CE4024 Lecture Notes – Part 2

# Security of Networked Systems

* Networks are prone to wiretapping and tampering
  + **Wiretapping**: attacker tries to get data being sent over a network
    - To prevent, use encryption
  + **Tampering**: attacker tries to modify data being sent over a network
    - To prevent, use digital signatures

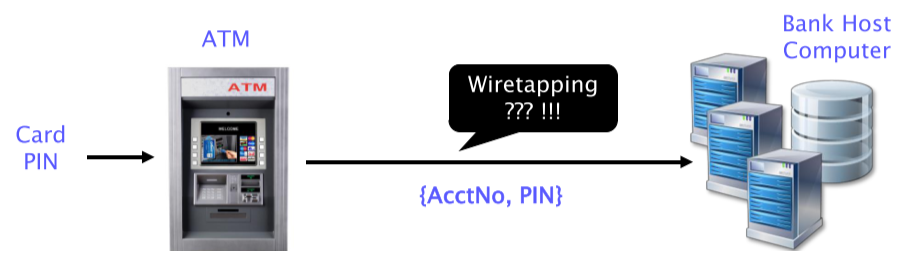
## Data and System Security Services

* Security requirements (what needs to be done):
  + Ensure messages aren’t modified maliciously
  + Ensure no unauthorized access of information
  + Ensure each party is identified correctly
* Security services (how to achieve security):
  + **Confidentiality** – private data is kept private
  + **Integrity** – data is unmodified/uncorrupted
  + **Authenticity** – verify data comes from correct source
  + **Non-repudiation** – proof integrity/origin of data
  + **Accountability** – hold parties accountable for their actions

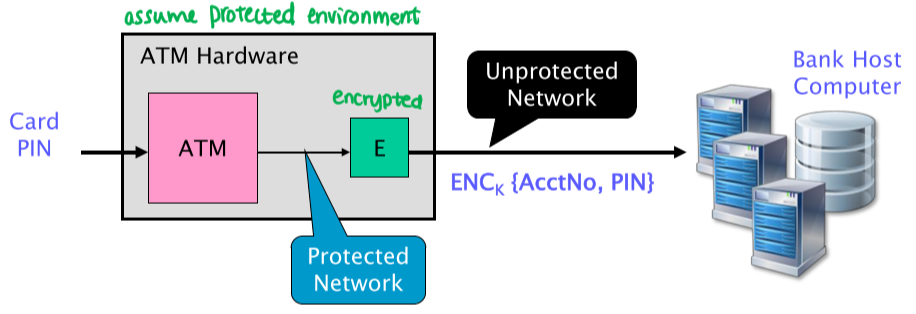
## Example: Bank ATM Terminals

### Confidentiality

* Banking system PIN must be kept confidential
  + ATM is connected to bank’s host computer through network
  + ATM uses PIN to authenticate customers
  + Account number/PIN information sent over network is prone to **wiretapping** – key storage/security?



* For security, account number and PIN are **encrypted** as close to the ATM as possible
  + Only encrypted data is sent into the public network



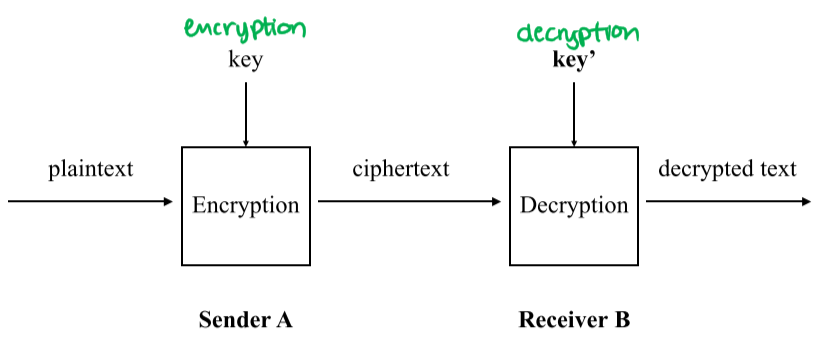
### Integrity and Authenticity

* The integrity and authenticity of transaction instructions on the ATM should be maintained
  + Customer PIN is verified by bank’s host computer
  + Bank’s host computer transmits payment instructions (e.g. withdrawals) to the ATM over a network – prone to **tampering**
  + ATM dispenses money upon receiving host computer’s instructions

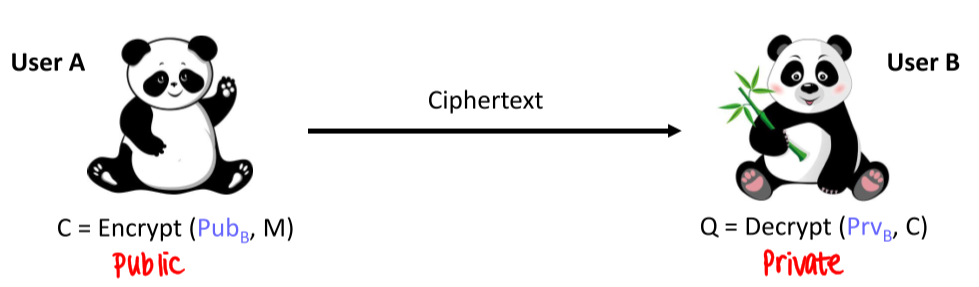


* To protect against tampering, site uses certificates (certified by **trusted third party**) to authenticate connection to bank

# Asymmetric Cryptography



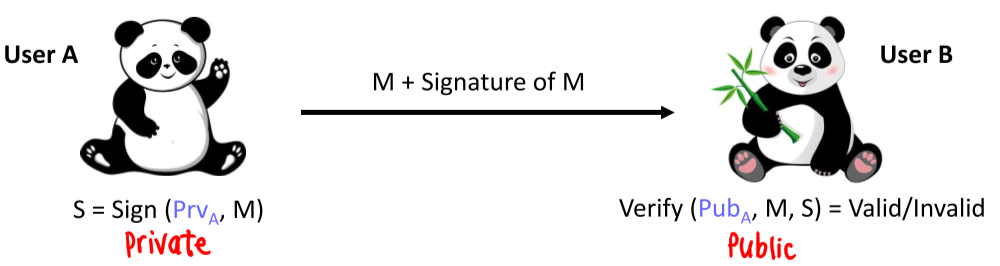
* In asymmetric cryptography (e.g. RSA), the sender and receiver use **different keys**
* **Confidentiality** depends on the secrecy of the decryption key
  + Encryption and decryption use different keys
    - **Encryption**: public key
    - **Decryption**: private key
  + Based on the assumption that it is computationally infeasible to derive the decryption key from the encryption key – ensured by difficulty of integer factorization
  + This removes the secrecy requirement of encryption keys



* Asymmetric ciphers **simplify key management**
  + Easier to distribute public keys
  + A single key can support secret communication with many parties – don’t need to agree on secret keys separately with each party
  + Asymmetric knowledge can be used to provide non-repudiation through **digital signatures**

## Digital Signatures

* **Sign**: private key
* **Verification**: public key

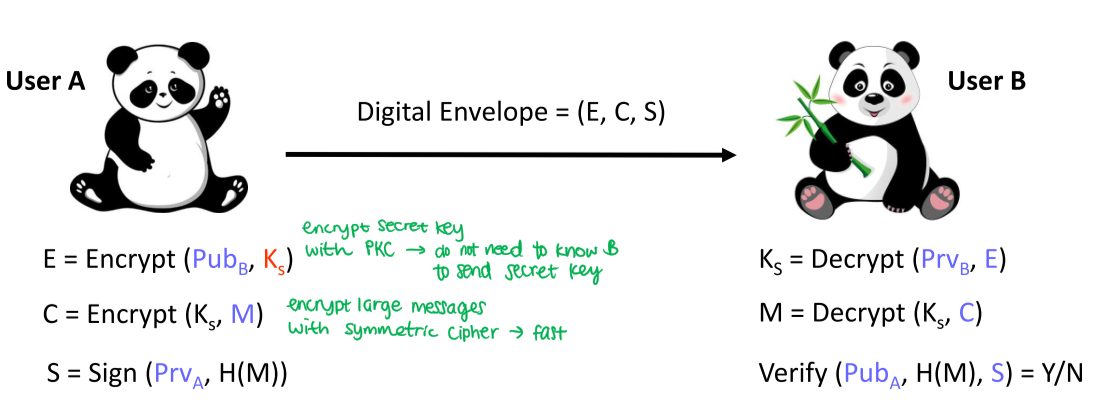


* Signature has following requirements:
  + Cannot be constant, or it can be forged (e.g. copy/pasted)
  + Must be a **function** of the entire document – if any part of the document changes, the signature becomes invalid
* Practical issues:
  + Public Key Cryptography (PKC) is relatively slow – big integer exponentiation is slow compared to bitwise operations
  + Signature must be a function of the entire document
    - Cannot break the document into smaller blocks and sign on the blocks, because an attacker can substitute or shuffle blocks, disrupting data
    - It isn’t desirable to apply a digital signature to a full, long message
    - Instead, use **cryptographic hash function** that “summarizes” document, and sign on the hash
      * If any part of the document is modified, the hash changes

### Practical Aspects of PKC

* Based on hard mathematical problems:
  + Discrete logarithms
  + Integer factorization
* Two major practical problems:
  + Security
    - Hard problems use worst case complexity – computationally infeasible means based off of current computing power
      * Hard problems can also have easy instances, which can lead to insecure implementations
      * Thus, we care about **key generation** in addition to key management
        + Cannot generate keys that can be easily factorized
  + Efficiency

