**Part1: Information Security:**

Other = risks (supported by business decisions, processes, additional checks)

**Confidentiality** = preventing unauthorized disclosure

**Integrity** = prevent unauthorized modifications

**Availability** = prevent unauthorized withholding of info/resources

**Accountability** = cannot prevent improper action, securely identifying users, logging, audit trail

**Reliability/Dependability** = systems perform properly in adverse conditions

**Privacy** = personal data/PII, control and requirements on data holders

Others = Reporting, awareness, BCP, management

Security assessment = sec-related product to be assessed/certified with standards (PCI DSS)

**Functionality VS Assurance**

* F = Sec facilities it provides
* A = guarantees offered as F claims

Security **Threats Analysis**

combating perceived threats (not all are worth it, Costs-Benefits)

**Risk Analysis**

Importance of each threat (probability, severity) (if it should be combated)

**Providing Security**

as strong as the weakest link

Design > Adding after

**Focus of Control (Data Ops/User/Both)**

* How data handled
* Ops can be performed on which data
* Which user perform which action

**Location** (lower = more computer, higher = more user orientation)

Physical Sec (unauthorized access)

Data Comms (Crypto and MAC)

**Assurance VS Complexity**

High Assurance = Low Complexity e.g. Trusted Kernels

**Bypassing Security Controls**

-attacks privileged apps = bypass protection mech  
-e.g. gain access to insecure backup, access data before transmission

**Security Management**

Company Sec pol, BCP ISO/IEC 27002 = ISM

**Security Policies**

Set of rules specifying how it should be enforced (domain)  
- BCP, sec education, sec incident reporting

**UK Data Protection Act**

Tell users what data they hold  
- protect individual’s personal data

----------------------------------------------------------------

**Part 2: Elements of Cryptography**

Cryptography = study of secret writing

Cipher (enc algo) = method of transforming data so cannot be recovered by unauthorized (e)

Encryption = process of transforming (m) into unintelligible form using (e)

Decryption = process of recovering (m)

Decipher = (d)

Secret key = (k)

Plaintext = message into cipher (m)

Ciphertext = result of applying cipher to plaintext (c)

Cryptanalysis = deciphering a message by unauthorized party

Cipher = transform plaintext into ciphertext (with a secret key known to sender and reciever)  
c = ek(m)  
m = dk(c)  
typically (e) = public (hidden = fallacy)

Secrecy of (m) -> Secrecy of (e)

**Properties**

no. of possible keys = large  
- prevent exhaustive search  
worst case assumptions  
- full knowledge of (e)  
- no. of (c), all using same (k)  
- known (m) according to (c)  
- chosen plaintext attack (can keep encrypting)

**Analysis of security**  
- worst case assumption, try to break it (be cryptanalyst)  
- believed to be strong (best attempts of experienced ones cant break them)

**Broken Ciphers** - **Caesar Cipher**- each letter = number  
- add (k) = (mod 26) if (k) = 3 , "HELLO" becomes "KHOOR"  
- simple substitution cipher (easy break, 25 keys)

**Simple substitution ciphers**- Key = permutation of letters   
- more secure than Caesar cipher (but can be broken easily by hand)  
- 26! = 4 X 10^26  
- some letters are more common  
- enable guesses to be made

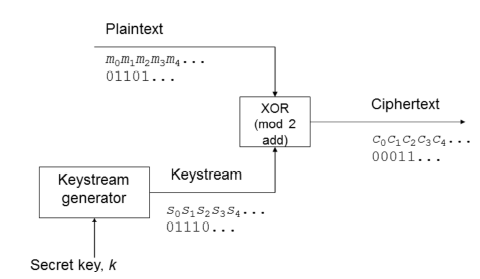
Period = 1, no Random, linear equ = 1

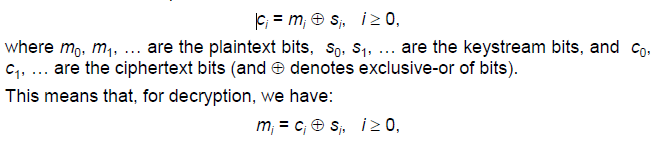
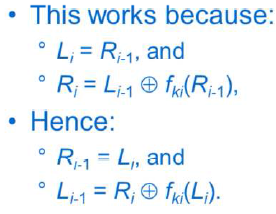
**Broken Ciphers - Vigenere Cipher**- polyalphabetic substitution cipher  
- take the number of (m) and the number of (k) and add them tgt, do it for the next letter in the (k) for next (m)  


Period = length of word

random = length of word,

**Modern Encryption Algo**- stream (bit by bit)  
- block (block by block)

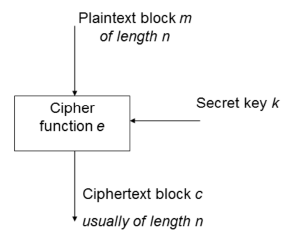
**Stream**  
- simple and fast to implement



Period = >264  
Uses 2 or more Linear Feedback Shift Register

**Keystream properties**  
- to be secure  
 - long period (length before repeat itself)  
 - pseudo-randomness  
 - look random  
 - large linear equivalence  
 - degree of complexity  
\*not sufficient to be strong, given one part, others must be COMPUTATIONALLY INFEASIBLE  
  
**Properties of Stream**-encryption = v fast  
-no error propagation  
-no protection against message manipulation (integrity)  
-same key twice = same keystream  
 - need for message/session keys  
(so will send a random msg, and the cipher text)

**One Time Pad (vernam cipher)** = special type of stream  
- pseudorandom keystream is replaced with random binary sequence, used ony once.  
- proven unbreakable (1949)  
- key management is hard  
 - key = as much as data

**Block Ciphers**

* Encrypt blocks (64/128/256 bits)
* Key (same size)
* c = ek(m)  
  m = dk(c)
* key > 64 to prevent dictionary attack (attack the block. Not the key)

**Simple sub cipher as a block cipher**

* can, block length =1
* too small to be secure

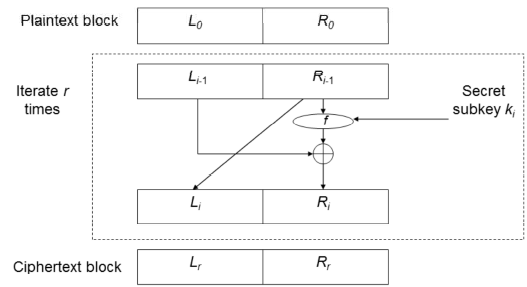
**Modern Block Ciphers**

* DES (Data Encryption Standard) defacto
* AES (Advanced ES) 2001

**Iterated block ciphers**

* Repeated use of round function (takes n-bit block to another n—bit block)
* No. of rounds = r (uses a subkey, from the main key)
* Decryption= subkey must be invertible (inverse function)

**Feistel Cipher**

* 

**DES**

* Feistel with 16 R
* block = 64, key = 56, subkey = 48

**Why not 2DES?**

* 2keys = 22n  , 1 key = 2n
* 2\* 56 = 112 keylength
* Single en = 2^3 = 8 ops (if key = 3)
* Double en = 2^6 = 64 ops (if key = 3)  
  \*But only used 18 ops (using MITM)
* It’s not more secure, it’s just longer to break

**2DES - Meet in the Middle Attack \*not in the exams!**

* C=ek1(ek2(m))
* Needs (m) and (c)
* Compute ek1(m) with all possible keys, Key1
* Then compute dk2(c) with key2
* Check for matching pair   
  (ciphertextK1 = plaintextK2)

****

* Using 2 pairs, to double check if the matching one is correct

**Summary**

* Theory = 64 ops, Practical = 18 ops
* 2^(56+1) of operations
* **Takes twice the effort, (not a lot stronger)**
* **Use 3DES!**

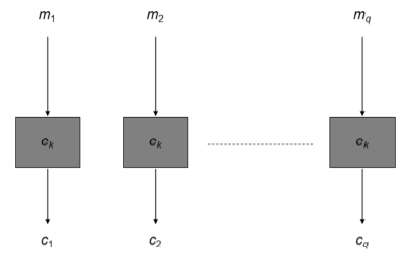
**3DES**

* C = ek1(dk2(ek3(m)))
* Simplifies migration if k1 = k2
* 3key variant = C = ek3(dk2(ek1(m)))

**AES**

* NIST initiative, new block cipher standard, oct01
* Must be 128, free public, variable key length, faster
* Rijndael = winner, not a Feistel

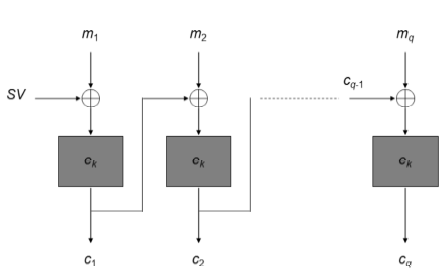
**ECB (Electronic Code Book) (Said smt is in impt here)**

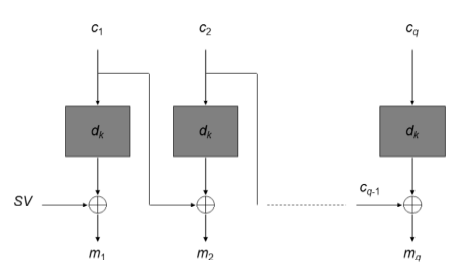
* Simplest
* ****
* Problem = mi=mj, then ci=cj

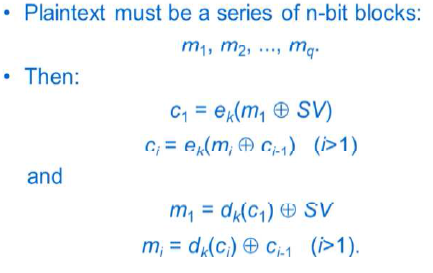
**ECB Cut and Paste Attack**

* “Kon goes” “to Gin” “Fin goes” “to Gem”
* Atk to “Fin goes” “to Gin” “Kon goes” “ to Gem”

**Cipher Block Chaining (CBC)**





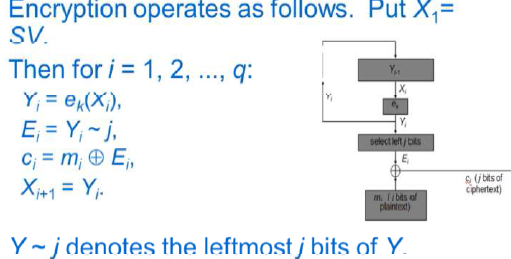


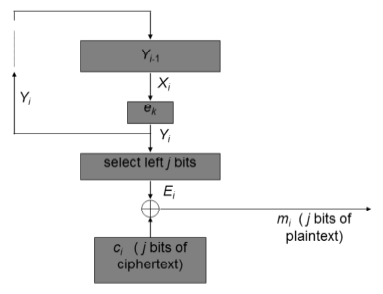
* Weakness = error propagation
* SV = starting var, diff for every msg

