Information Security

Personal notes based on lecture material and assigned readings from Princeton's COS 432: Information Security, taught by Ed Felten.

Table of Contents

Important Concepts

Types of Security
- Confidentiality
- Integrity
- Availability

Message Integrity
- Use a MAC (message authentication code)
  - Alice sends Bob message m and f(m)
  - Bob receives (a, b) and accepts iff f(a) = b
- Properties of a secure MAC
  - Deterministic
  - Easily computable for Alice (to generate) and Bob (to verify)
  - Not easily computable by Mallory
- Solution: use PRF $f(k, m)$ (see below)
- Message order
  - Problem: Mallory could change message order or resend old messages
  - Solution: Append sequence number (nonce) to each message

Pseudorandom Functions (PRF)
- "As good as random" – indistinguishable from a random function that maps all inputs to 256-bit outputs generated by flipping 256 coins (truth table)
- Public family of functions $f_0, f_1, f_2, \ldots$
  - Use f_k, where k is a secret key
- Theorem: If f is a PRF, then $f(k, m)$ with random(?) k is a secure MAC
  - Proof: If f is not a secure MAC, then f is not a secure PRF (contrapositive)
- Example: HMAC-SHA256

Pseudorandom Generators (PRG)
- Randomness often point of weakness in a security system
- PRGs use a small "seed" that is truly random
- Generate a long sequence of "good enough" (pseudorandom) values
  - Pseudorandom ≅ unpredictable
  - PRG is *secure* if output indistinguishable from truly random values
- Important property: "hidden state"
- Desirable property: forward secrecy
  - If Mallory compromises hidden state of generator at time t, Mallory cannot backtrack to reconstruct past outputs of generator
- Examples
  - PRG that is secure, but lacks FS
    - init: (seed, 0)
    - adv: (seed, k) -> (seed, k+1)
    - out: f(seed, k)
  - PRG that is secure AND FS

        - init: seed
        - state S
        - adv: f(S, 0)
        - out: f(S, 1)
        - Crux: state is overridden and *f is not (feasibly) invertible*

Encryption for Confidentiality

- Goal: ciphertext should not convey anything about plaintext
    - Semantic security (weaker than perfect secrecy)
        - Mallory chooses two plaintexts, we encrypt one of them
        - Mallory cannot do better than random guessing
- Alice encrypts with key k, Bob decrypts with key k
- First approach: one-time pad
    - Alice and Bob jointly determine long random string k (the pad)
    - Alice computes $E(k, x) = k \oplus x$
    - Bob computes $E(k, y) = k \oplus y = k \oplus (k \oplus x) = (k \oplus k) \oplus x = x$
    - Issues
        - Cannot reuse key
            - $(k \oplus a) \oplus (k \oplus b) = (a \oplus b)$
            - Easy to determine a and b with knowledge of English text distributions
            - Used in stream cipher attack (see below)
        - Need really long key (as long as sum of message lengths)
    - Strengths
        - Distribution of ciphertexts is random
        - Provably secure (Shannon 1949)
- Improvement: stream cipher
    - General idea: use PRG to "stretch" a small key into pseudorandom keystream
    - Start with truly random, fixed-size seed k
    - For each message, use a unique nonce (not necessarily secret)
        - Seed PRG with (k, nonce)
        - XOR message with output of PRG (like in one-time pad)
    - Critical: don't reuse (k, nonce) pair!
    - Issues
        - Proof of security associated with one-time pad no longer holds

Confidentiality *and* Integrity

- Possible approaches
    - $E(x \,||\, M(x))$
        - Used by TLS/SSL (Transport Layer Security, Secure Sockets Layer)
        - Must decrypt ciphertext to check integrity (no integrity on ciphertext)
    - $E(x) \,||\, M(E(x))$
        - Used by IPSec (Internet Protocol Security) – winner!
        - Can determine integrity without decrypting ciphertext
    - $E(x) \,||\, M(x)$

        - Used by SSH
        - Must decrypt ciphertext to check integrity
- Theorem: If E is semantically secure cipher and M is a secure MAC, then #2 is secure
    - Strategy of choice: encrypt plaintext, then append MAC of ciphertext
        - Bob first integrity checks, then decrypts
        - Important: Use separate keys for confidentiality and integrity
        - Important: Use separate pair of keys for Alice -> Bob and Bob -> Alice
    - This is authenticated encryption/decryption implemented in Assignment 1

Pseudorandom Permutations (PRP)

- Both encryption algorithm E and decryption algorithm D accept two inputs:
    - Block of size n bits
    - Key of size k bits
- Both E and D yield n-bit output block
- E_k is one of $(2^n)!$ permutations over the set of 2^n possible n-bit input blocks
- D is defined to be E^{-1}

	PR function	PR permutation	PR generators	Hash
Input	Any	Fixed-size	Fixed-size	Any
Output	Fixed-size	Fixed-size (equal)	Any	Fixed-size
Has key?	Yes	Yes	Yes (seed)	No
Invertible?	No	With key	No	Depends
Collisions	Yes, but can't find	No	No	Yes, but can't find

Block Ciphers

- Properties of a good block cipher
    - Efficiently computable (both E and $E^{-1} = D$)
    - Highly nonlinear ("confusion")
        - Hard for adversary to invert
    - Mix input bits together ("diffusion"/ "avalanche effect")
        - Small changes in input create large/complicated changes in output
- Feistel network (type of block cipher)
    - Operates in d rounds, typically between 12 and 16
    - In each round i
        - Input is split into two halves, L_i and R_i
        - $L_{i+1} = R_i$
        - $R_{i+1} = L_i \oplus f(k_i,\ R_i)$
        - Final ciphertext: $R_d \,||\, L_d$ (no switch on last round)
    - Theorem: If f is a PRF, then 4-round feistel network is a PRP
- DES (Data Encryption Standard)
    - 64-bit blocks, 56-bit key
    - 16 (weak) Feistel rounds
    - History

