0.0.1 Risk analysis

Quantitative	Assign real numbers to costs	Annual loss exposure	probability of an event occurring
Qalitative	Judges relative risk to threads	Based on intuition, judgement, experience	subjective, no hard numbers

Avoid risk	implement a control or Δ design	**Assume**	detect, recover. Plan 4 fall out
Transer risk	Δ design to introduce $\neq$ risk $\vee$ Buy insurance	**Reduce likelihood**	reduce chance of risk
Reduce conseq	Reduce the downside of risk occurrence.		

0.0.2 Cyphers

Summary Symmetric Encryption Algorithms

(bits)	**DES**	**3DES**	**AES**	**Type of attack**	**Know to cryptanalyst**
Plaintext block	64	64	128	Ciphertext only	Ecryption algorithm (EA), Ciphertext (C)
Ciphertext block	64	64	128	Known plaintext	EA, C, $1 \geq P - C$ pairs formed w key
Key size	56	112 or 168	128, 192 or 256	Chosen plaintext	EA,C, P chosen by crypta, corresponding C
				Chosen ciphertext	ciphertext chosen by crypta. and decrypted P gener
				Chosen text	EA, C, chosen P and chosen C

Fiesel Cipher structure: It is a structure that many symmetric block encryption algorithms use. The inputs to the enc. alg. **DES:** $|P| = 64b, |K| = 56b$. $\exists$ 16 rounds of processing. From the original $56b$ K, 16 subkey are generated, which are used for every round. Decryption is essentailly the =. Use the C as input, but subkeys K_i in R order. **3DES:** $C = E(K_3, D(K_2, E(K_1, p))) \wedge P = D(K_1, E(K_2, D(K_3, C)))$. 3DES is very strong, however it is very slow! **AES:** Iterative cipher, does not use Fiesel network. Its algorithm processes data as 4 groups of 4 bytes, called states. then applies 9/11/13 rounds in which state undergoes: *Byte substitutions, shift rows, mix columns, add round key.* All this operations can be combined into XOR or table lookups, so its very fast. Description is the same but inverse. **RC4 Alg.** RC4 is a stream cipher designed by RSA. It is a variable-key-size stream cipher with B-oriented operation. The algorithm is based on the use of a random permutation. Its period is > than 10^{100}. **RSA algorithm**:

Key generation d s.t. $de \bmod \phi(n) = 1$	p,q prime, $p \neq q$ PK= $[e, n]$	$n = pq$ PK= $[d, n]$.	$\phi(n) = (p-1)(q-1)$	e s.t. $gcd(\phi(n), e) = 1$
Encryption	Plaintext: $M < n$	C: $C = M^e \bmod n$	**Decription**	Plaintext= $M = C^d(\bmod n)$

Diffie-Hellman:
the purpose of the alg. is to enable two users to exchange a secret key securely that can then be used for subsequent encryption of messages. The alg. itself is limited to the exchange of the keys.

Global public elements	**UA K Generation**	**UB K Generation**	**Gen. Secret K by UA**	**gen. Secret K by UI**
p prime	Priv. $X_a < q$	Priv. $X_b < q$	$K = (Y_B)^{X_A} \bmod q$	$K = (Y_A)^{X_B} \bmod q$
$\alpha < q$ primite root of q	Public $Y_A = \alpha^{X_A} \bmod q$	Public $Y_B = \alpha^{X_B} \bmod q$		

Birthday attack: Works like this. Opponent generates $\frac{2^m}{2}$ variations of a valid message w same meaning. Then, generates $\frac{2^m}{2}$ variations of a desired fraudolent message. Chance he finds a collision is $\frac{1}{2}$ by b-day paradox.
Secure Hash function requirements **(1)** item H can be applied to a block of data of any size. **(2)** H produces a fixed- length output **(3)** $H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical **(4)** For any given code h, it is computationally infeasible to find x such that $H(x) = h$. A hash function with this property is referred to as **one-way** or **preimage resistant**. **(5)** For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$. This is called **second preimage resistant (6)** It is computationally infeasible to find (x, y) s.t. $H(x) = H(y)$

	Sha-1	**Sha-256**	**Sha-348**	**Sha-512**
Message size	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block size	512	512	1024	1024
Security (b-day attack)	80	128	192	256

Cipher Block Modes of Operation A symmetric block cipher processes one data at the time. You need a ways of breaking the plaintext into blocks.

