**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

**Assurance VS Complexity**

High Assurance = Low Complexity e.g. Trusted Kernels

**Security Policies**

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

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

**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

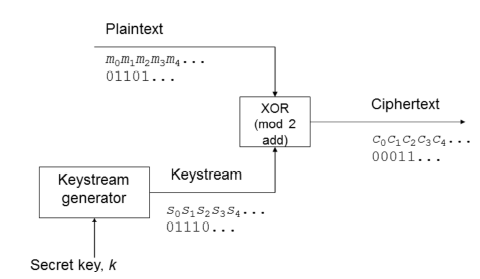
Period = 1, no Random, linear equ = 1

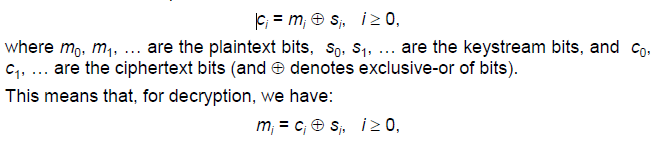
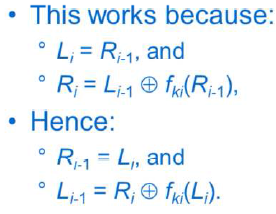
**Broken Ciphers - Vigenere Cipher**- polyalphabetic substitution cipher  
- take the number of (m) and the number of (k) and add them tgt, do it for the next letter in the (k) for next (m)  


Period = length of word

random = length of word,

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

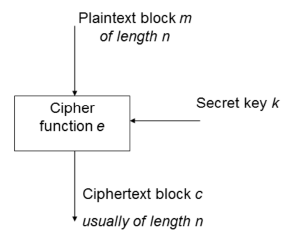
**Stream**  
- simple and fast to implement



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

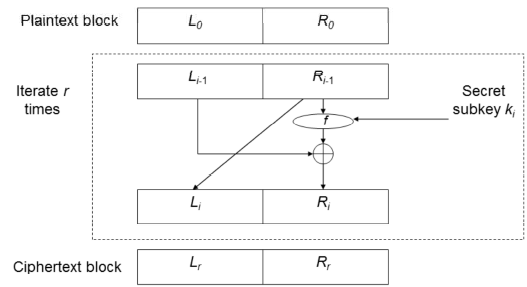
**Block Ciphers**

* 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)

**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

**Summary**

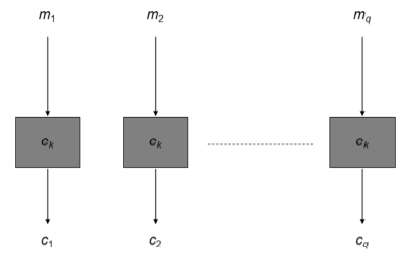
* Theory = 64 ops, Practical = 18 ops
* 2^(56+1) of operations
* **Takes twice the effort, (not a lot stronger)**
* **Use 3DES!**

**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

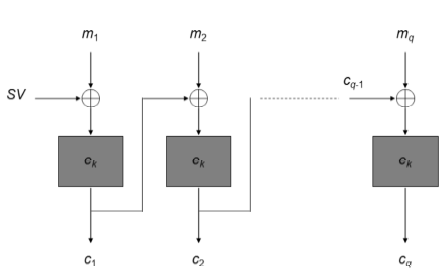
**ECB (Electronic Code Book) (Said smt is in impt here)  
-** Simplest  
****

