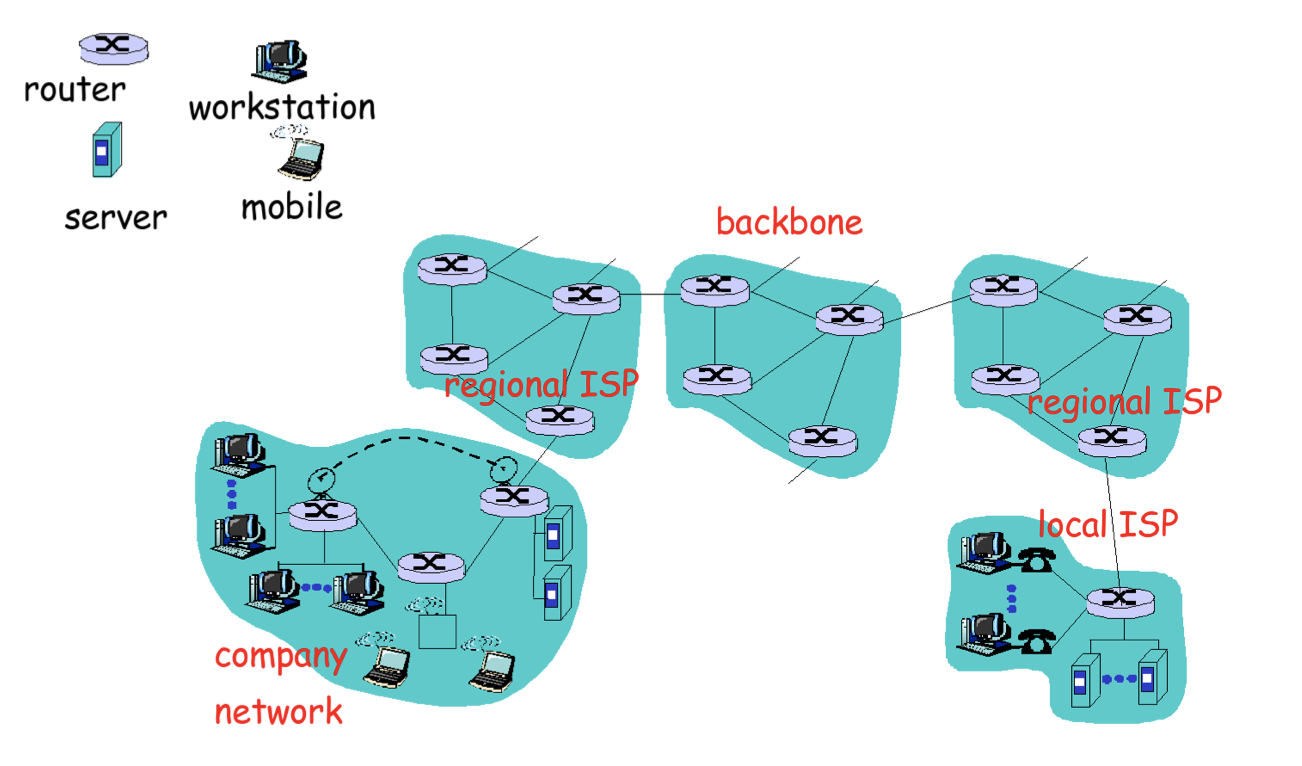
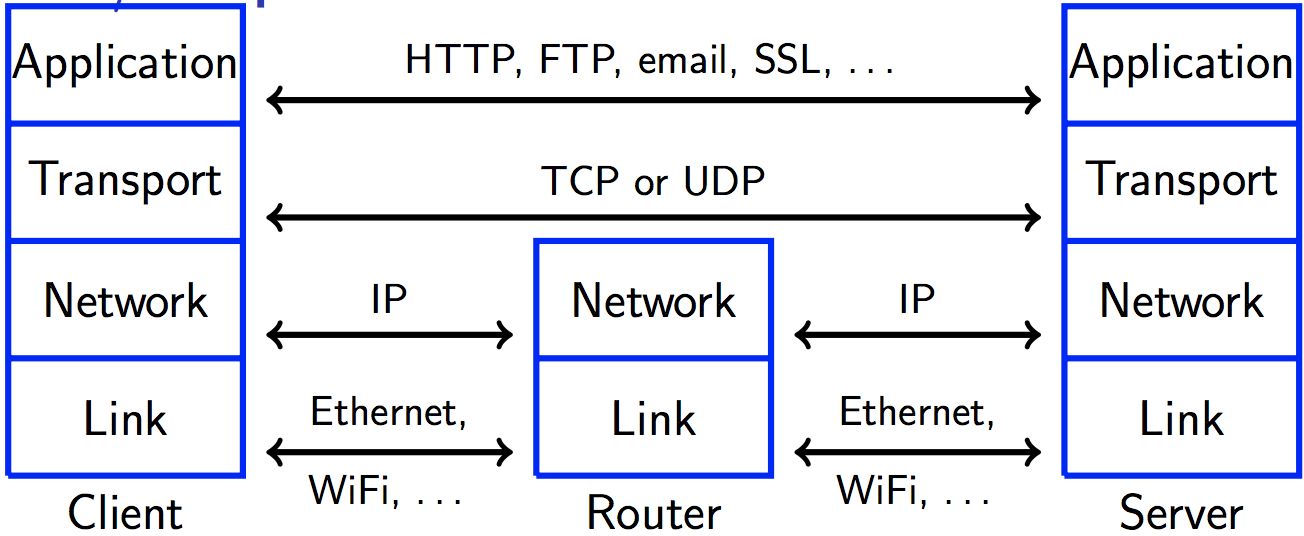
CS 458 Notes

# **Module 4 – Network Security**

1. **Network Concepts**
   1. Architecture of Internet



* 1. Characteristic of Internet
* No single entity control the network
* Traffic flows through nodes controlled by different entities
  + End node cannot control which nodes should traffic flow
  + Traffic is split up to individual packets
    - Each packet could be routed on different path.
* Nodes
  + Server
  + Laptop
  + Router
  + UNIX
  + Windows
* Communication Links
  + Wired or Wireless
* TCP/IP suite of protocols
  + Packet format
  + Routing of packets
  + Dealing with packet loss
  1. TCP/IP Protocol Suite
* Participants knew and trusted each other
* Design to address non-malicious error – packet drop
  + Not design for malicious errors



