# INFS 766
## Internet Security Protocols

## <u>Lectures 3 and 4</u>
## Cryptography in network protocols

**Prof. Ravi Sandhu**

# CRYPTOGRAPHY

# CRYPTOGRAPHIC TECHNOLOGY

**SECRET KEY**

**Symmetric Key**
**Single Key**
**Conventional**

**PUBLIC KEY**

**Asymmetric Key**
**Two Key**




# CRYPTOGRAPHIC TECHNOLOGY

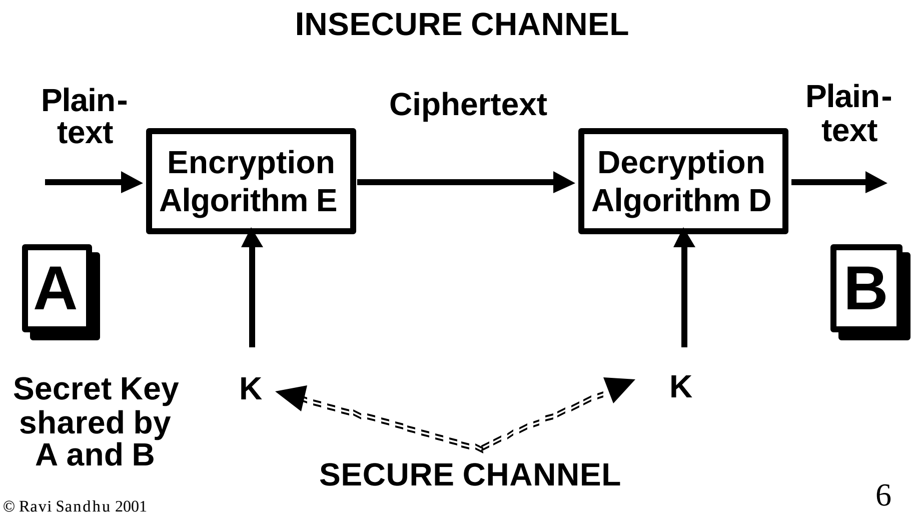
- **Secret-key encryption**
- **Public-key encryption**
- **Public-key digital signatures**
- **Public-key key agreement**
- **Message digests**
- **Message authentication codes**
- **Challenge-response authentication**
- **Public-key certificates**

# CRYPTOGRAPHIC SERVICES

- confidentiality
  - traffic flow confidentiality
- integrity
- authentication
- non-repudiation




# SECRET KEY CRYPTOSYSTEM



# SECRET KEY CRYPTOSYSTEM

- confidentiality depends only on secrecy of the key
  - size of key is critical
- secret key systems do not scale well
  - with N parties we need to generate and distribute N*(N-1)/2 keys
- A and B can be people or computers




# MASTER KEYS AND SESSION KEYS

- long-term or master keys
  - prolonged use increases exposure
- session keys
  - short-term keys communicated by means of
    - long-term secret keys
    - public key technology

# CRYPTANALYSIS

- ciphertext only
  - cryptanalyst only knows ciphertext
- known plaintext
  - cryptanalyst knows some plaintext-ciphertext pairs
- chosen plaintext
- chosen ciphertext




# KNOWN PLAINTEXT ATTACK

- 40 bit key requires $2^{39}$ » $5 * 10^{11}$ trials on average (exportable from USA)
- trials/second time required

| trials/second | time required |
|---|---|
| 1 | 20,000 years |
| $10^3$ | 20 years |
| $10^6$ | 6 days |
| $10^9$ | 9 minutes |
| $10^{12}$ | 0.5 seconds |

# KNOWN PLAINTEXT ATTACK

- **56 bit key requires $2^{55}$ » 3.6 * $10^{\wedge 16}$ trials on average (DES)**
- **trials/second time required**

| trials/second | time required |
|---|---|
| **1** | **$10^{9}$ years** |
| **$10^{3}$** | **$10^{6}$ years** |
| **$10^{6}$** | **$10^{3}$ years** |
| **$10^{9}$** | **1 year** |
| **$10^{12}$** | **10 hours** |