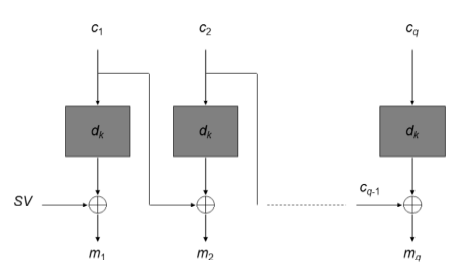
* Problem = mi=mj, then ci=cj

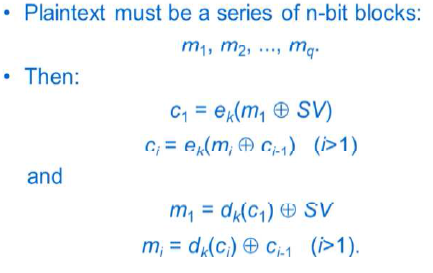
**ECB Cut and Paste Attack**

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

**Cipher Block Chaining (CBC)**





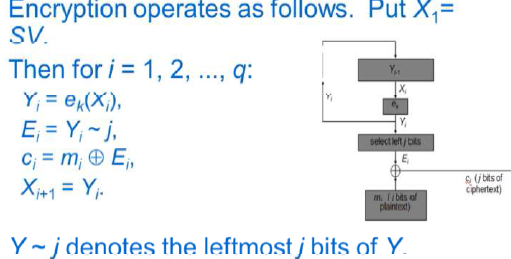


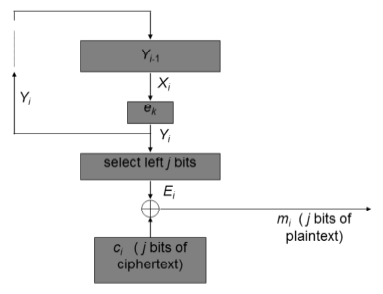
* Weakness = error propagation
* SV = starting var, diff for every msg

**Starting Value**

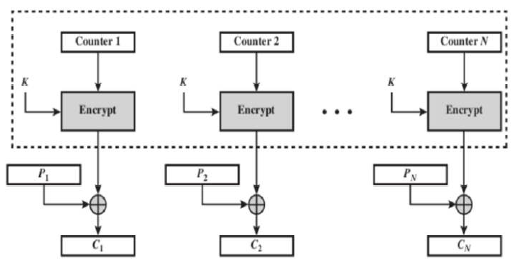
* protects identical (m) having same (c)
* SV = unpredictable
* Random(k) and encrypt a counter to produce SV
* Transmit in clear. Atker won’t know (k) to encrypt the counter and the SV

**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)

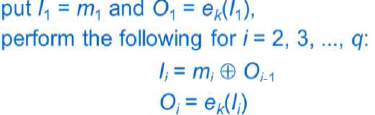
Diff = crypto algo, and use of keys (shared secret)

**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)

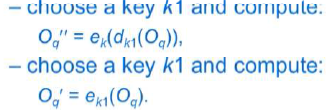
Operations

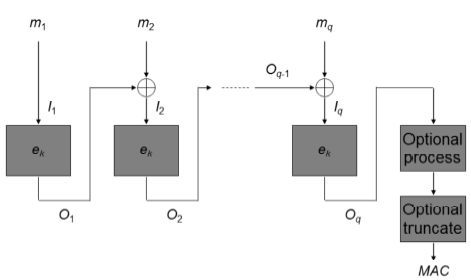
1. Pad data
2. encipher data using CBC



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

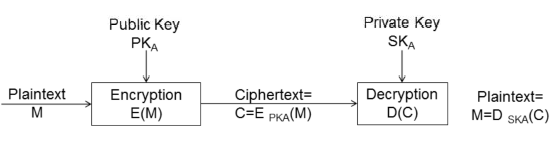
Public Key

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

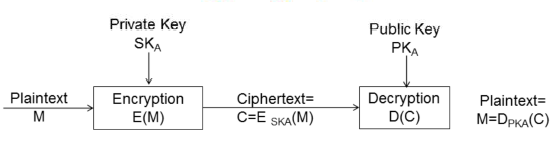
Private Key

* 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

**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)

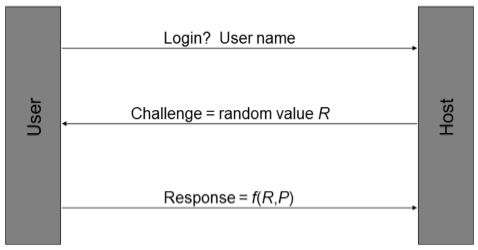
Problem in Unix

* 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

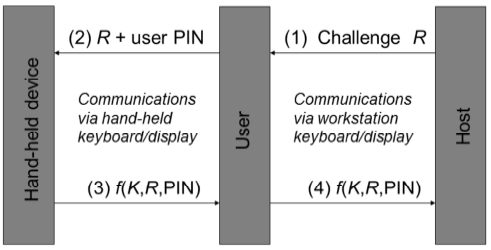
* Smart card/ Magnetic card
* Can be copied.