### Practical Implementation



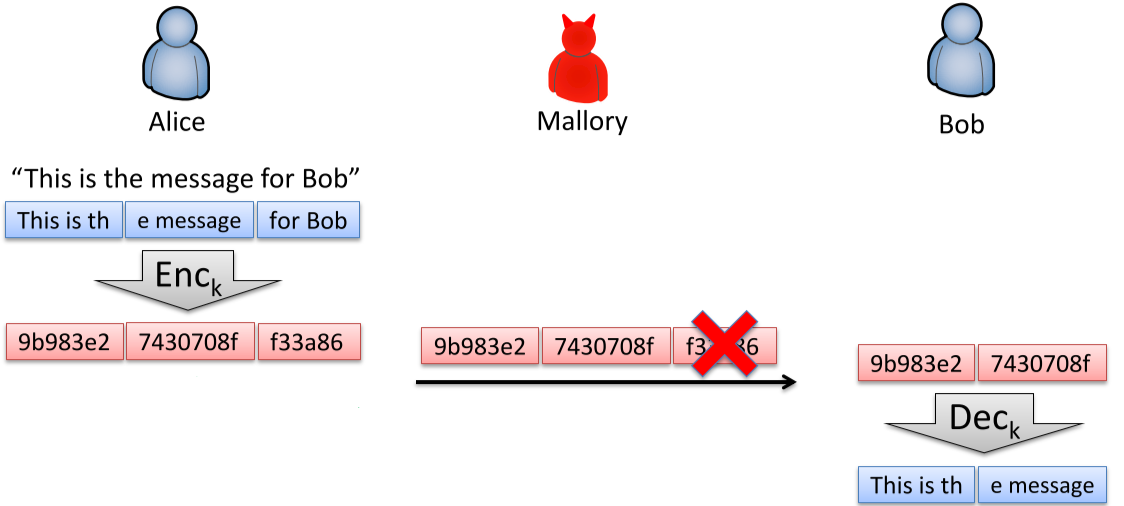
* In practice, symmetric ciphers and PKC are usually used together
  + **Symmetric**: fast, but both parties use the same, secret key – key distribution is difficult
  + **Asymmetric**: slow, but parties don’t need to know each other to send data
* Steps:
  + Encrypt secret key with PKC – this way, can send the secret key to parties without knowing them
  + Encrypt large messages with the symmetric secret key
  + Sign message with private key and cryptographic hash function of message

# Message Authentication Code

* MAC is a **cryptographic checksum** – similar to a hash, computes a value based on the original message, so that if any part of it is modified, the value changes
* Provides a way to **check the integrity** of information transmitted or stored over an unreliable medium
  + If any part of the message changed, the MAC changes
* If data is encrypted using a symmetric cipher, confidentiality is achieved; is integrity achieved at the same time?
  + Yes, because if the data was modified, would not be able to decrypt it using the secret key
  + With symmetric ciphers, encryption results in **both** **confidentiality and integrity**
* If data is encrypted using an asymmetric cipher, confidentiality is achieved; is integrity achieved at the same time?
  + No, because anyone with the public key can encrypt a message – unless a signature exists, **cannot verify origin** of message
  + When would you only want to have integrity protection but not confidentiality?
    - E.g. sensors that detect nuclear activity – want to detect activity, but do not want data to be encrypted and sent out (to prevent additional spying)
    - Sometimes, you want both confidentiality and integrity, but sometimes only one is necessary
* Can also be used in challenge-response identification protocols for computing responses based on a function of both a secret key and the challenge message
  + Hash functions summarize the entire data
  + **Keyed hash functions** encrypt only the summary, not the entire message
* The purpose of a MAC is to authenticate the **source of a message** and its **integrity**
  + Consists of two distinct parameters:
    - **Message** **input**
    - **Secret key** (known only to sender and intended receivers)
  + Sender uses MAC function to compute MAC based on the secret key and the message input; the MAC is sent along with the message
  + Receiver computes MAC with secret key and received message, and compares to the received MAC

### Can symmetric encryption be used as a MAC?

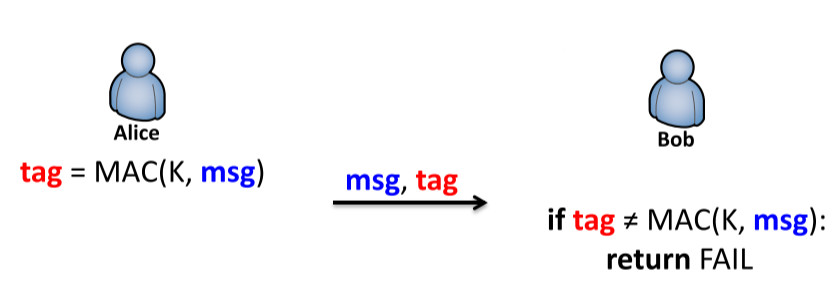
* Can we use symmetric ciphertext of the message as its authenticator?
  + No, there are a couple of problems:
    - **Not everything can be encrypted** (e.g. packet headers)
    - If symmetric encryption is used to encrypt individual blocks, it provides confidentiality but **not** **necessarily integrity** – blocks can be removed by an attacker undetected



* Can we use the hash of the entire message as an authenticator?
  + No, it **doesn’t provide** **authentication** because anyone can compute the hash
  + To solve this problem, need a secret component to the hash
    - Can mix a secret key with the hash
    - Can sign on the hash with the secret key

### MAC Function

* **Keyed hash function**: generates a **tag**: a small, fixed-size block of data using a shared secret key K and an input message M



* This guarantees authentication because:
  + Bob ensures message has not been altered – without K, impossible to generate correct tag for an altered message
  + Bob ensures that message is from Alice – only she knows K
  + **Sequence number/timestamp** can be used to guarantee freshness

### MACs Based on Block Ciphers

* CBC-MAC (cipher block chaining MAC) is used to construct a MAC from a block cipher
  + Message is encrypted with block cipher algorithm to divide message into a chain of blocks
  + Every block is encrypted using the same key and the proper encryption of previous block
    - i.e. last block depends on encryption of all other blocks
  + This ensures that any change to the message will cause the final encrypted block to change in a way that can’t be predicted without knowing the block cipher key

# Security and Insecurity of Crypto

* Security:
  + In theory, possible to have perfect secrecy
  + In practice, unrealistic
* Insecurity:
  + In theory, any imperfect cipher is breakable
  + In practice, a cryptanalyst has limited resources
    - Computing resources/manpower – needs a lot of computing power to get all keys
    - Time allocated – it would take too long to try all keys
* **Theoretically secure** systems may be vulnerable when used in a practical situation, due to system constraints
* **Practical security** is typically realistic enough to achieve
  + In practice, can utilize breakable ciphers based on attacker’s constraints
  + All ciphers that are currently used are breakable
  + Due to practical considerations, a theoretically insecure system can provide enough practical security, due to cryptanalysts’ constraints
* What does this mean for designing security systems?
  + **Conditional security**: all security depends on current technologies/computing power/resources
  + **Risk management (risk-based approach)**: accept that security can be broken; need to be able to detect security problems and know how to respond/manage it
  + **Cover time**: the amount of time that a system needs to be secured for
* Typically, key management is outside the model of crypto algorithms
  + Designing encryption/decryption doesn’t tell you how easy/hard it is to manage keys

## Key Management

* **Open system security** is based on cryptography, which is controlled by keys
  + Hence, **key management** is important
    - Long key lengths are more secure because they can prevent exhaustive search
    - Long keys aren’t easy to manage
  + **Tamper-resistant** design is crucial to solving the key management problem
  + Example: banks
    - Backend: have a machine where cryptographic keys are stored, and cryptographic operations are performed
      * No one, including system administrator, can access the data inside the machine
      * However, these machines have limited capacity
      * Standard: FIPS 140-2
    - User-end: tamper-resistant chip smart cards that store keys and protect it
      * Standard: ISO 7816

### Key Length Risk Consideration

* Shorter key:
  + Less secure (easier to perform exhaustive search)
  + Easier to manage
* Longer key:
  + More secure
  + More computing overhead for keys – problem for small platforms, which cannot do large calculations
  + More space to store
* Trade-off between security and convenience:
  + E.g. if PIN is too long, people cannot remember it, but if key is too short, easier for attackers to guess
* The security strength of an algorithm with a particular key length is measured in bits, and measures the difficulty of compromising the protection offered by the algorithm and key
* Appropriate security strength/key length depends on sensitivity of data being protected

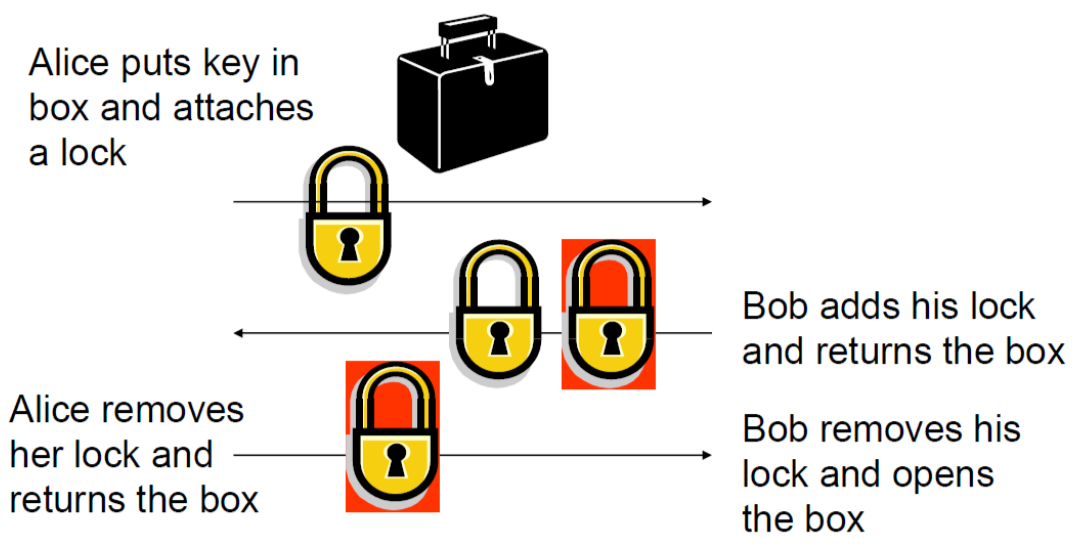
### Key Agreement and Key Transport

* **Key agreement**: both parties contribute information that forms the key, together
* **Key transport**: one party decides on a key, provides it to the other party
* SP 800-568B specifies that RSA can be used for both schemes if the key length >= 2048
  + As computing power increases, key length needs to be increased