**Starting Value**

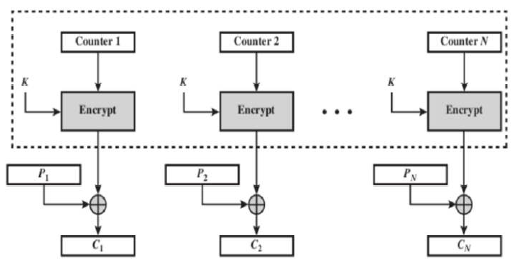
* protects identical (m) having same (c)
* SV = unpredictable
* Random(k) and encrypt a counter to produce SV
* Transmit in clear. Atker won’t know (k) to encrypt the counter and the SV

**Output Feedback (OFB)**OFB = Defers slight from CFB mode (Uses the part before XOR as the IV for the next one)   
MAIN DIFF = no error propagation





**CTR Mode**



**Properties of Block Ciphers**

* Error Propagation (depends on mode)
* Message manipulation protection (depends on mode)
* Message/session keys needed (depends on mode)

**Cryptanalysis of ciphers**

* Exhaustive search
* Pre-computed tables
* Divide and conquer
* Chosen plaintext (input special data, reveal properties of (k))

**Integrity Mechanisms**Protect against

* Alteration/Deletion/Insert/Replay
* Changing the order/Falsifying the origin

2 types (1 here, 1 with dig signature)

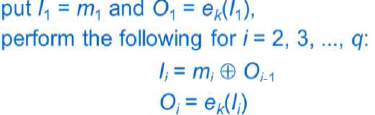
Diff = crypto algo, and use of keys (shared secret)

**Message Authentication Code MAC**

Key + function, MAC =fk(m)  
sends m and MAC  
Widely used = CBC-MAC  
Generated using a block cipher using CBC  
(standards not so impt)

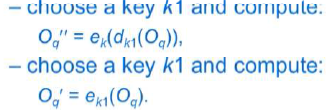
Operations

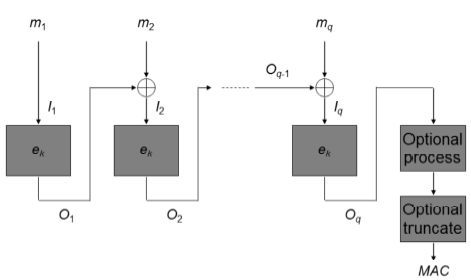
1. Pad data
2. encipher data using CBC



1. take final block as the MAC
2. Optional process/truncate

**Optional (encrypt hash)**

****



**Padding**

1. Adds as many zeros necessary
2. Adds single 1, then adds as many zeros
3. M1, add extra block with msg length

Length known, use M1, Else, use M2/3 (detects malicious addition/deletion of zeros)

**Attack for MAC**

1. Key Recovery ( learn secret key)
2. MAC forgery (work out Mac for new msg)
3. Cut and Paste (Forgery) Attack  
   - Use hash, add last block, and hash it  
   (not in Exam)

**Hash**

* add (k) to (m) and input into hash function
* Flawed, security problem

**HMAC – Hash-MAC**

* Much better than hash
* HMAC = h(K1 || h (K2||D))
* K1 and K2 are variants of Secret (k)

**MDCs – Manipulation Detection Code**

* Use with cipher
* Function of the whole data (no key)
* Concatenated with data, then encrypted
* Function chosen properly (cant just XOR)
* Cannot use stream cipher
* Problem arise with known plaintext atks
  + with (m) and (c), reveals keystream
  + new(m) can be (e), message forged

**Integrity and Confidentiality**

* MDC+(e)
* (e) + MAC
* But MDC = serious problem, avoid
* Use proposed (e) modes (OCB/OCBv2) = fast
* Atk on IPsec/SSH, confidentiality not enough
* Integrity = use MAC
* Confidentiality = use MAC + (e)
  1. It should be (e) + MAC, then use OCB

**Public Key Crypto**

Sender & receiver do not share a secret key

Sender = encryption key, Receiver has decryption key

Every user generates a key pair (decrypt, and encrypt)

Everyone send them a secret message (1 can decrypt)

**Asymmetric algo** = 1 key for encry, 1 key for decryp

* Decryption key does not reveal encryption key
* Either 1 can be used for encrypt, other for decry

**Symmetric algo** = same key

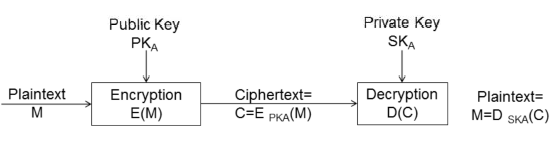
Public Key

* Encryption
* Authentication: used to verify a message (encrypt by private)

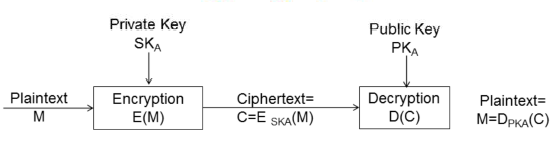
Private Key

* Secret
* Used of owner to decrypt (encrypt by public)
* Used by owner to provide authentication and Proof of origin

**Encrypt with Public Key**

****

Receiver with private (only owner can decrypt)

**Authentication with Public Key** (encrypt with priv)

Decrypt with public key (only sender has private)

**Assumptions**

* Computationally easy to generate keypair
* Easy to generate message with PK
* Easy to decrypt with SK
* Infeasible for attacker knowing PK and C to find SK or message

**RSA**

Based on the idea of difficulty of finding the prime factors of large numbers

Multiplication = feasible, factoring = (believed) difficult

Input easy to compute output,   
output hard to determine input

Generate = (at least 512) prime *p, q*. *n= p\*q*

Select *e* (greater than 1, less than *(p-1)(q-1)*

No num div neatly into *e* and into *(p-1)(q-1)* except 1

Publish (*n, e*)  
Private key *d* from *p, q, e*

ed = 1 mod (p-1)(q-1) (maths part not in exams)

C = Me mod n

M = Cd mod n

If P = 47, q = 71, n = 3337

****

**Security of RSA**

Depends on difficulty of finding d given n and e

If factorization = not hard, RSA = broken

Typical length = 1024, 2048, 4096

**Key Distribution**

Distributed by channels which no need privacy

Must be genuine

Removed the need for secrecy in key distribution (but still have authenticity)

Solution = use of certificates

**Properties of RSA**

* Special type of Block Cipher
* Much slower to implement than DES/AES
* Unsuitable for long message
* Encrypt session keys
* Need for formatting (before applying RSA)
  1. Add randomness – otherwise predictable

Others PKC

* RSA widely used, well-trusted
* Elliptic curve scheme (eclipse RSA)

PGP Not in Exams!!!

**Digital Signature**

= integrity mechanism

Unlike MAC

* Proof of origin
* Non-repudiation (sender cannot deny)

**Keys of Dig Sig**

Choose 2 keys

* Private signing transformation
* Public verification transformation

Private signing to m (adds a signature)

Sends m and the sig

Applies the verification transformation to m

Check if the sig is valid or not

Valid = message integrity + non repudiation

**Need for redundancy**

To verify if message is in natural language (not possible to check msg is natural every time)

Needs a mechanical way for verification that message is authentic. (to add redundancy)

**Other Sig Schemes**

* RSA not the only one
* ElGamal, Elliptic Curve (discrete logs)

**Long Messages**

Need break m into blocks, add serial num + redundancy

Time consuming!

Current model = recoverable

Other main type = Verify without recovery

**Digital Signature in Practice – Similar to MAC**

No message recovery

Gives Boolean (true/ffalse)

**Certificate Authority**

* Everyone registers with CA and gets CA public signature verification
* Every user submits their public key with CA
* Put all info tgt and generate sig (public key cert)
* Anyone with CA’s public key can verify user public key
* More than 1 CA = cross-cert (sign each other)

**Hash – Functions**

1-way function

* Prevent cryptanalyst from matching sig to wrong message
* Birthday paradox attack though
* 2n/2 trails to find collision or sqrt(m) samples
* Must use > 128 bits long

Make RSA into sig without msg recovery

Hash it then sign the hash

Collision-Resistant hash (if another msg =/= same hash)

**Hash-Functions in Practice**

SHA-1 used (280 ops, weak) SHA-0 = weak

SHA-2 (512 bits, strong)

SHA-3 KECCAK (best, new)

MD4/MD5 used (though weak)

**Uses**

MAC = Hash (Key + message), similar to HMAC

Protect integrity of large files

**Part 3: Identity Verification**

Need

* Computer access
* Entry
* Financial Security

Diff between identification info, and information needed to verify a claimed identity

Diff = identification & authentication

**Classification**

* You know
* You possessed
* Physical characteristic
* Result of involuntary action

First 2 with the last 2

**Passwords (you know)**

Security procedures

* Accountability
* Don’t write them down
* Hard to guess

Alternative = OTP

Storage

* How?
* Unencrypted = readable by staff
* Usual = hide using one-way function (easy to compute, difficult to invert)
* Check password by applying function + compare

Unix

* Uses slow encryption
* Salt (dict attack difficult, every password has its own unique salt value)

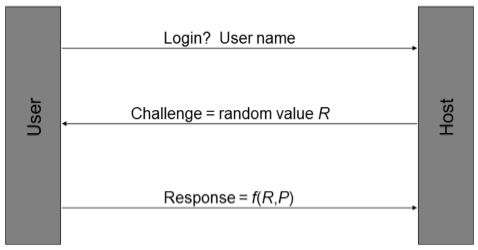
Problem in Unix

* Slow enc = not slow
* Cheap data storage = dictionary attack
* Public domain packages exist = run against password files (effective)
* Password = not guessable
* Unix “hide” the password file (root can see)

Transmission of password

* Insecure = vulnerable to interception
* Encryption = no working ( can replay )

Solution = Challenge-response



Properties

* User and system must know P
* One way function (must be quick)
* Insecure if not enough password (brute)

Solution = Tokens

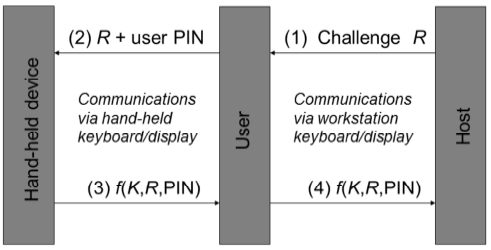
* Smart card/ Magnetic card
* Can be copied.