# KNOWN PLAINTEXT ATTACK

- **80 bit key requires $2^{79}$ » 6 * $10^{23}$ trials on average (SKIPJACK)**
- **trials/second time required**

| trials/second | time required |
|---|---|
| **1** | **$10^{16}$ years** |
| **$10^{3}$** | **$10^{13}$ years** |
| **$10^{6}$** | **$10^{10}$ years** |
| **$10^{9}$** | **$10^{7}$ years** |
| **$10^{12}$** | **$10^{4}$ years** |

# KNOWN PLAINTEXT ATTACK

- **128 bit key requires $2^{127}$ » $2 * 10^{38}$ trials on average (IDEA)**
- **trials/second time required**

| trials/second | time required |
| --- | --- |
| **1** | **$10^{30}$ years** |
| **$10^{3}$** | **$10^{27}$ years** |
| **$10^{6}$** | **$10^{24}$ years** |
| **$10^{9}$** | **$10^{21}$ years** |
| **$10^{12}$** | **$10^{18}$ years** |




# DICTIONARY ATTACKS

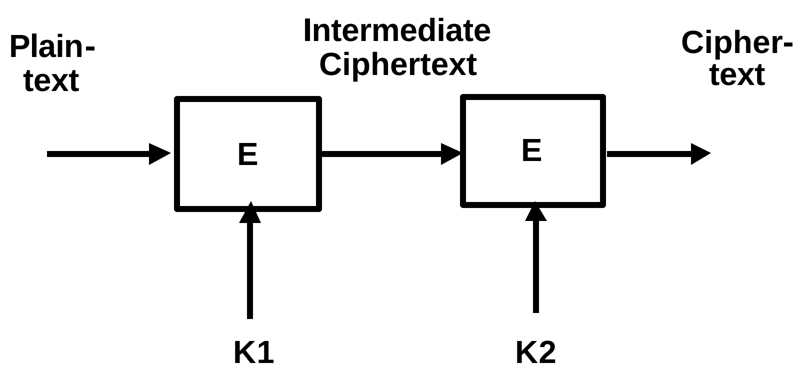
- **if keys are poorly chosen known plaintext attacks can be very simple**
- **often the user's password is the key**
  - **in a dictionary attack the cryptanalyst tries passwords from a dictionary, rather than all possible keys**
  - **for a 20,000 word dictionary, 1 trial/second will crack a poor password in less than 3 hours**

# CURRENT GENERATION SECRET KEY CRYPTOSYSTEMS

- **64 bit data block size**
  - **DES: 56 bit key**
  - **Triple DES: 112 bit key**
  - **Triple DES: 168 bit key**
  - **Skipjack: 80 bit key**
  - **IDEA: 128 bit key**
  - **RC2: variable size key: 1 byte to 128 bytes**
  - **many others**