        - Designed in secrecy by IBM and NSA (1978)
        - U.S. government standard
        - Adopted by private sector
    - Problems
        - Vulnerable to differential cryptanalysis (not publicly known)
        - Designed to be slow in software to discourage implementation
        - 56-bit key size: sufficient then, can be brute-forced now
- AES (Advanced Encryption Standard)
    - 128-, 192-, 256- bit versions (input, output, and key)
    - Ten rounds
        - Not feistel design
    - Symmetric-key algorithm
        - Same key used to encrypt and decrypt
    - Adopted by U.S. government and used worldwide (superseded DES)

Encryption of Variable-Sized Messages

- Padding
    - Plaintext not a multiple of blocksize
- Cipher modes
    - Multi-block messages
    - Schemes
        - ECB (Electronic Code Book)
            - Each block encrypted independently
            - Not semantically secure!
                - Does not hide data patterns
                - Subject to replay attacks
        - CBC (Cipher Block Chaining)
            - Each block of plaintext is XORed with previous ciphertext block before being encrypted
            - First block?
                - Generate random initialization vector (IV)
                - Treat as C_{-1} (prepend to final message)
                - XOR with first plaintext block before encryption
            - Decent solution
                - Identical messages -> different ciphertexts due to IV
                - Single bit errors propagates due to chaining
        - CTR (Counter mode)
            - Input to block cipher: messageID (nonce) || counter
            - Output of block cipher XORed with plaintext
            - Best solution!
                - Identical messages -> different ciphertexts due to nonce
                - Efficient to compute (parallelizable (enc/dec)ryption)
            - Note: If messageID is unique, can reuse key
            - Note: Not forward secret as a PRG

Asymmetric (Public) Key Cryptography

- Problems with symmetric key crypto
  - Integrity
    - Alice sending message to Bob, Charles, Diana (all share key k)
    - Bob can forge a message from Alice to Charles and Diana by computing MAC with shared key k
    - If n people are communicating, $\binom{n}{2}$ keys must be used
  - Confidentiality
    - Maybe only one party (e.g. Alice) should be able to decrypt message
  - Must exchange (secret) key in secure way
- Asymmetric scheme (idea first conceived by Diffie-Hellman-Cox in 1976)
  - Two different keys for encryption/decryption or signing (MAC)/verifying
  - One key is kept public and other is private

RSA Algorithm (Rivest-Shamir-Adleman 1978)

- Define N = pq, where p and q are large, randomly-chosen secret primes
- Pick e to be any value less than and relatively prime to $(p-1)(q-1)$
  - Can be small (3, 17, 65537 often chosen)
- Find d such that $ed \bmod (p-1)(q-1) = 1$
- Public key: (e, N)
  - e is the public key exponent
- Private key: (d, N) and (p, q)
  - d is the private key exponent
- Sending a message with RSA
  - Message M from Alice to Bob converted to integer $0 \leq m < n$
    - Use agreed upon, reversible padding scheme
  - Encryption: ciphertext $c \equiv m^e \ (mod\ N)$
    - Modular exponentiation is efficient
  - Decryption: plaintext $m \equiv c^d \ (mod\ N)$
- Relies on difficulty of integer factorization and "RSA problem"
  - Factorization
    - Factor modulus N to determine p, q
    - Compute $(p-1)(q-1)$
    - Determine d from $ed \bmod (p-1)(q-1) = 1$
  - RSA problem
    - Take e^{th} root of ciphertext modulo composite N
- Why is symmetric key crypto still used at all?
  - RSA is slow
    - Computationally weightier operations (~1000x slower)
  - RSA keys are big
    - N is the product of two large primes (~4k bits)
- Applications
  - Confidentiality ("your eyes only" message)
    - Encrypt with public key of recipient
    - Recipient decrypts with private key

    - Integrity ("digital signature")
        - Sign by encrypting with private key
        - Verify by decrypting with public key
- Issues
    - Encrypting small messages with small e
        - If m^e is strictly less than N, ciphertext can be decrypted by taking e^{th} root of ciphertext
    - Chosen plaintext attack
        - Encrypt likely plaintexts under public key
        - Test if equal to ciphertext
        - Result: RSA not semantically secure
        - Why? RSA is a deterministic encryption algorithm
    - Chosen ciphertext attacks/malleability
        - Product of two ciphertexts is equal to encryption of product of respective plaintexts
            - Attacker asks private key holder to decrypt unsuspicious ciphertext $c' \equiv cr^e \ (mod\ N)$
                - r is chosen by the attacker
            - c' is the encryption of $mr \ (mod\ N)$
            - Attacker can multiply mr by r^{-1} to find m
        - Occasionally want a malleable cipher (not in RSA!)
    - Same plaintext -> same ciphertext (minor weakness)
        - RSA is deterministic
    - No built-in integrity check (minor weakness)
- Optimal Asymmetric Encryption Padding (OAEP)
    - Preprocessing step added before encryption to address all issues
    - Reverse OAEP used as a postprocessing step after decryption
    - Benefits
        - Adds element of randomness to deterministic RSA
        - Prevents partial decryption of ciphertexts/information leakage
- Encrypting larger messages
    - Cipher modes
        - CTR mode doesn't work because of randomization
        - CBC mode works, but is inefficient (overhead, very slow)
    - Hybrid encryption
        - Generate random symmetric encryption key k
        - Encrypt message with k
        - Encrypt k with RSA
        - Transmit both encrypted text and encrypted key

Digital Certificates

- Certifies the ownership of a public key by the named subject of the certificate
    - Address the impostor problem
- Chain of trust
    - Bob signs a message saying "Alice's public key is …"

        - Works if Bob is known and believed to be trustworthy and competent
    - If we do not know, must ask Charlie to verify Bob's identity
- Certificate authority
    - Universally trustworthy third party lists verified public key holders
    - Everyone knows CA's public key
    - Customers of a CA are generally server administrators who need to present certificate to their clients