**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)

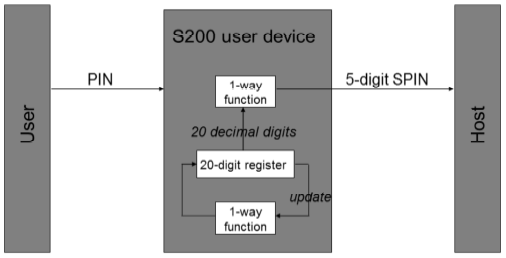
Watchword Protocol (user key, PIN, one-way f)



* liveness check

Safe 200 System

* New 5 digit password every identification



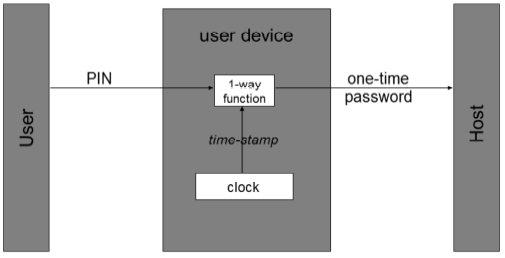
* Allows 3 iterations (before/after due to sync)

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

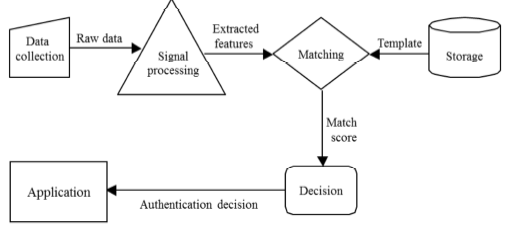
* Fingerprint
* Retinal
* Iris
* Hand geometry

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

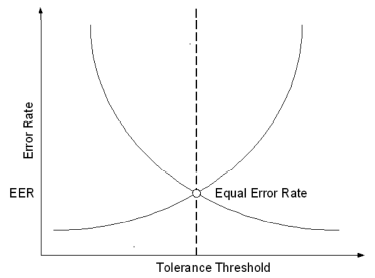
Matching subsys

* 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
* 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

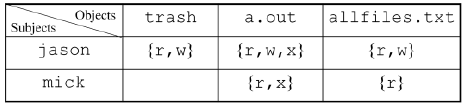
**Delegation**

* **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**

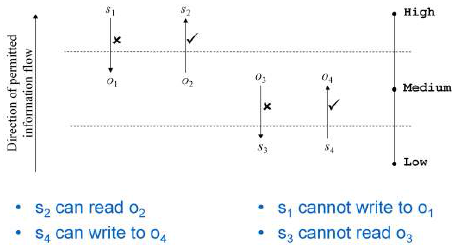
* Column in a matrix
* Each entry includes a userid and access mask
* Bit pattern where each bit represent accessright
* 111(R,W,X) then 100(R)
* Focused on objects
* Cannot check based on subject

**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

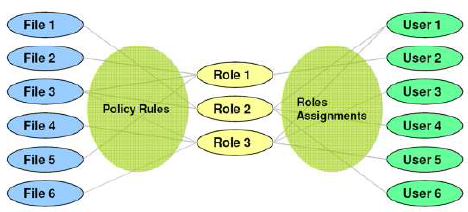
Prevents:

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

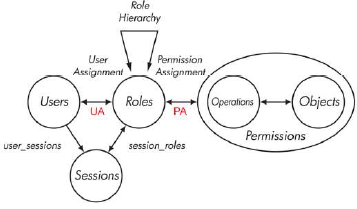
**Role-Based Access Control (RBAC)**

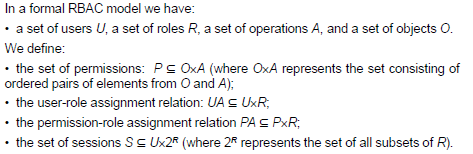
Core idea = Roles (priv management)

