What you have
    - Token, physical key, passport…
- What you are
    - Biometrics

Weak Authentication

Password-based authentication
- Most popular user authentication technique
    - Low cost, user-friendly, easy to implement
- Not very secure

Assumes communication is confidential. If server is compromised, user passwords would be revealed.

Password File Stored in Server
Server maintains password file which contains information regarding each user account and corresponding password.

To safely store passwords, server chooses arbitrary message M and creates password file. To register new user, server inserts new record to password file
Record
- User id
- Encrypted M with user's password

To verify
- Server encrypts M using submitted password
- Server compares actual value vs expected value

In practice, hash functions are more commonly used instead of encryption.
- More logical since there is no need to decrypt the message stored in password file

Advantage: Server does not store any secret!

Attack on Passwords

Whether to involve server

- Online attack (server is involved)
- Offline attack (server is not involved)

Based on how to guess

- Brute force attack (guess randomly)
- Dictionary attack (guess from dictionary)

Online/Offline Attacks

Online attacks can be easily countered

- Choosing long passwords
- Limiting number of tries
- Slowing down log-in process

Offline attacks are problematic

- Adversary will…
    - Obtain encrypted password file
    - Tries every possible password and encrypts them
    - Compares ciphertext with each tried password to all passwords in password file
- Does not involve authentication server

Brute Force Attacks

- Assumes passwords are randomly chosen
- Tries every single possible passwords
- Is extremely ineffective

In reality, passwords are not randomly selected.

- User id?
- User's name, birthday, phone number, etc
- Common word list or patterns
- Can be guessed by social engineering
- Thus, a more structured attack would be more efficient… like dictionary attacks

Dictionary Attacks

- Adversary will…
    - Create a dictionary of commonly used passwords
    - Pre-computes password file of the dictionary
    - Compares pre-computed password file with real password file
- Fast, but only works for weak passwords

How to Combat This?

Salting

- Extra data value unique for each value called the salt
- Prevents dictionary attacks as common words as pre-computed password file would not have matching encryptions / hashes despite same password

Choose high-quality password

- Use a pass-phrase that is easy to remember and appears random
- Eg: I'll **m**ake **h**im **a**n **o**ffer **h**e **c**annot **r**efuse.
- Password: mhaOhcR139 (Seems random, but is not in reality)

Drawbacks of Weak Authentication

- User has to reveal secret
    - Implies requirement for secure communication channel
- Vulnerable to replay attacks
    - Replay attack: Attacker can resend whatever user has sent in prior authentication sessions without knowing actual data

Strong Authentication

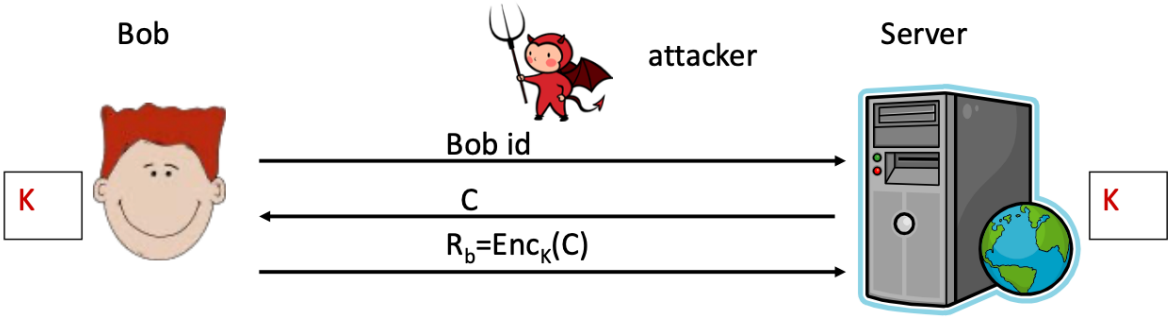
Shared key-based authentication

- User does not reveal secret to server
- Objectives
    - Defeat replay attacks
    - Remove need for encrypted channel
    - Mitigates brute force and dictionary attacks because user uses key instead of password
- Two methods
    - Challenge-response protocol
    - Time-stamp protocols