**Hand-held ID devices (alternative to smart/mag cards)**

* Key and display
* Key/password storage
* Crypto calculation facility

✓Can use with PC (no need card reader)

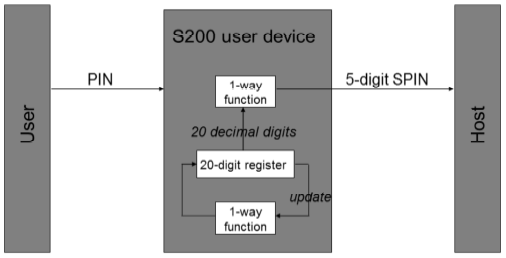
Watchword Protocol (user key, PIN, one-way f)



* liveness check

Safe 200 System

* New 5 digit password every identification



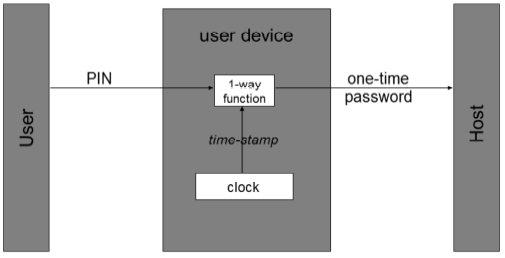
* Allows 3 iterations (before/after due to sync)

OTP – RC2289

* Shared-secret = pwd || seed
* Hash Chain  
  Apply F N times to (k) to get 1st password then apply N-1 times to get 2nd password
* Challenge = seed and N of times
* Response = Fn-1(f…(f1(s)))
* Attack = intercept challenge, and change N

Time-based SecurID

* Uses a clock value and secret key



**Biometrics**

* Passwords may be revealed/guessed
* Tokens may be lost/stolen
* Harder to forge
* System must be trusted

How?

* Person = unique
* Distinguish traits (What? How? How diff?)

Verification (1..1) VS Identification (1..\*)

Biometrics

* Universality – all should have char
* Uniqueness
* Stability – does it change over age
* Collectability – possible to measure
* Performance – accuracy, speed
* Acceptability – environment (ppl ok?)
* Forgery resistance

Static (physiological) Vs Dynamics (behavioral)

Static

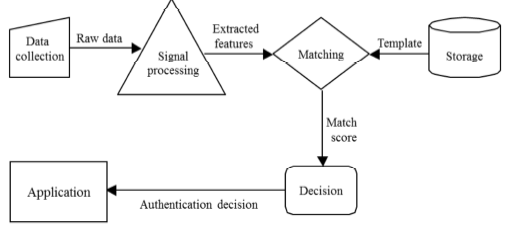
* Fingerprint
* Retinal
* Iris
* Hand geometry

Dynamic

* Signature
* Speaker
* Keystroke

Major Components of biometric system

* Data Collection
* Signal Processing
* Matching
* Decision
* Storage
* Transmission



Enrolment

* Data Collection and feature extraction
* Stored in DB/token
* Might need several iterations

Requirements (Enrolment)

* Secure enrolment procedure
* Quality/matchabiltiy
* Binding of template to enrolee
* Check:
  1. Already enrolled
  2. Similar to existing temp

Requirements (data collection)

* Sampled = enrolled template
* User require training
* Adaption of templates (if physio change)
* Sensors must be similar (consistent)
* Changes
  1. Feature may change
  2. Presentation may change
  3. Performance may change
  4. Environment may change

Signal processing subsystem

* Feature extraction
* Raw file from data collection subsys
* Transforms data required by matching subsys
* Discriminating features extracted from raw
* Filtering may apply to reduce noise

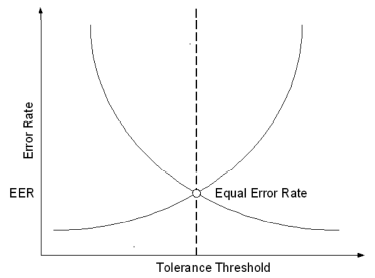
Matching subsys

* Receives data from signal and storage
* Measures similarity of claimant and reference
* Match score

Decision subsys

* Interprets the match score with matching subsys
* Threshold defined
* Binary decision: YES/NO

Possible Decision Outcomes

* Genuine = accepted
* Genuine = rejected ( Type I error, false rej)
  1. False non-match rate (FRR)
* Imposter = rejected
* Imposter = accepted (Type II error, false accep)
  1. False match rate (FAR)
* Balance required (in Type I and II)   
  Application dependent (jail/company)
*   
  Equal error rate (EER)

Storage subsys

* Maintains templates
* Physically protected
* Normal db, or stored in token/smartcard

Transmission subsys

* Logically separate (some physically)
* Vulnerable during transmission

**Biometric Technologies**

**Liveness Detection**

* Authentication with live user

**Fingerprint Recognition**

* Arch, loop Whorl, Ridge
* No liveness
* Minor injuries = problem
* Processing
  1. Original
  2. Orientation
  3. Binarised
  4. Thinned
  5. Minutiae
  6. Minutiae graph
* Advantages/Disadvantages
  + Mature
  + Easy to use/non-intrusive
  + High Accuracy
  + Long-term stability
  + Ability to enroll
  + Low Cost
  + Some users cannot enroll
  + Skin condition
  + Dirty sensors
  + Associated with forensics app

**Hand Geometry**

Width, length of fingers, etc

* Advantages/Disadvantages
  + Mature tech
  + Non-intrusive
  + High user acceptance
  + No negative associations
  + Low accuracy
  + High cost
  + Large readers required
  + Difficult for some users (child)

**Eye Biometrics**

**Retinal Scanning**

Retinal Vascular pattern inside of eyeball

* Advantages/Disadvantages
  + Potential for high accuracy
  + Long-term stability
  + Feature protected (from environment)
  + Genetic independence (even twins)
  + Difficult to use
  + Intrusive
  + Perceived health threat
  + High sensor cost

**Iris Scanning**

Colored portion & pattern of the eye surrounding pupil

High degree of randomness = very accurate

Fast processing

* Advantages/Disadvantages
  + Potential for high accuracy
  + Long-term stability
  + Resistance to imposters
  + Genetic independence (even twins)
  + Fast processing
  + Very low error rates
  + Liveness check
  + Intrusive
  + Perceived health threat
  + High sensor cost

**Face Recognition**Visible spectrum: inexpensive

Approach: eigenfaces, feature analysis

* Advantages/Disadvantages
  + Non-instrusive
  + Low cost
  + Ability to operate covertly
  + Affected by appearance/environment
  + Low accuracy
  + Identical twins attack
  + Privacy abuse

**Facial thermogram**

Capture heat emission patterns

* Advantages/Disadvantages
  + Non-instrusive
  + Stable
  + Not Affected by appearance/environment
  + Identical twins resistent
  + Ability to operate covertly
  + High cost
  + New technology
  + Potential for privacy abuse
  + Affected by state of health

**Signature Recognition**

Handwriting, Trained reflex

Variety of Characteristics

* Angle of pen, Pressure, signing time, velocity, acceleration, geometry

Advantages/Disadvantages

* + Resistance to forgery
  + Widely accepted
  + Non-intrusive
  + No record of signature
  + Signature inconsistencies
  + Difficult to use
  + Largish templates
  + Problem with trivial signatures

**Speaker Verification**

Acoustic patterns, anatomy (size/shape of mouth and throat), behavioral (voice pitch, style)

* Advantages/Disadvantages
  + Use of telephony infrastructure/mic
  + Easy to use/non-intrusive/hands-free
  + No negative association
  + Pre-recorded attack
  + Variability (ill or drunk)
  + Background noise
  + Large template
  + Low accuracy

**Choosing biometrics**

Identification VS authentication

Collection Pt Attended VS Unattended

Used to the biometrics?

Covert VS Overt

Co-op VS Non Co-op

Storage requirements constraints?

Performance requirements strict?

Acceptable to the users?

**PART 4: ACCESS CONTROL**

* keys distributed to users
* list of trusted users

**Fundamental Techniques**

* Authentication
* Authorization (access control)
* Memory protection (process cannot R/W mem of other process, aka buffer overflow)

**Terminology**

* OBJECTS Resources (files, dir, printers, sockets)
* SUBJECTS Active entities (Process, threads)
* PRINCIPALS Entities ( user, group, roles, keys)
  + Can create subjects
* Trusted Computing Base (TCB)
  + Responsible for enforcing security
  + Poor software implementation/config = compromise security
* Reference monitor (entity mediates access requests by subjects)
* Security kernel (hardware/firmware/software of TCB implementing Reference monitor)
  + Mediate all access
  + Protected from modification
  + Verifiable as correct

**Authorization (access control)**

* Relies on Authentication
* Decision = **security context** of the process
  + Inherited from user (that initiated it)
  + Sec of User = which user group

**Access Control**

* Process that controls interaction between user and resources
* Implemented security policy based on
  + Organizational requirements
  + Statutory requirements (PII)
* Access control policy includes
  + Confidentiality (restrictions on read)
  + Integrity (restrictions on write)

**Why access control?**

* **Prevent** users from having unlimited resources
* **Limit** access of **unauthorized**

**Reference monitor**

* Establishes validility of request
* Returns decision of granting/denying access



**Access Modes**

* Accessing object = flow of information
* Write = subject to object
* Read = object to subject
* Execute
  + Execute program
  + Execute access to unix directory
    - Read = list directory

**Access Rights**

* Way of accessing an object
* Interpretation of access rights depends on
  + Operating system
    - Write = R/W
    - Append = W only
  + Object type
    - Win treats everything as object

**Access control policies**

* **Discretionary** policies are based on identities
* **Mandatory** policies independent of identity

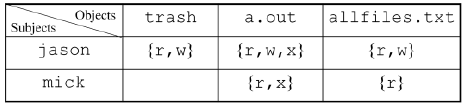
**Delegation**

* **Perform access** of another process
* Delegation of access rights (win=impersonation)

**Access control models (Ref Monitor)**

* Includes elements used to represent the systems, such as sets, relations and functions (entities, relations between entities and operations that can be performed)
* Deduce formal results about security of a sys
* Rules that help with implementation

**Access control matrix**



Subject S wants to access object O using access right A

* Represented as (S, O, A)
* Granted if A belongs to access matrix corresponding to subject S and object O

**Disadvantages (matrix)**

* Not suitable for direct implementation
  + Matrix sparse, implementation inefficient (Empty cells)
* Management of matrix = difficult (too large)

**Access Control List**

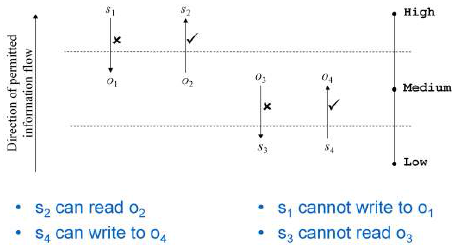
* Column in a matrix
* Each entry includes a userid and access mask
* Bit pattern where each bit represent accessright
* 111(R,W,X) then 100(R)
* Focused on objects
* Cannot check based on subject

**Capability list**

* Row in access control matrix
* Focus on subjects
  + Services, and app software
* Object identifiers and access masks
* Cannot check based on object