* 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

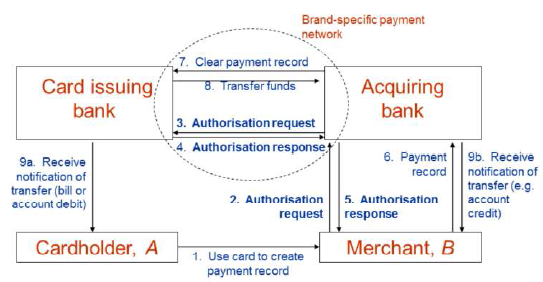
Payment model = pull model

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

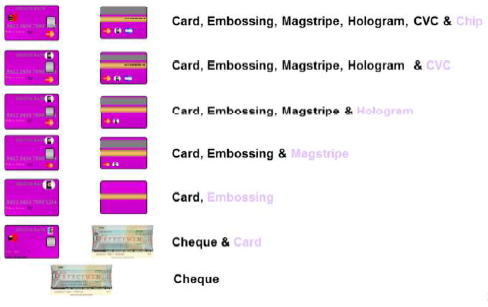
(not major, difficult to eavesdrop)

**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

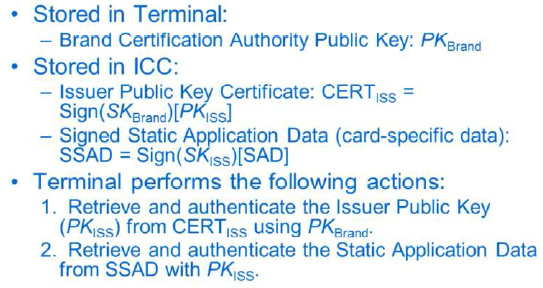
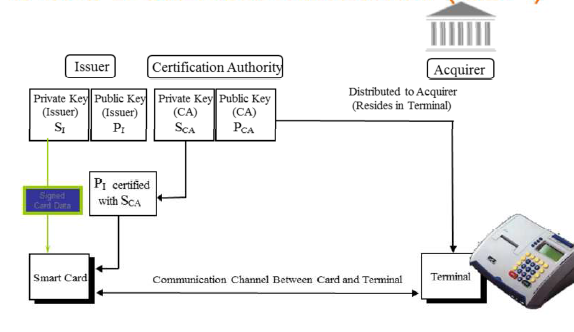
**EMV - EMV specifications**

ISO/IEC 7816 (international standards)

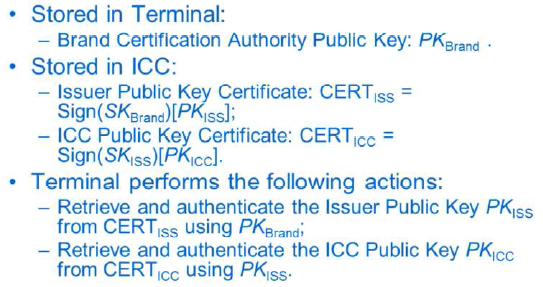
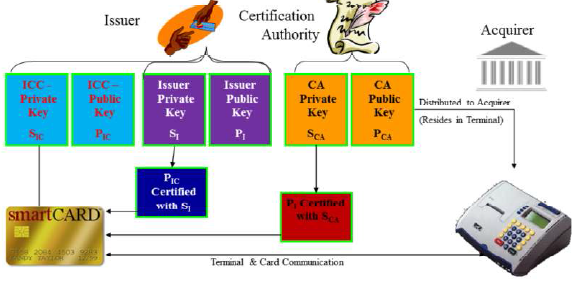
1. Application independent ICC to Terminal Interface Requirements
2. Security and key Management
3. Application Specification
4. Cardholder, attendant and Acquirer Interface