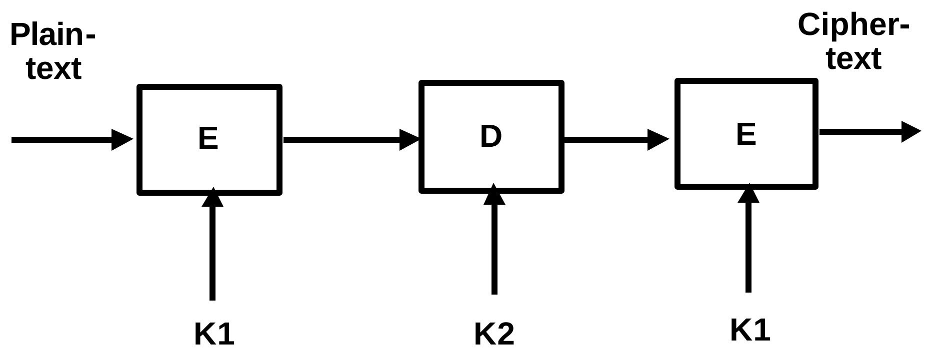



# DOUBLE DES




- **effective key size is only 57 bits due to meet-in-the-middle attack**

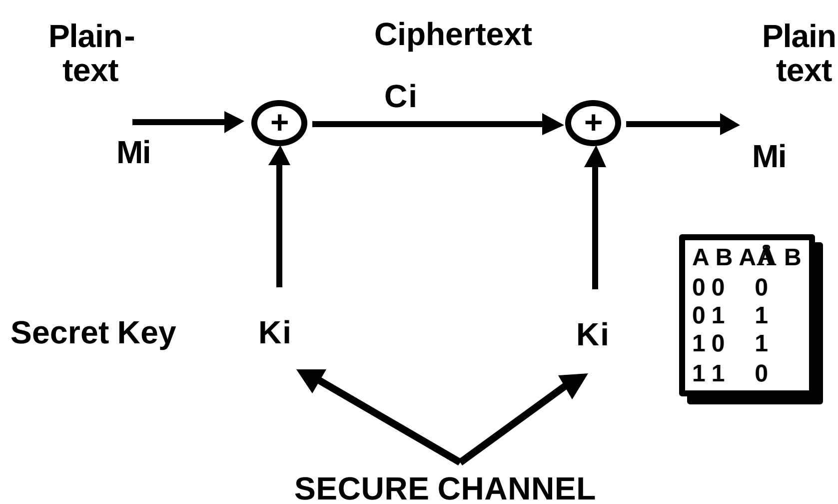
# TRIPLE DES




- **effective key size is 112 bits due to meet-in-the-middle attack**




# PERFECT SECRECY
# VERNAM ONE-TIME PAD



## PERFECT SECRECY VERNAM ONE-TIME PAD

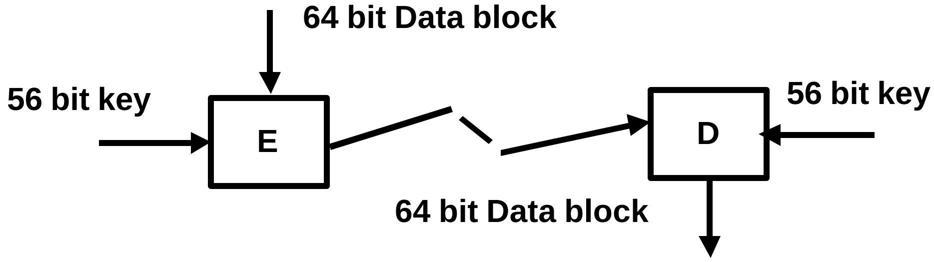
- known plaintext reveals the portion of the key that has been used, but does not reveal anything about the future bits of the key
- has been used
- can be approximated




## ADVANCED ENCRYPTION STANDARD

- New Advanced Encryption Standard under development by NIST
  - must support key-block combinations of 128-128, 192-128, 256-128
  - may support other combinations
- selection of Rijndaehl algorithm announced in 2000
- will be in place in a couple of years

