Key principles of designing secure systems:

- Economy of mechanism (KISS): don't use a hammer when a gentle shove is enough
- Fail-safe defaults: most users don't change default settings, make defaults most secure
  - E.g. UNIX based on Access Control Lists deny access entirely if subject not listed on ACL
- Don't rely on security by obscurity: don't use your own functions, it will get leaked
- Complete mediation: every access to every object is checked by reference monitor
  - E.g. users have no direct access to disk (ask OS to provide access), can't snoop on others
- Least privilege: want users to have just enough powers to do what they need to do
- Separation of duty: no single person has enough power/need multiple to agree to launch
- Defense in depth: multiple walls; plan for defenses to fail, even if one does, exist more

	Symmetric (share secret k)	Asymmetric
Confidentiality, Secrecy	Stream/Block cipher	Public key encryption
Integrity, Authenticity	Message Authentication Code (MAC)	Digital signature scheme

Code Identity: measurement (captures behavior of system; binaries/libraries (f) + configuration files/inputs to f), allows inferring of proper control flow, type safety, correct information flow
Traditionally, CI information includes measurements of the entire software stack of every piece of code that was run since boot → simplify by using **Policy Enforcement strategy**: give some trusted piece of software (e.g., OS) a policy stating that it cannot run, e.g., any programs that aren't signed by Microsoft (we can ignore the specifics of the applications and drivers that are run; anything above the OS is being enforced by the OS and does not need to be in the TCB)
BUT: OS is large with possibly many bugs → **Minimum Security Layer**: rely on a smaller VM monitor (once VMM takes over, never gives up control) → either trust VMM + VM it's executing, or embed policy in VMM to only run approved VM (remove VMs from set of trusted components)
Threat Model includes **assets** (what I am protecting/what matters most), **system goals** (functionality, security), **adversary definition** (risk assessment, includes goal + capability)

- Network: attacker has complete control over the network (read, intercept, inject messages)
- System: modify software on disk, reset machine, cannot break hardware protections
- Cryptography: cannot break cryptography (CRHF, signatures unforgeable)

Bootstrapping trust: what function f does machine compute?
Secure Boot: enforces a trusted starting state, but does not mitigate runtime vulnerabilities

- Use hardcoded public key (stored in root of trust) to authenticate allowed list of software
- List of approved software measurements + signature over list on regular (untrusted) storage
- Software X measures X+1, if measurement on allowed list (signature verified) execute, else halt

Trusted Boot: record CI and store in secure storage in hardware via hash chain

- X measures X+1, hashes new code with previous value $V_{N+1} \leftarrow \text{Hash}(V_N, SW_{X+1})$, executes
- V_0 is hardware-defined default value; attacker cannot create malware that produces same $V_N{}'$
- Local access control (sealed storage), remote attestation (convince 3rd party device is good)

Network: best-effort delivery of packets (structured sequence of bytes consisting of **header** (metadata used by network) and **payload** (data to be transported)) between hosts
Network layers: application (HTTPS), transport (BGP, TCP), network (IPv4), data link (ethernet)

Firewalls block/filter/modify traffic at network-level, installed at perimeter of the network

- If service compromised in DMZ, cannot affect internal hosts

Firewall placement

- Host-based: faithful to local configuration and host state, travels with you, but no network correlation, must be installed, configured, maintained on every host
- Network-based: can correlate among nodes, protects every host in the network, but unknown host app state, and must replicate host network state

Types of firewalls

- Stateless packet filtering: inexpensive, low fidelity, filter by packet header (IP fields, protocol, flags, payload up to single packet) – since firewall at network level, all headers accessible
- Stateful inspection: add storage across packets; added state (+ obligation to manage)
- Application proxy: check protocol messages directly

Transmission Control Protocol: reliable stream delivery service operating over IP, provides host-to-host connectivity, uses 3-way handshake (SYN (sequence) + ACK)
Intrusion Detection System: policy based (e.g. regex/snort rules), anomaly based (using distribution of "normal" events, e.g. working sets; learning good distribution is hard)
State holding (assume stateful TCP policy): SYN flood, exhaust firewall resources, sneak packet
Overlapping fragment: relies on data fragmentation, reassembly based on offset
Time to Leave: to prevent packets from circulating, each packet has a TTL beyond which it'll disappear; break up "ATTACK" with packets with low TTL, bypass policy

$$\text{Intrusion rate} = \Pr[I] = \frac{|I|}{|\Omega|}, \qquad \text{Alert rate} = \Pr[A] = \frac{|A|}{|\Omega|}$$

$$\text{Detection rate} = \Pr[A \mid I], \qquad \text{Bayesian dr.} = \Pr[I \mid A]$$