**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**

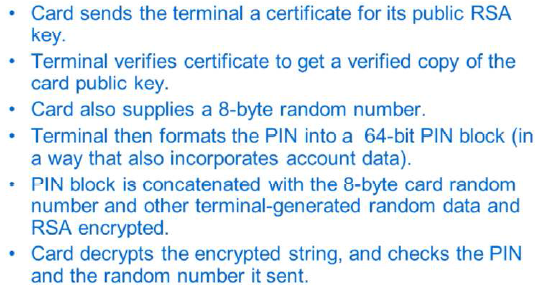
RSA and SHA-1

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**

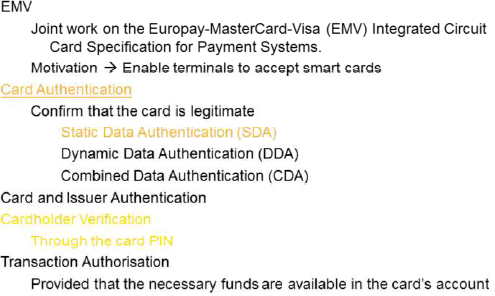
* MACs computed on messages (Protect integrity)
* Using secret keys between card and issuers
* Protect online authorization and trans records

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

**Other Vulnerabilities**

Card Skimming

**Summary**

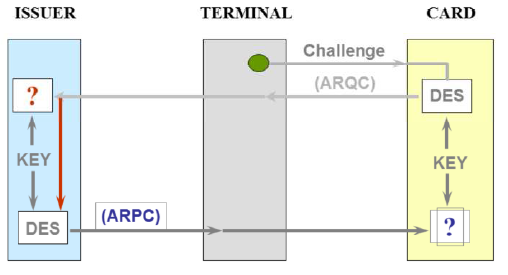


**SDA Vulnerabilities**

Weakness = card cloning

BUT Card authenticate (using PIN)

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

BUTTT discoverable if online or use DDA/CDA cards

**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

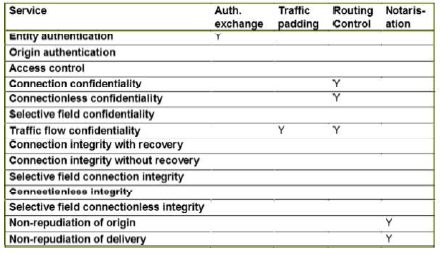
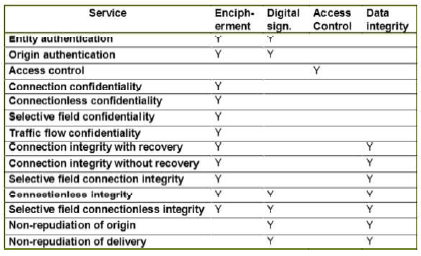
* conceal real volumes
* Traffic flow confidentiality
* Only effective with other mechanisms

Routing control mechanisms

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

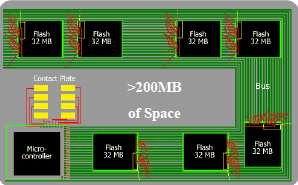
Notarisation mechanism

* 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

Software Vulnerabilities

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

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)

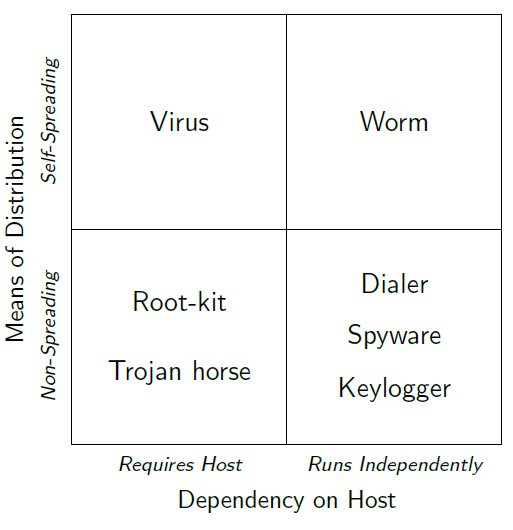
Fail securely

* 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

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

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

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

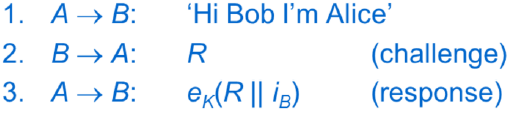
* Make it stronger by
  + 1 way encrypted pass file
  + Slow down encryption
  + Add salt
  + Access control on pass file
  + Lock after failed attempts
  + OTP