# Symmetric Ciphers

* Sender and receiver have symmetric knowledge – they know the same secret key
* Symmetric ciphers have many problems:
  + **Key distribution**: they must both know the key in advance
    - This problem is greater in the Internet, due to network size
  + **Dispute**: both parties can claim to have generated the message – third party cannot tell who did
    - No accountability, since both sender and receiver have ability to generate and send a message
    - This makes symmetric ciphers not suitable for electronic transactions in general, since disputes must be resolvable

## Key Exchange Without Secret



* **Authenticity problem**: how can we ensure the recipient is correct?

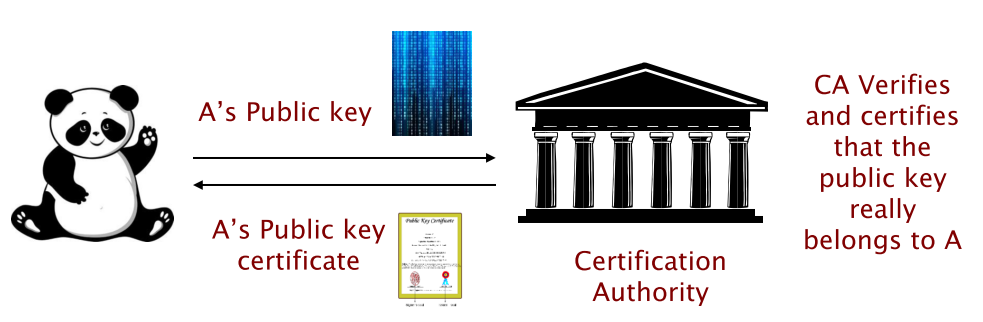
# Asymmetric (Public Key) Cryptography

* Sender and receiver have asymmetric knowledge – they use different keys
* Confidentiality depends on **secrecy of decryption key**
  + Encryption and decryption use different keys
  + It is computationally infeasible to derive decryption key from encryption key
    - Based on hard problem of big integer factorization
* Each user has a pair of keys:
  + Public key for encryption
  + Private key for decryption
* This removes the secrecy requirement of encryption keys
* Asymmetric ciphers simplify key management
  + Easier to distribute published keys
  + Only need one key support secret communication with multiple parties
* How it works:
  + Sender A uses receiver B’s **public key** **to encrypt**
    - This means that anyone can send B a message
  + Receiver B uses his **private key to decrypt**
    - Only B can decrypt the messages
* The asymmetry can also provide non-repudiation, through digital signatures
  + Sender **signs with private key**
  + Receiver **verifies identity with public key**
* How do you know the public key belongs to the correct person?
  + Need to **ensure the** **integrity** of the public key – if the key is incorrect or modified, cannot decrypt message

## Public Key Infrastructure (PKI)

* Since Public Key Cryptosystems (PKC) use much longer keys, secure storage and distribution is more challenging
* Main PKC key management issue: **integrity of public key must be assured**
  + Can be provided by a **trusted third party (TTP)**
  + For each user with public key , TTP generates a **public key certificate**:

* + - = type/ID of signature algorithm
    - = name of certification authority (TTP)
    - = key owner
    - = timestamp
    - = lifetime
* Anyone who wants to send data to a receiver, first retrieves their public key certificate
  + How can you check the authenticity of the certificate?
    - Verify the signature
  + How do you verify the signature?
    - Using the CA’s public key, which is **usually embedded in software** (e.g. OS, browsers)
    - This means that the **root of the trust** lies in the software provider
      * Cryptographically, trust the CA, but ultimately, need to trust the software that packages the CA’s public keys
  + Why do we need the and ?
    - All current technologies are **conditionally secure** – depends on limitations of current technologies
    - To protect against new technologies that can break security, by **forcing a renewal** after the lifetime
  + Who can be the CA/TTP?
    - Anyone that the sender and receiver both trust, can be the CA/TTP
* Security requirement of public key certificates:
  + **Confidentiality is not required** – key is public
  + **Integrity** **is ensured** – signed by CA
* PKI enables the use of PKC in distributed systems, through **solving the** **integrity problem**
  + Enables senders to ensure that they are using the correct public key
  + TTP certifies the binding between a user and their public key



* **Distribution** of public keys:
  + Through a directory service
  + Sent directly between users
* The CA’s public key must be public available, and its correctness is very important
  + Must the CA be highly available?
  + How about the availability of the directory service?
  + What if the corresponding private key is compromised?

# Secure Key Management

* The strength of cryptography depends on the level of **key protection**
  + Keys must be protected from external hacking and internal fraud
* How should we store keys securely?
  + In real-world systems, encrypt keys with a **key encryption key** (master key)
* If the purpose of the cryptographic key is to encrypt data, why do we need a key encryption key?
  + Two ways to protect keys in a file system:
    - Access control
    - Encrypt key
  + In real-world systems, cannot have enough secured filesystems to store all the user keys
  + Instead, encrypt user keys with master key, securely store the master key, and store the encrypted keys in regular filesystems
* Software-based solutions are vulnerable
  + Decrypting sensitive data in unsecured memory makes the data vulnerable to attacks
* **Hardware-based solutions** are essential for secure key management
  + Separate hardware encasement that safeguards all cryptographic algorithms and keys from unauthorized access
  + All cryptographic processing takes place within a **tamper-resistant**, physically secured environment
  + Longer encryption keys need faster CPU – what problems arise when using faster CPU?
    - Overheating – in security design, it is an issue, because needs ventilation to combat, which means holes that could become attacker entry points
      * Without ventilation, to combat overheating, the CPU will slow down

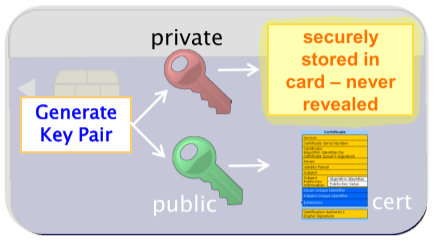
## Chip Cards

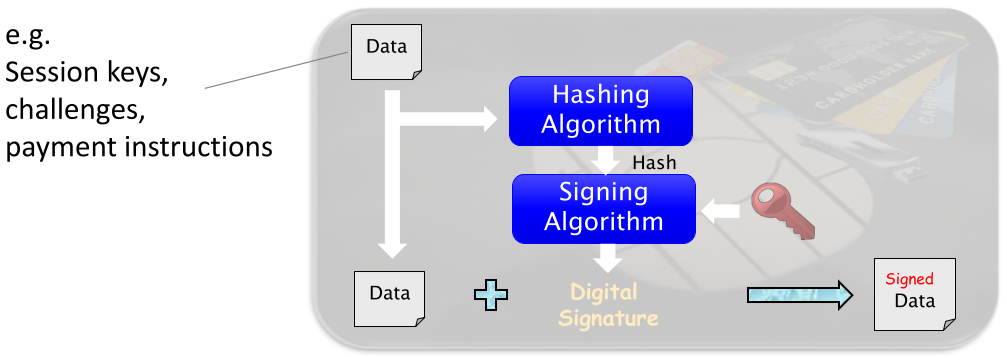
* Typical applications:
  + Digital identification
  + Multi-factor authentication
    - What you know (e.g. password)
    - What you own (e.g. the card itself)
    - What you are (biometrics)
  + E-payment
* Microprocessor in the card:
  + How is it powered?
    - Electromagnetic field generates current that powers the microprocessor, upon contact
  + Chip card provides tamper-resistant environment to store cryptographic keys
    - Card generates cryptogram to prove its identity
    - Key: RSA private key – provides decryption and digital signature
    - What is the advantage of doing on-card key generation?
      * If the card is tamper-resistant, that means that the **key will never be leaked**
    - What is the disadvantage?
      * If the key is generated inside the card, then key management problem – if the card is lost, so is the key; then, **data is effectively lost forever**, since no one else can decrypt it
  + Supports single or multiple applications
  + Some support biometrics
* **Tamper-evident**: knowledge of tampering
* **Tamper-resistant**: cannot tamper, or data is wiped during tampering

## File System Architecture

* File types:
  + **Master file** (MF) – root directory
  + **Dedicated file** (DF) – directory
  + **Elementary file** (EF) – data/records
    - Cryptographic data are stored in these types of files
* Each file has an access control list (ACL), which defines its allowed operations (read, write, etc.)