Sound $\Leftrightarrow A \subseteq I$, complete $\Leftrightarrow I \subseteq A$, base rate fallacy

https:// *(Protocol)* mysite.com/ (*Domain*) *translated to IP address by DNS* folder/page? (*Path*) *resource hosted on server* id=5 (*Query string*) *passed to dynamic script*

GET: (officially) to request static content pages/resources, can encode arguments in URL
POST: form submissions/logins, typically encodes arguments in HTTP request (URL also can)
Cookies: create a persistent state to introduce notion of a session (HTTP is a stateless protocol)
Same Origin Policy: isolate content from diff. origins (scheme, domain, port, based on URL), secrecy (script from evil.com cannot read data from bank.com) integrity (…cannot modify).

- Browser goals: safe to visit evil.com, visit two pages simultaneously, safe delegation (A open B)

Information disclosure: fail to have complete mediation
Password guessing: (online) defend via limiting guesses, minimize use of popular passwords; (offline) steal hashed passwords, defend via slow hash functions, better pwds (harder to guess)
Cross-site scripting (XSS): bypass SOP, website display user-supplied content laced with malicious HTML/JS: Attacker takes advantage of browser's trust in web server

- <img src = "bad" onerror="document.location = …/document['cookie']>
- **Reflected XSS**: attack string included as part of crafted URI or HTTP parameters, improperly processed by the application, and returned to the victim *after they click the link*
  - Less vulnerable if website expects parameters to arrive via POST
- **Stored XSS**: server stores JS-infused input, users who view it are exploited (work with POST)

Cross-site request forgery (CSRF): takes advantage of server's trust in browser (that requests are initiated by user); browser executes unwanted state-changing actions while authenticated

- Defense via **secret token validation**, e.g. (HMAC of) session identifier; harder with POST

Shellcode injection: ip=127.0.0.1 & netcat -v -e '/bin/bash' -l -p 31337&submit=submit
SQL injection: 1' or true order by *n*; … union select username, password, 3, 4 from users; #
- **Blind** (server responds with generic error message): ask T/F, exploit side channels
- **Defense**: don't take user input directly; parameterized queries with bound parameters

Unintended file inclusion (local or remote): user inputs passed into file include commands
Mass assignment vulnerabilities: attacker guesses common sensitive field (…&admin=true)
CSS history probing: use JS + CSS to check which links were visited
Clickjacking: malicious site tricks user into clicking on some element on page unintentionally
- "Like" button needs to access cookies for domain FB; should be isolated from rest of page
- IFrame (parent + embedded page): by SOP, DOM (doc. obj. model) from one domain cannot access DOM from one domain → host cannot click button/check if it's clicked
- **Framebusting**: page stops functioning when included in a frame

Phishing (malicious links), **Tracking**

Malicious Browser in the Middle:	**Malicious Browser + Vendor**:

Lampson's Gold Standard: access control for web security
- **Authentication**: e.g. passwords/secure hardware, multi-factor mechanisms
- **Authorization**: policy specified by **Lampson's Access Matrix**
  - State of system: current values for all resources of the system
  - Protection states P: subset of state that deals with protection
  - Security policy: characterizes authorized states in $Q \subseteq P$
  - Access control matrix: precise representation of P
  - Security mechanism: prevents system from entering $P \backslash Q$
  - $S \subseteq O$ = set of active objects (protected entities relevant to system, e.g. files/dir/memories)
  - Entry $M[s_i, o_j] = \{r_x, \dots, r_y\} \subseteq R$ means subject s_i has such rights over object o_j
  - **Mechanisms**: Discretionary Access Control (user can set access control mechanism to allow or deny access to an object, e.g. UNIX), Mandatory Access Control (system mechanism or admin controls/mandates access to an object and an individual user cannot alter that access) Role–Based Access Control (access based on role not identity, orthogonal to above)
  - **Implementation strategies** (ACM is precise but huge):
    - Access Control Lists (columns of ACM, set of pairs (s, rules), no rights by default),
    - Capabilities (rows, pairs of (o, rules), stored in kernel (protected) memory and only accessible indirectly OR memory words with associated *tag* bits only modifiable in kernel mode OR encrypted and cannot be modified w/o detection by user processes)
- **Audit**: did A delete x; evidence for decisions, trails can help track attacks