Mode	**Description**	**Typical Application**
Electronic Code Book (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	Secure transmission of single values
Cipher block Chaining (CBC)	The input to the enc. alg. is the XOR of the next 64 bits of plaintext and the preceding 64 bits of cipher text	General-purpose block-oriented transmission, Authentication
Cipher Feedback (CFB)	Input is processed s bits at a time. Preceding ciphertext is used as input to hte enc. alg. to produce pseudoramdom output, which is XORed with plaintext to produce next unit of ciphertext	General-purpose block-oriented transmission Authentication
Counter Mode	Each block of plaintext is XORed with an encrypted counter. The counter is ++ for each subsequent block	General-purpose block-oriented transmiss High-speed requirements

0.0.3 Access Control

Authentication: verification that the claimed identity of a user or other system entity is valid. **Authorization**: Granting of a right or permission to a system entity to access a system resource. Determines trust **Audit**: Independent review and examination of sys records and activities to test for adequacy of sys controls, ensure compliance and detect breaches. **Discretionary Access Control** Scheme in which an entity may be granted access rights that permit the entity,to enable another entity to access some resource. An **Access matrix**, tell who can do what to which files. If cut by columns, then **Access control lists**. If cut by row, **Capability tickets**. A **subject** is an entity capable of accessing objects. **Role-Based Access Control** RBAC is based on the roles that users assume in a system rather than the user's identity. It redefines a role as a job function within an organization. Then, those roles are put in an ACM like in DAC.

Models	Hierar	Const	Entity	Definition
$RBAC_0$	No	No	**User**	individual that has access to this comp. sys. Each user has an user ID
$RBAC_1$	Yes	No	**Role**	Named job function within the organization. $\forall$ jobs $\exists$ a description of the authority of u
$RBAC_2$	No	Yes	**Permission**	Approval of particular mode of access to one or more jobs.
$RBAC_3$	Yes	Yes	**Session**	A mapping btw a usr and an activated $\subset$ of the {} of roles to which the user is assigne

Constrains: Means of adapting RBAC to the specifics of administrative and security policies in an organization. A constrain a a defined relationship among roles $\vee$ a condition related roles. $\exists$ following types:

Mutually exclusive roles	user can be assigned only one role in the {}. Can be static $\vee$ dynamic. Thus, non overlapp
Cardinality	Setting a max. # with respect to roles. Usr can only be assigned to a given role. Also, set max # of ro
Prerequisite	Usr can only be assigned to a particular role if it has some prerequisites. Useful to give

0.0.4 Database security

Making Queries	Select Name, UID, Financial aid from Students	**Create Views**	*create view* view_name *as* query_def
Grant sytanx	*grant* privilege_list *on* resource *to* user_list;	**Revoke**	*revoke* privilege *on* resource *from*
Row-Level AC	*create view* employee *as select * from* employee *where* name = 'Carol'	*grant select on*	employee_Carol *to* carol;
Delegate Policy	*grant* privilege *on* resource *to* user *with grant option*	**Inference**	Use non-sensitive data to deduce sens
Indirect Inference	work with statistical results to obtain data	if set too small	server doesn't give data.

Database encryption makes data to be stored and retrieved encrypted. Queries are made over encrypted data, that never gets decrypted on the server. The problem though is that key management in such a model is simply a mess, as well as queries.

0.0.5 SSL-IPsec

SSL is a transport layer security, designed to procide confidentiality, integrity and authentication at the end points. The internet protocol version is called **TLS**.

SSL Session	SSL Connections	Decomposition
Association btw client and server	transport-L connection	Ethernet frame Header(H)
Created by Handshake protocol	Protentially many connections per session	IP Header
Defines set of crypto. sec. parameters	Share crypt. par. of session	TCP Header TCP data stream, encrypted and authenticated

0.0.6 Key management

Digital Signatures: Scheme for demonstrating the authenticity of a digital message or document. You can do so by picking good session key, which are used for bulk encryption. However, it is hard to do so, because they are based off *pseudo-random numbers*. Another way of doing this is **Trusted Third Party**, which allow you to exchange key with ease.
Kerberos: Authentication protocol, which works on the basis of "tickets" to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner. its designers aimed primarily at a *client-server* model, and it provides *mutual authentication*. Kerberos protocol messages are protected against eavesdropping and replay attacks.
Web of trust: Used in PGP, GnuPG, to establish the authenticity of the binding between a public key and its owner. Its decentralized trust model is an alternative to the centralized trust model of **Public Key Infrastructure (PKI)** which relies exclusively on a certificate authority. However, you still have the problem of how to manage the root.