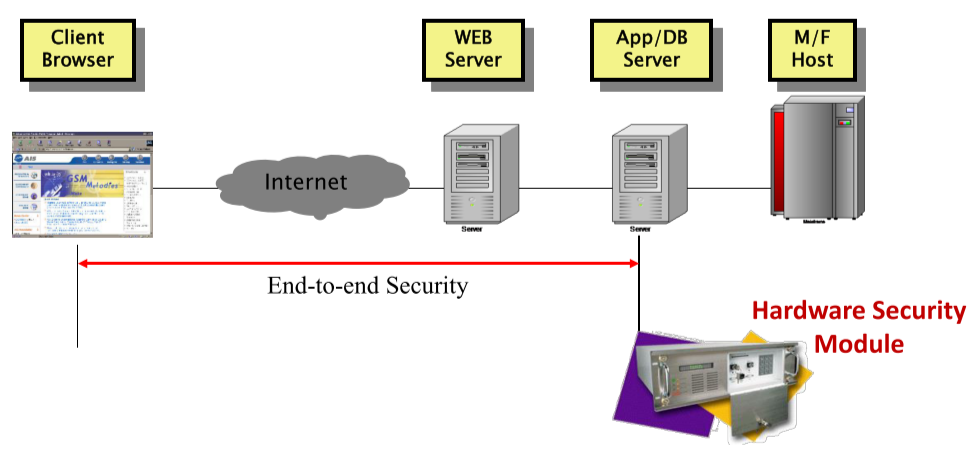
## On-Card Crypto Functions and Objects





* Card generates public-private key pair
  + Public key is made available for data encryption and signature verification
  + Private key is never revealed
* Store user’s public key certificate on smart card – why is this necessary?
  + Portability – provides integrity verification to devices that it communicates to, so that those devices don’t have to search for the certificate
* **Hash computation** to create message digest
  + When do we need to use message digests?
    - So that we can digitally sign (RSA) on entire documents efficiently
  + Why do we need to store the hash function on the card?
    - Signing needs to be done inside the card, so the hashing must be too

## Server-End Secure Key Storage



* Application server is vulnerable to insider fraud, if not other security issues
* HSM is a tamper-resistant hardware for key storage and management
  + Contains cryptographic processing – encrypted passwords/keys are sent to the box for processing
  + Private key never comes out of the box

# Key Management/Security Objectives

* Depending on value of data, security is a **business requirement** – how much you’re willing to spend to protect the data’s value
  + Level of security
  + Type of security
* Security needs to meet management objectives such as:
  + Public confidence
  + Accountability
  + Liability protection
  + Minimize uncertainty
* Need to prevent **insider fraud** – use tamper-resistant key management server

## Security Enforcement

* Three categories:
  + **Technology** – e.g. smart cards, cryptography, biometrics, PKI, firewalls, VPNs
  + **Processes** – e.g. security system integration, risk analysis
  + **Control** – e.g. security reviews/audits/monitoring
* Conventional cryptography and PKI are widely adopted to enforce security requirements:
  + Confidentiality
  + Integrity
  + Authenticity
  + Non-repudiation
  + Accountability
* With growing concern for insider fraud, it is equally important to enforce **accountability** of government officers
* Conventional cryptography and PKI may **not** address accountability effectively
  + Conventionally, to enforce accountability:
    - All transactions are digitally signed
    - PKI enables digital signature
    - PKI is enhanced by smart cards
    - Smart cards are protected by passwords
  + Can we guarantee that the signature is really performed by the authorizing officer?
    - In security design, assumption is that **the key is you**
      * Possible for password to be stolen
      * Possible for card to be stolen
* **Strong non-repudiation** is required to enforce accountability:
  + Need to also enforce process of generating the digital signature – i.e. the authorizing officer must be personally involved in the signing process
* Issues – can have one of the other (physical presence or digital signature):
  + Conventional cryptography doesn’t enforce physical presence
  + Conventional biometrics cannot be used to provide digital signatures
* Key issue: **link between key and key owner is assumed** in software design
  + Traditional security protocols assume “the key is you”
  + The real relationship between the key and you is crucial to the accountability issues of security systems
    - Need to require strong assurance that the virtual identity is really linked to the physical identity

## User Identities in Security

* User identities are fundamental to a range of system security issues
* Typically, user identities are established by:
  + **Identification**: system identifies the user
  + **Authentication**: user claims their identity, system verifies the identity by authentication
* Types of authentication:
  + **Simple authentication**: prove knowledge of secret by providing secret
  + **Strong authentication**: prove knowledge of secret without giving up secret

# Distributed Authentication

* Computers run programs that are owned by users
* When a program runs, the users delegate their rights to the program, so that the program will have the same privileges and resource access rights as the user
  + User is responsible for any mistakes that the program makes
  + By passing your secret key to a program and authenticating, user gives the program its user privileges

## Distributed System Security

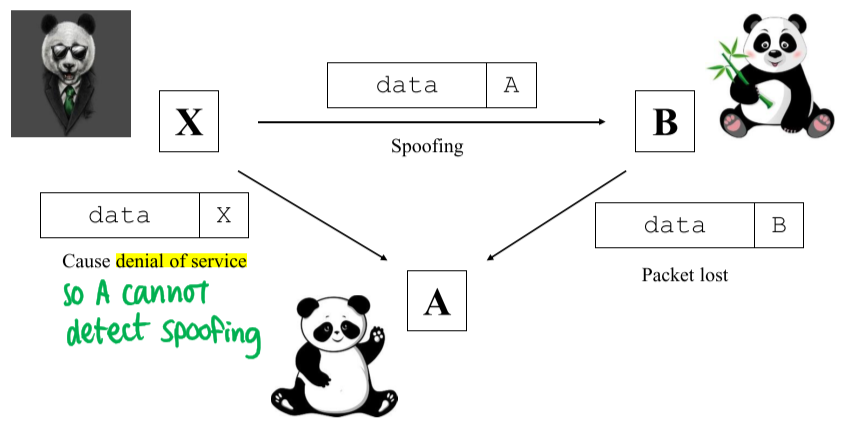
* **Inter-process communication** (between computers connected over a network) is a basic need of networked systems
* In closed environments, there is a level of trust between devices
* IPC over open networks introduce complexity to security problems
  + Anyone can try to connect to the network
  + Cannot trust any other devices, only yourself
* Cryptography is a basic mechanism for communication security
  + Problem outside scope of cryptographic algorithms: key management
    - Key exchange – how to exchange keys with the correct party?
    - Key distribution
* **Strong authentication** protocols are commonly used
  + **Entity authentication** – access control
  + **Key exchange** – transaction confidentiality and integrity through establishing shared secret
* The Internet is much more open than telephone networks, despite having same underlying structure
  + The **interface** **to the Internet** is usually through ISPs, either government or commercial
  + ISPs have access to all network communication – can you trust them?

## IP Packet Attacks

### Wiretapping

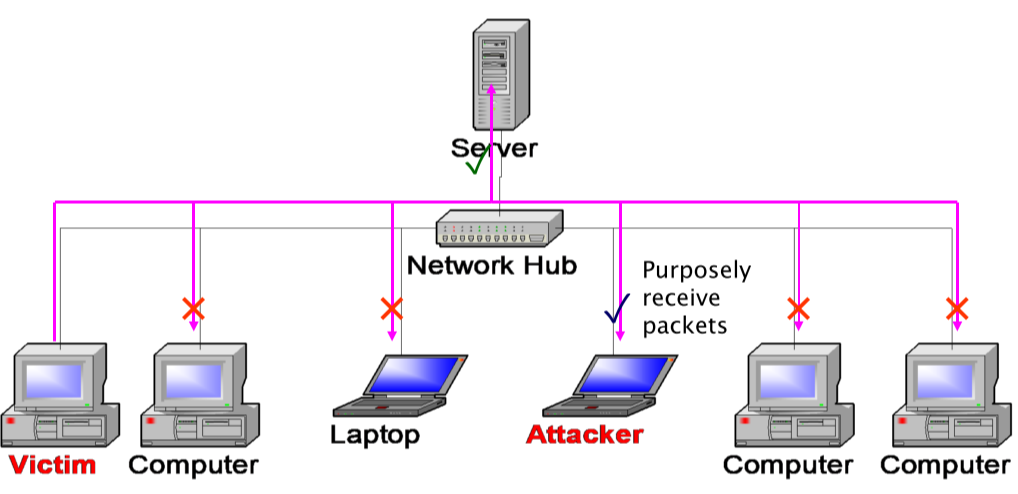
* The format of IP packets is well-known
  + Header contains source and destination IP addresses
  + Address information from the header is sometimes used for authentication
* Tapping of packets can use well-known information, e.g. impersonation

### Spoofing



* X impersonates as A and sends data to B – this will work if relying on IP address for authentication
* X floods network of B to cause denial of service, so that A cannot detect spoofing

### Sniffing



* Most local area networks (LANs) are **broadcast networks** – server broadcasts received data to all computers, irrelevant devices choose to ignore the message
  + Attacker can listen in on broadcasts and receive packets
* Cross-country traffic must connect to wide area networks (WANs) – so why is LAN security still a problem?
  + Need to connect to an ISP, which means connecting to a LAN (which then contains a router to connect to WAN)
    - Every WAN needs to connect to a LAN
  + Hence, sniffing of network data can occur inside an ISP