AAA on the web: **authenticate** public keys by CA (**DHKE vuln. To MitM attacks**) (certificate transparency (maintain public, append-only log) for earlier detection of mis-issued or rogue certs. → **Audit**!), **authorize** website actions in the browser
Side-channel attacks: covert communication channel via cache (prime + probe, leak RSA keys), transient execution (Spectre, exploit speculative execution), Hertzbleed (more bit flips → higher power consumption → lower CPU frequency → less cycles → longer timing)
Trusted Execution Environments: isolate a lower privileged component (e.g., application) from a higher privileged one (OS/VMM) → harder to hack applications even if OS/VMM compromised

Adversarial ML (attacks on classifiers): **evasion** (classifier trained normally; attacker manipulate test-time input to cause misclassification), **poisoning** (attacker manipulates training data, test-time input normal but classifier give wrong result), **model theft** (classifier trained normally on private or secret data, attacker interacts with classifier, reconstructs training data)
k-anonymity: linking on quasi-identifiers (set of attributes that can be linked with external data to uniquely identify individuals) yields at least k records for each possible value of QI
alternatives: L-diversity, M-invariance, T-closeness
Impossibility result: for reasonable "breach", if sanitized database contains information about database, then some adversary breaks this definition (e.g., X is the height of an average man)
Differential privacy: computed statistic similar if any individual's record is replaced in the DB/ participation in DB does not increase risk of revealing secret/no worse off from records included
Randomized sanitization function $\kappa: D \to S^*$ has ε-differential privacy if for all data sets D_1 and D_2 differing by at most one element and all subsets S of S^*, $\Pr[\kappa(D_1) \in S] \leq e^{\varepsilon} \Pr[\kappa(D_2) \in S]$
Composition theorem: If κ_1 is ε_1-differentially private and κ_2 is ε_2-differentially private then $\kappa_1; \kappa_2$ is $(\varepsilon_1 + \varepsilon_2)$-differentially private; repeated querying degrades privacy gracefully
Sensitivity $S_F = \max_{n(x,x')} |F(x) - F(x')|$, achieve ε-DiffPriv by $\kappa(DB) = F(DB) + \text{Laplace}\left(\frac{S_F}{\varepsilon}\right)$
No security without usability → insecurity from (1) assume knowledge or abilities users don't have, (2) unreasonable effort, (3) unhelpful/confusing UI, (4) unqualified user makes choice
Usable security hard: (1) too many terms (mental model mismatch of users and professional's understanding of security differs), (2) complexity of policies, (3) security not priority, (4) habituation if overwhelmed, (5) users are different (different threat models)
Strategy: (1) make invisible (when possible), (2) offer better user interfaces (affordances (natural course of action), mappings (relationship between controls and their effects), mental models → design UIs to match what people expect) (3) train users (where necessary)
Bad UI/warnings: (1) no clear statement of risk (e.g., "The server might not be who it claims"), (2) missing explanation of consequences (e.g., "A third party could gain access to the information you create or share with this application"), (3) no instructions on how to make situation better (e.g., "Check the certificate and decide if you trust this site…")
Fair Information Practice Principles (1973):
- There must be no personal data record-keeping systems whose very existence is secret.
- There must be a way for a person to find out what information about the person is in a record and how it is used.
- There must be a way for a person to prevent information about the person that was obtained for one purpose from being used or made available for other purposes without the person's consent.
- There must be a way for a person to correct or amend a record of identifiable information about the person.
- Any organization creating, maintaining, using, or disseminating records of identifiable personal data must assure the reliability of the data for their intended use and must take precautions to prevent misuses of the data.

Privacy protection models: [EU] comprehensive model, privacy as fundamental human right [US] sectoral model, self-regulatory, mostly sector-specific laws, with relatively minimal protections (patchwork quilt), no explicit constitutional right to privacy (*also: co-regulatory*)
Crypto wars: export of tech. classed as 'critical' required a license, crypto considered military