**Information flow policy**

* Enforces confidentiality requirements
* Every object and subject = security label)
* Can access if a<= b



HML = security clearance

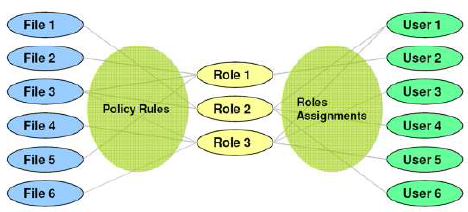
Prevents:

* Information leak due to inappropriate reads
* Information leak due to inappropriate writes

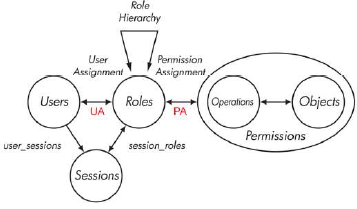
**Role-Based Access Control (RBAC)**

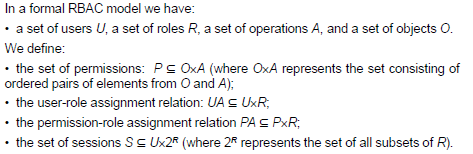
Core idea = Roles (priv management)

* Acts as bridge between users and objects
* Reduces complexity of configuring authorization



* Users associated with >1 roles
* Permission associated with >1 roles
* User activates a session (security context) by selecting 1>roles associated with user
* An access request is granted if >1 roles associated with user has permission





**Hierarchical RBAC**

Help further reduce administrative burden

* Match organizational structure
* More senior, inherit rights of others☺

User assigned to a role r (via UA) is assigned to all roles (lower level)

Permission assigned to a role r (via PA) is assigned to all roles (higher level)

**Constrained RBAC**

Separation of Duty

Static

* Finance and HR cannot be assigned to the same user

Dynamic

* Check conflicts roles upon request

**Administration (of RBAC)**

* Assign user/permission to role
* Add a role
* Add an edge from role hierarchy

2 possible approaches

* Assign special admin permission to admin roles
  + Issues may arise
* Use of role hierarchy to limit power of admin roles

RBAC = widely accepted as best practice

**Unix Security mechanism**

Root = UID 0

* /etc/passwd = anyone can read, no password
* /etc/shadow = password

Process spawned by shell = associated with user ID

**Unix Permissions**

RWX

Owner Group Others

111 101 101

**Unix Files**

File = owner and group (owner need not be member of group)

Device = treated as files

**Windows Security Mechanism**

Winlogon = authentication

Success = access token returned.

Access token bounded to authenticated user

Windows Authorization

When accessing file  
 -> access token given to file system service

-> file sys service forward req, access token, file ACL to security reference monitor (SRM

-> SRM compares identities

**Access masks**

Everything kept in 32-bit access mask

SRM constructs granted access mask  
if same = access is granted.

**Access control Entries**

Every file has security descriptor

Most important = Discretionary access control list DACL

DACL = list of access control entries (ACEs)

Each ACE = security identifier and an access mask

SRM examines ACE

If access token contains an SID that match the ACE SID :

* Matching entries in the requested AM and the ACE AM are added to the granted AM
* Updating the granted AM involves performing logical OR with (requested AM AND ACE AM)

Requested and Granted are compared

**CS1 : EMV (**Europay, Mastercard , Visa)

* ‘chip and PIN’

**Terminology**

* Issuing Bank = issues the card
* Acquiring Bank = process card transaction

**Authorization**

= exchange of msg between merchant terminal & issuer

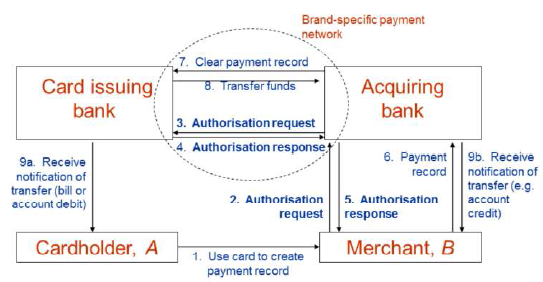
Payment model = pull model

Decision (online authorization) = based on floor limit

**Card Brand**

Each card brand operates a payment network

Acquiring & issuing banks use this network



**MOTO/Card not present (Mail Order, Tele Order)**

* Via phone/online/Mail

**Chargeback**

* Cardholder may dispute a transaction
* Merchant to prove transaction (else reversed)
* Major problem for MOTO
  + Use online authorization to protect

**Primary threats**

* Unauthorized card use
* Use of fake cards
* Transaction repudiation
* Eavesdropping of transactions
* Manipulation of transactions
* Loss of card number confidentiality
* Manipulation of data transfers

**Countermeasures**

* Use of handwritten signatures
* CVC
* Online authorization
* Use of PIN

**Countermeasures (cloning) + above**

* Use of holograms
* Card-based cryptogram generation &   
  card authentication

**Countermeasures (transaction repudiation)**

* use of handwritten signatures
* use of PIN

**Countermeasures (eavesdropping/manipulation)**

* cryptographic protection

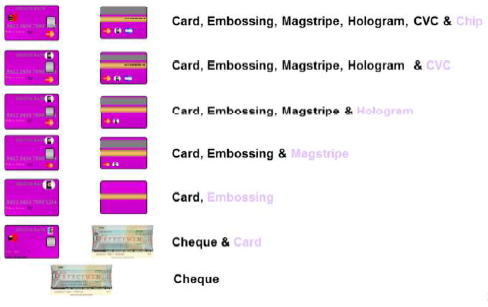
(not major, difficult to eavesdrop)

**PKIs**

* Use to distribute public key
* Use of CA
* **Open and closed PKIs**
  + Closed = single policy agreed by all
  + Open = general use

**EMV- migration to chip cards**

* Motivation = increasing crime levels



* Cost for online authorization (telecomm)
  + Risk VS cost
* Business case (fraud, processing savings)
  + Reduce fraud cost
  + Reduce/maintain online issuer authorization level
* Increased security using
  + Card authentication method (CAM)
  + Cardholder verification method (CVM)
  + Selective/offline authorizations

**EMV - development of EMV**

* EMV joint to create industry standard to minimize merchant cost

**EMV - EMV specifications**

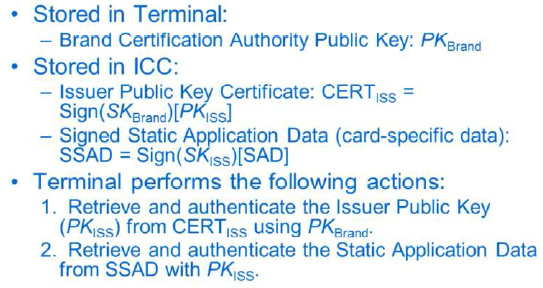
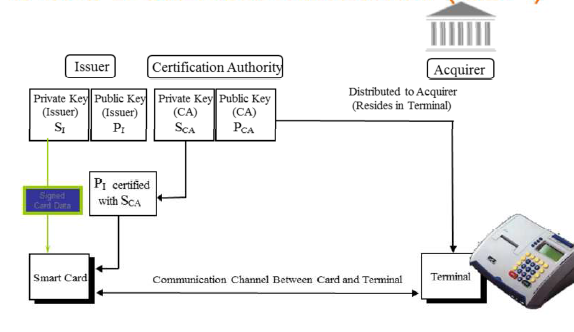
ISO/IEC 7816 (international standards)

1. Application independent ICC to Terminal Interface Requirements
2. Security and key Management
3. Application Specification
4. Cardholder, attendant and Acquirer Interface

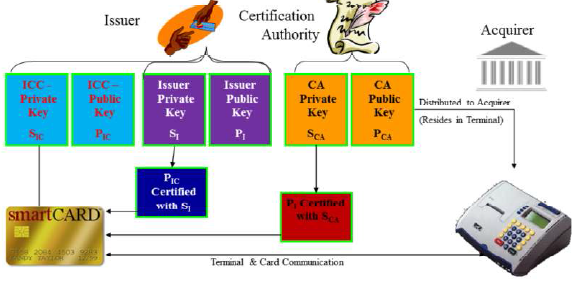
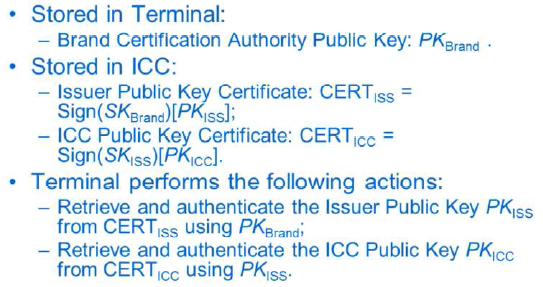
**Offline Data Authentication** (EXAM!!!)

* STATIC = card specific data is pre-generated by issuer and stored in chip at personalization time (still vuln against copy/replay)
* DYNAMIC – Authentication provided during transaction, function of card data and challenge (data from terminal)

**SDA (based on Digital Signature to verify static data)**



**DYNAMIC**

Prevents counterfeiting of the ICC 

**CA heirachy**

Static = 2 layers (brand, issuers)

Dynamic = 3 layers (brand, issuers, cards)

**Digital Signature Scheme**

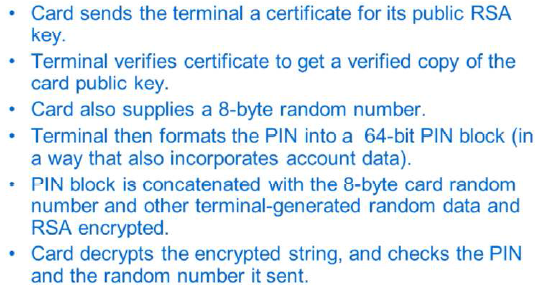
RSA and SHA-1

message recovery (cos u get the public key)

**CDA (Combined Data Authentication) <not so impt>**

* prevents certain ‘wedge’ attacks

**EMV - EMV Cardholder Verification**

* PIN number
* Online PIN verification (ATM)
* Offline PIN verification (encryption)

**EMV – EMV Risk Management and Cryptograms**

* Risk is performed by both card and terminal (either request = go online)
* Terminal = random
* Card = after max of offline amount

**Cryptograms**

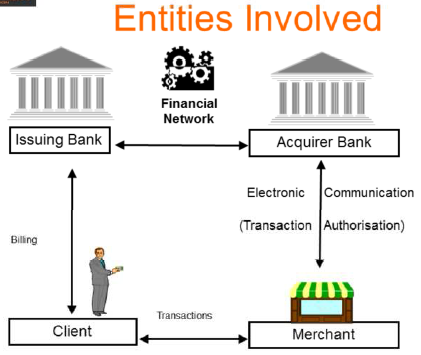
* MACs computed on messages (Protect integrity)
* Using secret keys between card and issuers
* Protect online authorization and trans records