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

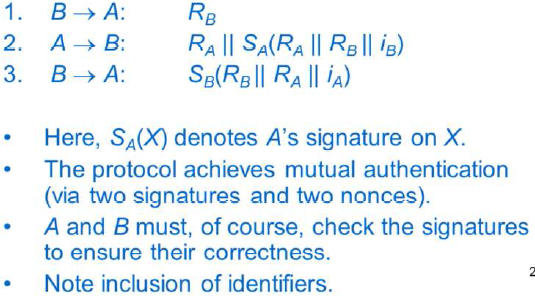
Include current value and increments it

Needs a pair of secret seq num for communicating parties

**Signature-based entity authentication**

Challenge/Signature

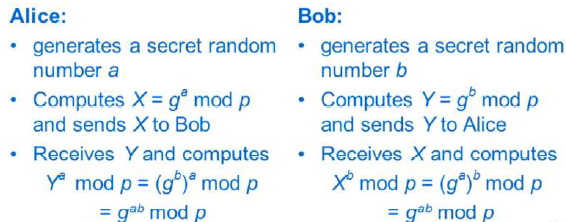
Nonce/ time-stamp for freshness

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

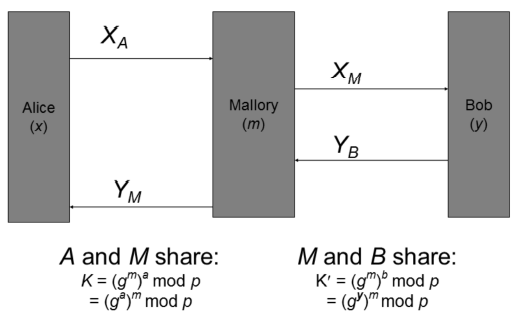
**Diffie-Hellmann Key Agreement**



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

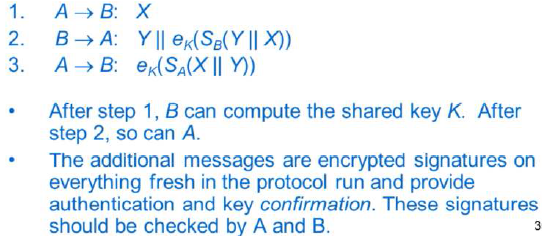
Secure against passive adversaries

**Man in the middle attack**



**Station-to-Station Protocol**

Adds authentication (via signatures) to DHKey



**Kerberos (how messages work)**

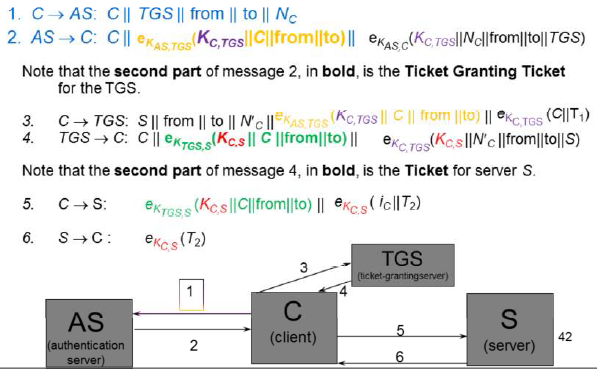
TTP-aided authentication protocol

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

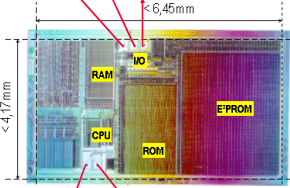


Kerberos uses symmetric encryption and MDC

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

**Smart Card**



ROM = OS

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

RAM = OS Workspace7

SC File System (Hierarchical)

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

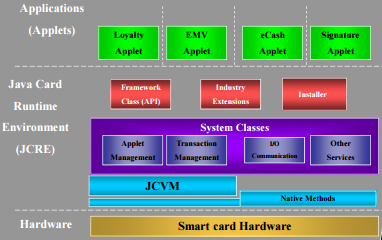
Why Java? Standard language, OOP, security, sandbox, portability, and interoperability