Key Management

- For symmetric ciphers, 128-bit keys are sufficient
- Need larger key for PFF/hash function
    - Finding a collision is more efficient than finding (exact) key
    - Birthday attack
        - If b is the bit-length of the hash, generate $2^{\frac{b}{2}}$ items at random
            - $\sim 50\%$ probability of finding collisions!
        - This attack takes only $O(2^{\frac{b}{2}})$ time and $O(2^{\frac{b}{2}})$ space (also possible in constant space)
    - PRF output size is typically 2x cipher output size (256 bits)
- Principles
    - Key management is usually the hard part
    - Keys must be strongly (pseudo)random
    - Each key should have a different purpose
    - Vulnerability of a key increases with
        - Usage
        - Places (copies) stored
        - Time
- Implications
    - Change keys periodically
    - Use "session keys"
        - Long-term keys used to negotiate session keys
        - Session key used temporarily
    - Erase keys when no longer needed
    - Keep keys out of long-term storage (if possible)
    - Keep keys in inaccessible places
        - Offline, locked in a safe
    - Protect against compromise of old keys (forward secrecy)

Diffie-Hellman Key Exchange (Diffie-Helman 1976)

- Relies on difficult of discrete-log problem (given $g^x \ mod \ p$, find x)
- Algorithm
    - Public: large prime p and primitive root g modulo p
        - p is often chosen to be $2q + 1$ where q is prime ("safe prime")
    - Alice selects random, secret a, $1 < a < p - 1$
    - Bob selects random, secret b, $1 < b < p - 1$

    - Alice transmits $g^a \ mod \ p$ to Bob
    - Bob transmits $g^b \ mod \ p$ to Alice
    - Alice and Bob raise received values to their respective secret number
        - Arrive at shared secret $g^{ab} \ mod \ p$
    - In practice: use H($g^{ab} \ mod \ p$) as shared secret key
- Insecure if adversary can modify messages (no authentication of communicating parties)
    - Man in the middle (MITM) attack
        - Instead of transmitting $g^a \ mod \ p$ to Bob, Mallory transmits $g^v \ mod \ p$
        - Instead of transmitting $g^b \ mod \ p$ to Alice, Mallory transmits $g^u \ mod \ p$
        - Alice ends up with $g^{au} \ mod \ p$ and Bob ends up with $g^{bv} \ mod \ p$
        - Mallory can forge messages between Alice and Bob by decrypting, modifying, and reencrypting
    - Solution: digital signature scheme
        - Server releases a public key to which it holds private key counterpart
        - Server signs hash of its copy of key with its private key
        - Client decrypts server signature with server's public key
        - If decryption matches hash of client's copy of key, then client concludes key is indeed shared
        - Client accepts messages from server if hashes match, otherwise closes channel
- Bad key values
    - If $g^a \ mod \ p$ or $g^b \ mod \ p$ is 1, shared key will be 1
    - If $g^a \ mod \ p$ or $g^b \ mod \ p$ is $p - 1$, shared key will be 1 or $p - 1$
    - Insecure values as adversary can guess key
    - Alice and Bob should reject if receive 1 or $p - 1$
        - Alice and Bob agree to reselect random number if mod power result is 1 or $p - 1$
    - Theorem: If p is a "safe prime" (see above), then 1 or $p - 1$ are the only insecure values
- Diffie-Hellman and forward secrecy
    - Alice and Bob have a shared key and want to negotiate new key
    - Alice and Bob conduct D-H key exchange protected by old key to get new key
        - Old key prevents adversary from modifying messages in D-H protocol
        - Alternative to using digital signature to *check* that keys match
    - If adversary *later learns* old key, cannot determine new key
        - D-H does not save or transmit information sufficient to determine key

Password Security

- Dictionary attack
    - Guessing attack using a precompiled list of likely options ("Password," "Computer," etc.)
    - Dictionary versus brute force attack
        - Brute force attack probes entire keyspace

        - Brute force generally used against encryption, while dictionary attack is used against passwords (user generated)
- Storing the hash of a password in a database is better than storing the password
- Storing the *salted hash* of a password is better than storing just the hash
    - Two identical passwords will hash to the same value, so if an attacker cracks one password, the other is known to her as well
    - Solution: append a unique (not secret) value, called a *salt*, to a password before computing its hash
    - Store salt with the hash in the database
    - To verify a password, add salt to password, compute hash, and check against stored hash value
- Rainbow table attack
    - Precomputed table of hashes for possible plaintext passwords
    - Attack requires access to database of password hashes ("offline attack")
    - Salts frustrate this attack, because for each possible password must compute hash corresponding to each salt in the database (adds dimension to table)

Challenge-Response Protocols

- Wish to authenticate user without revealing password p in protocol
- Procedure
    - User sends (user)name to server
    - Server asks user to encrypt random number r
    - User returns $PRF(p, r)$
    - Server verifies if user has correct password
- Used by HTTP (Web's Hypertext Transfer Protocol)

Public Key Infrastructure

- Hardware/software/people/policies/procedures needed to create/manage/ distribute/store/validate/revoke digital certificates
- Binds public keys with user identities through certificate authority (CA)
- Components of a PKI
    - Certificate authority (CA)
        - Root of trust in PKI, authenticating individuals, computers, network entities
        - CA issues own public key in self-signed CA certificate
            - Signs certificates with corresponding private key
        - Issues signed (encrypted) digital certificates
            - Alice requests certificate from CA
            - CA verifies Alice's identity
            - CA computes hash of certificate contents and signs hash with own (CA's) private key
            - CA appends signed hash to original certificate
            - CA makes Alice's certificate publicly available
        - Verification process
            - Bob retrieves Alice's certificate

            - Bob decrypts signed hash with CA's public key
            - Bob compares hash of certificate with decryption
            - If match, knows Alice's public key is valid
    - Registration authority
        - Role 1: subordinate CA
            - Certified by root CA to issues certificates for specific uses permitted by root
        - Role 2: verifies identity of users requesting info from CA
    - Central database (server)
        - Holds certificate requests and record of issued/revoked certificates
    - Certificate store (on local computer)
        - Saves issued certificates and record of pending/rejected requests
    - Key Archival Server
        - Saves encrypted private keys in certificate database in case of loss
- Issues
    - Standards exist (X.509) but no government body enforcing standards
    - Provides chain of trust, but PKI is only as strong as weakest link
        - If one CA is compromise, security of entire PKI is at risk
        - 2011: Web browser vendors forced to blacklist all certificates issued by Dutch CA DigitNotar after 500 fake certificates discovered

Access Control

- SUBJECT wants to do VERB on OBJECT
    - Subject – active entity that requests access to an object or data within object
        - E.g. user, program, process, etc.
        - In this case, assume *running program*
    - Verb – action subject wishes to perform on object
        - In this case, assume *operation/API call*
    - Object – passive entity or resource that contains the information
        - E.g. computer, database, file, program, network connection, etc.
        - In this case, assume *system resource*
- Policy: set of allowed (S, V, O) triples
- Approaches
    - Access Control Matrix (ACM)
        - Table of Subjects v. Objects
        - Intersection contains allowed Verbs
        - Simple but inefficient implementation (matrix will be very sparse)
    - "Profiles"
        - For each user, store allowed permissions
    - Access Control List (ACL)
        - For each object, store (SUBJ, VERB) pairs
            - Alice: read, write; Bob: read; …
        - Small and simple in practice
            - Most common approach (Parse!)