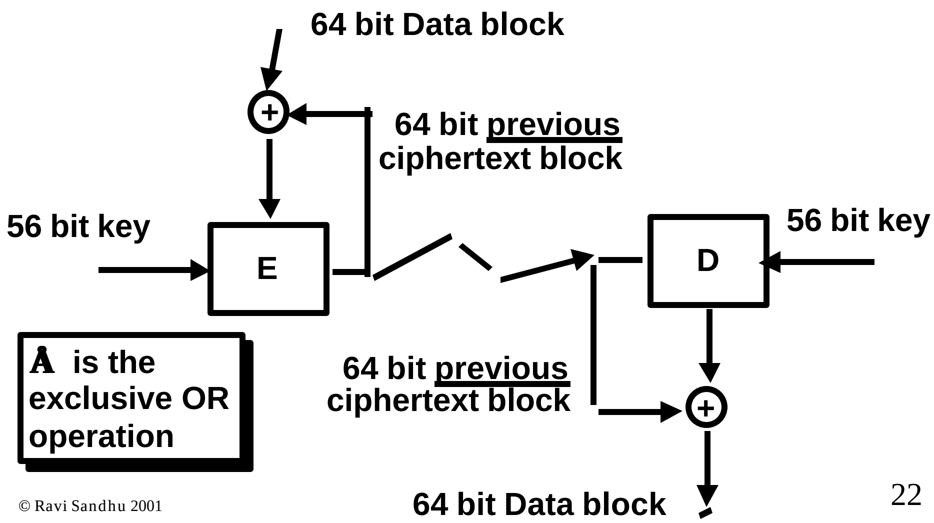
# ELECTRONIC CODE BOOK (ECB) MODE




- OK for small messages
- identical data blocks will be identically encrypted




# CIPHER BLOCK CHAINING (CBC) MODE

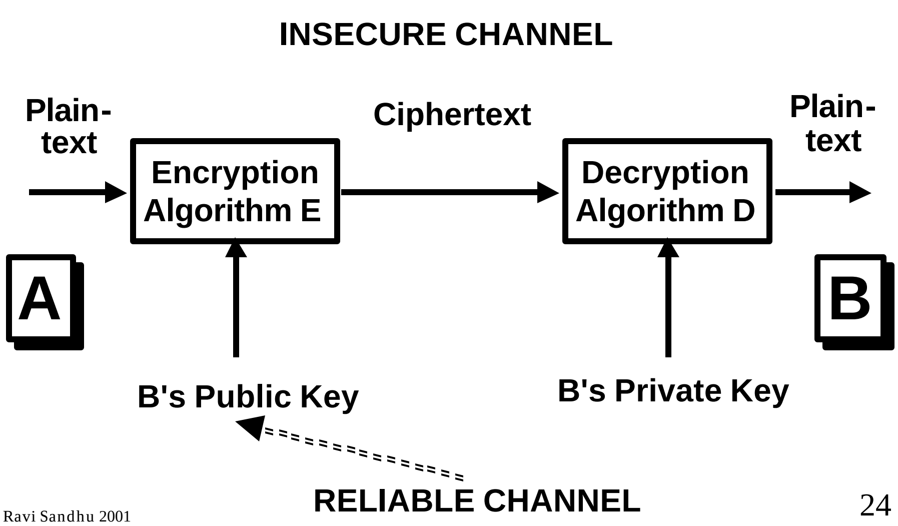


# CIPHER BLOCK CHAINING (CBC) MODE

- **Needs an Initialization Vector (IV) to serve as the first feedback block**
- **IV need not be secret or random**
- **Integrity of the IV is important, otherwise first data block can be arbitrarily changed.**
- **IV should be changed from message to message, or first block of every message should be distinct**




# PUBLIC KEY ENCRYPTION



# PUBLIC KEY CRYPTOSYSTEM

- **solves the key distribution problem provided there is a reliable channel for communication of public keys**
- **requires reliable dissemination of 1 public key/party**
- **scales well for large-scale systems**




# PUBLIC KEY ENCRYPTION

- **confidentiality based on infeasibility of computing B's private key from B's public key**
- **key sizes are large (512 bits and above) to make this computation infeasible**

# SPEED OF PUBLIC KEY VERSUS SECRET KEY

- Public key runs at kilobits/second
  - think modem connection
- Secret key runs at megabits/second and even gigabits/second
  - think LAN or disk connection
- This large difference in speed is likely to remain independent of technology advances




# RSA

- public key is (n,e)
- private key is d
- encrypt: $C = M^e \bmod n$
- decrypt: $M = C^d \bmod n$

## GENERATION OF RSA KEYS

- choose 2 large (100 digit) prime numbers p and q
- compute n = p * q
- pick e relatively prime to (p-1)*(q-1)
- compute d, e*d = 1 mod (p-1)*(q-1)
- publish (n,e)
- keep d secret (and discard p, q)




## PROTECTION OF RSA KEYS

- compute d, e*d = 1 mod (p-1)*(q-1)
  - if factorization of n into p*q is known, this is easy to do
- security of RSA is no better than the difficulty of factoring n into p, q