0.0.7 Authentication

Authentication: Binding of identity to subject. Involves two steps: Identification: present identifier to security system. Verification: Present authentication information to bind identity.
Markov Model: Reduce space requirements of bad password list. Create model that represents the bad password database, created from trigrams from words in bad password list.
Bloom Filter : $\neq$ way of encoding bad pswd dictionary in a small cheap DS. Create N bit array, and use k independent hash functions which hash into a space of 0 to $N-1$. $\forall\ h(x)$, set the corresponding bit in the hash table to every hash value.
Rainbow Table : Is a precomputed table for reversing cryptographic hash functions, usually for cracking password hashes. Tables are usually used in recovering plaintext passwords. Trades space for processor.
Salt Value : Random number added to the password to make offline cracking harder.
Shadow password files: Hashed passwords kept in a separate files from the user ID.
One-Time pswd: pswd that can only be used once, than invalidated. Problem is synchronization of user, system, generation of good random pswd and distributing them.

0.0.8 Mandatory Security Policies

DAC: Normal usr can Δ AC state assuming they have permission. It is at the discretion of the user. **MAC:** Access decisions cannot be Δ by normal rules. Generally enforced by system wide set of rules. Normal usr cannot Δ ACS. Stronger protection.**Confidentiality Policy:** Prevent the unauthorized disclosure of information. **Simple Security P.:** No read up. Subj. can only read iff level(O) ¡= level(S).**No write down:** Star p., subject can only write to Level(S) ¡=Level(O). **Adding Sec. clearance flexibility:** Define max and current level for subjects.**Principle of tranquility** Raising object's security level. Information once available so some subjects is no longer available..**Strong tranquility** The clearances of subjects and the classification of objects do not Δ. **Weak tranq** The clearance Δ according to specified policy. For the next table: Subjects S, objects O, integrity levels I, relation $\leq I \times I$ holding when

second dominates

Biba integrity model	**Strict Bell LaPadula**	**Low-Water-Mark policy**
$min : I \times I \to I$= lesser of lvls	$s \in S$ can read $o \in O$ iff $i(s) \leq i(o)$	S can w to o iff $i(o) \leq i(s)$
$i : S \cup O \to I$ gives integrity of entity	$s \in S$ can w to o iff $i(o) \leq i(s)$	if s reads o, then $i'(s) = min(i(s), i(o))$
$r \subseteq S \times O$ means can read.	s_1 can ex s_2 iff $i(s_2) \leq i(s_1)$	s_1 can ex $s_2 \in S$ iff $i(s_2) \leq i(s_1)$.

0.0.9 Trusted systems and examples

Definition: System that employs sufficient HW and SW to assure mechanisms to allow its use for simultaneous processing of a range of sensitive or classified info. Implements strong security mechanisms.**Trusted Computing Base** TCB contains elements of HW and SW that enforce security. Must be tamper proof, and cannot be circumvented. **Memory Protection rings:** Assign different security levels to memory, one for each program. **Privilege levels:** CPU enforces on memory access and Δ of control between $\neq$ programs of $\neq$ privilege. Also enforced by HRDW, so malicious code can't jump into kernel space. **Stack switching** Happens when calling more privileged code. Prevents less privileged code from passing in short stack and crashing more privileged code.**Hardware rings** usually only two installed. Implements limiting memory access.**Trusted Computing Group** Consortium developing standards for computer architectures using secure co-processors. **TPM basics:** TPM stores a number of key pairs. Has protected storage and can be used to boot strap security locally. **Certification services** They measure values of data, as well as the digest of hashes. **Integrity reporting** ensures the authenticity of stored values based on trusted platform identities.**Usage scenarios:** Store root secrets in secure co-processor. In an enterprise, IT group is responsible for machien admin. Ensure platform is in particular configuration.