Zero-day Attack (Vulnerability)

- Exploitation of a previously unknown (to developers) vulnerability in a computer application or operating system
- Conducted in the time window between the discovery of the hole and the release of the security patch/update

Secure Information Flow

- Output of a program $P(v, s, r)$ should not leak any information about secret input s over all possible values of public/visible input v
- Cannot enforce by simply watching output ("dog that didn't bark" problem)

Network Security

- Internet is a network of networks
    - Each network is an "autonomous system" (AS)
        - AS is a collection of routers (formally: IP routing prefixes)
        - Under the control of a single administrator, maintaining a clearly defined routing policy
    - AS's connect together at exchange points
    - 47,000 AS numbers assigned by mid-2014
- Border Gateway Protocol (BGP)
    - Protocol designed to exchange routing and reachability information between autonomous systems (AS) on the Internet
    - Makes routing decision based on paths, network policies, and rule-sets configured by network administrator
    - Not actual routing protocol (that's the Internet Protocol (IPv4, IPv6, …))
- Shortest Path Routing
    - Simplified version of BGP
    - Attacks can lie about length of path to another router/host
        - Changes shortest path between routers A and B, diverting information through malicious router
    - Prefix hijacking attack
        - Also called IP hijacking or BGP hijacking
        - Involves announcing shorter route (either non-existent or tunneled) to redirect traffic
        - Pakistan's attempt to censor Youtube (2008)
            - Accidently leads to worldwide shutdown
        - China Telecom announces 37,000 prefixes not belong to them (2010)
            - Worldwide impact, but local traffic most affected
        - Malice *highly unlikely*
            - Can't bypass application-level encryption
            - Can't store all the traffic
            - Easily detectable
        - Defense
            - Cryptography can prevent an AS from lying about other nodes
            - Cannot prevent lying about their *own* links and costs

    - Bottom line: relies on trust between small number of ASs
        - ASs can sever connections with a rogue node
        - Unlike application-layer security
- Layered network stack
    - Application
        - BGP — on top of TCP
        - Domain Name System (DNS) — on top of UDP
    - Transport (TCP or UDP)
    - Network (IP)
    - Physical and data link
- IP Packets and Spoofing
    - IP Packets contain source and destination IP addresses
    - Source can be spoofed, but destination can't
        - Return message will then be sent to spoofed address
    - Nodes cannot verify claimed source address
        - If A -> B -> C, C does not know if package originate from A or B
        - Node only knows local origin
    - Defenses
        - Ingress filtering
            - Discard an incoming packet if Source IP is inside network
        - Egress filtering
            - Discard an outgoing packet if Source IP is outside network
    - Distributed denial of service (DDoS) attack
        - Easy if lot of zombie nodes
        - Interesting: attack from single machine with bandwidth ~ as target
        - Smurf attack
            - Attacker sends broadcast ECHO request to network
                - Internet Control Message Protocol (ICMP) ping
            - Return address (source) is spoofed to be victim's address
            - All network hosts reply to victim
- Domain Name System (DNS)
    - Hierarchical, distributed naming system for devices connected to Internet
    - Translates domain names to IP addresses (DNS name resolution)
        - Process initiated on client side by DNS resolver
        - If a particular DNS server cannot translate, will ask another server
            - Recursive process
        - Caches recent translations
    - "Cache poisoning" attack (DNS spoofing)
        - Basic DNS does not use crypto
        - Attacker supplies incorrect translation of a domain name
            - Incorrect translation is cached (poisoning the cache)
            - Subsequent requests for translation of that domain name return address of server controlled by attacker
        - Solution: DNSSEC

- Provides: authentication of DNS data, authenticated denial of existence
- Does not provide: availability or confidentiality (no encryption)
- Answers from DNSSEC protected zones are digitally signed
    - DNS root servers used as root of trust
    - DNS hierarchy used as chain of trust
        - Parent domain (DNS zone) verifies DNSKey record in subdomains
    - Procedure
        - Domain owners generate their own keys
        - Upload them with DNS control panel to domain name-registrar
        - Keys pushes via secDNS to zone operator
        - Zone operator (i.e. Verisign for .com) signs and publishes keys in DNS

- Cryptography in the Network Stack
    - Can be incorporated into different layers (application level, transport level, network/IP level, etc)
    - SSL/TSL
        - Application layer (or between transport and application layers)
        - Authenticates: hostnames (server identity)
            - Server usually well-known entity
        - Encrypts: sessions over TCP layer
            - Allows secure communication (confidentiality and integrity) between server and client
    - IPSec
        - Network layer security
            - Goal: integrity/confidentiality at level of IP packets
        - Authenticates: IP addresses
        - Encrypts: IP packets
        - Problems
            - IP is stateless, but keeping state required for encryption
                - Communication consists of independent (request, response) pairs
                - Does not require server to retain session information over multiple requests
            - Many security problems are application-specific

POODLE and SSLv3 Vulnerability

- TSL/SSL are encryption protocols used to protect communication between websites and computers
    - Represented by small padlock icon in browser
    - Protects information from being intercepted, spied upon, or modified by attackers between user and service provider