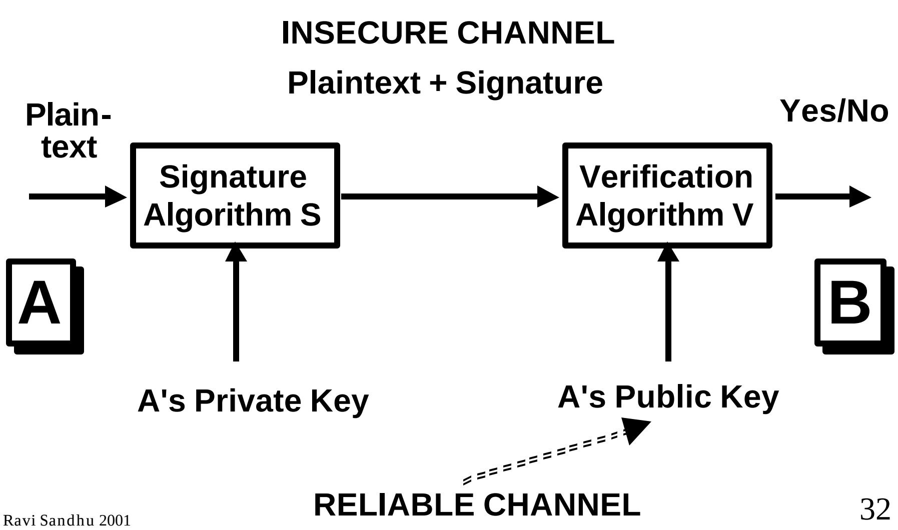
# RSA KEY SIZE

- key size of RSA is selected by the user
  - casual 384 bits
  - "commercial" 512 bits
  - "military" 1024 bits

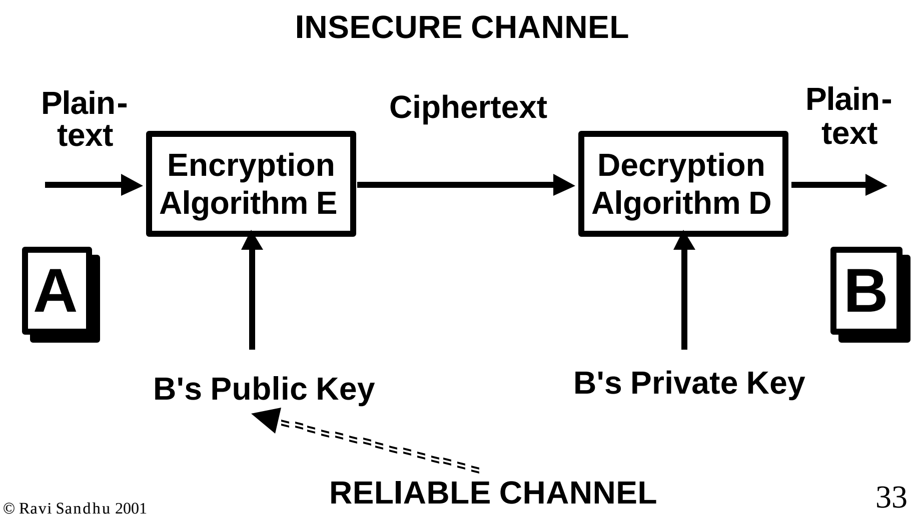




# DIGITAL SIGNATURES



# COMPARE PUBLIC KEY ENCRYPTION







# DIGITAL SIGNATURES IN RSA

- RSA has a unique property, not shared by other public key systems
- Encryption and decryption commute
  - $(M^e \bmod n)^d \bmod n = M$ encryption
  - $(M^d \bmod n)^e \bmod n = M$ signature
- Same public key can be use for encryption and signature

# EL GAMAL AND VARIANTS

- encryption only
- signature only
  - 1000's of variants
  - including NIST's DSA




# NIST DIGITAL SIGNATURE STANDARD

- System-wide constants
  - p 512-1024 bit prime
  - q 160 bit prime divisor of p-1
  - g $g = h^{((p-1)/q)} \bmod p, 1<h<p-1$
- El-Gamal variant
  - separate algorithms for digital signature and public-key encryption