0.0.10 Malware overview

0 Day Exploit exploit that has not been patched yet.**Malicious code:** set of instructions that cause a site's sec. pol. 2 be violated.

Virus	Attaches to prgrm and copies to other prgrm	**Trojan Horse**	Unexpeced, additional f.
Logic Bmb	Triggers action when condition occ.	**Logic bomb**	Triggers action when time occurs
Trap door	Allows unauthorized access to funct.	**Worm**	Propagates copies through a network
Rabbit	Replicates itself w/o limit to exhaust resources	**Netbot**	Trapdoor prgr controlled through network
Root Kit	hooks standard OS calls to hide data		

Rootkit: replace function table entries, Δ syscalls. Hard to defend. **Virus parts:** (1) Infection mechanism (2) Trigger (3) payload.**Macro virus** Macro code attached to some data file. Interpreted by program using the file. **Virus scanners**:

1st	Check signature virus by patterns of code. Rec only known sign	**2nd**	use heuristics rather than signatures
3rd	Track virus by action. Look 4 anomalous behaviour	**4th**	Use scanning, Acces control, Behavioural analysis

Worms: Spread themselves through the network. Uses scanning to send packets to random addresses. **Botnets:** Install on compromised machines. Master sends commands to bots, can control through P2P masked connections.

0.0.11 Network and DoS

DoS: Basic: use simple flooding ping. Higher capacity link floods lower capacity link. **Address spoofing** Sender can put any source address in packets he sends. Some routers can catch some spoofing by using reverse path or egress filtering. **ARP** unsed to discover mapping of neighboring ethernet MAC to PI addresses. **ARP cache poisoning** bootstrap problem w respect to security. Use man-in-the-middle attack. However, can encrypt traffic to avoid problem. **Smurf attack**: an amplification DoS. Send ICMP echo request to IP broadcast addresses. Spoof the victim's address as he source. **Transport level - TCP and UDP** Service to service communication. **Syn flood** resource DoS attack focused on the TCP three-way handshake. flood the SYN table w messages. **DNS problems:**DNS open relays Makes it look like good DNS server is authoritative server to bogus name. DNS Cache poisoning Δ the name of addrs mapping to attacker address. **DNS communication** Use UDP. Requests and responses have matching 16 bit. Requests and responses have matching 16 bit transaction Ids.

0.0.12 Software security

Sec in design: Concerns must be considered up front, as they impact system arch. Can't do it in test phase.**Defensive programming:** The goal is to provide continued functioning of software in spite of unforeseeable usage of the software. Don't make any assumption about ur system. **Handling User Input:** must verify the input, and its length. can use a white list of expected results.**Injection attacks:** Error in input handling that results in unexpected execution flow. Often occurs in scripting languages.**XSS** Inject malicious code into web pages. Need to validate data supplied in order to avid this kind of attack. **Input fuzzing** Generate "random" inputs to test the program. **Writing correct code** Is you algorithm correct? TCP session hijacking. **Correct use of memory** Ensure no memory leaks, no malloc errors. use heap randomization and tools to track heap utilization

0.0.13 Buffer Overflow

Overwrite saved return address to its convenience. **Shellcode**: hex value of a program to call a shell w high privileges. Cannot use bytes w NULL value. **Defenses:** Compile-time include: Programming language choice, extensions/ safe libraries and stack protection. Run-time include executable address space protection, address space randomization.

0.0.14 Intrusion Detection

Intruders! Masquerader: they pretend to be legitimate users. Misfeasor: legitimate user behaving badly. Clandestine: seizes prvileged control of the system. Able to clan up audit logs and evade access controls. **Intrusion detection** *Anomaly:* use statistical techniques to determine unusual behaviour. *Mis-use* use signatures to determine occurrence of known attacks. Detection can be performed on host data &,or network data.**Bad detections** false positive: detect activity as an intrusion. Reduce by loosening intrusion detection rules. False negative miss reporting bad behavior as intrusion. **IDS architecture** agent/sensors run at @ lowest level gathering data. Perform basic processing. Then, sends to analyzer that performs more significant processing of the data. Potentially there is a hierarchy of agents and directors.**Data sources:** Direct: network packets, system calls.indirect syslog, events from other intrusion, network info generated

by routers. **Mis-use detection** uses viruses detection techniques to detect bad behaviour. signatures are created by analyzing historical data. **Intrusion protection systems** Requires very fast signature handling. it is a type of NIDS. **honey pot:** deplu a fake system, observes it being attacked. It is a resource management.