    - Prevents someone sharing Wi-Fi in Starbucks from spying on your bank transactions
    - TSL has now replaced SSL, except in cases of backward compatibility
- TSL clients will downgrade protocol used to lower version of TSL and then SSL if dealing with legacy servers ("downgrade dance")
    - First handshake attempt: offers highest protocol version supported
    - If handshake fails, retry will earlier protocol versions
    - If attacker interferes with client-server negotiations, can downgrade to SSL 3
        - Attacker is MITM between client and server
- Encryption in SSL 3.0
    - Uses either RC4 stream cipher or block cipher in CBC mode
        - RC4 has biases
            - If same secret is sent over many connections and encrypted with many RC4 streams, information about secret will leak
        - CBC encryption
            - Block cipher padding is not deterministic, nor covered by MAC
            - Integrity of padding cannot be fully verified when decrypting
    - Attacker can decrypt "secure" HTTP cookies
- Solution
    - Disable SSL 3.0 in browser
        - Can prevent communication with legacy systems
    - Use TLS_FALLBACK_SCSV

Firewalls and VPNs

- Intranet
    - Private network internal to company
    - Private IP space (typically)
        - Internal view different from external view (http://benefits/)
    - Principle: don't connect (most) machines directly to Internet
- Network Address Translation (NAT)
    - Machines assigned IPs from reserved spaces
        - Examples: 192.168.*.* and 10.*.*.*
    - Network shares single "real" IP address
    - NAT keeps translation table of inside IP address to outside IP equivalents
        - A router acts as an agent between the Internet and local network
        - No publicly visible IPS for local machines
            - Can't accept incoming connections (directly)
- Firewalls
    - Perimeter defense for a network
        - Separate outside from inside
        - Monitor boundary
        - Block questionable incoming traffic
        - Centralize security policy for easy administration
    - Types
        - Network/IP layer: packet filtering

                - Stateless packet filtering
                    - Block all incoming connections
                - Stateful packet filtering
                    - To block *some* incoming connections
                    - Only allow incoming packet if in response to previous outgoing packet
                    - Remember TCP sequence number and acknowledgement number
                    - Allows more sophisticated policies
            - TCP layer: circuit-level gateway
                - Goal: allow servers to run on inside of firewall
                - Allows more sophisticated filtering than Network/IP layer
                - Components
                    - SOCKS: TCP-level proxy protocol
                    - Client library: internal machine
                    - Client program: supports proxying
            - Application level: proxy server
                - Even more sophisticated filtering
                - Need separate proxies for each service
        - Complications
            - Firewall blocks incoming DNS replies
                - Use DNS proxy (application-level gateway)
            - Need to serve web and other content
                - Put servers outside firewall (DMZ)
            - Firewall blocks incoming email
                - Drop server outside firewall
                - Siphon mail in after filtering
- Virtual Private Networks (VPNs)
    - Extend the perimeter
        - Goal: make branch offices behave as if on same private network
    - VPN server is on the firewall
        - Dual interface with Internal IP and External IP
    - Example process
        - User working from home can use VPN client to entire private network
            - Authenticates to VPN server using username/password
            - Obtains shared session key
            - VPN server assigns intranet IP to client
                - Adds mapping of intranet IP – external IP to NAT table
            - "Tunnel" established
- DMZ
    - Firewall configuration used to secure local area networks (LANs)
    - Most computers run behind firewall connected to public network
    - One or more computers run outside firewall, in DMZ
        - Intercept traffic and broker request for rest of LAN
    - Another firewall separates these computers from rest of Internet

Web Security

- Browser (based on OS/hardware) interacts with website (based in network)
  - Browser sends requests
  - Website replies
- Two sides of web security
  - Web browser
    - Can be attacked by any website it visits
    - Attacks can lead to malware installation (keyloggers, botnets), document theft, loss of private data
  - Web application
    - Runs at website
    - Written in PHP, ASP, JSP, Ruby
    - Many potential bugs: CSRF, CSS, SQL injection
    - Attacks lead to stolen credit cards, defaced sites, etc.
- Web attacker
  - Entices user to visit malicious website (i.e. attacker.com)
    - Can easily obtain SSL/TSL certificate for his site ($0)
    - Uses phishing email, enticing content, appears in search results, is placed by ad network, etc.
  - Network attacker
    - Passive: wireless eavesdropper
    - Active: evil router, DNS poisoning (see network security notes)
  - Malware attacker
    - Attacker control's user machine
      - How? Convinces user to install malicious content
      - Masquerades as antivirus program, codec for new video format, etc.
    - Exploits application bugs (e.g. buffer overflow)
- JavaScript
  - Language executed by browser
    - Scripts embedded in Web pages
    - Can run before HTML is loaded, before page is viewed, while it is being viewed, or when user is leaving the page (any time)
  - Used to implement "active" web pages
    - AJAX (asynchronous JavaScript and XML) allows Web apps to send/retrieve data from server in background
    - Note: JSON often used instead of XML, need not be asynchronous
  - Origin of many security issues
    - Allows attacker to execute code on user's machine (browser)
  - Security model
    - Script runs in "sandbox"
      - No direct file access (on user's computer)
      - Can cause browser to load remote pages/resources like scripts or images, which may be cached locally by browser

        - Can technically only store cookies
    - Same-origin policy
        - Can only read properties of documents and windows from same *server*, *protocol*, and *port*
        - Does not apply to library imports
            - Scripts loaded in enclosing frame from external site
                - `<script type = “text/javascript> src=“http://www…”> </script>`
            - Script runs as if loaded from site that provided page
- Cookies
    - Small piece of data sent from website and stored in user’s browser
        - Every time user loads website, browser sends cookie back to server
    - Purpose: allow websites to remember state (items in a shopping cart) and/or record user’s browsing activity (clicking buttons, logging in, etc)
    - Can store form content such as: passwords, credit card number, address
- Three attacks
    - Cross-Site Request Forgery (CSRF)
        - Malicious script (from user’s visit to malicious site) makes forged request to “good” site with user’s cookie for good site
            - Changes Netflix accounts settings, steals Gmail contacts
        - At risk: web applications that perform actions on input from authenticated users without requiring authorization of specific action
            - User authenticated by cookie
            - Browser tricked into sending HTTP request to target site
        - Can force user to perform state changing requests:
            - Transferring funds out of bank account
            - Changing email address/password
        - *Cannot* directly see website’s response to browser’s forged request
            - Attacker must find URL that has side effects or online form
            - Unless…attacker uses cross-site scripting
        - Defenses
            - Secret validation token (synchronizer token pattern)
                - Secret and unique token embedded by web app into all HTML forms and verified on server side
                    - Must ensure unpredictability and uniqueness (i.e. using hash chain of random seed)
                    - Example: `<input type=“hidden” name=“…” value=“Kby…”>`
                - Can be difficult on web apps that heavily use AJAX
            - Referer validation
                - Check HTTP Referer header to ensure request is coming from authorized page
                - May cause issues with browsers that omit Referer header for privacy reasons (too strict a policy)