## Distributed System Security Problem

* Given open network IPC, self-protection is required
  + Self-protection requires cryptography
  + Cryptography requires shared keys
  + Shared keys require authentication protocols
    - Authentication
    - Key exchange
* Due to the open nature of the Internet, distributed systems are vulnerable to:
  + **Wiretapping**: packets can be read from the network
  + **Tampering**: fake packets can be injected into the network
* Why is this possible? Due to the lack of **central administration**
  + Different machines can have different administrators
  + The same user can have different identity assignments on different machines – i.e. when a user logs in, they can assign their own identity; this is liable to **impersonation**
  + Different machines can have protection policies – even if administrator can be trusted, they might not have the technical capabilities to protect the machine; this is liable to **unauthorized disclosure** of data
* Program components run on different machines
  + Application programs may need to access resources over open networks
  + Data resources may be delivered over open networks
  + Message passing may occur over open networks
    - Encryption is necessary to prevent third parties from reading data
* **Open connectivity**:
  + Identity is authenticated on one machine and accepted by another
    - Identity should not be trusted unless authentication is done on own machine
  + Data is stored on one machine and processed on another
    - Do not want to trust the other machine(s), because they could be under different administration
* Security control enforced by one computer cannot rely on the identification procedures performed by others
  + **Encrypt network packets** if security is desired
  + Each transacting client needs to be **authenticated by the server**
    - Distributed systems require strong security
      * Authentication
      * Key exchange
      * Secure data exchange

### Example: Internet Banking

* Non-repudiation to protect the bank, through **strong authentication**
  + Bank **authenticates user** **identity** to correctly identify customer
    - When user enters PIN, it is encrypted such that the bank server can verify the password without anyone seeing it
      * This protects against impersonation
      * Only the customer knows the PIN, so we assume that they are responsible for the transaction
      * Assume customer protects themselves by keeping the PIN secret
  + Ensure **transaction integrity**
    - At the end of the authentication, secret key is established; need to ensure transaction data is not modified maliciously
    - Use key to generate keyed MAC code, to protect the integrity of transaction data

# Key Exchange Protocols

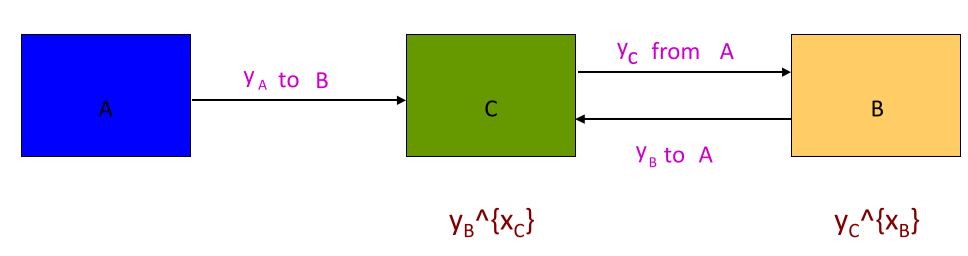
* Key exchange protocols support:
  + **Key exchange** for bulk data transfer
  + **Strong authentication** for distributed access control
    1. Authentication – establish identity
    2. Access control – server checks if user has rights to access data
* Both are essential security requirements of distributed systems
* Some protocols may only support one or the other

## Key Distribution

* Security depends on protection of keys
  + Symmetric encryption/decryption
    - Problems: Fast, but difficult to share secret key
      * **Secure distribution** of symmetric keys
  + Asymmetric decryption/digital signature
    - Problems: Slow, need to ensure integrity of public key
      * Impractical to protect **bulk data transfer**
  + Solution: user asymmetric cipher to encrypt symmetric key, and symmetric key to encrypt data

## Diffie-Hellman Key Exchange Scheme

* Key exchange scheme only – does not provide authentication
  + Based on the **discrete log** problem
* Two users, and :
  + Agree on a large prime number , and a primitive root
    - Every member of finite field can be expressed by an exponent of
    - Hence, every element in can be expressed as
  + Publish and (public parameters)
* and each choose a private random number, and
  + computes , B computes
* Key exchange mechanism:
  + and exchange and
    - and are kept private, and are made public
      * This is possible because it is computationally infeasible to derive and from and
    - Thus, knows and , and knows and
  + computes and computes
* The shared secret:
* What assumptions did we make?
  + We assumed the **integrity** of the public component – assumes that each party received the correct , hence end up with the correct secret
* What is the security problem of the Diffie-Hellman scheme?
  + The public component can be tampered with, making it vulnerable to a **man-in-the-middle attack**



* + - Attacker intercepts , sends to – this results in unknowingly establishing a secret key with
* Can we use DH for digital signatures?
  + Can only be used for key exchange – since the key is shared in the end, it cannot be used for digital signatures (in its basic form)

### Discrete Log Problem

* **Discrete logarithm problem (DLP)**: given , , and , need to find the discrete log so that
* Exponentiation is a one-way function when the DLP is difficult

# Authentication Protocols

* **Strong authentication**: prove your identity without disclosing the secret
* Two types of authentication protocols:
  + Based on symmetric ciphers
    - E.g. Needham-Schroeder Protocol
    - Requires presence of an **Authentication Server (AS)** – trusted third-party
      * The AS shares a key with every user in the system
  + Based on public key cryptosystems
    - E.g. X.509 is an authentication protocol using PKC
    - Requires the presence of a **Certification Authority (CA)** – trusted third-party
    - The CA’s public key is publicly known

### Design Criteria

* **Reciprocity**: one-way or two-way (mutual) authentication
  + One-way: server authenticates user identity
  + Two-way: user and server authenticate each other’s identities
* **Key freshness**: protection against replay attacks
  + Attacker can intercept message, make a copy of encrypted username and password, and then resends the packet, impersonating as you
  + To protect against replay attacks, need some dynamic data – add time-varying parameter/nonce/sequence number to the encrypted data
* **Key control**: who generates the key?
  + If one party chooses the key with specific properties, some attacks are possible
* **Third party requirements**: is a TTP involved? Are they offline or online?

## Needham-Schroeder Protocol

1. sends authentication request to server :  
   I am , I want to talk to , nonce
2. responds to with message encrypted with key shared between and : (’s nonce)when you talk to use key send this to (key shared between and ):
   * provides with to identify itself to
   * Only and know
3. sends to message encrypted with key shared between and :use this keyto communicate with
   * tells that whoever knows is
4. sends to : nonce
   * Use to encrypt nonce – challenges to see if it knows
5. sends to :
   * Proves that it knows , by decrypting the nonce and encrypting with

* What does the protocol achieve?
  + Mutual authentication – authenticates and vice versa
* What is the security concern of the protocol?
  + Attacker can **replay message 3**, since there is no time-varying parameter chosen by B

## Challenge-Response Protocol

* Ensure the freshness of a response through issuing a challenge to the sender, to guard against replay attacks
  + is a nonce created by as a challenge to , to ensure **timeliness** (prevent replay attacks)
  + ’s response to is well-protected because it contains the nonce provided by
  + However, is created without B’s knowledge – subject to replay attack because it doesn’t contain a time-varying parameter chosen by B
  + Attacker can wiretap the authenticator, obtain , and impersonate to communicate with

### Nonce

* A value that **cannot be used more than once** **for the same purpose** – a time-varying parameter
  + Guarantees freshness
  + Prevents against replay attacks
* In some applications, such as session keys, nonces have to be unpredictable
* Three ways to generate nonces:
  1. Random numbers – large number, to avoid repeats
  2. Timestamps
  3. Sequence numbers

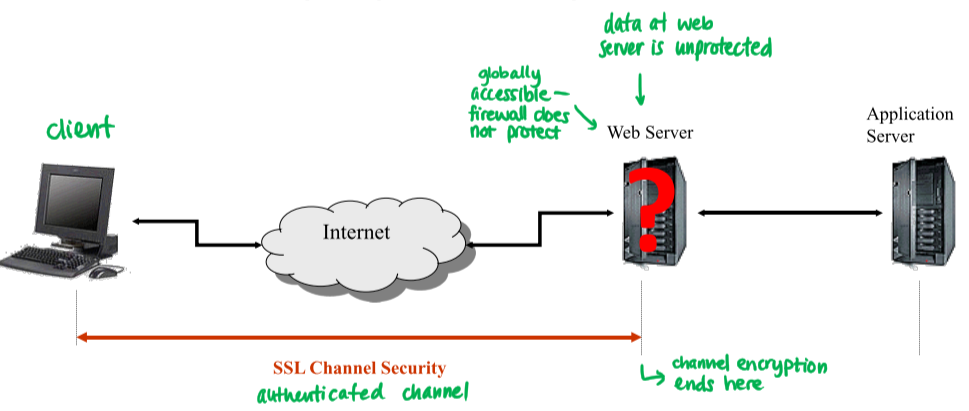
## X.509 Authentication Protocol

* Authentication only, no key exchange

1. nonce generates ’s session key, encrypts with ’s public key ’s digital signature on the message
2. nonce generates ’s session key, encrypts with ’s public key ’s digital signature on the message, with ’s nonce
3. signs on ’s nonce

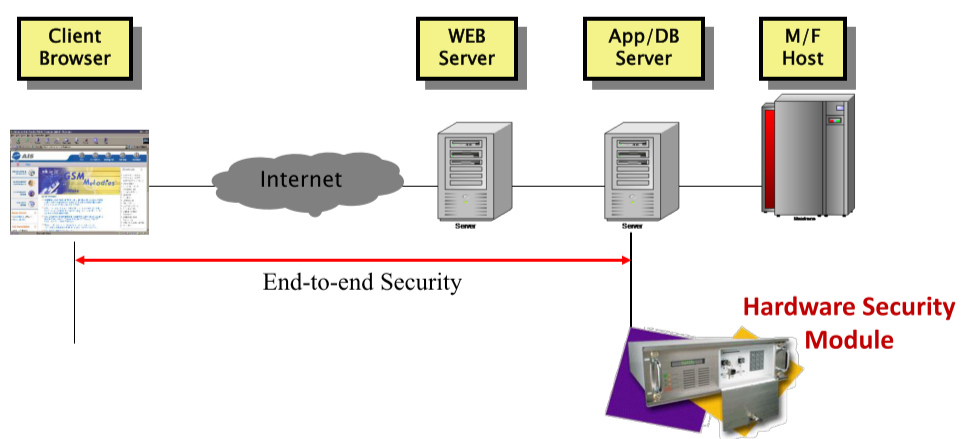
* What does the protocol achieve?
  + Mutual authentication between and
* What is the security concern?
  + Should never sign on a random value – the value can have meaning, e.g. a message digest (hash value)
    - could be certifying a transaction with its signature