1. **Threats in Network**
   1. Port Scan

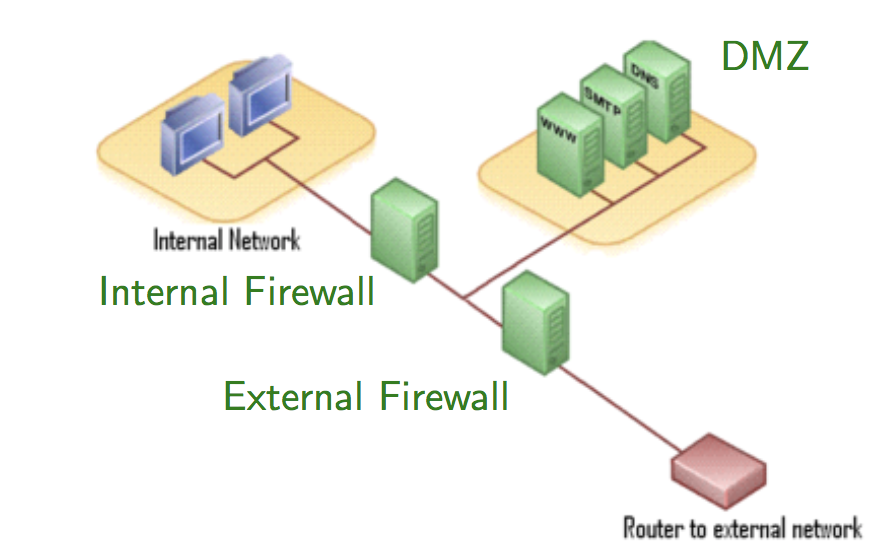
* Loose lipped system reveal information that could facilitate an attack
  + Login application reveal information about OS/user is valid
  + Server return version information
  + E.g: Ashley Madison ‘s password reset form
    - Invalid validate of email entered, will send password reset to emails in database is valid
* Nmap tool can identify many applications
  + Useful to attackers and system administrator
* Attacker Goal: find application with exploitable flaw
  1. Intelligence
* Social Engineering
  + Gather sensitive information from person
  + Exploits thru person’s willing to help
* Dumpster diving
* Eavesdropped thru oral communication
* Google
  + Find information that shouldn’t be there
  1. Eavesdropping and wiretapping
* Owner of Node can monitor communication flow
  + Aka eavesdropping / passive wiretapping
  + Require modification or fabrication of communication
* Always assume communication is wiretapped
  1. Communication Media
* Copper wire
  + Allow a physically close attack to eavesdrop without physical contact
* Optical fibre
  + No inductance
  + Any signal loss by splicing is detectable
* Microwave / satellite communication
  + Signal path at receiver tend to be wide
* Attack are feasible in practice, but require physical cost and effort
* Wi-Fi
  + Easily intercepted by anyone
    - Don’t need additional hardware
  + Other security issues
    - Physical barriers help against random devices being connected to a wire network
      * Useless in wireless network
    - Need authentication mechanism to defend
  1. Misdelivered Information
* Imagine computers connect within a company
  + Packet sent to multiple nodes, not only to intended receiver
  + Attacker can use packet sniffer to capture these packets
* Wrongly addressed emails
  1. Impersonation
* Impersonate a person by using stolen password
  + Thru guessing attack
  + Exploit default password that wasn’t change
  + Sniff password
* Exploit trust relationship between accounts
  1. Spoofing
* Object masquerades(falsify) as another one
  + Node, person, URL, email, Wi-Fi, etc.
* URL Spoofing
  + Exploit typo: [www.uwaterlo.ca](http://www.uwaterlo.ca)
  + Exploit ambiguities
  + Exploit similarities
* Used in phishing attack
* Spoofing is also used in session hijacking and man in middle attacks
  1. Session hijacking
* TCP protocol sets up state at sender and receiver end nodes and use this state for exchanging packets
  + Attack by hijack session and
* Web server have piece of data – cookie to re-identify client for future visits
  + Sniff and steal cookie and act as client
  1. Traffic analysis
* Existence of communication between 2 parties is sensitive and should be hidden
* TCP/IP has each packet include unique address
  + For sender and receiver end nodes
  + Make traffic analysis easy
  + BUT attacker can learn these address by sniffing packets
  1. Integrity attacks
* Attackers modify packet
  + Change payload
  + Change address
  + Replay previously seen packets
  + Delete or create packets
* Caused by line noise, network congestion/software errors
* E.g.: DNS cache poisoning
  + DNS maps host names to addressed, stored in packets
  + Attacker can create wrong mappings
  1. Protocol Failures
* TCP/IP assume nodes implement protocol faithfully
  + Node to slow down if network is congested
* But attacker could ignore these requests
* Caused by implementation do not check if packet is well formatted
  1. Website vulnerabilities
* Web site defacements
* Access URL with web server return HTML code
  + Tell browser to display web page
  + How to interact with web server
* Attacker send malicious URL to web server
  + Exploit buffer overflow
  + Invoke shell or another program
  + Feed malicious input
  + Access sensitive files
* HTTP protocol is **stateless**
  + Web server ask client to keep state when return a web page to submit this state when accessing next web page
    - Use of cookie or URL
  + Attacker can submit modified state information
* Cross-site (XSS) / request forgery(CSRF) attacks
  + Form of code injection
  + Attackers add HTML code to someone web page
    - Users download and execute code when downloading the web page
  + XSS – steal sensitive information contained in the web page
  + CSRF - perform malicious actions at web site
    - Money transfer
  1. Denial of Service
* Cutting a wire / jamming a wireless signal
  + Flood node by overloading its connection
* Ping flood
  + Node receive ping packet is expected to generate a reply
  + Attacker could overload victim
* Smurf attack
  + Spoof source address of sender end node in ping packet
    - Setting it to victim address
  + Broadcast ping pocket to all nodes in a LAN
* Exploit knowledge of implementation details about a node
  + Make node perform poorly
* SYN flood
  + TCP initialize state by having 2 end nodes exchange 3 packets
    - SYN, SYN-ACK, ACK
  + Server queue SYN from client and remove it when receiving ACK is received
  + Attackers to send many SYNs but no ACKs
* Craft packets such that its hashed into same bucket in hash table
* Black hole attack
  + Packet drop attack
  + Route packets in internet based on distributed protocol
  + Router inform one another cost to reach set of destinations
  + Malicious router announces low cost for victim destination
    - Discards any traffic destined for victim
  + Happen due to router misconfiguration
* DNS Attack
  + DNS cache poisoning
    - Lead to packets being routed to wrong host
  1. Distributed Denial of Service
* DoS use a single attacking machine
  + Possible to identify machine to have router discard its traffic
* DDoS = more difficult due to more attacking machines
* Attacking machines participate without knowledge of owner
  + Attacker break in using Trojan, buffer overflow then install malicious software
    - Make machine = zombie/bots
    - Wait for attack commands from attacker
  + Network of bots = botnets
  1. Reflection & Amplification DDoS
* Attack where victim is flooded with legitimate looking traffic
  + Originate from unsuspecting network nodes on the internet
* Amplification:
  + Vulnerable network node
    - Home Wi-Fi router runs a service SNMP
  + Response to queries with much more data than query itself
* Reflection
  + Attacker spoof source address of its queries to victim
  + Make it vulnerable network nodes send (reflect) response to the victim
* Hard to combat
  + Response traffic is coming from innocent nodes
  + Hard to identify real source (bots) due to spoofing
  1. SNMP Reflected Amplification Attack
* Enabled on home routers and similar devices
* Have bad default values for security settings
  + Default settings for most routers is “public” to max potential reflectors
    - Allow devices to identify them
  + SNMP allow “GetBulkRequest” query
    - Send back an order of magnitude more data as the request
  + Botnet of such device generate large amount of data
    - Short period of time
    - DDoS the victim
  1. New Generation Botnets
* Virus, worm, Trojan for propagation
  + Exploit vulnerabilities
* Stealth to hide from owner
* Code morphing = harder detection
* Bot usable for different attacks
* Distributed, dynamic & redundant control infrastructure
  + P2P system for distributing updates aka “Fast Flux”
    - Single host maps hundreds of addresses of infected machines
    - Proxy to malicious websites
    - Machines constantly swapped in/out of DNS – more difficult to track
  + Domain Generation Algorithm
    - Infected machine generates large set (50k) of domain names that changes daily
    - Contacts a random subset of these names for update
    - Control botnet = authorities take control of 50k domain names
  + Earlier worms – Nimda, Slammer written by hacker for fame
    - Goal = spread worm as fast as possible (cause disruption and help detection)
  + Today: botnet controlled by crackers looking for profit = rent them out
  + Spread more slowly infected machines might not be detected for weeks
  1. Active Code
* Reduce load on server = ask client to execute code on behalf
  + Java, JS, ActiveX, Flash, etc
* Invoke another application
  + Word, iTunes, Steam
* These may be inadvertently
  + XSS attack
* Dangerous for client
  + Java 1.1 ran in sandbox with limited capabilities
    - No writing to a file
    - No talking to random network nodes
  + Java 1.2 can break out of sandbox by user’s permission
  + Java 7 run signed applets out of sandbox by default
    - 3th Party applications
      * problem for all browser
      * malicious input parameters, macros, etc.
    - Privileged
      * Applications run with unrestricted access
    - Sandbox
      * Run with restricted access
      * Intended to protect computer and personal information
  1. Script kiddies
* Download scripts and raise an attack with minimum effort
* Tools allow easy building of individual attacks (based on all previous points) from existing exploits

1. Network Security Controls
   1. Design & Implementation

* Use controls against security flaws
* Always check for input
  + Never trust input from client
  + use a list of allowed characters
  + banned those on black list
  1. Segmentation and Separation
* Don’t put company server all on single machine
* Deploy on multiple machines
  + Based on functional and access requirements
* If machine is broken into = some service affected
  + Better than all
* E.g.: Web server of a company need to be accessible from outside
  + More vulnerable
  + Should not be trusted by other servers of the company and deployed outside company firewall.
  1. Redundancy
* Avoid single point of failure
  + Can be caused by attackers or natural events (disk crash, power failure, earthquake)
* Deployed in a redundant way on multiple machines
  + Different software = genetic diversity at different lcoations
* Do constant backup
  + Keep backup at safe place
  1. Access Controls
* ACLS on router
  + Traffic to company goes to a few nodes
  + Prevent flooding attack
    - ACLs drops packet with source/destination address
  + Expensive for high traffic routers
  + Difficult to gather logs for forensics analysis
  + Source address of packets might be spoofed (not effective)
* Firewalls
  + Designed to filter traffic
  + Based on other criteria than just packet address