# NIST DIGITAL SIGNATURE STANDARD

- to sign message m: private key x
  - choose random r
  - compute v = ($g^r$ mod p) mod q
  - compute s = (m+xv)/k mod q
  - signature is (s,v,m)
- to verify signature: public key y
  - compute u1 = m/s mod q
  - compute u2 = v/s mod q
  - verify that v = ($g^{u1}$*$y^{u2}$ mod p) mod q




# NIST DIGITAL SIGNATURE STANDARD

- signature does not repeat, since r will be different on each occasion
- if same random number r is used for two messages, the system is broken
- message expands by a factor of 2
- RSA signatures do repeat, and there is no message expansion

# DIFFIE-HELLMAN KEY ESTABLISHMENT

**A** $y_A = a^{x_A} \bmod p$ public key

$y_B = a^{x_B} \bmod p$ public key **B**

private key $x_A$

private key $x_B$

$$k = y_B^{x_A} \bmod p = y_A^{x_B} \bmod p = a^{x_A * x_B} \bmod p$$

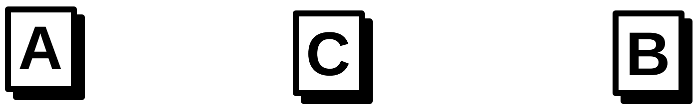
**system constants: p: prime number, a: integer**




# DIFFIE-HELLMAN KEY ESTABLISHMENT

- **security depends on difficulty of computing x given $y=a^x \bmod p$**
  **called the discrete logarithm problem**

# MAN IN THE MIDDLE ATTACK







# CURRENT GENERATION PUBLIC KEY SYSTEMS

- **RSA (Rivest, Shamir and Adelman)**
  - **the only one to provide digital signature and encryption using the same public-private key pair**
  - **security based on factoring**
- **ElGamal Encryption**
  - **public-key encryption only**
  - **security based on digital logarithm**
- **DSA signatures**
  - **public-key signature only**
  - **one of many variants of ElGamal signature**
  - **security based on digital logarithm**

# CURRENT GENERATION PUBLIC KEY SYSTEMS

- **DH (Diffie-Hellman)**
  - **secret key agreement only**
  - **security based on digital logarithm**
- **ECC (Elliptic curve cryptography)**
  - **security based on digital logarithm in elliptic curve field**
  - **uses analogs of**
    - **ElGamal encryption**
    - **DH key agreement**
    - **DSA digital signature**




# ELLIPTIC CURVE CRYPTOGRAPHY

- **mathematics is more complicated than RSA or Diffie-Hellman**
- **elliptic curves have been studied for over one hundred years**
- **computation is done in a group defined by an elliptic curve**

# ELLIPTIC CURVE CRYPTOGRAPHY

- **160 bit ECC public key is claimed to be as secure as 1024 bit RSA or Diffie-Hellman key**
- **good for small hardware implementations such as smart cards**




# ELLIPTIC CURVE CRYPTOGRAPHY

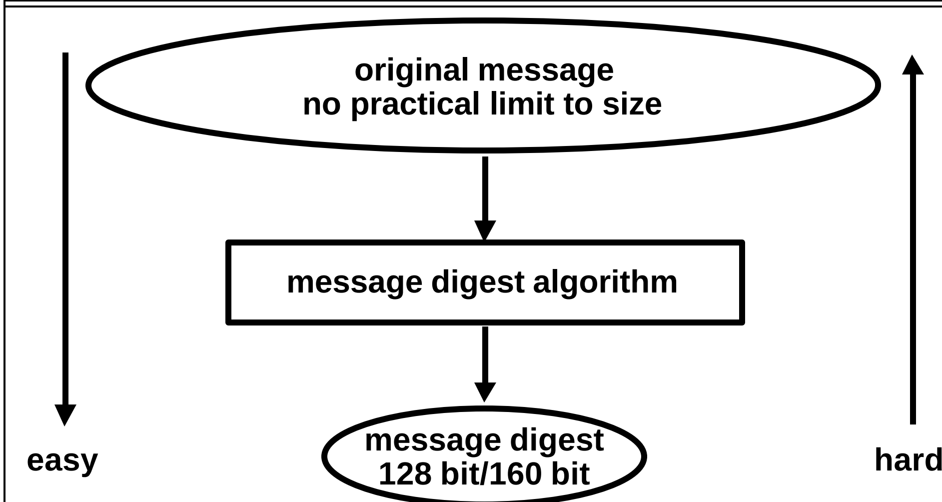
- **ECDSA: Elliptic Curve digital signature algorithm based on NIST Digital Signature Standard**
- **ECSVA: Elliptic Curve key agreement algorithm based on Diffie-Hellman**
- **ECES: Elliptic Curve encryption algorithm based on El-Gamal**