            - Must calibrate leniency of policy
        - E.g. `Referer: http://www.facebook.com/home.php`
    - Custom HTTP header
        - Set custom headers for each REST request
        - Attacker cannot set custom header by script via form, image, iframe, etc.
            - Unless using JavaScript XMLHttpRequest or Flash
            - JavaScript same-origin policy prevent cross-site requests
        - Verify request's header contains `X-Requested-By: XMLHttpRequest` or `X-Requested-With`…
            - If no header, drop request
- Recommendations
    - Strict referer validation for login forms or bank sites (info submitted over HTTPS)
    - For other sites, use Ruby-on-Rails or other framework that implements secret validation token (correctly)
    - Another type of header?
- Cross-Site Scripting (CSS/XSS)
    - Attacker injects malicious code into link to (supposedly) trustworthy source (sent to Alice via email, etc)
        - When user visits vulnerable site, embedded script is submitted as part of client's Web request (i.e. Google search)
            - If server-side application reflects user input (CSS vulnerability exists), browser will run reflected script
            - Bypasses same-origin policy test
    - Attacker can retrieve user authentication cookie and learn sensitive data, or hack web application itself
    - Defenses
        - HTML-escape all user input
            - Browser displays but does not run HTML-escaped input
        - Sanitize input by stripping of tags
- SQL Injection
    - Input validation vulnerability
        - User input in HTTP GET request could contain termination of line followed by (malicious) SQL script
            - E.g. `'; DROP TABLE USERS; --`
                - Eliminates all user accounts
    - Prevention
        - Input validation
            - Filter characters with special meanings
            - Check data types
        - Whitelisting
            - Blacklisting "bad" characters doesn't work

        - Could forget to filter out some characters
        - Could reject valid input
    - Allow only well-defined set of safe values
        - Implicitly defined through regular expressions
- Limit privileges
    - Prevent leakage of database schema
    - Encrypt sensitive data stored in database

Web Privacy

- The market for software that respect user privacy is a *lemons market*
- "Third party" online tracking
    - Sites other than the one you are visiting tracking your browsing history
    - Typically invisible to users
    - 64 independent tracking mechanisms on typical top-50 sites
- Tracking techniques
    - Tagging
        - Placing data in your browser
        - Includes: HTTP Cookies, HTTP Auth, HTTP Etags, Content cache, IE userdata, HTML 5 protocol & content handlers, HTML5 Storage, Flash cookies, Silverlight storage, TLS session ID & resume, Browsing history, window.name, HTTP STS, DNS cache
    - Fingerprinting
        - Observing your browser's behavior
        - Includes: User-Agent, HTTP ACCEPT headers, Browser plugins, MIME support, Clock skew, installed fonts, cookies enabled?, browser add-ons, screen resolution
        - Browsers are unique enough
            - User agent string, plugins, etc. can uniquely identify
        - Panopticlick
            - Browser fingerprinting service/experiment
            - User-agent string: 10 bits of entropy, 84% of fingerprints unique (with Flash/Java, 94% unique)
- Anonymity?
    - Not quite
        - Third party is sometimes a first party
            - Facebook may be a third party to the site you are visiting, but if its "like" button is on the page, Facebook knows…(?)
        - Leakage of identifiers
            - `GET http://ad.doubleclick.net/adj/...`
              `Referer: http:/`
              `/submit.SPORTS.com/…?email=jdoe@email.com`
              `Cookie: id=35c192bcfe0000b1...`
            - If the email appears in the referer, identity has been compromised now and in the future
        - Third party buys your identity

        - Hacks and bugs
            - Google spreadsheet (see github notes)
        - Cookie synchronization
            - Third party X sends its cookie to third party Y
            - X and Y exchange data about user
            - ```
              GET      http://tracker2.com/?uid=ghaihtn3
              Referer: http://tracker1.com/…
              Cookie:  id=35c192bcfe0000b1...
              ```
    - Pseudonymity
        - Can tell when same person comes back (to website, etc) but doesn't know real-life identity
        - This is not true anonymity
            - Anonymity: shouldn't be able to track you under a pseudonym in a different session
        - Possible to connect online pseudonym with real-life identity
- Solutions
    - Referer blocking
        - Drawback: many sites check referer header for CSRF defense
            - Blocking referer indiscriminately will break sites
        - Drawback: can't prevent cooperative tracking
    - Third party cookie blocking
        - Advantage: does not break security systems
        - Drawback: doesn't prevent fingerprinting
        - Safari blocks third party cookies unless:
            - User is submitting a form
            - Browser already has cookie from same party
    - Do Not Track
        - Preference that can be set in web browsers
    - HTTP Request blocking
        - Compile and maintain list of known trackers
            - Semi-automated analysis
            - Based on domains and regular expressions
        - Sequence of events
            - User installs browser extension
            - Downloads list
            - Block request to objects on the list
        - Drawback: false positives and false negatives
        - Drawback: need to trust list
    - Blocking tools
        - Ghostery
        - Adblock Plus
        - Drawback: doesn't work by default – user must install
        - Drawback (Adblock): user needs to install blocklist separately to block all trackers like analytics and social widgets
              ```