## Example: E-Banking Security



* Browser and web server have a secure, authenticated channel
  + Channel encryption **ends at web server**
  + Web security provided by SSL/TLS is **infrastructure-level authentication** – ends at web server
    - Data is automatically decrypted at web server – unprotected
* Cannot protect web server with firewall – it would block access to it

Result: end-to-end security with tamper-resistant HSM



# Secure Network Architecture

* In practice, there is a lot over overhead required to use cryptography – and so, can we have practical security without using cryptography?
* **Open-system environment**: cannot trust anyone – data must be self-protected: must be encrypted when sent over a network, so that confidentiality, integrity, and authenticity is preserved
  + With a massive amount of data, infeasible to encrypt it all
  + If we can create a closed environment, then cryptography (which is a very costly operation) can be avoided
* **Closed-system environment**: if environment can be controlled correctly,
  + E.g. inside a datacenter
  + How do we create a closed environment?
    - Physical security
    - Network access control – control who can have remote access to the datacenter

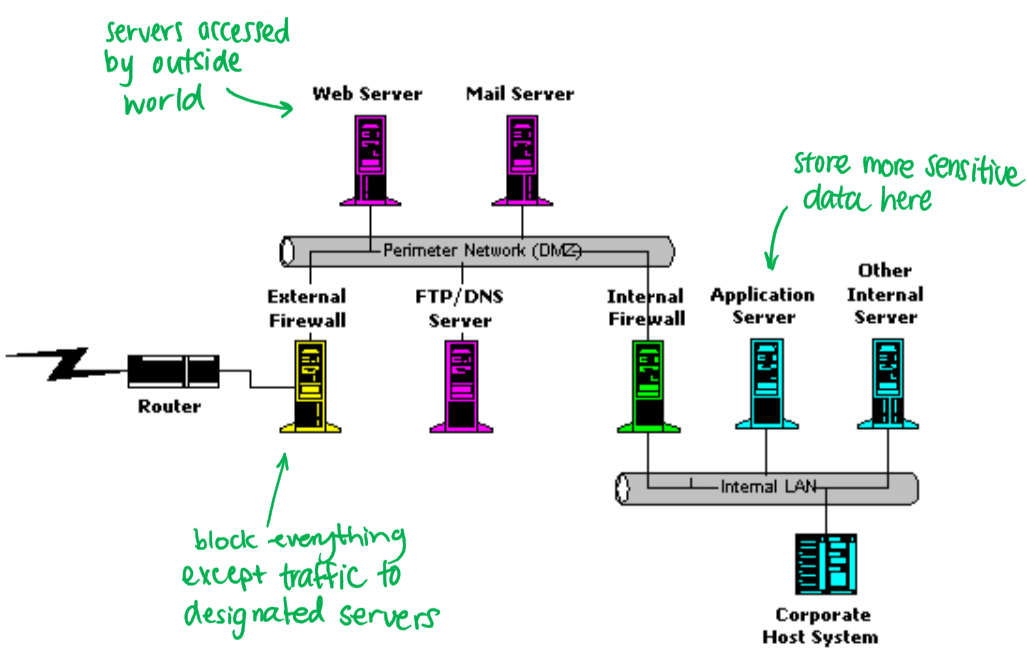
## Closed System Approach

* The Internet is insecure due to the open nature of the network
  + A connection to the Internet introduces **risks** and **uncertainties** to your network
    - Who is attacking?
    - What losses will be incurred?
* Open system security is costly to provide
  + In the real world, we try not to do cryptography unless it is absolutely necessary
  + Hence, the typical approach is to use a combination of open and closed system approach
* Internal network: closed system approach
  + Closed system created by **firewalls** – limit which servers can connect to the outside
    - E.g. in a banking system, only the web server can connect to the outside; other internal servers in the same network are blocked from outside communication
  + Firewalls based on **packet filtering**
    - Check packet headers to see if destination is legal/communication is allowed
  + Packet filtering based on **filtering rules**
* Data still needs to flow between the outside and the protected server
  + Need **end-to-end security**: data is protected from client to server
    - This requires cryptography
    - Cryptography requires access control and authentication
    - Remembering the keys for access control and authentication may use smart cards

## Network Security Plan

* Need to design a secure network infrastructure that
  + Provides connection for access to outside world
  + Protects internal network from outsiders
  + Allows easy maintenance of security – e.g. filtering rules, firewall updates
    - When new applications or servers are introduced, prevent conflict between new and existing rules; otherwise, after awhile, network will become insecure
  + Avoids use of cryptography when possible
* Approach:
  + Use **firewalls** to create closed system
  + Firewalls support **packet filtering**
  + Divide network into **subnets**

### Network Security Example



* Web server and mail server accessible to outside world
  + Usually should not store sensitive data here
  + Gateway – no processing is done
* Where should password file (sensitive data) be stored?
  + When web server receives authentication request, it forwards the request through the internal firewall, to the **application server** to process login
* Which server is the most sensitive?
  + Customer account balance and transaction records are the most sensitive data that a bank stores
    - Integrity and confidentiality requirement
  + **Corporate host system** stores the most sensitive data
    - To attack it, would require hacking into external firewall 🡪 take control of web server 🡪 hack internal firewall 🡪 take control of application 🡪 take control of corporate system
      * But then, corporate host typically uses different message format/communication protocols – harder to hack

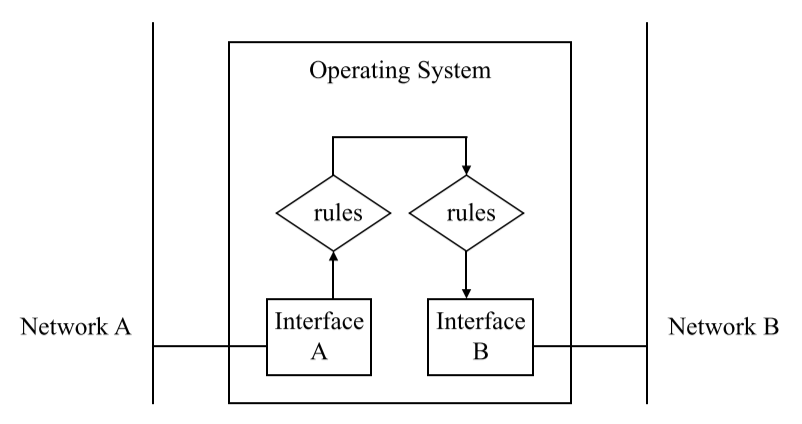
### Typical Enterprise Security Framework

|  |  |  |  |
| --- | --- | --- | --- |
| Secure Information Systems | | | |
| Application Security | Firewall  and  VPN | Intrusion Detection | Identity and Privilege Management |
| Transaction Security |
| Access Control |
| Security Protocols |
| Cryptographic Algorithms |
| Operating System Security | | | |
| Hardware Security | | | |

* IPC: access from one machine to another is through the network
  + Cryptographic algorithms enable security protocols (authentication and key exchange protocols)
  + Security protocols enable access control, which allows authentication into the application server
  + Transaction security requires encryption and digital signatures – using cryptographic algorithms – to form secure transactions
    - Server makes use of layers to verify signature, decrypt data, and pass to application layer to execute transactions
* **Security is meaningless without knowing what you need to protect**
  + Application security depends on value of data determined by application

## Firewall for Closed Systems

* Firewalls restrict access between a protected network and the Internet, or between networks
  + Allow access to global Internet
  + Controls/limits outside access to local network (LAN)
* All traffic must pass through the firewall
  + **Security policy** defines authorized traffic – specified by **filtering rules**
  + Only authorized traffic can pass through the firewall



* Firewalls are computers with two network interfaces
* A program controls the flow of packets through the interfaces
  + When a packet comes in, it is dropped if it doesn’t pass through filters, otherwise allowed to go through to the other network