1. Firewalls

* Castles of internet ages
* All traffic go in/out must go thru gates (choke points)
  + Wireless access point should be outside of firewall
* Choke point examine traffic
  + Especially Incoming and might block access
* 2 strategies
  + Permit everything unless explicitly forbidden
  + Forbid everything unless explicitly allowed
* Do not protect attacks from internal nodes
  + Need multiple layer to defend
  1. Packing Filtering gateways
* Simplest type
* Make decision based on header of packet
  + Header contain
    - Source
    - Destination address
    - Port number
      * Can be used to infer type of packet
      * 80 = web
      * 22 = SSH
      * etc.
* ignore payload of packet
* can drop spoofed traffic
  1. Stateful Inspection firewalls
* More expansive than packet filtering
* Keep state to identify packets that belongs together
  + Client within the company open a TCP connection to server outside the company
  + Firewall must recognize response packets from server and let them thru only
* Application layer protocol (FTP) require additional (expensive) inspection of packet content to figure out kind of traffic.
* IP layer can fragment packets
  + Firewall might have to re-assemble packets
  + For Stateful inspection
  1. Application proxy
* Client talk to proxy, proxy talk to server
  + Specific application for communication
    - Email, web, etc.
  + These are not transparent as packet filtering or Stateful inspection
  + Intercepting proxy require no configuration by client
    - Other traffic is blocked
* For users within the company wanting to access a server outside the company
  + Forward proxy
* And vice versa
  + Reverse proxy
* Proxy have full knowledge about communication
  + Can do sophisticated processing
    - Limit types of allowed database queries
    - Filter URLs
    - Log all emails
    - Scan for virus
* Can also do strong user authentication
  1. Personal firewalls
* Firewall runs on home computer
  + Important for computers that are always online
* Forbid everything unless explicitly allowed
  + Define for communication originating from other computers
  + Maybe also for communication originating on user computer
* Protect against attacks on servers running on computer
  + Servers that are wrongly configured and that allow access from other computers (cannot be configured to disallow this)
  + Servers that have exploitable bug
  1. DMZ
* Subnetwork that contains an organisation external services
  + Accessible to the internet
* Deploy external and internal firewall
  + External firewall protects DMZ
  + Internal firewall protects internal network from attacks lodged in DMZ



1. Honeypots / Honeynets
   1. Intro

* Set up an unprotected computer or an entire network as a trap for an attacker
* System has no production value
  + Any activity is suspicious
    - E.g.: email = spam.
* Observe attacker to learn about new attacks
  + Identify and stop attacker
  + Divert attacker from attacking real system
* Attacker should not be able to learn that attacked system is a honeypot
  + Cat & mouse game
* Attacker might break in using honeypot
  1. Types
* Low interaction
  + Daemon that emulates one or multiple hosts
    - Running different services
  + Easy to install and maintain
  + Limit amount of information gathering possible
    - Easier for attacker to detect than high interaction honeynet
* High Interaction
  + Deploy real hardware and software
    - Use stealth network switch or key logger for logging data
  + More complex
    - Can capture lots of information
    - Can capture unexpected behaviour by attacker

1. Intrusion Detection System
   1. Intro

* Firewalls! = protect against inside attackers or insiders making mistakes and can be subverted.
* IDSs are next line of defence
* Monitor activity to identify malicious or suspicious events
  + Receive events from sensors
  + Store and analyse them
  + Act if necessary
* Host-based and network based IDSs
* Signature-based and heuristic/anomaly based IDSs

* 1. Host-based and network based IDSs
* Host-based
  + Run on host to protect this host
  + Can exploit lots of info (packets, disk, memory)
  + Miss out on information available to another attacked host
* Network-based
  + Run on dedicated node to protect all hosts attached to a network
  + must rely on information available in monitored packets
  + Typically, more difficult to subvert
  + Distributed IDSs combine the two of them.
* Distributed based
  + Use both host-based and network based.
  1. Signature based IDSs
* Each known attack has its signature
  + SYNs to ports that are not open could be part of a port can
* Signature-based IDSs try to detect attack signatures
* Fail for new attackers or if attacker manages to modify attack such that its signature changes
  + Polymorphic worms
* Might exploit statistical analysis
  1. Heuristic/anomaly based IDSs
* Look for behaviour that is out of ordinary
* Model good behaviour and raise alert when system activity no longer resembles this model
* By model bad behaviour and raise alert when system activity resembles this model
* Overtime, IDS learn to classify unknown events as good or suspicious
  + Due to machine learning.
  1. IDS overall
* Stealth mode
  + 2 network interface, one for monitoring traffic, another for administration and for raising alarms
  + first have no published address
    - do not exist for routing purpose
* Responding to alarms
  + Types of response depends on impact of attack
  + From writing a log entry to calling a human
* False positive / negative
  + Might lead to real alarms being ignored
  + Need to strike balance between 2 interface
  + Need to be monitored to be useful.

# **Module 5 – Internet Application Security and Privacy**

1. Basic of Cryptography
   1. Intro

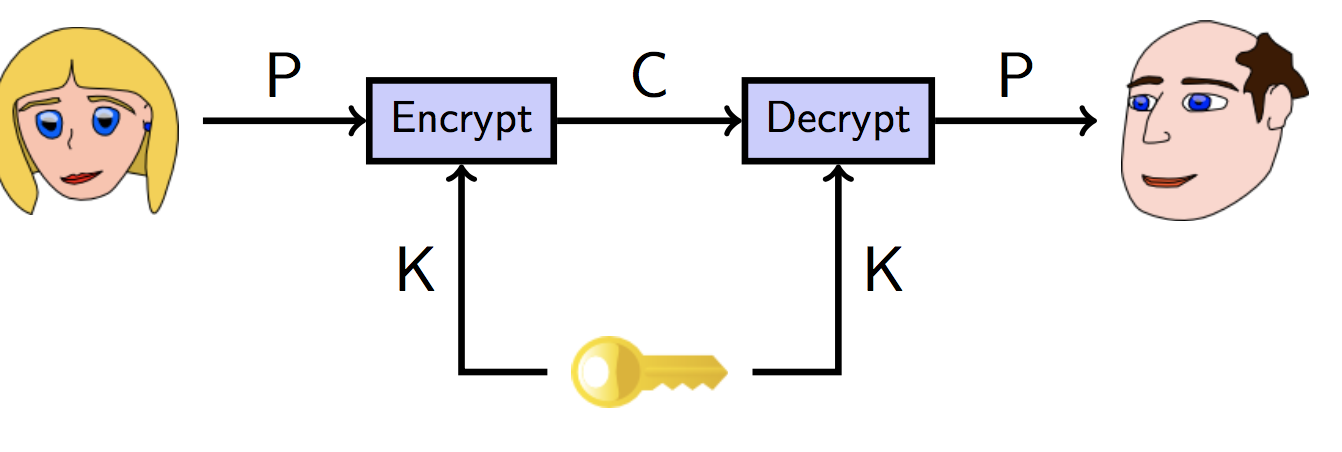
* Cryptography = secret writing
  + Turing plaintext to cipher-text
* Cryptanalysis = breaking secret messages
  + Recover plaintext from ciphertext
* Send secure message over an insecure medium
  1. Common terminology
* Good people
  + Alice, Bob, Carol, Dave
* Bad people (passive eavesdropper)
  + Eve
* Active man in the middle
  + Listen to, and modify, insert or delete transmitted message
  + Mallory
* Trusted third party
  + Trent
  1. Building Block

3 components:

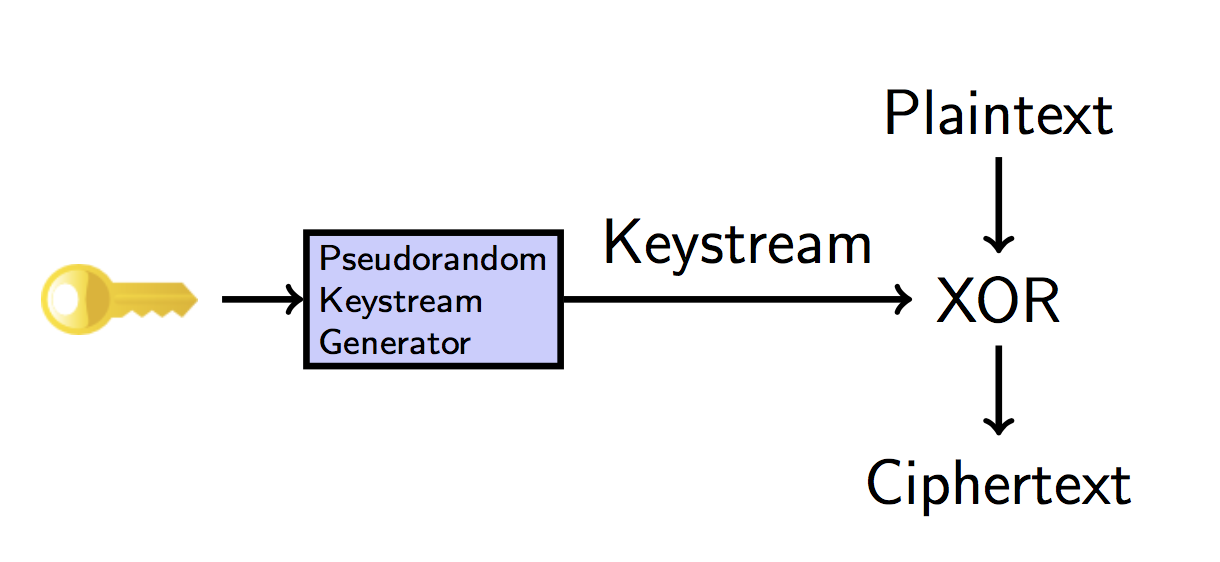
* Confidentiality Components
  + Prevent Eve from reading Alice’s message
* Integrity Components
  + Prevent Mallory from modifying, deleting, insert Alice’s message
* Authenticity Components
  + Prevent Mallory from impersonating Alice
  1. Kerckhoffs’ Principle
* Security of a cryptosystem should not rely on a secret that is expensive or hard to change
* Do not have secret encryption methods
  + Use large class of encryption methods
  + Make class public information
  + Use a secret key to specify which you are using
    - Easy to change key
* The system is at most secret as the number of key
  + If Eve can try them all = she finds the right one
  1. Strong cryptosystems
* What information do we assume attack has when trying to break the system?
  + Know algorithm
  + Know part of plaintext or number of corresponding plaintext/ciphertext pairs.

1. Secret key encryption

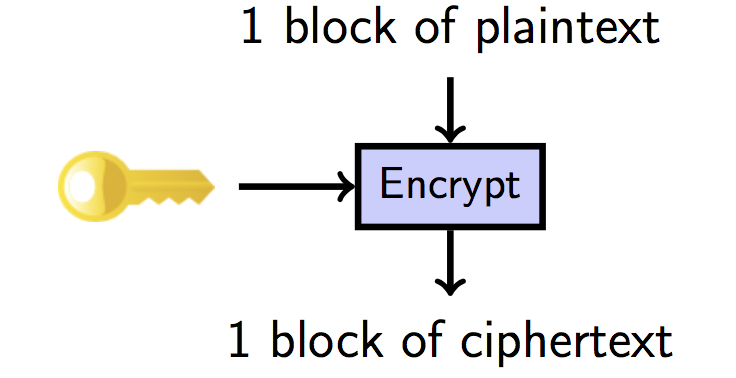
* Simplest form of cryptography
* Aka symmetric encryption
* Key Alice use to encrypt the message = key bob use to decrypt



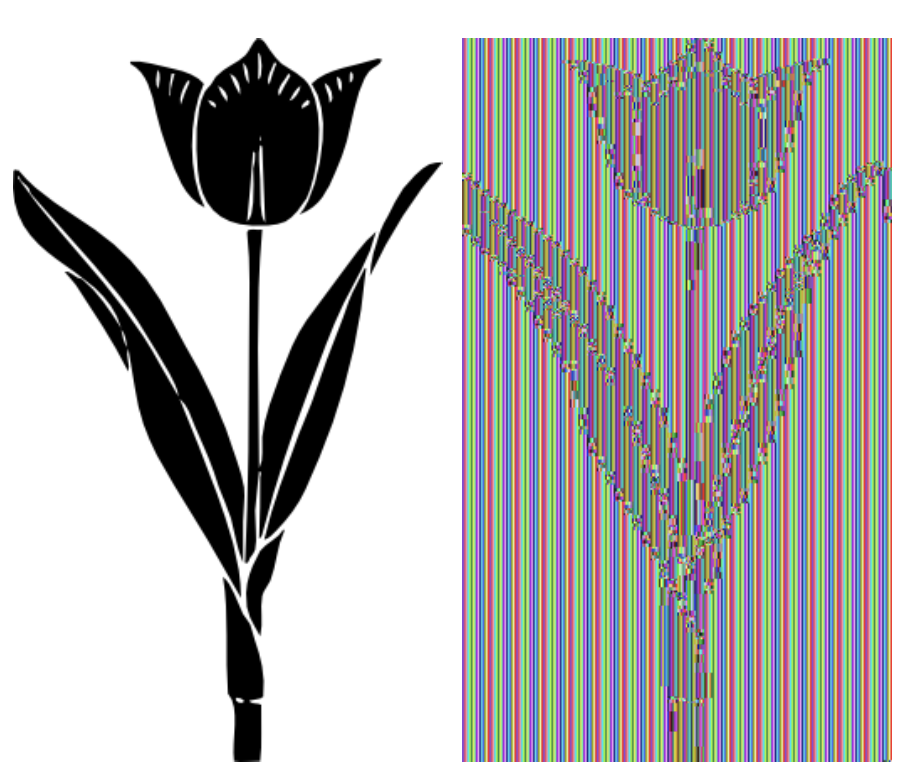
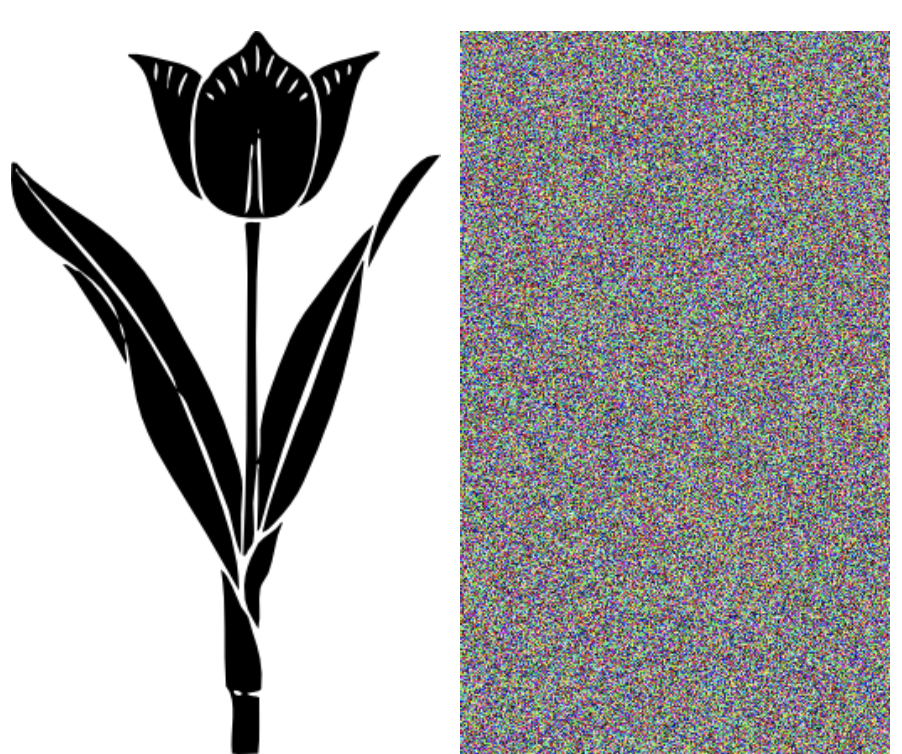
* 1. Secret Key encryption
* Eve not knowing the key = should not be able to recover the plaintext
  1. Perfect secret key encryption
* Unbreakable cryptosystem
  + 1. One time pad
  + Key is truly random bit string of the same length as the message
  + Encrypt and decrypt function are jus XOR
  + Hard to use correctly
    - Key must be truly random
    - Not pseudorandom
    - Key must never be used more than once
  1. Computational security
* Contrast to OTP’s “perfect” security
* Most cryptosystem have a computational security
  + Certain they can be broken
  1. 40 bit crypto
* 240 = 1,099 = 511,627,776 possible keys
  + Computer = 18 hours to break
  + One lab = 11 mins
  + BOINC = 5ms
  1. 56 bit crypto
* US Government Standard for a long time
  + DES
* 256 = 72,057,037,927, 936 possible keys
  + Computer = 134 year to break
  + One lab = 16 months
  + BOINC = 5 minutes
  1. 128 bit crypto
* modern standard
* 2128 = 340,282,366,920,938,463,463,374,607,431,768,211,456 possible key
  + Computer = 635 thousand million x3 years to break
  + One lab = 6 thousand million x3 years
  + BOINC = 49 thousand million x2 years
  1. Other better strategy
* Don’t break the crypto at all
* Always weaker parts of the system to attack
  + Principle of easiest penetration.
* Make sure in cryptography, information transfer is not weakest link
  1. Types of secret key cryptosystems
     1. Stream ciphers