**EMV – Residual Vuln (visited later)**

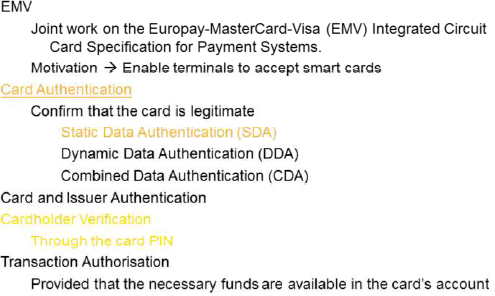
**Other Vulnerabilities**

Card Skimming





**Summary**

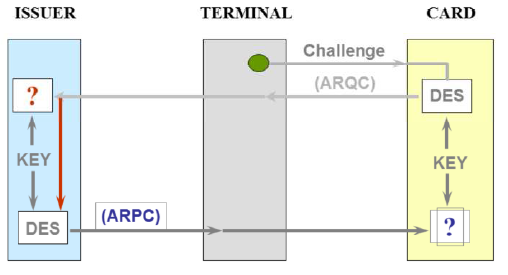


**SDA Vulnerabilities**

Weakness = card cloning

BUT Card authenticate (using PIN)

BUTT = got “YES” cards (accept any pins)

BUTTT discoverable if online or use DDA/CDA cards

**Bitcoin**

* 1st decentralized digital cash
* Works on P2P
* Own value based on supply and demand
* Makes anonymous payment

**Started**

* Open source project
* Activated in Jan 2009

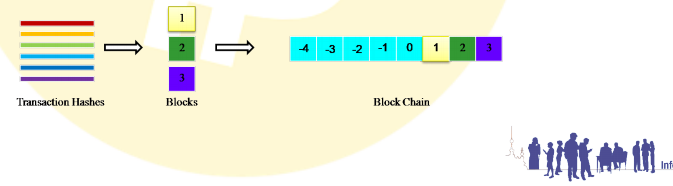
**Address**

* Unique Bitcoin address using Elliptic Curve Dig Signature
* To receive: pass the public key over
* To send: put inside using private key

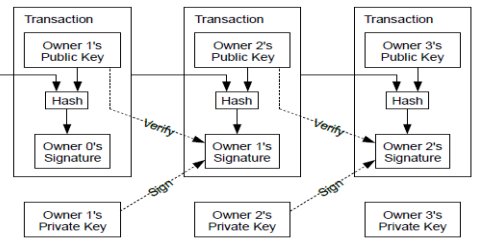
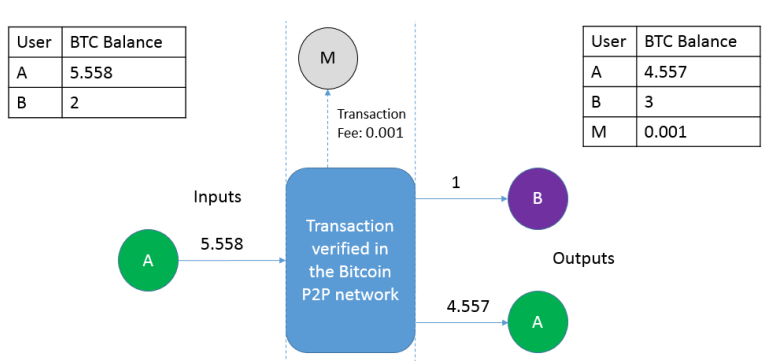


**Block Chain**

* Public available ledger (stores all transactions)
* All transactions are put through hash functions
* Single block contains a no. of successful bitcoin transaction hashes
* Records permanently the transaction history
* Security based on correctness of block chain



**Transaction**

**Block Chain =** Detects double spending

**Miners**

Authorizing transaction and create new blocks

CPU to generate random hashes that matches a target

ASIC miners = type of dedicated hardware used for hashing and very popular power.

* Rewarded bitcoins

**Proof of work**

Requires a user to exhaust some computational power and time before task is completed

Generate hashes until a hash value is generated with a certain number of zero bits infront of hash

Work = increases exponentially (more zeros)

**Transaction Authorization**

Process of verifying and not an attempt to double spend

Comparing hashes against transaction history

Verification assigned randomly to miners

After checking, transaction is hashed and added to block

**Creation of new blocks**

* Coinbase (pays 25 Bitcoins)
* Rewards reduces every 4 years
* Reward ~ 4000 pounds
* Meets ‘proof of work’

**Advantages and Disadvantages**

1. Secure your wallet (strong password, backup wallet)
2. Bitcoin are irreversible
3. Currency fluctuates
4. Transaction are publicly broadcast

**Bitcoin Wallet**

* Creating a wallet (bitcoin-qt)
* Need to download the whole block chain of (45 GB)

**To get bitcoins**

* Request bitcoin as payment
* Bitcoin exchange to buy bitcoins
* Buy individually
* Mining (totally anonymous)

**Make Payment**

* enter bitcoin address
* enter amount
* send

**Receive Payment**

* generate new key
* give to the party

**Simple miner**

* joining the mining pool
* (slush’s pool), (GUI miner)

**Good vs Bad**

* Tor browser
* Silkroad

**Other issues**

* Bitcoin Hoarding
* Transfers to wrong address accidentally where no corresponding key-pair
  + Stucked in account forever
  + If removed -> deflates value
* If bitcoin private keys are lost, become unspendable forever
* Consumer could not request a refund

**Proposed solution**

* Research papers

**Operating Systems Security**

* Intermediary between user software and hardware
* Abstraction layer providing idealized view of hardware
* Set of services to assist user and applications

Common Functions

* User Interface
* Memory, peripheral, CPU, I/O
* **Security**

Needs to control:

* Sharing
* Provide an interface to all the access
  + Identification & authentication for access control

Communication security

* app data sharing
* communication between apps
* communication between apps and storage
* Encrypted comms with external entities

Access Control

* Identification, authentication, authorization
* Declaration of policies above

Privacy

* Strong separation between app
  + Prevent personal data leakage

Sandboxing

* Programs can execute cannot affect rest of machine
* Apps= own set of resources
* App = OS executing one app only
* Isolation = easy to conceptualize, hard to implement

Whole Disk Encryption

* Allow encryption to whole disk/ user data
* Avoid data being accessed by unauthorized
* Avoid data loss if theft of physical tampering

**OS Security**

* Ensure OS CIA

**Potential Exploits**

* Unprotected system access resources illegitimately
* Attack via vulnerable app
* Privilege escalation
* Convert channels (transfer high security to low)
* App read other app memory address/data
* Virus/worm/Trojans

Higher level security useless if no OS security

**Requirements of Secure OS**

**Mandatory Security**

Built in mechanism within OS (implementing policy)

* Policy independent security labeling and decision making logics
* Enforcement of access control for operations
* Implements controls like permission, access authorization, authentication, crypto

**Trusted Path**

Trustworthy relationship between user and app software- with the OS

* User/app interact with trusted software
* Mutually authenticated channel (prevent impersonation)
* Extensible (addition of trusted service/app)

**Support Diverse Security Policies**

* Separation of security policy logic from policy - enforcement mechanism
* Support for Policy definition/changes
* Provide default security behaviour

**Assurance**

* Process/methodology to verify design & implementation of sys (from what it claims)

**Multics and Hydra (secure)**

SELinux (NSA Recommendations)

**Certifying OS**

**Orange Book**

* Set of secure systems levels D to A1

Common Criteria

* Everyone merged into 1
* Protection Profiles
  + Security feature to achieve
* Security Feature
  + How it achieve the security goal
* Assurance level
  + Confidence level in implementation and effectiveness of security feature

Analysis of OS Security

* Formal requirement analysis
* Formal risk analysis
* Formal OS security analysis
* Formal specification of how security countermeasures implemented
* Practical testing each of implemented countermeasures
* All required to have certifiable confidence

Bad Model, no sales

* Certification process = too long, too exp
* Certification is static, security landscape changing
* No commercial value

**OS – another software**

1000 lines, 7 bugs

Large piece of software

Vuln too complicated to find, will be found

**Patch**

* Necessary vulnerabilities
* Review patch doc, issues with patch
* Use a patch staging environment
* Put in place required resources to minimize downtime, and any other risk
* Define, document

**OS Hardening**

Limit damage of internal and external attack

**Disable non-essential services**

* Services = potential route of attack
* Understand any interdependencies between services

**Update and patch**

* Delay = attackers time to exploit vulnerability
* Patching not in conflict with business servers

**File Directory protection/encryption**

* Access control, encryption

**Logging**

* Track attacker’s activities
* Assist in recovery
* Gather required for legal proceedings
* Alerts to suspicious activities

**System hardware**

* Unused ports/drivers

**Logging**

* Security mechanism can fail
* Logging on essential services
* Too much = storage, privacy
  + Actual attack = might be rare
  + Analysis fatigue
* Predefined sig required (must detect first)
* Logging =/= real time (days after attack.
* Regular analysis, and trimming
* Protect log files
* Processing log files

**SELINUX  
implementation of modifications to Linux kernel that deploys MAC using Linux Security Modules (LSM) framework**

Discretionary Access Control (DAC)

* DAC WAS only security Framework for linux
* Based on user’s identity and ownership
* No protection against malicious/flawed software
* System/priv services must run with coarse-grained priv, or full root access
* 2 major categories: Admin and others

MAC

* Strong separation of security domains
  + Based on CIA
* System/app/data integrity
* Ability to limit program privileges
  + Can run uncertain code
  + Prevent priv escalation

Why Secure OS?