# PKCS STANDARDS

- de facto standards initiated by RSA Data Inc.




# MESSAGE DIGEST



# MESSAGE DIGEST

- for performance reasons
  - sign the message digest
  - not the message
- <u>one way</u> function
  - m=H(M) is easy to compute
  - $M=H^{-1}(m)$ is hard to compute




# DESIRED CHARACTERISTICS

- <u>weak hash</u> function
  - difficult to find M' such that H(M')=H(M)
- given M, m=H(M) try messages at random to find M' with H(M')=m
  - $2^k$ trials on average, k=64 to be safe

# DESIRED CHARACTERISTICS

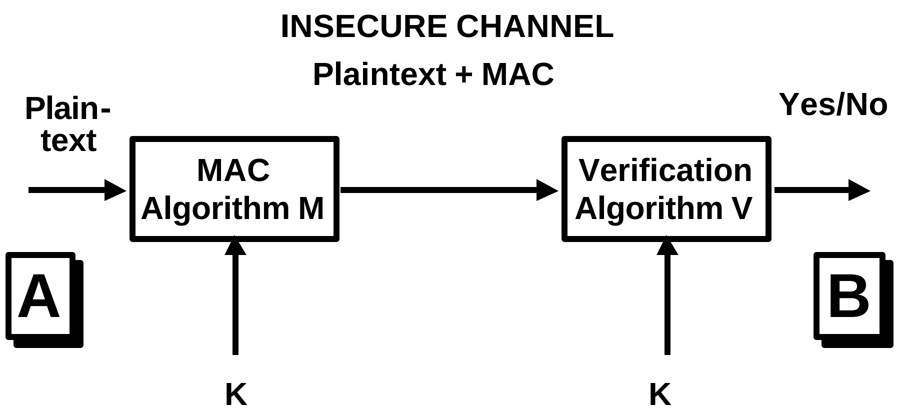
- **strong hash function**
  - **difficult to find any two M and M' such that H(M')=H(M)**
- **try pairs of messages at random to find M and M' such that H(M')=H(M)**
  - **$2^{k/2}$ trials on average, k=128 to be safe**
  - **k=160 is better**




# CURRENTT GENERATION MESSAGE DIGEST ALGORITHMS

- **MD5 (Message Digest 5)**
  - **128 bit message digest**
  - **falling out of favor**
- **SHA (Secure Hash Algorithm)**
  - **160 bit message digest**
  - **slightly slower than MD5 but more secure**

# MESSAGE AUTHENTICATION CODES




MAC = MD of plaintext + K





# CURRENT GENERATION MAC ALGORITHMS

- **HMAC-MD5, HMAC-SHA**
  - **IETF standard**
  - **general technique for constructing a MAC from a message digest algorithm**
- **Older MACs are based on secret key encryption algorithms (notably DES) and are still in use**
  - **DES based MACs are 64 bit and not considered strong anymore**