* Stream cipher is what you get if you take the OTP
* But use a pseudorandom keystream instead of a truly random one
* RC4 is most common used stream cipher on internet today
  + But its deprecated
* Advantage
  + Fast
  + Useful to send lot of data securely
* Disadvantage
  + Tricky to use correctly
    - E.g.: encry*pt* the 2 message with same key
    1. Block ciphers

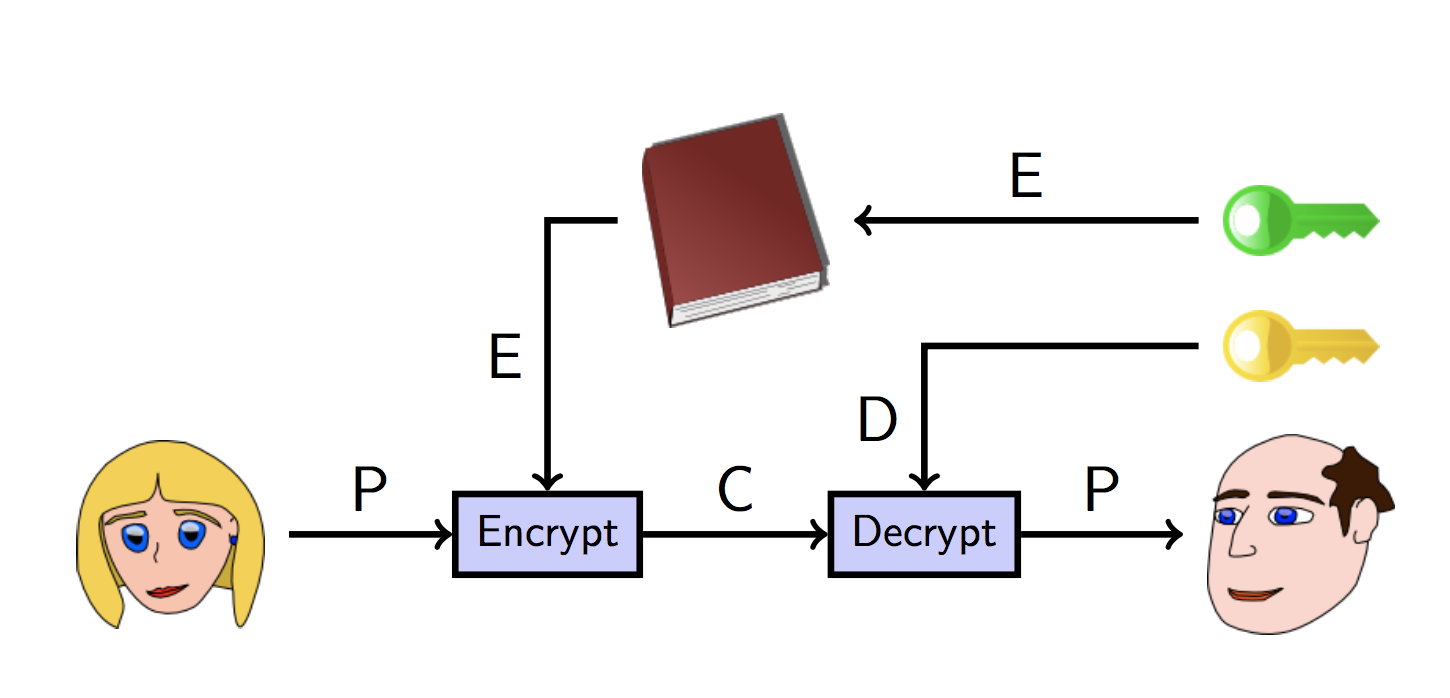


* stream ciphers operate on message one bit at a time
* block ciphers operate on the message one block at a time
  + 64 or 128 bit long
* E.g.: AES
* Mode of operation
  + If plaintext is larger than one block
    - Choice of what to do with multiple blocks is called the mode of operation of the block cipher.\
  + Simplest thing to do = encrypt each successive block separately
    - AKA Electronic Code Book mode
    - Disadvantage: if there are repeated blocks in plaintext, you will see a repeating pattern in the ciphertext.
  + Better modes:
    - Cipher block chaining (CBC), Counter (CTR), Galois Counter (GCM)
  + Patterns in plaintext are no longer exposed
    - Need an IV (initial value) which acts like a salt

* 1. Key exchange
* Use an algorithm.

1. Public key encryption



* 1. Intro
* Invented in 1970
* Aka asymmetric cryptography
  + Alice send message to Bob without any pre-arranged shared secret
  + Secret key cryptography = same encrypt & decrypt key
  + NOT FOR PUBLIC KEY CRYPTO
    - One key for encryption
    - Different key for decryption
* RSA, ElGamal, ECC, NTRU, McEliece
  1. Public key sizes

|  |  |  |
| --- | --- | --- |
| AES | RSA | ECC |
| 80 | 1024 | 160 |
| 116 | 2048 | 232 |
| 128 | 2600 | 256 |
| 160 | 4500 | 320 |
| 256 | 14000 | 512 |

* 1. Hybrid cryptography
* Longer key, public key crypto takes long time to calculate compared to secret key cryptography
  + Large message = too slow encryption
  + Solution: hybrid approach
* E.G.
  + Pick random 128-bit key K for a secret key
  + Encrypt large message with key K
    - Using AES
  + Encrypt the key K using a public key cryptosystem
  + Send encrypted message and encrypted key to Bob

Note:

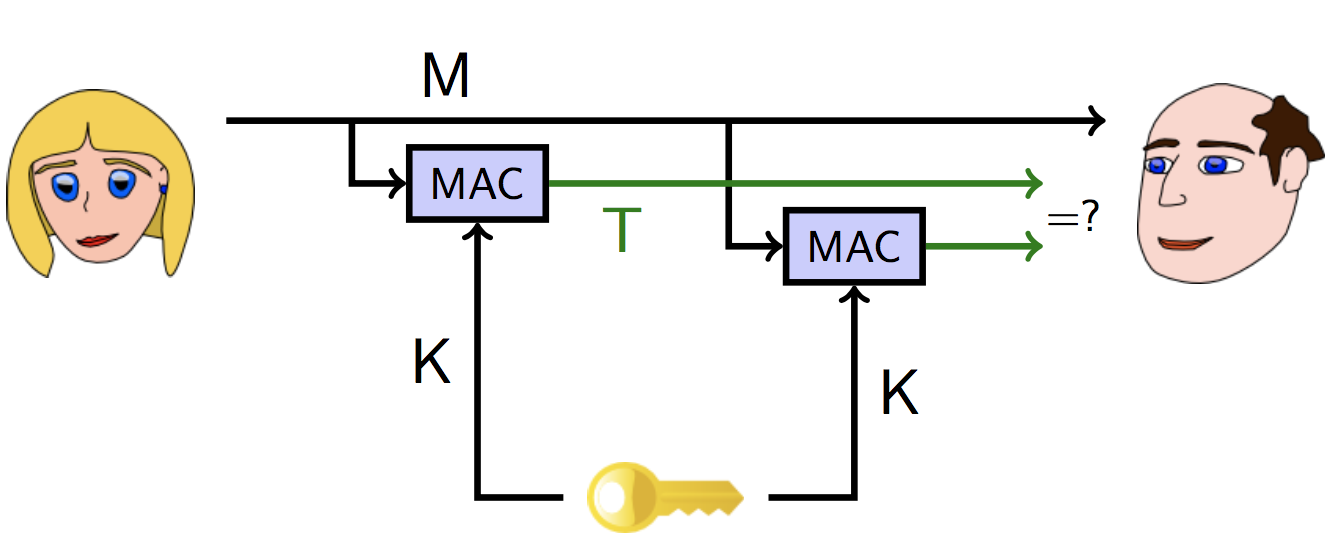
* Safe from Eve, but need to be aware of Mallory
  + Can modify them in transit
  + Doesn’t need to know content but can change it in an undetectable way.
    - Bit flipping attack on stream ciphers