Challenge-Response Protocol

Server and client shares secret key K

- Upon authentication request, server sends client a random challenge C
    - ** Protocol is not secure if C is not random!
- Client responds with response (either encrypted or hashed with key)
- Server computes response the same way client does and performs equality check for authentication

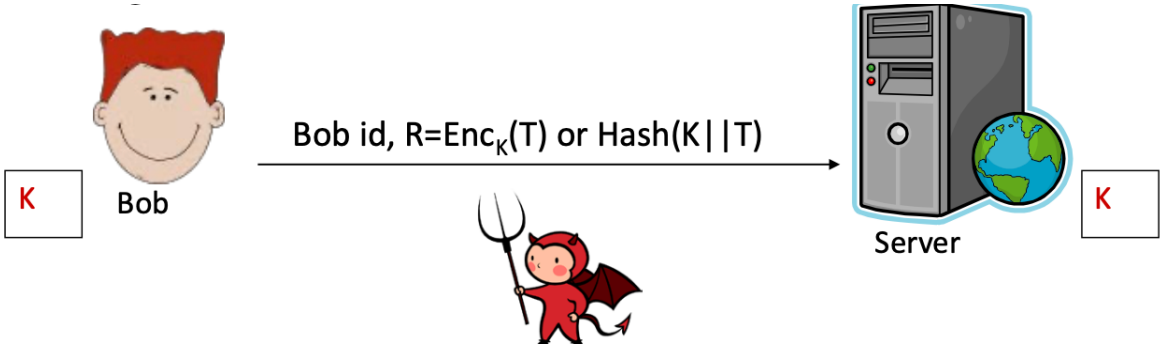


Time-Stamp Protocol

Server and client keep synchronized clocks and shared key K

- Time is used as one-time challenge

- Client sends server their ID, and encrypted / hashed with key clock reading T
- Server verifies whether response using client's key K and server's own clock reading T
- Has a time window, since message's time stamp will not match server's current time stamp, since the message takes time to transmit from client to server



Challenge-Response vs Time-Stamp

Challenge-Response

- Requires 3 message flows
- Need to generate random number

Time-Stamp

- Requires 1 message flow
- Time synchronization is difficult
- Time window for attacks
    - Eavesdropper can replay response within specified time window…
    - Can be remedied by blacklisting already-used responses

Challenges of User Authentication

- Not all service used by user are managed by the same party
- User needs to create an account for each service
- Difficult to do authentication for a lot of servers

Solution: Authenticate to dedicated authentication server

- Authentication server issues credentials that can be presented as proof of authentication…
- Thus, Single Sign On (SSO)

Single Sign On (SSO)

SSO maintains identities and authentication codes for all independent applications. When user wants to access an application, SSO Trusted Third Party (TTP) authenticates on behalf of the user.

Terminologies: Asserting Party
System or administrative domain that asserts information about subject
- Asserts that user has been authenticated and has been given associated attribute

Terminologies: Relying Party
System or administrative domain that relies on information supplied to it by asserting party
- Up to relying party whether or not to trust assertions provided to it
    - Trust: Assertion really comes from asserting party
    - Local access policy defines whether subject may access local resources
- Pre-established trust between relying parties and asserting parties

How it Works
- User wants to access resource from relying party
- User is authenticated by asserting party
- If successfully authenticated, asserting party sends artifact to user
    - Artifact: Base64 encoded string consisting of unique identity of source site and unique reference to assertion
- User requests from relying party with artifact in query
- Relying party sends request with artifact to asserting party
- Asserting party responds with assertions about user
- Relying party determines if user is allowed to access what they want to access

Communications between asserting party and relying party use Security Assertion Markup Language.