* Increasing risk to valuable info
* Attack Don’t require a corrupt user
* Application can be circumvented
* Process in clear

Mandatory Access Control (MAC)

* Administrative set security policy
* Control over all process and objects
* Decisions = security-relevant info

SELinux (Flexible MAC)

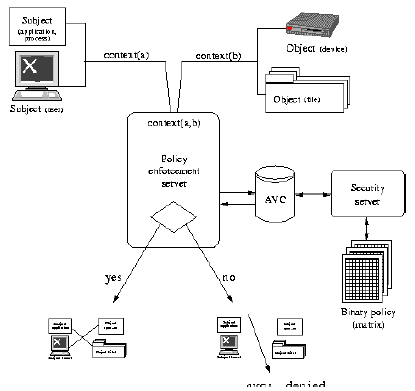
* RBAC
* Type enforcement (similar type then can communicate)
* Optional multi-level security
* Highly configurable

Linux Security Module LSM framework

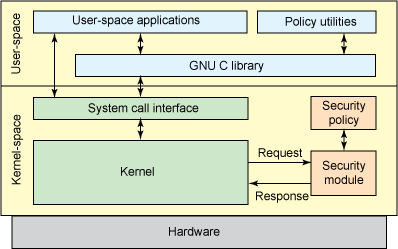
* Linux does its normal security check
* Kernel needs to decide if access granted, asks a security module if action is ok too
* Administrator picks the policy

Flux Advanced Security Kernel (FLASK)

* Supports flexible security policy (syntax)
* Separates policies from enforcement
* Using more info when making access control decisions



1. Subject attempts to perform ops on object
2. Gathers security ccontext of both > sends to security server (policy decision maker)
3. Checks Access Vector Cache (AVC) and returns decision  
   Goes to security server if cache no have  
   checks binary policy  
   returns to enforcement server
4. Policy permits/deny the operations (log if necessary)



**How SELinux Works?**

* Labeling for everything

User:role:type:level(optional)

* Type enforcement

Specifies the part of the policy where which context can access which context

Security Advantage

* Policy + Enforcement
* Very fine-grained AC
* Tiny violations can be monitored
* Management by policy, not user actions
* Minimizes dmg user mistakes can cause
* Forces users to follow policy

Others (AppArmmor)

* File-system agnostic

Others (TrustedBSD)

* Build a trusted OS extensions
* Basic Security Module (BSM) audit implementation
* Audit API (file format that supports audit trail)

Operating System Virtualization

* OS hosts multiple isolated user-space instances
* Restricts the capabilities of an application
* Nothing can modify kernel parameters or other user instances
* Linux VServer, OpennVZ, Solaris, BSD

**Network Security**

Security Threat = security policy may be breached

* Person/thing/event/idea poses danger to assest (CIA)

Security service = measure can put in place (address a threat)

* Safeguard
  + Communications security safeguard
  + Physical
  + Personnel
  + Administrative
  + Media
  + Emanations (radio freq)
  + Life cycle controls

Security mechanism = means to provide a service

Attack = realization of threat

Vulnerabilities = weakness in safeguard

**4 Fundamental Threats (Based on CIA + legitimate)**

\*Deliberate/Accidental

* Information leakage
* Integrity violation
* DOS
* Illegitimate use

**5 Attacks (enabling threat)**

Penetration Threats

* Masquerade
* Bypassing controls
* Authorization violation

Plantings Threats

* Trojan horse
* Trap door

**Security Service Classification (ISO7498-2)**

* Authentication
  + Entity authentication  
    checking of claimed identity at a point of time  
    address masquerade and replay
  + Origin authentication  
    verification of source (not against replay)
* Access Control

Unauthorized use of resources

* Data confidentiality  
  unauthorized disclosure of info
  + Connection
  + Connectionless
  + Selective field
  + Traffic flow
* Data integrity  
  threats to validity of data
  + Connection integrity without recovery
  + Connection integrity with recovery
  + Selective field connection integrity
  + Connectionless integrity
  + Selective field connectionless integrity
* Non-repudiation  
  denying that data was sent

**Security mechanism**

* 2 classes
  + Specific security
  + Pervasive security (not specific to provision of individual security service)

Encryption mechanisms

* data and traffic flow confidentiality
* authentication key exchange

Digital signature mechanisms

* signing/verification
* non-repudiation, origin authentication, data integrity
* authentication key exchange

Access control mechanisms

* using info associated with client and server who gets access
* ACL, capabilities, security labels

Data integrity mechanisms

* Against modification of data
* Data integrity, origin authentication
* Authentication key exchange
* MAC, single data unit
* complete data seq (replay, selective deletion, reorder)

Authentication exchange

* provide entity authentication service
* authentication protocols

Traffic padding mechanisms

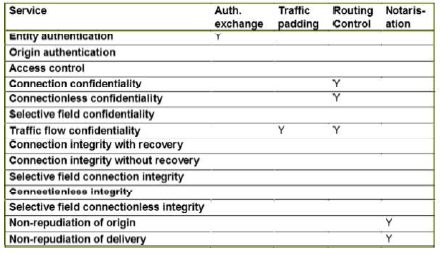
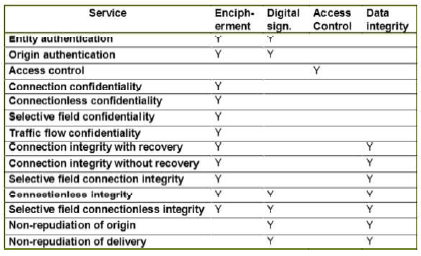
* conceal real volumes
* Traffic flow confidentiality
* Only effective with other mechanisms

Routing control mechanisms

* Prevent sensitive data using insecure channels
* Security services, confidentiality and integrity

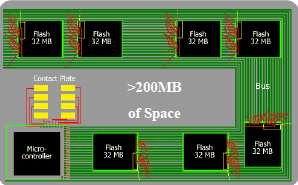
Notarisation mechanism

* Integrity, origin, destination of data (guaranteed by 3rd party)
* Notary apply crypto transformation
* Non-repudiation



**Smart Card attacks**

Contact Card (Gold contacts)

Contactless Card

Memory not an issue

**Evolution**

* Separate of OS and application

Goal = Security is absence of risk.

Smart Card = Control of risk

Smart Card CIA (Aim to Prevent, Detect, Recover)

* Hardware tamper resistance
* OS logical security
* Organizational and overall system security

Smart Attack Points

* Communication
* Processing
* Data Structures

Major Smart Card Attacks

* Social Attacks
* Hardware Attacks

Hardware (silicon)

* Decapsulation (chemical to dissolve plastic)
* Passivation layer (prevents oxidation)
* Protective layer (no metal shield, cant access) But can drill holes
* Card destroyed, chip not destroyed
* No global secrets -> no big threats
* Expensive equipment

Hardware (smart card probing)

* Monitor data buses (keys, sensitive data, PIN, app)
* Obtain the complete running program
* Physical access to microprocessor
* Countermeasures
  + Active shield (chip don’t operate upon modification)
  + Obfuscated ogic and buried buses in multiple layers
  + Encrypted buses
  + Individual chip line scrambling (too expensive)
* Very expensive and difficult to attack

Hardware (Side Channel)

* Needs power and clock rate
* Fault Attacks
  + Light, voltage, clock. Temperature, …
  + Simplified (15 grams of coin vault)
* Glitch Attacks
  + High/Low frequency induce errors in processing
* Power Analysis
  + Looks at the power consumption
  + Threat = what is happening in microprocessor
  + Power consumption differs from diff data
  + RSA use 0 and 1 (1 must square and multiply)
  + Power trace of crypto algo

E-passports

* ICAO 9303 RFID
* Basic Access Control (BAC)
  + Machine readable zone (MRZ) printed on the card
* Access the passport through 128 bit key
  + Entropy smaller (used personal details as keys)
  + Countermeasures
    - Random passport numbers
    - Seed through MRZ optional field
    - Faraday cages
    - Larger entropy

Transport

NXP, MiFare classic (CRYPTO 1)

* Karsten Nohl, 2007
* Security through obscurity

Countermeasures

* Kerckhoffs’ Principle
* Peer reviews and open systems
* Security evaluations

Satellite TV

Protection

* Smart Card receives encrypted messages containing the Keys
* Decrypted by smart card and keys are delivered to STB for signal decryption

SAT TV Counter Attack

* “Black Friday” or “Black Sunday”
* Transmit a counter attack code
* Target hacked card
* Kill command
* SUCCESS -> Hacked cards were looped

Hackers

* Loader/Unlooper
* Introduce glitches to
  + Overcome endless loops
  + Make cards re-programmable

Cardsharing attack

* One legitimate user allow n-number of bad users to provide unauthorized access

Game Console Security

* Floppy disk (purposely have bad sectors)
* Catridges (initially, difficult to copy)
* Playstation (mod a chip on motherboard)
* Xbox cracked
  + Chip in motherboard
* Xbox 360
  + Designed to be piracy proof
  + Legitimate DVDs with Bad sectors
  + Rest Glitch Hack

SIM Card Root Attack

* DES on most card
* “Send a msg”, “error msg” < signed with DES
* Use rainbow table to get DES key
* Use SMS filtering, better algo

Host Card Emulation (HCE)

NFC Skimming/Cloning

BOTNET Threat

Responsible for

* Large-scale network
* Launching DDOS
* SPAM
* Click fraud
* Information theft
* For fun -> Profit

Torpig

Trojan Horse:

* Distributed via Mebroot
* Injects itself into 29 apps as DLL
* Steals sensitive info (passwords)
* HTTP injection for phishing
* Use “encrypted” HTTP as C&C protocol
* Domain flux for C&C server

Mebroot

* Spreads via drive-by downloads
* Sophiscated rootkit (overwrites MBR)

Domain Flux (BotNet Resiliency)

* Taking down a single bot has little effect on master
* C&C servers vulnerable to take down
  + Static IP= ppl will block/remove
  + DNS = ppl will block/remove
* Domain flux
  + idea = periodically generate new C&C domains

TorPig, Domain Flux

* Same DGA
* 3 fixed domains, if all else fails
* Weekly (WD), daily (DD)
* Every 20 minutes bot attempts to connect
  + Wd.com, wd.net, wd.biz, fail = dd.com, dd.net, dd.biz, fail 3 fixed
* Wd.com/wd.net (mostly)

Sinkholing Trojan Horse

* Reverse engineering DGA
* Check if domains unregistered (then registered)
* Controlled Botnet for 10 days (new binary on 02/04)
* Data = 8.7GB apache logs, 69GB pcap
* Purchased 2 hosting (don’t care complain)
  + 1 suspended
* Setup apache web servers to receive bot request
* Recorded all network traffic
* Automatically downloaded and removed data from our hosting providers
* Enabled hosts a week early, immediately received data from 359 infected machines.

Data Collection Principle

1. Minimize the damage
   1. Always with okn message
   2. Never sent new/blank config file
2. Collect enough info to enable notification & remediation of affected parties
   1. Worked with law (FBI & DoD cybercrime units)
   2. Worked with bank security officers
   3. Worked with ISPs

Software Vulnerabilities

* No more format strings
* Stack -> heap smashing
* Heap is difficult to exploit
* Exploitation is harder

Drop in reports

* 0 day private market
* Bounty programs
* Black market

Penetrate and Patch

* Software vuln = taken by major software vendor
* Not designed with security perspective
* Patching – not used routinely, but last resort
* Considered at design stage, no need for freq patches
* Problem
  + Vulnerabilities must be found
  + Fast written patches = new vuln
  + No guarantee, users will install patches
  + Lazy approach (users test, then release)
  + Damage to reputation
  + (still….) need routine patching (applies to open and closed source)

Open Source security implications

* Code open to be abused (those looking for vuln)
* May not be exposed to experts in security
* Development may be piecemeal

Closed source security implications

* Hidden = harder, but obscurity is a questionable security principle
* No guarantee of good design
* May be possible to reverse engineer code

Principles of Secure software

Outset

* Penetrate and patch = security not a high priority in design
* Best way = design from the outset
* Security considerations integral to SE lifecycle

Least Exposure

* Ensure is not exposed to unnecessary risks
* Isolate code for security critical ops in separate modules/library, easy to analyze and control
* Ensure principle of least privilege (min access necessary to perform an ops)
* Minimize the possible attack surface (turn off unnecessary functionality and services)

Fail securely

* World readable core dump = expose security critical information
* Firewall continues to operate when log disk = full = failure to detect important security critical event

Secure software by simplicity

* Design perspective (easy to analyze and test)
* Usability (appear simple, user wont ignore/avoid)
  + Wont use if over-designed from security

Buffer Overflow

* Buffer = Set of contiguous mem location
* Overflow = store more elements that can contain
* Detects anomalous situation, blocks operation
* Cannot block anomaly = operation executed

Why?