1. Integrity
   1. Components

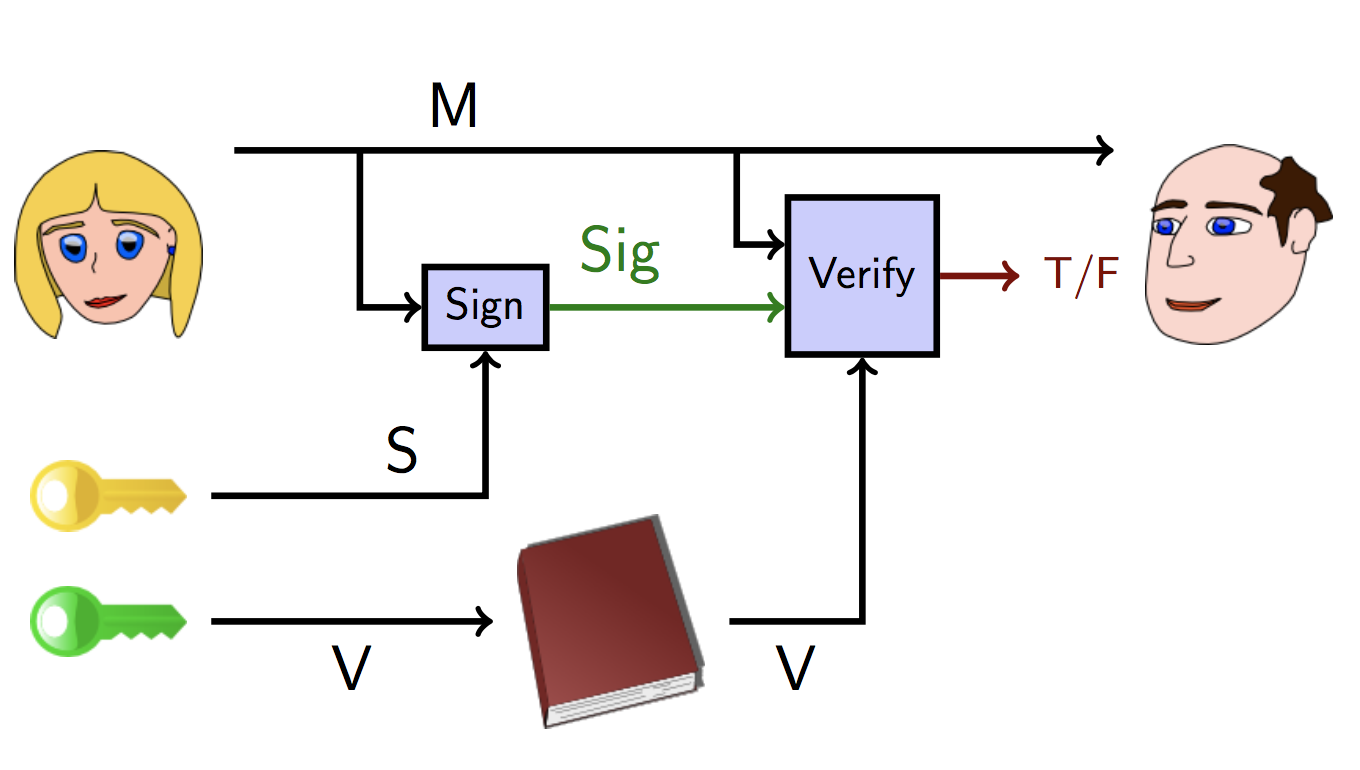
* How to tell if message hasn’t changed in transit
  + Use checksum
    - Add up all bytes of a message
    - Last digits of serial numbers of Credit Card, ISBN are checksum
      * Alice compute the checksum of the message
      * Stick it to the end before encrypting to Bob
      * Bob receive and check the checksum andverifies if its correct
  1. Why it doesn’t work
* With checksum method, Mallory can easily change the message in such a way that checksum stays the same
  + Need a cryptographic checksum
  + Hard for Mallory to fund a second message with the same checksum
  1. Cryptographic hash functions
* Hash function h takes an arbitrary length string x and computes a fixed length string
  + Y = h(x) aka message digest
* E.g: MD5, SHA-1, SHA-2, SHA-3
* 3 key properties
  + Preimage resistance
    - Given y, it’s hard to find x such that h(x) = y
    - A preimage of x
  + Second preimage-resistance
    - Given x, it’s hard to find x’ != x such that h(x) = h(x’)
    - Second preimage of h(x)
  + Collison-Resistance
    - Hard to find any 2 distinct value values x, x’ such that h(x) = h(x')
* What is “hard”
  + SHA-1 takes 2160 to find a preimage/second preimage
    - 280 to find a collision using brute force
    - collisions are always easier to find than preimages or second preimages due to birthday paradox
  + can’t send an unencrypted message and its had to get integrity assurance
* Hash functions provides
  + Integrity guarantees only when there is a secure way of sending/storing message digest
  + E.G.
    - Bob publish a hash of his public key (message digest) on his business card
    - Putting the whole key on there would be too big
    - Alice can download bob’s key from the internet hash itself
      * Verify result that matches the message digest on bob’s card

1. Authentication
   1. Message Authentication codes (MAC)

* Same trick for encryption
  + Have a large class of hash functions and use a shared secret key to pick the correct one
* Only those who know the secret key can generate, check the computed hash value – aka tag
* Also, known as MACs



* 1. Combining ciphers and MACs
* Often need both confidentiality and message integrity
* There are multiple strategies to combine a cipher and MAC when processing a message
  + Encrypt then MAC, MAC then encrypt, Encrypt and MAC
* Encrypt then MAC is recommended
* Crypto library already provides an authenticated encryption mode
  + Securely combines two operations
  + Don’t have worry about getting it
  + GCM, CCM, OCB mode
  1. Repudiation
* Suppose Alice and Bob share a MAC key k
* Bob receive message M along a valid tag T
  + Computed using k
* Then…
  + Bob can be assured that Alice is the one who sent the message M
  + Hasn’t been modified since she sent it
    - Signature on the message
  + Bob can’t show M and the tag T to Carol to prove Alice sent the message
* Alice can claim Bob made up message M and calculate tag T himself
* WANT TO AVOID THIS CASE – repudiation
* Some interaction should be repudiable
  + Private conversations
* Some are non-repudiable
  + Electronic commerce
  1. Digital Signature
* For non-repudiation
  + We want is a true digital signature
* Key properties
  + If Bob receives a message, with Alice’s digital signature, then
    - Alice, not an impersonator, sent the message like a MAC
    - The message has not been altered since it was sent like a MAC
    - Bob can prove these facts to third party
      * Additional property not satisfied by a MAC.
* To arrange this, use similar techniques to public key cryptography
* Making digital signature
  + Remember public key crypto
    - Separate keys for encryption and decryption
    - Give everyone copy of encryption key
    - Decryption key is private
  + To make digital signature
    - Alice sign the message with her private key
  + To verify Alice’s signature
    - Bob verifies the message with his copy of Alice’s public verification key.
    - If it verifies correctly = signature is valid