## Practical System Security Approach

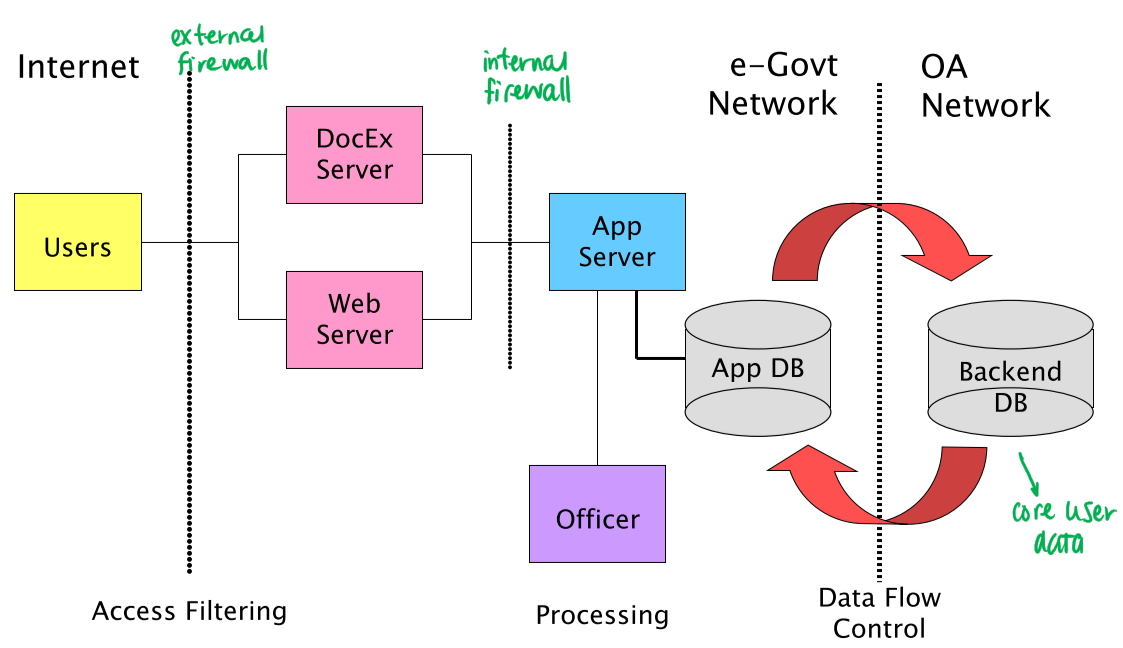
* **Risk analysis**
  + Identify what you want to protect – based on application, determine information assets, their value, and related threats
  + How valuable is the data to you – can you afford to lose it? How valuable is the data to the attacker?
* **Security policy**
  + Based on risk analysis
  + Who can access the data, confidentiality/integrity constraints
  + Needs to meet security objectives, fit business needs, and regulatory requirements
* **Security architecture**
  + Network and system security
    - Firewall, network segmentation, system hardening
  + Application and software security
    - Secure application design/programming
* There is no perfectly secure system:
  + Cryptographic algorithms can break
  + Security features may not be designed or used correctly
  + Security components may not be implemented or configured properly
* Security throughout the entire lifecycle of information systems (design, development, implementation, operation, disposal) need to be managed carefully
* Practical design manages risk
  + **Conditional security**: depends on computing power of threats, and how they change
  + **Risk-based security systems**: identify risks, analyze risks, control and manage risks, evaluate implementation, and update as necessary
    - Need to write down all assumptions – if environment changes and assumptions become invalid, then the system becomes insecure

### Example: E-Government Systems

* Two general types of e-government systems:
  + Online access to government services
  + Internal operation automation of government agencies
* **Form submission**: simple transactions
* **Document submission**

# Application Architecture

* In general, use **multi-tier architecture**
  + Better security: can filter access at middle layer
  + Better interoperability: standard external interfaces
* **External interfaces**: support form submission and document submission
* **Internal interfaces**: integrates with backend operations, usually uses proprietary protocols

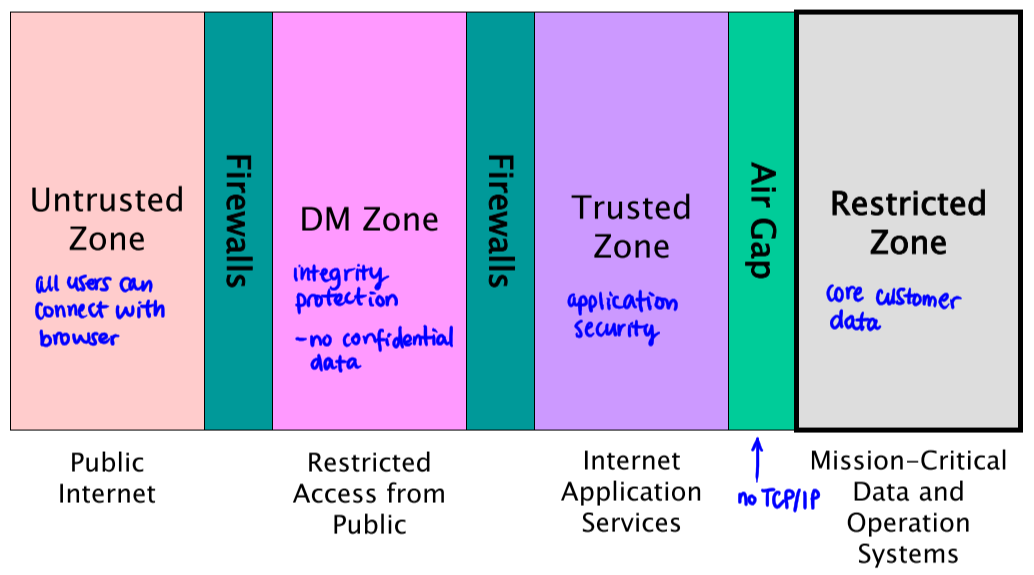


* Application DB stores application-specific, non-sensitive data
* Core user data stored in Backend DB
* OA network – office automation network

## Web Hacking

* Most web hacking is done by exploiting web server loopholes
  + Software bugs:
    - Invalid input (buffer overflow, Unicode)
    - Executable scripting
  + Stealing data
  + Web defacement – modify web pages

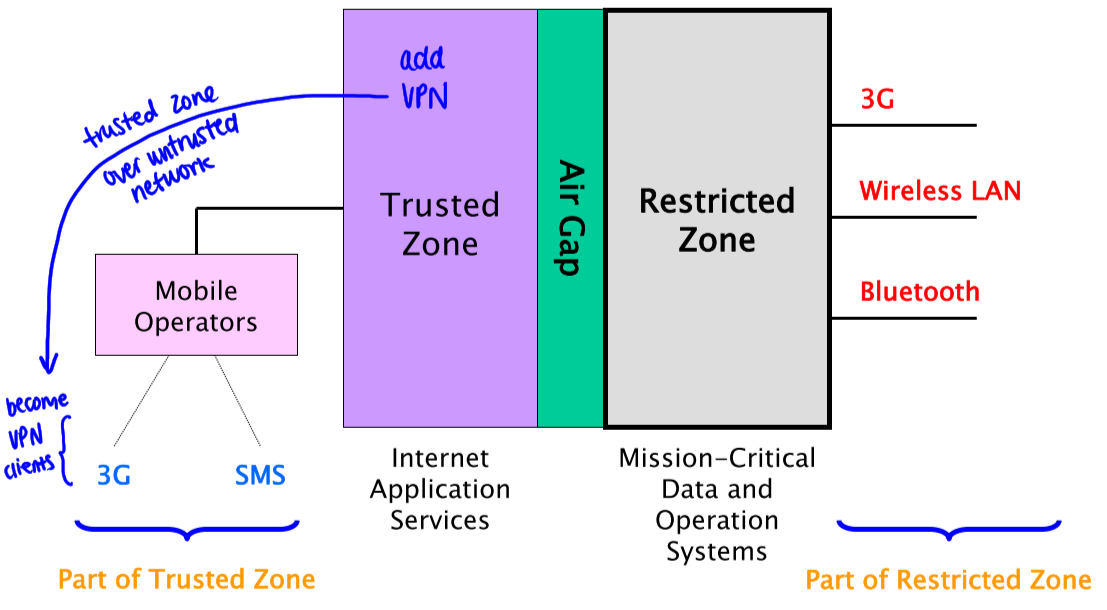
## Defence in Depth



* **Untrusted zone** – Internet with public access
* **DM zone** – restricts public access using firewall
  + Integrity protection (e.g. against defacing of the website)
* **Trusted zone** – application services
* **Restricted zone** – mission-critical data and operations
  + Separated by **air gap** – no TCP/IP communication allowed through the gap

## Evolution of E-Government Systems

* Mobile technologies used to ensure pervasiveness of government services
  + Both public access and workforce internal operations
* New network security enhances pervasiveness, but also introduces security issues
  + **Technology is more dynamic** – may change more rapidly
  + **Systems are** **more open** (instead of proprietary)
    - More open protocols, media access, and operating environments
* As a result, control is more difficult to enforce – more uncertainty, less confidence

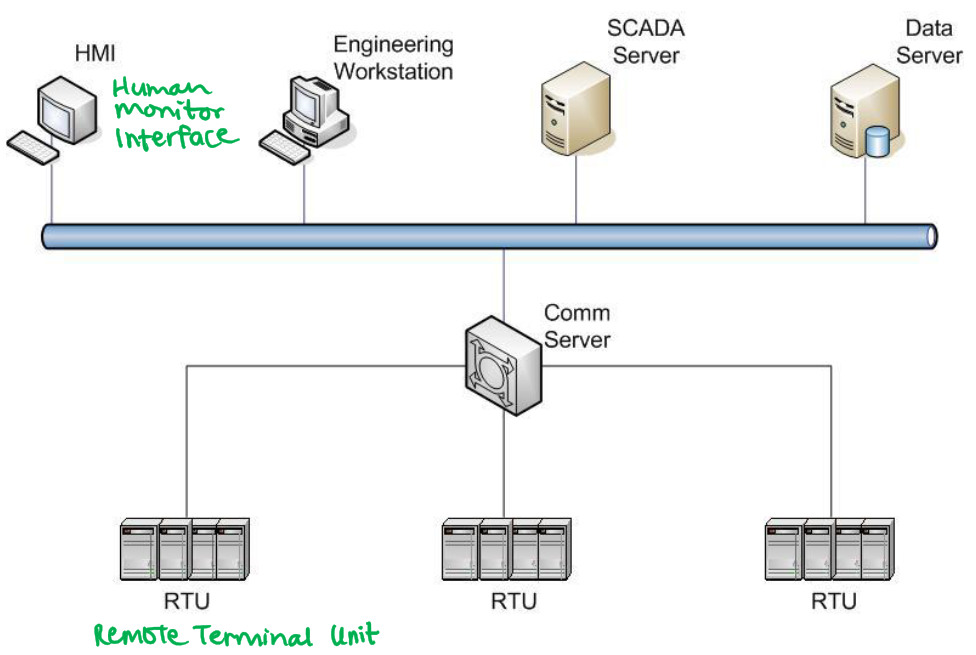


* How to connect to trusted zone through public (untrusted) networks?
  + Add **VPN** to trusted zone
  + Mobile operators become **VPN clients** – untrusted network becomes part of trusted zone