* Languages = lack of proper boundary checks
* Channeling problem, data and channel share same channel
  + Channel = stack, heap
  + Data
  + Control elements (pointers)
* One of the Most popular attack
  + OS dependent
  + Exploited locally, remotely
  + Modify data and execution

Family of Attacks

* Stack-based overflows
  + Shellcode injection
  + Return into libc
  + Return-oriented programming (ROP)
* Heap overflows
* Integer overflows
* User-controlled format strings

Stack = grow towards lower memory address

ESP = top of the stack (lowest valid address)

Stack = composed of frames

Frames pushed (cos of function calls) PROLOGUE

Address of current frame = Frame Pointer (FP) register %EBP

Each frame = actual parameter, return address, pointer to previous frame, local variables

Process Structure

* Environment Section (environment, command line data)
* Stack section (local params, saving processor status)
* Heap section (dynamically allocated data)
* Data Section
  + Initialized variable (.data)
  + Uninitialized variable (.bss)
  + Code/Text Section (.text) usually RO

Stack Overflow

* Data Copied without checking boundaries
* Overflows ‘preallocated’ buffer and overwrites RET address
* Causes a segmentation fault
* Crafted correctly = overwrite RET to user-defined value
* Jump to user-defined code
  + Shell Code, Shell code address
  + NOP Sled
* Code = part of overflowing data
* Executed with the priv of running program

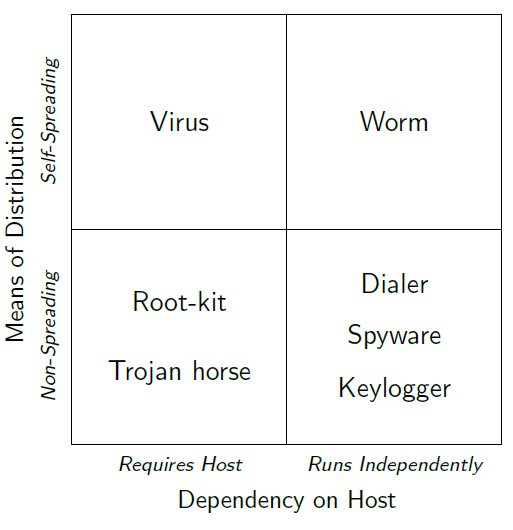
Overflowing Functions

* Gets()
* Strcpy(), strcat()
* Sprint(), vpsrintf(), scanf(), sscanf(), fscanf()

Stack Guard

* Writes a canary value before RET on the stack
  + Terminator NULL, CR, LF, EOF
  + Random canary
  + XOR canary: XOR of RET
* Verify value before RET
* Modified function PROLOGUE/EPILOGUE
* Overhead, RET not only the interesting point

Malware



Static/Dynamic Analysis

Extract/generalize malicious behavior (host/network)

Generate and deploy detection models

* Lack of definition of Malicious behavior
* Cat and mouse, attackers = too much freedom
* Victims (unwittingly) = help attackers

AV

* No big effort to collect samples
* Reverse engineer
  + Byte/instruction level sign
  + Regex
* Heuristics
  + Code execution starts in last section
  + Incorrect header size
  + Patched import table
* Signature based detection
  + Hash

Nowadays

* Reputation
* String signatures
* Suspicious behavior
* Malware prevalently obfuscated (polymorph, metamorph, packed)
* Hard for sig based detection
* Focused on dynamic behavior
  + Behavioral taint-enhanced sub-graphs isomorphism
  + Machine learning

**Authentication and Key establishment Protocol**

Protocol = set of rules for exchanging **messages** between 2 **principals**

* Message format
* Rules for handling
* Security protocol can be at diff TCP/IP layer

**Security Protocol**

2 properties

* Acts honestly, achieve aim of protocol by authenticating user
* Neither passive eavesdropper or active adversary can defeat the objective

Alice and Bob = authenticate one another, share key

Trent = trusted 3rd party (e.g. CA)

Eve = passive eavesdropper

Mallory = active adversary

Give them idealized crypto mechanism, force them to use on untrusted network

Authentication

* Origin or entity authentication
* Exchange of crypto message = authentication p

Entity authentication = entity is as claimed

Unilateral authentication = entity authentication giving one entity assurance of the other’s identity, but not vice versa

Mutual authentication = entity authentication providing both entities with assurance of each other’s identity

Basis = something you know/have/are

Weak = passwords and pins

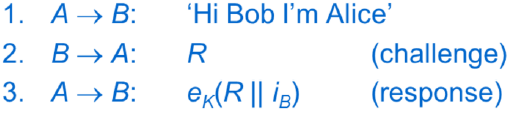
* Make it stronger by
  + 1 way encrypted pass file
  + Slow down encryption
  + Add salt
  + Access control on pass file
  + Lock after failed attempts
  + OTP

Strong = proves its identity by showing knowledge of a secret, without revealing secret AKA challenge-resposne

Encryption-based unilateral authentication

Alice and Bob share secret K, and Bob needs to authenticate Alice

* Alice sends initiating message
* Bob sends CHALLENGE
* Alice sends RESP with identifier B (encrypted with secret key K)
* Bob decrypts Alice Key and checks RESP



Eve sees CHALLENGE and RESP

Mallory can impersonate Bob

Unilateral authentication of Alice to Bob

**Replay Attack**

Challenge R must be unpredictable, or else Mallory can masquerade as Alice



**Freshness/Liveness**

Freshness: assurance that message has not been used previously, and originated within an acceptably recent timeframe

Liveness: assurance that message sent by a principal within an acceptably recent timeframe

**Nonce**

Nonce = Number used once, random challenge

“one-time” property, nonce needs to be unpredictable(Random from large set)

**Time Stamps**

To check for freshness

Less messages in protocol

Securely synchronized clock = non-trivial

* Clock drift
* Window of acceptance = (clock drift + variable propagation time)
* Log recent messages to prevent Mallory exploiting window with replay attack

**Logical time stamp**

Provide alternative to clock

Alice and Bob use pair of sequence num in their comms

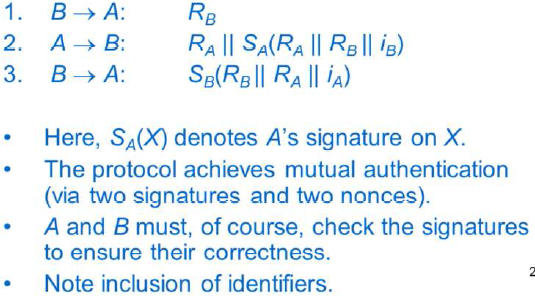
Include current value and increments it

Needs a pair of secret seq num for communicating parties

**Signature-based entity authentication**

Challenge/Signature

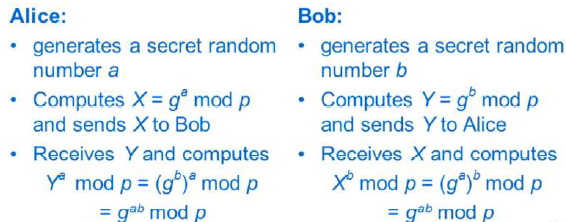
Nonce/ time-stamp for freshness

Instead of shared key, Bob has authenticated version of Alice’s signature verification Key (vice versa for mutual)

**Key Agreement**

Session key = by-product of authentication protocol

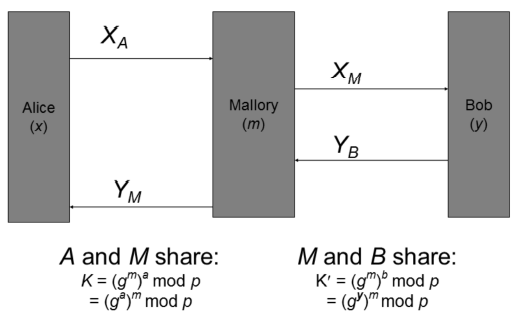
**Diffie-Hellmann Key Agreement**



Eva can see public values X and Y, but hard to compute key from values

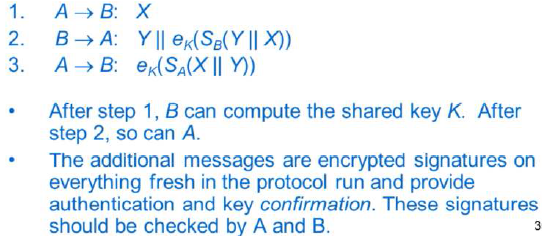
Secure against passive adversaries

**Man in the middle attack**



**Station-to-Station Protocol**

Adds authentication (via signatures) to DHKey



**Kerberos**

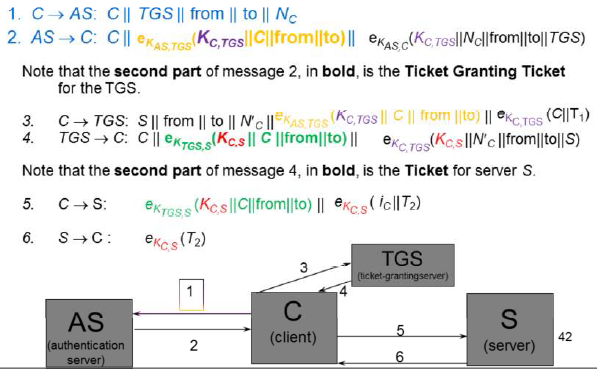
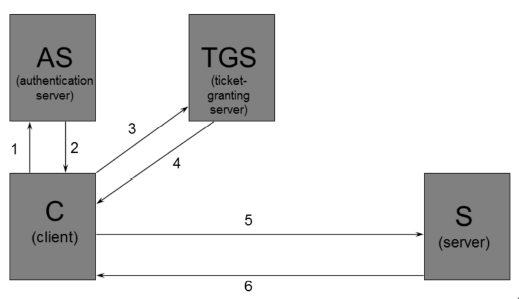
TTP-aided authentication protocol

Authentication Server (AS)

Ticket Granting Server (TGS)

Motivation

* Long term secret key into host for min time
* Short term key can be erased from host
* Minimizes risk of exposure of the long term secret key



Kerberos uses symmetric encryption and MDC

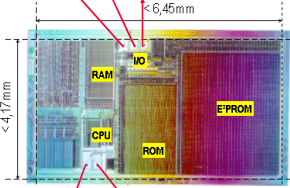
Issues:

* Revocation: TGT valid till expired (10 hours)
* Long term keys needs to be established between AS and TGS , TGS and server and AS and client
* Requires synchronous clocks, must be protected against attack
* Cache of recent messages
* Need AS and TGS = online (no eavesdropping)
* Client-AS long term key = (vulnerable)
* Short term key, TGT = unprotected
* DoS

Windows network authentication relies on the extended version of Kerberos

OTHERS NOT IN EXAM!!