* 1. Hybrid Signatures
* Like public key crypto
  + Sign large message is slow
* Can hybridize signatures to make them faster
  + Alice sends the unsigned message and a signature on a hash of the message
  + The hash is much smaller than the message
  + Faster to sign and verify
* Authenticity and confidentiality are separate
  1. Combining public key encryption and digital signatures
* Alice has 2 different key pairs
  + Encryption, decryption key pairs
  + Signature, verification key pairs
* Alice use Bob’s encryption key to encrypt a message destined for Bob
* Alice use her signature to sign the ciphertext
* Bob use Alice’s verification key to check the signature
* He uses his decryption key to decrypt the ciphertext
  1. Relationship between key pairs
* Alice’s signature, verification key pair is long-lived.
* But encryption, decryption key pair isn’t
  + Give perfect forward secrecy
* When creating a new encryption, decryption key pair,
  + Alice use her signing key to sign the new encryption key
  + Bob use Alice’s verification key to verify the signature on this new key
  + If Alice’s communication with Bob is interactive
    - Can use secret key encryption and does not need an encryption, decryption key pair at all.
  1. Key management problem
* Hardest problem of public key cryptography
* How to find verification key?
  + Know it personally (manual keying)
    - SSH
  + Trust a friend to tell him (web of trust)
    - PGP
  + Can trust some 3th party (Certificate Authority)
    - TLS/SSL
  1. Certificate authorities
* CA = trusted third party who keep a directory of people/organisation verification keys.
  + Alice generate signature and verification key pair, send verification key to CA with bunch of information, both signed with Alice’s signature key to CA.
* CA ensure personal information and Alice’s signature is correct
  + generate a certificate consisting of Alice’s personal information and verification key
  + Certificate is signed with CA’s signature key
* Everyone is assumed to have copy of CA’s verification key
  + Verify signature on the cert
  + Multiple level of certificate authorities
    - Level n CA issues certificates for level n+1 CAs
    - Public key infrastructure (PKI)
* Need to have only verification key to root CA to verify certificate chain.
  1. Summary
* Put these knowledge into protocol
* Hard
  + Pieces all work individually
  + Does not mean build out of them will
* Common mistake
  + Use the same stream cipher key for 2 message
  + Assuming encryption also provides integrity
  + Falling for replay attacks or reaction attacks

1. Security Controls using Cryptography
   1. Intro

* Used for some separation, with key need to be available to legitimate users but not adversaries
  + If web browser can decrypt file containing saved passwords
    - Adversary can read your web browser too
  1. Program and OS security
* Using secret key crypto can be problematic
  + But public key is OK
    - Local machine need access to public part of the key
    - Only encryption and signature verification
    - App can be installed only if digitally signed by vendor or upgraded by original developer
    - OS may allow execution of programs if signed (iOS)
  1. Encrypted code
* Research into processors that will only execute encrypted code
* Processor will decrypt instructions before executing
  + Decryption key is processor-dependent
* Malware won’t be able to spread without knowing a processor’s key
  1. Encrypted data
* Hard drive encryption protects data when laptop got lost/stolen
* Do not protect data against other users who use laptop
  + Or people installing malware on laptop
  + Or people physically extracting the decryption key from laptop’s memory
  1. OS authentication
* Authentication mechanism used
  + Salted hashes
* Require hardware token
  + People are bad at doing cryptography in their heads
  1. Network security and privacy
* Primary use for cryptography
  + Separate security of the medium from security of the message
* Entities you can communicate with over network are inherently less trustworthy
* Used at every layer of network stack for security and privacy:
  + Link – WEP, WPA, WPA2
  + Network – VPN, IPsec
  + Transport – TLS/SSL, Tor
  + Application – SSH, PGP, OTR

1. Link-layer security
   1. Intro

* Intend to protect local area networks
  1. WEP
* Wired Equivalent Privacy
* Enforce 3 security goals: not actually enforced
  + Confidentiality
    - Prevent leakage of content
  + Access Control
    - Prevent loss of controls from infrastructure
  + Data integrity
* Brief description
  + Alice and Bob shared secret key k
    - Either 40 or 104-bitlong.
  + To transmit message. Require a checksum c(M)
    - Do not depend on k
  + Pick a random IV v and generate a keystream
    - RC4(v, k)
  + XOR [M, c(M)] with keystream to get the ciphertext
  + Transmit v and ciphertext over wireless link
  + Upon receipt of v and ciphertext
    - Use v and shared k to generate keystream RC4(v, k)
    - XOR the ciphertext with RC4(v, k) to get [M’, c’]
    - Check c’ = c(M’), if true = accept message transmitted.
  1. WEP data integrity
* If checksum used is CRC-32
  + Poor choice – already a CRC in the protocol to detect random errors
  + CRC can’t help protecting against malicious errors
* CRC has 2 important properties
  + Independent of k and v
  + Its linear: c(M XOR D) = c(M) XOR c(D)
  1. WEP access control
* If adversary wants to inject a new message F onto WEP-protected network
  + Need a single plaintext/ciphertext pair
    - Can derive a value of v and corresponding keystream RC4(v, k)
  + Then C’ = [F, c(F)] XOR RC4(v, k) and transmit **v, C’**
  + C’ = correct encryption of F = message must be accepted
  1. WEP authentication protocol
* How did adversary get the single plaintext/ciphertext pair required for the attack?
  + Authentication protocol give it to adversary for free
  + Disaster in design
* Authentication protocol supposed to prove that a certain client knows the shared secret k.
* Authentication protocol
  + Access point send a challenge string to client
  + Client sends back the challenge, WEP-encrypted with the shared secret k.
  + Wireless access point checks if the challenge is correctly encrypted,
    - If true, accept client
* Adversary seen both plaintext and ciphertext of the challenge
* Another challenge: enough not only to inject packets,
  + Also execute authentication protocol itself.
  1. WEP decryption
* The ability to modify and inject packets also leads to ways the adversary can decrypt packets
  + Access point knows k,
  + Turns out, adversary can trick into decrypting the packet
* Note that none of the attack used stream cipher was RC4
  + Recovered k.
  1. Recovering WEP key
* Series of analyses of RC4 in particular
  + Problem: turns out RC4 is used with similar keys, the output keystream has a subtle weakness
    - Reason why WEP use RC4
* Observations have led to program that can recover either 104 bit or 40-bit WEP key in under 60 second, most of the time
  1. Replacing WEP
* Wi-Fi Protected Access (WPA) was rolled out as a short-term patch to WEP
* Replacement protocol (IEEE, later call WPA2) were being developed
* WPA
  + Replaces CRC-32 with a real MAC
    - Avoid confusion with a Media Access Control address
  + IV is 48 bits
  + Key is changed frequently (TKIP)
  + Ability to use 802.1x authentication server
    - Maintains less-secure PSK (pre-shared key) mode for home users
  + Able to run on most older WEP hardware
* 802.11i standard was finalized in 2004
  + call WPA2 required for all products calling themselves “Wi-Fi” since 2006
* WPA2
  + Replace RC4 and MIC algorithm in WPA with CCM authenticated encryption mode using ARD
    - Considered strong except in PSK mode
    - Dictionary attacks still possible

1. Network layer security
   1. Intro

* Support every link in network had link-layer security
  + Not enough
  + Need security across networks
    - End to end
  1. Virtual Private Network – VPN
* Connect two or more network that are physically isolated and make them appear to be a single network
  + Alternately: connect single remote host (often a laptop) to one network
* Goal: adversary between networks should not be able read or modify the traffic flowing across the VPN
  + DoS and some traffic analysis still usually possible
  1. Setting up VPN
* One host on each side is the VPN gateway
  + Could be firewall or DMZ
  + In laptop scenario, it will of course be the laptop itself on its side
* Traffic destined for other side is sent to local VPN gateway
* local gateway use cryptography encryption to send traffic to remote VPN gateway
  + done by tunnelling
* remote gateway receives, decrypt and send them to appropriate destinations
  1. Tunnelling
* Send message of one protocol inside message of another protocol, out of their usual protocol nesting sequence
* TCP-over-IP is not tunnelling
  + You are supposed to send TCP (a transport protocol) over IP (network protocol; one layer down in the stack)
* But IP-over-TCP is tunnelling
  + Going up the stack instead of down
  + Same as IP over IP – same place in the stack
  + Same as PPP (a link layer protocol, bottom of the stack) over DNS (application layer protocol, top of stack)
  1. IPsec
* One standard way to setup a VPN is by IPsec
* Many corporate VPNs use this open protocol
* 2 modes
  + transport
    - useful for connecting a single laptop to a home network
    - only the contents of the original IP packet are encrypted and authenticated
  + Tunnel mode
    - Useful for connecting 2 networks
    - Contents and the header of the original IP packet are encrypted and authenticated
    - Result is placed inside a new IP packet destined for the remote VPN gateway
  1. Other style of VPNs
* Microsoft PPTP
  + An older protocol
  + Many design flaws as WEP
  + Users migrating to IPsec
* VPNs based on ssh
  + Tunnel PPP over ssh
  + Efficiency concern but extremely easy to set up on a standard unix/linux box.
  + OpenSSH v4 supports IP over SSH tunnelling directly.