Electronic Voting

- Types of voting machines
  - Hand-counter paper, punch cards, lever machines, optical scan ballots, electronic voting machines, touch-screen terminals, hybrid schemes
- Paper ballot attacks
  - Chain voting
    - Attacker obtains blank ballot and stands at entrance of voting booth
    - Attacker marks ballot as desired
    - Intimidates voter to take marked ballot and deposit it
      - Instructs voter to bring back blank ballot to attacker outside
      - Allows attacker to continue process
- "Receipt-free" secret ballot
  - Key aspect: cannot prove to 3rd party how you voted
- Proxy re-encryption
  - Bob wants to reveal of contents of message sent to him (encrypted with his public key) to Chris without revealing private key to Chris
  - Designates proxy to re-encrypt message
    - Generates new key that Chris can use to decrypt message
    - Proxy cannot read Bob's messages
- ElGamal encryption
  - Asymmetric key encryption algorithm based on Diffie-Hellman key exchange
  - Typically used in hybrid cryptosystem
    - Message encrypted using symmetric cryptosystem
    - ElGamal used to encrypt *key* used for symmetric cryptosystem
  - Algorithm
    - Each user has a private key x
    - Each user has three public keys: prime modulus p, generator g, and public $Y = g^x \ mod \ p$
  - Performance
    - As an asymmetric scheme, ElGamal is quite slow
    - Probabilistic
      - Advantage: single plaintext can be encrypted to many possible ciphertexts
      - Disadvantage: produces 2:1 expansion in size from plaintext to ciphertext
  - Security
    - Rests on the difficulty of the discrete log problem
    - Unconditionally malleable (not resistant to chosen ciphertext attack)
- End-to-end verifiability
  - Voter can confirm that vote was 1) cast as intended and 2) counted as cast
    - Does not have to trust election equipment or personnel
  - Should still be a secret-ballot
  - Goal: end-to-end verifiable elections while protecting voter privacy
    - Plan: use reencryption mix scheme
  - Two phases

        - Voters publish their names and encrypted votes
            - Public "bulletin board" of ciphertext ballots
        - At end of election, administrators publish tally of votes
            - Include cryptographic proof that tally matches set of (published) encrypted votes
    - Two possible paradigms
        - Anonymized ballots (mix networks)
        - Ballotless tallying (homomorphic encryption)
            - Includes RSA, ElGamal, Benaloh, etc.

Email Protocols

- Traditional mechanism
    - User composes message using email client on computer
        - Headers: to, from, date
        - Body: can encode different types of media
    - User hits "send" button
        - Email text and attachments uploaded to Simple Mail Transfer Protocol (SMTP) server as outgoing mail
    - Outgoing messages wait in outgoing mail queue
        - SMTP server communicates with DNS to find location of recipient's email server
    - Messages are downloaded from recipient server to recipient's email client
        - Uses Internet Message Access Protocol (IMAP)
    - Examples: Thunderbird, Postbox, Outlook (desktop clients)
- Webmail
    - Uses HTTP(S) to upload messages to sender's mail server and to download from recipient's mail service
        - Still use SMTP to transfer mail from sender to receiver servers
    - Examples: Gmail, Yahoo! Mail, AOL Mail

Anonymous Communication

- Is Internet anonymous?
    - No. IP addresses necessary for routing
        - Best case: pseudonymous
        - Worst case: identified
    - Encryption does not hide identities
- Tor (Onion Router)
    - Internet networking protocol designed to anonymize data relayed across it
        - Protects against Internet surveillance form known as "traffic analysis"
    - Data bundled into encrypted packet when it enters Tor network
    - Unlike normal internet connections, Tor
        - Strips away part of packet header
            - Separates addressing information that could identify sender
        - Encrypts rest of addressing information (packet wrapper)
    - Modified/encrypted data packet routed through many relays

        - Each relays decrypts only enough of each data packet wrapper to know which relay data came from/which relay to send it to next
        - Encryption keys different for each hop along circuit
        - Last hop, from exit node to receiver, usually not encrypted
            - Cannot assume receiver is using Tor
    - Goal: should not be able to trace data packet's path through Tor

SSH

- Cryptographic network protocol for secure data communication
    - Typically used to log into a remote machine and execute commands
- Connects, via secure channel over insecure network, a server (running SSH server) and a client (running SSH client)
- Uses public-key crypto to authenticate remote user
    - SSH only verifies whether public/private key match
    - Does not match public keys to identities
        - For unknown public keys, must verify this
- List of authorized public keys stored in Unix home directory
    - File located at `~/.ssh/authorized_keys`
    - File should not be writable by anything apart from owner and root
    - SSH remembers key used by a server side over different sessions

Malware

- Taxonomy

	Requires host	Runs independently
Doesn't spread	Trojan, Rootkit	Keylogger, Spyware
Spreads	Virus	Worm

- Viruses
    - Definition: Reproduces own code by attaching itself to other executable files
        - When infected executable file is executed, virus code is also executed
        - Key points: self-replicating (spreads), infects files (requires host)
    - Classic viruses account for only 3% of all malware
    - What can act as a host?
        - Executable files
            - Either append code to file or overwrite parts of file code
            - Often take same name as existing files, with .exe extension
                - User might accidently click and execute virus code
        - Boot sector
            - Region of hard disk containing machine code to be loaded into RAM on computer boot
            - E.g. Pakistani Brain virus
        - Macros
            - Set of instructions within application used to automate tasks

- Macros can perform system operations, such as creating, writing to, deleting files (potential for great damage)
- Most macros written for Word, Excel, etc
- Macro viruses infect templates for new documents
  - Each time new document is created, virus replicates
- Cross-platform (not PC only)

- Virus lifecycle
  - Reproduction phase
    - Balances infection rate versus detection possibility
  - Infection phase
    - Viruses can stay resident in memory (dormant)
  - Attack phase
    - Attack on trigger
      - Jerusalem virus attacked on Friday the 13th
    - Delete files, change random data on disk
- Defenses
  - Antivirus software
    - Signature-based detection
      - Database of byte-level or instruction-level signatures that match virus (with wildcards, regular expressions)
    - Heuristics
      - Code execution starts in last section
      - Patched import address table
  - Sandboxing
    - Run untrusted applications in restricted environment
    - Default: do not run as administrator
- Variants
  - Encryption
    - Malware body encrypted with key
    - Decryption routine stored unencrypted
    - Decrypts upon execution
  - Polymorphic viruses
    - Change slightly with each infection
    - Encrypted payload
    - Different key used for each infection
    - Makes static string analysis difficult (impossible)
  - Metamorphic viruses
    - Different "versions" of code, but essentially same behavior