**Smart Card**

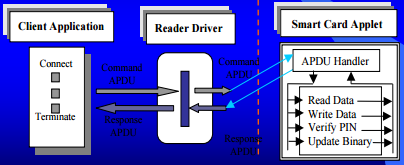


ROM = OS

EEPROM = App Data and OS extensions   
(1000x slower than RAM)

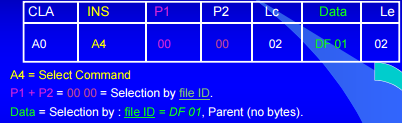
RAM = OS Workspace7

ISO7816 – standards

App written in C,C++,VB,Java (easy to program)

SC File System (Hierarchical)

APDUs



Response 9000 = success, 6A82 = File not found

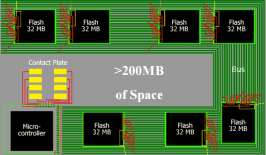
SCOS (Smart Card Operating Sys)

Many claimed multi-app support (but was not true)

* App struct agreed in advance
* Had to be installed in advance
* App designed for specific SC micro-processor
* All functionality embedded in SCOS

Separation of OS and app

Standard language of app dev



Java Card

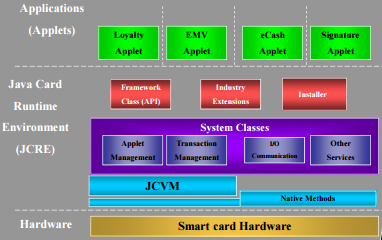
Why Java ? Standard language, OOP, security, sandbox, portability, and interoperability

Java card = smart card, capable of running Java program

Java card language = subset of Java Language

Java card VM = subset of JVM

Java card API = little resemblance to traditional Java API



V2

Javacard.framework -> APDU, Sys, and Util

Javacardx.framework -> ISO7816-4 File Sys

Javacardx.crypto -> export-controlled crypto functionality

Javacardx.cryptoEnc -> Basic crypto functionality

Portability and interoperability = issue

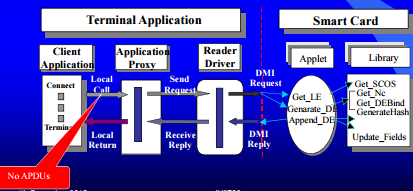
Class file conversion/download = vendor specific

V2.1

* Firewall and object sharing model redesigned
* Install method = interoperable
* New Exception functionality
* Multiple instances of single applet class possible
* AID = more general purpose
* Javacardx.framework extension deleted (can use internal objects to represent)
* Crypto extension packages restructured
  + Got security and crypto

V2.2.1

* Support for additional platforms
* Crypto extanded
* Object deletion mechanism
* Logical channels support
* More functionality (RMI)

Remote Method Invocation (RMI) - abstraction

Lifetime of Java Card

* Starts when the SCOS

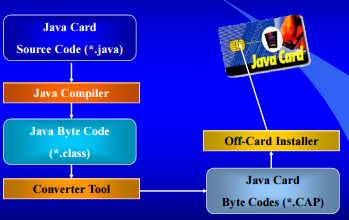
Lifetime of JCVM

* Runs forever unlike JVM

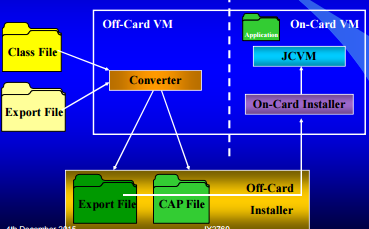
Lifetime of JC applets and objects

* When installed and registered in registry
* Ends when its removed from registry
* Space may reused (due to GC)
* Objects stored in EEPROM

Java Card Security

* Security policy = enforced in JCVM
* Objects = owned by applet that created them
* Objects = shared with other applets using sharing methods
* Obtained EAL4+, EAL5+, Common Criteria
* Java card protection profile (set of security requirements)
* 

Architecture of JCVM (off/on card)



Summary of Benefits of Java 2.X.X

* Interoperable
* Secure
* Multi-Application scope
* Dynamic
* Open
* Compatible with existing

Java 3

* Silicon better
* Software better
* Smart cards as networked components
* New Apps (run in parallel, web)
* Multiple interface (nfc, contactless, usb)
* Smart card programming (close to Java)

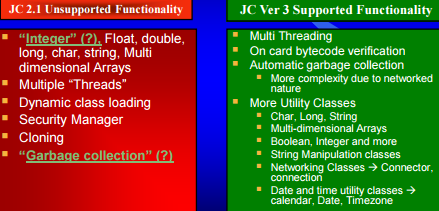
Aim: Retain (compatibility, interoperability, security, scalability)

2 types:

* Classic
  + Traditional JVM (off/on card)
  + Better crypto
* Connected (high-end, networked, enhanced JCVM)
  + Web=server
  + Multi-threading
  + Efficient and smaller
  + Load java class file without preprocessing
    - On card byte code verification
  + Auto GC

Both share security features





Web Application: Used as protocol for easy communication (via HTTP/HTTPS)



Typical Classical computer security threats apply.

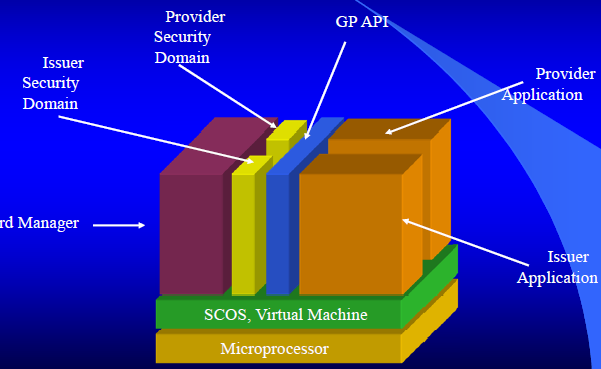
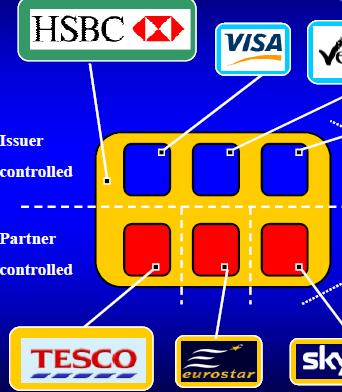
Global Platform (GP)

= independent organization, sets standards

= facilitate partnerships

* Provide a global, open multi-industry framework





SmartCard Micro-processor

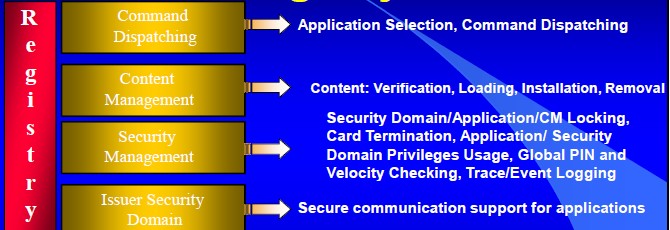
Uses Runtime Environment (RTE)

* SCOS
* VM
* API

Card Manager

* Communicate with off-card
* Initialize in secure manager
* Manage the whole card

Card Manager and Registry



Security Management

* Card Manager = center of security scheme
* Global PIN
* Card Locking
* Card Termination
* Event logging
* …

Security Domains

* Secure mechanism for add app
* Mechanism for issuers to assign priv app provider
  + Personalize app
  + Runtime crypto
  + Delegated management

Windows for smart card = Does not exist!

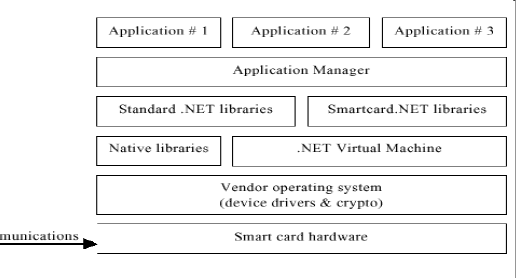
.NET card

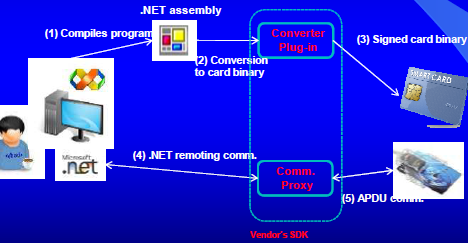
Smart card based corporate badges

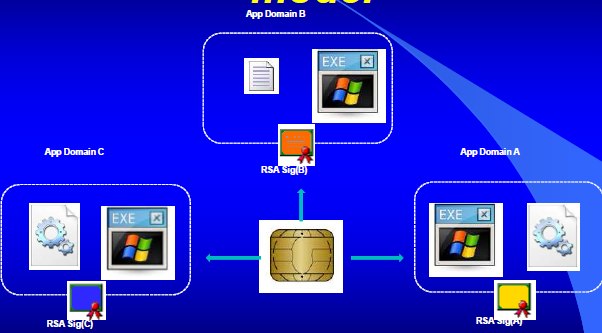
* Remote access control (VPN, cert based)

DoD

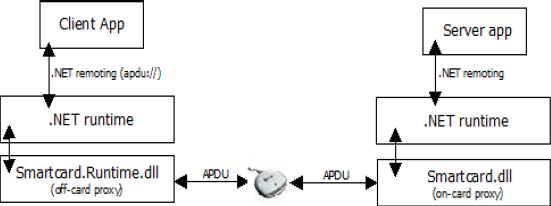
* US: combination of .NET and java card







RMI



EAL5+

BasicCard

Basic language, API for java and .NET

Very low memory requirements

Has crypto

BasicCard App Development

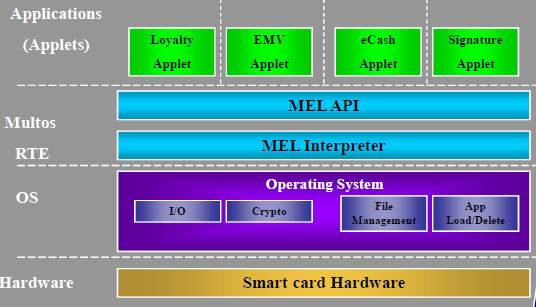
Environment

* App written in basic, java, .net
* Convert to pcode
* Vm for execution of Pcode

Transaction Manager – atomicity

* File ops or changes are single transaction

MULTOS = Operating system



Multos Application Development