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



Remote Method Invocation (RMI) - abstraction

Lifetime of Java Card

* Starts when the SCOS

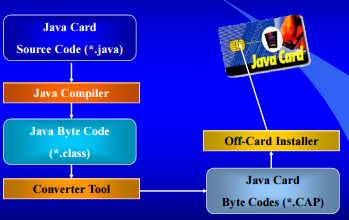
Lifetime of JCVM

* Runs forever unlike JVM

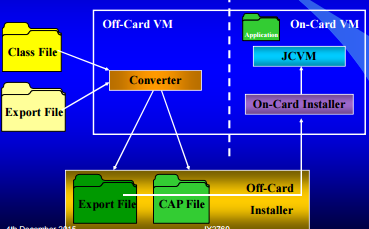
Lifetime of JC applets and objects

* 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
* 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

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

Both share security features



Typical Classical computer security threats apply.

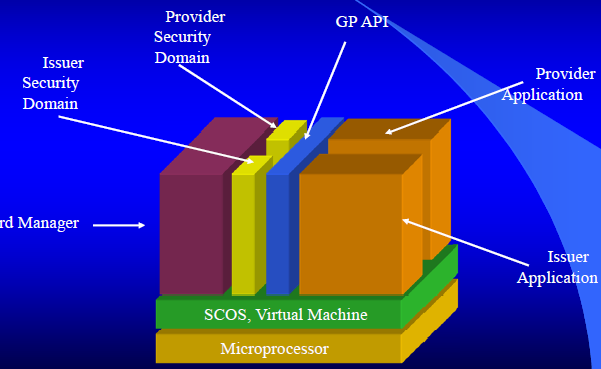
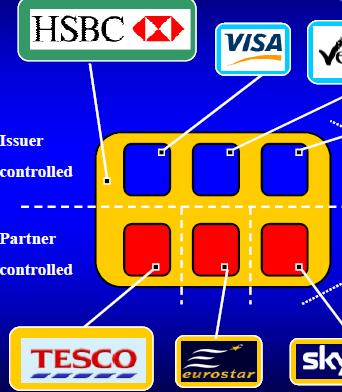
Global Platform (GP)

= independent organization, sets standards

= facilitate partnerships

* Provide a global, open multi-industry framework





SmartCard Micro-processor

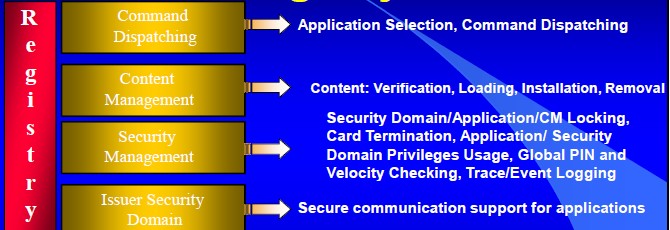
Uses Runtime Environment (RTE)

* SCOS
* VM
* API

Card Manager

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

Card Manager and Registry



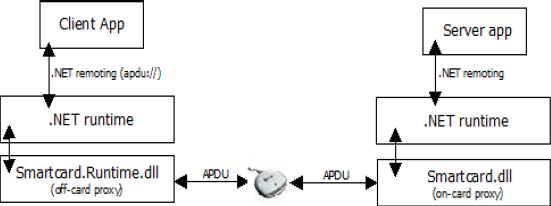
Security Management

* 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

RMI



EAL5+

BasicCard

* Basic language, API for java and .NET
* Very low memory requirements
* Has crypto

BasicCard App Development

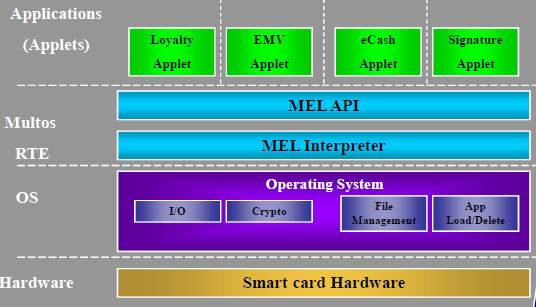
Environment

* 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