1. Transport layer security and privacy
   1. Intro

* Mechanism arrange to send individual IP packets securely from one network to another.
* Transport layer security mechanism transform TCP connection to add security
  1. TLS/ SSL
* Main transport-layer security mechanism
* Invented a protocol called secure sockets layer (SSL)
  + Meant for protecting HTTP connections
  + Could be used to protect any TCP-based connection
  + HTTP + SSL = HTTPS
  1. TLS at high level
* Client connects to server
  + Indicates it wants to speak TLS and which cipher-suites it knows
* Server send its certificate to client
* Contain
  + Host name
  + Verification key
  + Other administrative information
  + A signature from CA
  + Server choose which cipher-suite to use.
* Client validates server’s certificate
  + RSA authentication where one agreement of key
* Client and server run a key agreement protocol to establish keys for symmetric encryption and MAC algorithms from chosen cipher-suite
  + Server sign protocol message with signature key
* Communication now proceeds using chosen symmetric encryption and MAC algorithms
  1. Security properties provided by TLS
* Server authentication
* Message integrity
* Message confidentiality
* Client authentication
  1. Success of TLS
* Most successful Privacy Enhancing Technology (PET)
* Why?
  + Comes with your browser
  + Just works, without you having to configure anything
  + Most of the time
    - Protects the privacy of your communication
    - Increasingly important due to success of Wi-Fi.
  1. Privacy Enhancing Technologies
* Encryption to protect other stuff other than messages
  + Such as metadata
    - Who is it sending to
* Wants to hide the existence of the message
  1. Tor
* Successful privacy enhancing technology that works at transport layer
  + Hundreds of thousands of users
  + March 2015 = 2 million users
* Normally a TCP connection make on internet automatically reveals your IP address
* Tor allows you to make TCP connections without revealing your IP address
* Most commonly used for HTTP web connections.

* + 1. How it works
* Scattered around the internet are about 7k Tor nodes
  + Aka onion routers
* Alice wants to connect to a web server without revealing her IP address
  + She will pick 1 of Tor nodes (n1) and use public key encryption to establish an encrypted communication channel
    - Like TLS
  + Alice tell n1 to contact second node n2
    - Establish new encrypted communication channel
    - Tunnelled within n1
  + And carry on till last node
    - Usually 3 nodes
    1. Sending message with Tor
* Alice now share 3 secret keys
  + K1 for n1
  + K2 for n2
  + K3 for n3
* When Alice wants to send a message M,
  + She sends Ek1(Ek2(Ek3(M)))
* Node n1 use k1 to decrypt the outer layer
* Then pass result to n2
  + Ek2(Ek3(M))
* Like previous step, node n3 use K3 to decrypt and send M to client
  + 1. Replies in Tor
* If client replies with message R, it will send to node n3
* N3 will encrypt R and send Ek3(R) to n2
* Continue till it reach Alice
* Alice will decrypt with k1, k2,k3
  + 1. Advantage
* N1 know Alice is using Tor and her next node is n2
* But do not know what’s next
  + 1. Disadvantage
* Connection between n3 and website is not encrypted
* Need HTTPS for end to end encryption
  1. Anonymity vs pseudonymity (IMPORTANT)\*\*\*\*\*\*
* Tor provides anonymity in TCP connections over internet
  + Unlikely (long term) and linkably (short term)
  + It means no long-term identifier for a Tor user
  + If web serve gets a connection today and tomorrow
    - Do not know if they are from same person
  + Both connection in quick succession from the same Tor node are more likely to be from the same person
    1. Nymity Slider
* Place transaction both online and offline on a continuum according to level of Nymity.
  + Verinymity
    - Government ID, credit card
  + Persistent pseudonymity
    - blogs
  + Linkable anonymity
    - Prepaid phone card
  + Unlinkable anonymity
    - Cash payments
* If build a system at a certain level of Nymity
  + Easy to modify to have a higher level of Nymity
  + Hard to modify to have a lower level
* E.g.
  + Easy to add loyalty card to a cash payment
    - Credit card to a loyalty card
  + Hard to remove identity information if you are paying by credit card
* Design systems with low level of Nymity fundamentally
  + Adding more is easy

1. Application layer security and privacy
   1. Intro

* Many application would like end to end security
  1. Secure remote login (SSH)
* Usual usage
  + Client connect to server
  + Server sends its verification key
    - Client should verify that this is correct key
  + Client and server run a key agreement protocol to establish session keys, server sign its messages
    - Communication from here on is encrypted and MAC with session key
  + Client authenticate to server
  + Server accept authentication
    - Login proceed (under encryption and MAC)
  1. Authentication with SSH
* 2 ways
  + send password over encrypted channel
    - server need to know a hash of password
  + sign a random challenge with your private signature key (better)
    - server needs to know your public verification key
    - e.g.: encrypt email thru public key.
    - Better cause it’s not good to trust a single source
  1. Anonymity for email: remailers
* Tor allow anonymously communicate over the internet in real time
  + Easier problem
  + Implemented much earlier than Tor
* Anonymous remailers allow you to send mails without revealing own email address
  + Hard to have conversation (disadvantage)
  1. Type 0 remailers
* Known being anon.penet.fi
* How it works
  + Send mail to anon.penet.fi
  + Then forward to intended recipient
  + Replies is the anon address get mapped back to your real address and delivered to you.
  1. Anon.penet.fi
* Work if:
  + No one watching the net connections to or from anon.penet.fi
  + Operator of the service and machine remains trustworthy.
  + Mapping on anon address to real address is kept secret
  1. Type 1 remailers
* Removed the central point of trust
* Message now sent thru a chain (encrypted) of remailers
  + Dozen to choose
* Each step in the chain is encrypted to avoid overserves
  + Also delay and reorder messages
* Support for pseudonymity is dropped: no replies
  1. Type 2 remailers
* Constant length message to avoid overseer watching “big file” travel across network
* Protection against replay attack
* Require email client to construct message fragments
  1. Nym servers
* Recovering pseudonymity
  + Mapped pseudonyms to reply blocks
  + Contained a nested encrypted chain of type 1 remailers.
* Attaching message to end of one of these reply block cause it to be sent through the chain
  + Eventually delivered to nym owner
* Significant privacy issues with type 1 remailer system
  1. Type 3 remailers
* New
* Mixminion remailer
  + Native support for pseudonymity
    - No longer reply on type 1 reply blocks
  + Improved protection against replay and key compromise attacks
* not very well deployed or mature
  1. Pretty Good Privacy (PGP)
* Popular implementation of public key cryptography
* Many compatible programs
  + GNU privacy Guard (gpg), Hushmails, etc.
* What does it do
  + Primary goal: protect contents of email message
* How it works
  + Public key encryption to provide
    - Encryption of email messages
    - Digital signatures on email messages
* PGP main functions
  + create 4 kinds of keys
    - encrypt, decrypt, signature and verification
    - encrypt - encrypt message using someone else encryption key
    - decrypt- decrypt messages using own decryption key
    - sign messages using own signature key
    - verify signatures using someone else verification key
  1. Fingerprints
* Form of verifying public keys
* Cryptographic hash of a key
  + Shorter
* No way to make 2 different key with the same fingerprint
* Can try
  + Alice download bob key
  + Alice software calculate fingerprint
  + Alice verify with bob and ask him to read his key fingerprint to her
  + If match, Alice know she got authentic copy of Bob’s key
  1. Signing keys
* Once Alice verified Bob’s key
* Alice uses her signature key to sign bob’s key
* Effectively same as Alice sign a message:
  + Verify they key with fingerprint belongs to Bob
* Bob can attach Alice’s signature to the key on his webpage
  1. Web of trust
* Alice can now act as an introducer for Bob
* If another party (carol) do not know Bob
  + Carol downloads Bob’s key from website
  + See Alice’s signature
  + Use Bob’s key without having to check with Bob.
* PGP software handles it mostly automatically
  1. Perfect forward secrecy
* Suppose communication between Alice and Bob are recorded by bad guys
  + Key material is discovered
  + Attacker can decrypt past message
  + Cause by incriminating records
* Perfect forward secrecy ensure future key compromises should not reveal past communication
  + Use secret key encryption for a short-lived key
* Created by modified Diffie Hellman protocol
  + Discard session key after use
  + Secure erase from memory
  1. Deniable authentication
* Do not want digital signature
  + Non-repudiation is great for signing contracts
  + Not desirable for private conversation
* Do not want authentication
  + Can’t maintain privacy if attacker can impersonate our friends
* Use MAC
  1. No third-party proofs
* Shared-key authentication
  + Alice and Bob have same key
  + Key is required to compute MAC
* Bob cannot prove that Alice generated the MAC
  + Give a measure of deniability
  1. Signal Protocol
* Signal is an app
  + Provide forward secrecy, improve deniability