# Cyberspace and Cyber Security

* Cyberspace: the **virtual** **environment** of information and interactions between people and businesses
* Provides critical support for the economy, civil infrastructure, public safety, and national security
* How is cyber security different from network security?
  + **Network security**: protect packets of data
  + **Cyber security**: protecting people’s interactions/business – much more complicated

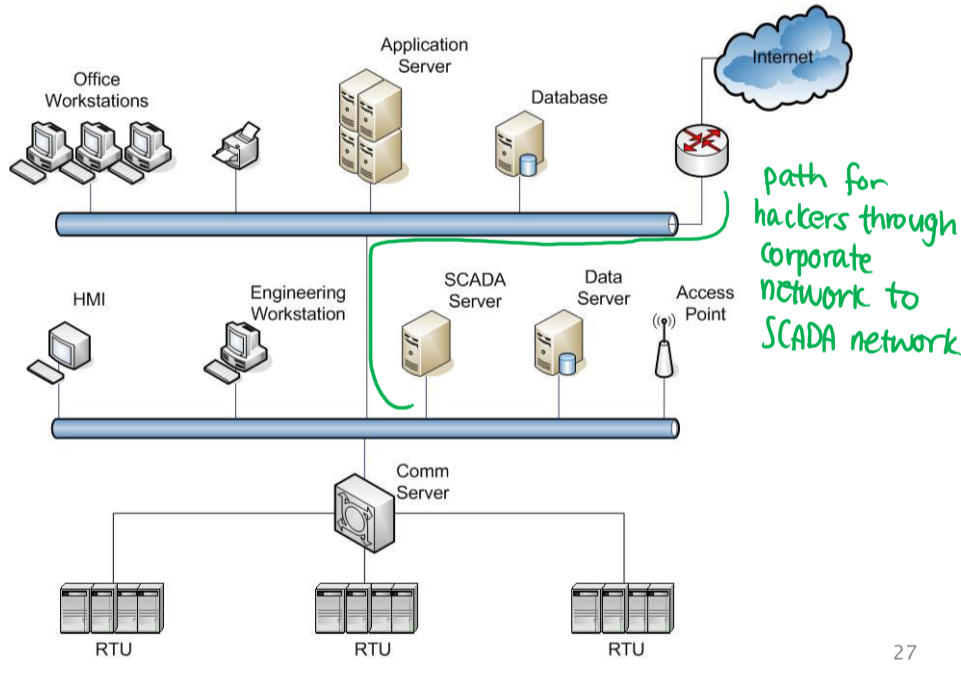
## SCADA (Supervisory Control and Data Acquisition System)



* SCADA networks monitor and impose controls on the environment
  + **Remote Terminal Units (RTU)** directly interact with environment
  + **Human Monitor Interface (HMI)** visualize all operations

### CII (Critical Infrastructure Security) Trend

* CII today
  + Promotes corporate connectivity and remote access
  + This means that most devices are I**P-based** – more vulnerable to hacking
  + Protocols and operating systems are more **standardized** – more open = more opportunities for attacks
* The trend is to support new IT capabilities, but provide less isolation
  + This increases possibility of cybersecurity vulnerabilities – creates greater need for security

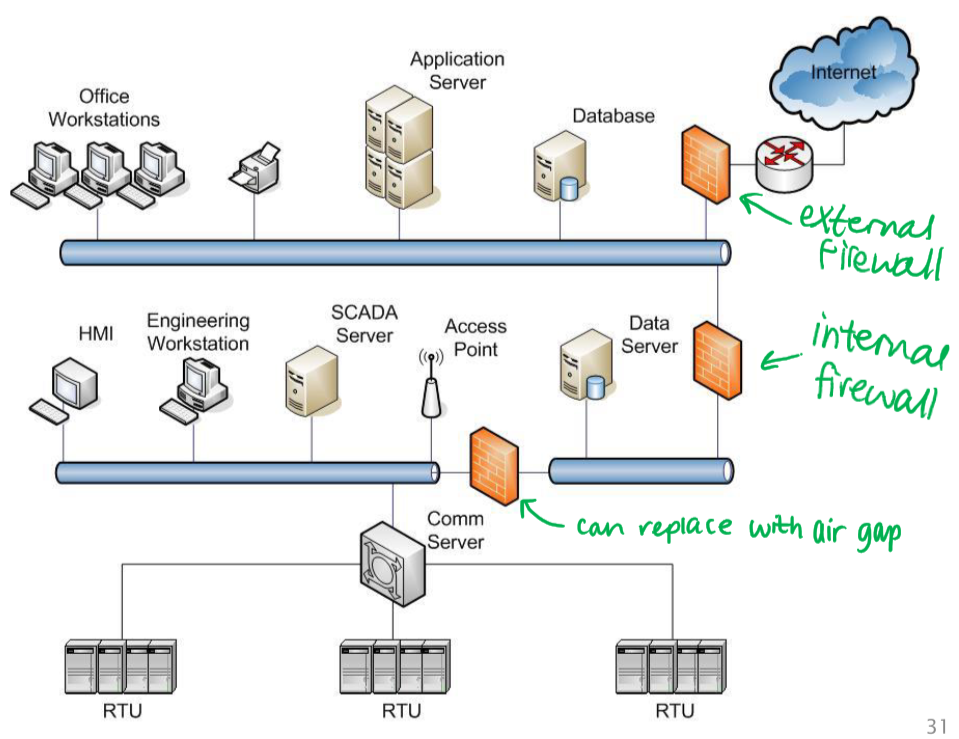


* In the past, SCADA servers are isolated
* Nowadays, they are connected to the corporate network; this provides a path for attack
* Typical security objective of CII is to prevent loss of life, endangerment of public health, and regulatory compliance
* Security characteristics of CII include:
  + Time-critical and deterministic responses
  + CII-controlled processes are continuous in nature
  + Security protection at network infrastructure level is very important

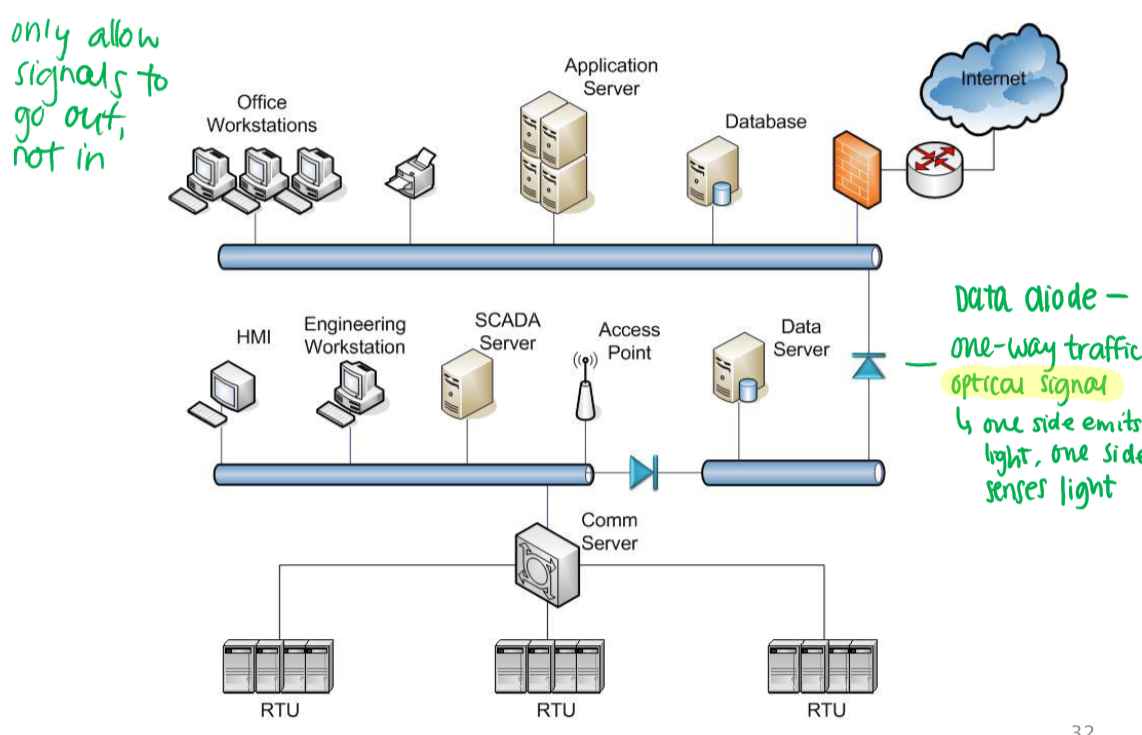
### Mitigation of Threats

* **Insider threats**
  + Identity management/accountability: access control, strong authentication
  + Data protection: cryptographic operations
  + Restrict use of connectivity (Wi-Fi, Bluetooth)
  + Personnel: background checks, immediately revoke privileges when not necessary
* **External threats**
  + DDoS attacks
  + Malware: firewalls, security updates

### SCADA with No Direct Traffic



### SCADA with One-Way Traffic



* Want to allow signals to go out, but not in (so SCADA server is not compromised)
* Data diode – use **optical signal** to enforce one-way traffic
  + Sender emits light, receiver senses light