- Worms
  - Definition: self-replicating program that propagates itself across networks
    - Key points: self-replicating (spreads), propagates itself (no host)
  - Components
    - Target locator
      - Email harvesting (scan address books, inbox of email client, Google searches, buy list of emails, IP addresses)

        - Infection propagator
        - Life cycle manager
        - Payload
            - Often a Trojan horse
    - Variants
        - Email-based
            - Forged from address
            - Hide executable extension (.exe) behind harmless ones (.jpeg)
            - Promise interesting pictures or applications
        - Exploit-based
            - Do not require human interaction
            - Spread using well-known network services (TCP, etc)
            - Spread can be modeled with classic disease model
                - Slow start, followed by exponential growth
    - Defenses
        - Virus scanners
            - Scan email attachments or other contents
            - Effective against email-based worms
        - Host level defense
            - Elimination of underlying software vulnerabilities
            - StackGuard: protect against buffer overflow
            - Randomize position of stack, heap, libraries in memory
        - Network level defense
            - Intrusion detection systems
                - Scan for known attack patterns
            - Rate limiting
                - Quota on number of outgoing connections
            - Personal firewall
                - Block outgoing SMTP connections from unknown apps
- Writing secure code
    - Careful coding, code audits, high-level languages, model checking, formal methods and protocol verification, fuzz testing, static analysis, dynamic analysis, taint analysis, comparison across implements, access control

Big Data and Privacy

- Goals
    - Make valid inferences about population as a whole from dataset
    - Cannot make valid inferences about individuals from dataset
- Semantic privacy
    - Given two datasets D and D', where D' is D with one datapoint removed, anything analyst can learn from D, they can also learn from D'
    - Theorem: Semantic privacy implies result of analysis does not depend on content of dataset
- Differential privacy
    - Property of a protocol A run on a dataset X producing output $A(X)$

        - A is a randomized algorithm
    - A gives ϵ-diffrential privacy if $A(X)$ and $A(X')$ give very similar results, where X and X' differ in the inclusion/exclusion of one element
- Post-processing
    - Theorem: applying an arbitrary function f to the output of a differentially private protocol A gives an output that is still ϵ-DP
- Achieving differential privacy
    - Output perturbation: add random noise to true answer of query
    - Use Gaussian distribution or (better) Laplace/geometric distribution
- Applications
    - Collaborative recommendation systems
        - "People who bought X also bought Y"
        - Privacy issues
            - Rare book X only Ed would buy
            - Collaborative recommendation links item X to another item Y
            - Recommendation gives hint about what (else) Ed bought
- Solutions
    - Look at algorithm internals
        - System generates covariance matrix
        - Correlation between purchases of all pairs of items
        - Add random noise to matrix to achieve differential privacy
    - Machine learning and DP queries
        - Machine learning algorithm exchanges DP queries and results
        - Can synthesize new dataset

Economics and Security

- Fundamental question
    - Does the market produce optimal security?
- Definitions of "optimal"
    - Strong Pareto Efficiency
        - Condition A is SP-superior to Condition B if everyone prefers A over B
        - Condition is SP-efficient if no SP-superior alternative available
            - Impossible to make any one individual better off with making at least one individual worse off
            - Pareto improvement: a change that could make one individual better off without making any other individual worse off
    - Kaldor-Hicks Efficiency
        - Condition A is KH-superior to Condition B if a set of zero-sum payments P among people (i.e. wealth transfers, redistributive taxes) exists such that A + P is SP-superior to Condition B
            - Note: payments need not occur in practice
        - Condition is SP-efficient if no KH-superior alternative is available
    - Theorem: a world with perfect information and perfect bargaining would be SP-efficient and KH-efficient
        - Proof by contradiction

        - Implication: since world isn't SP-efficient or KH-efficient, market failures must be occurring
- Market failure #1: negative externalities
    - Harm falls on third party (not seller or buyer)
        - Neither will invest in reducing harm to third party
    - Implication: underinvestment in security
        - Note: bargaining to fix externalities not possible in real world
- Market failure #2: asymmetric information
    - Hard for buyers to evaluate security of products
    - Producer knows more about security of product than buyer
    - "Lemons market"
        - Little incentive for producer to improve quality
    - Solutions
        - Add warranties to product
        - Seller reputation
- Network effects
    - Product becomes more valuable as more people use it
    - Tends to push markets toward monopoly (monoculture)
    - Benefits of having a dominant producer
        - Security is often more efficient with scale
        - As a producer whose product pervades society, some of the external benefits are in fact internalized (no longer true externalities)
        - Warranties and reputation matter more
    - Nuance: race to market
        - Network effects often tip toward early leader
        - Companies try to get MVP (minimum viable product) into market as soon as possible
            - Less incentive to work on security now
            - "Bolt on security later" approach
        - Solutions
            - Large customers can protect themselves
            - Market structures to improve information flow
                - Insurance companies, certification programs
            - Change in liability rules
                - Optimal rule: cost born by whoever can best prevent harm (this tends to be the producer)
                - Problems: hard to attribute blame, hard to measure harm, and high cost to adjudication (judging)
            - Public inspections
                - Large buyer demands ability to publicize security evaluations of products

Human Factors in Security

- Reasons for user error
    - Bad UI/UX leads to mistakes

        - If pilot makes mistake, system should change to make that mistake harder to make (blame system, not person in long run)
    - Rational ignorance
        - Reason: security/system is too difficult to understand
        - Cost of user informing him/herself seems higher than cost of breach
    - Heuristic decision making and cognitive biases
        - Could be exploitable by adversary
    - Relying on user intelligence/designing for yourself
- Wifi encryption
    - General recommendation: wifi networks should be encrypted
    - Reality: open wifi networks are not encrypted
        - PUWireless is a closed network that should be encrypted, but isn't
    - Problem
        - Key distribution to all devices using wifi network
        - Someone joins airport wifi access point to access internet, but doesn't know how to enter key
    - Possible solutions
        - Exploit physical proximity between devices
            - "Tap to pair this device"
            - Line-of-sight medium
        - Trust on first use policy (TOFU)