0.0.15 Firewalls

Goal: insert *after the fact security* ba wrapping or interposing a filter on network traffic. **Requirements:** all traffic btw section A and section B must pass through FWL. Only auth traffic is allowed to pass, and the FWL is immune to penetration. **Packet filter firewall** Can block traffic based on source, destination address, ports and protocol.**Stateful inspection FWL** evolved as packet filters aimed for proxy functionality.can also reconstruct layer 4 traffic. This type of FWL can be expansive.**Application Proxy FWL:** firewall software runs in apps space on the firewall. The traffic source must be aware of the proxy and add an additional header.

0.0.16 Design principles

Economy of Mechanism Keep design as simple as possible simpler = less can go wrong	**Fail-safe default** Base access on permission, no exclusion If prot. sys fails, everybody is denied.	**Complete mediation** $\forall$ access to $\forall$ obj must be checked if permission Δ after, get unauthorized access
Open design *The design should not be secret* don't depend on ignorance of attacker, on possession of specific keys or passwords	**Separation of Privilege** 2 keys is more robust than 1 key separation of duty defence in depth	**Least privilege** $\forall$ *program, user of sys should operate* *w least set of privileges to complete the job* minimal protection domain
Least Common Mechanism minim. amount of mech cmn to >1 usr shared mechnism have > insrnc than non shared	**Psychological Acceptability** Its esntl that the human interface be designed 4 ease of use, so can be user routinely	**Key points** require: careful analysis careful implementation

0.0.17 Privacy enhancing technologies

What is privacy? "The right to be let alone". **What is anonymity?** Unobservability, unlinkability, sender anonymity and receiver anonymity. **Chaum's MIX** uses public key cryptography for anonymous e-mail. Provides unlinkability. **MIX cascade** what if some of the mixes are controlled by adversaries? then a cascade of mixes can withstand N-1 adversaries. **Random routing**: hide message source through random routing. Routers don't know who the source message is. **Onion routing** sender chooses a random sequence of routers. Goal: hostile routers shouldn't learn Alice is talking to bob. Do so by using multiple encryption levels. **Crowds** Routers form a random path. Now, routers choose path, not sender. $\forall$ routers, if $(flip(p))$, the router forwards to another router, else forwards the message to the recipient.**Probabilistic notions of anonymity** Has to be beyond suspicion: the observed source of the message is no more likely to be the true sender than anybody else. Probable innocence and possible innocence: non-trivial probability that the observed source of the message is not the true sender. **Digital cash** anonymous payment system, which uses blind signatures in order to cover up.

0.0.18 Side channels

Side channel Information disclosure through *physical* properties of implementation. **Timing attacks** execution time of operations varies depending on data. Attacker can measure timing, recover secret information.**RF noise** Computers emit RF, with high-gain antenna, can monitor activity from a distance **Power analysis** computers electronics leak information through power. Power analysis is especially useful for smart cards. **Stenography** greek for covered writing. embed hidden message in other communication. can use: spaces, word length, timings, low bits of images/audio. **Secure deletion** how do you make sure a file is really gone? must eradicate all copies.**Disk level issues** magnetic infromation persits after overwrite. a better approach is to never write confidential data to disk.

0.0.19 Security Audit

Security audit Independent review and examination of a system's records and activities to determine the adequacy of system controls.**Trail** chronological record of system activities that is sufficient to enable the reconstruction and examination of the sequence of environments and activities.**Logging** Application or system logging events. Create security trail. things to log: —session initialization —Auth.info — Auth. decisions —Network connections. **Audit event analysis** Need to log enough info to review correct enforcement. Some events dictated by need to comply w laws. timestamped.**Network security events** Ntwk device generates syslog messages. Intrusion detection device creates events. **Ensuring applications audit** Review/rewrite application to insert audit log calls. Interpose library. Use dynamically linked library to audit then perform the original library call. **Audit analysis** originally envisioned to be directed human analysis. Baselining, you try to understand the time of audit events generated in normal situations.**Conlusion** audit trail is necessary, and tends to be overlooked. No direct functional benefit and essential to determine that things are operating per security requirement.