# HMAC

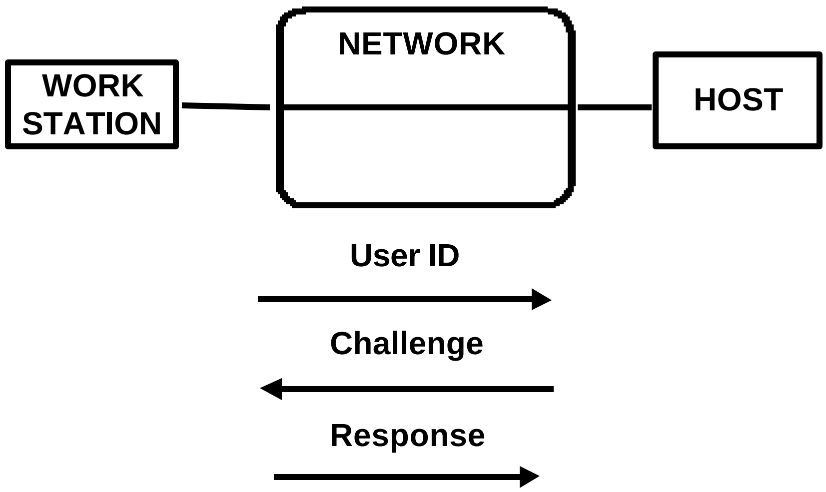
- **HMAC computation**
- **HMAC$_K$(M) = h(KÅ opad || h(KÅ ipad || M) )**
  - **h is any message digest function**
  - **M message**
  - **K secret key**
  - **opad, ipad: fixed outer and inner padding**
- **HMAC-MD5, HMAC-SHA**




# SAFE CRYPTOGRAPHY

- **Secret-key encryption**
  - **128 bit or higher**
- **Public-key**
  - **1024 bit or higher**
- **Message digests**
  - **160 bit or higher**
- **A large portion of what is deployed is much weaker**

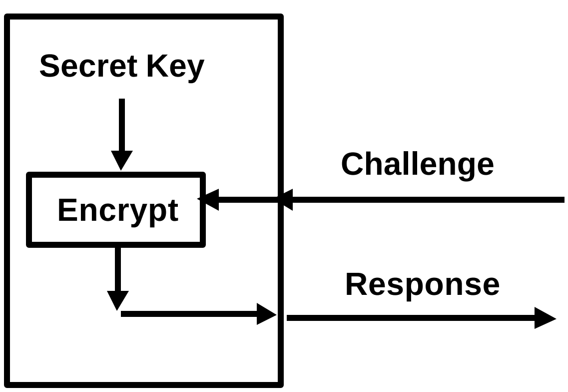
# STRONG AUTHENTICATION CHALLENGE RESPONSE



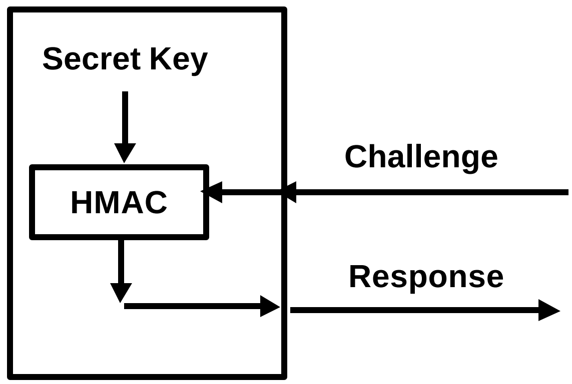





# CHALLENGE RESPONSE



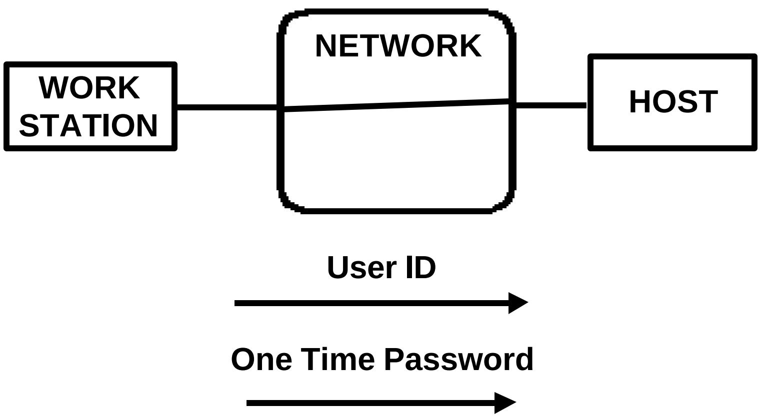
# CHALLENGE RESPONSE



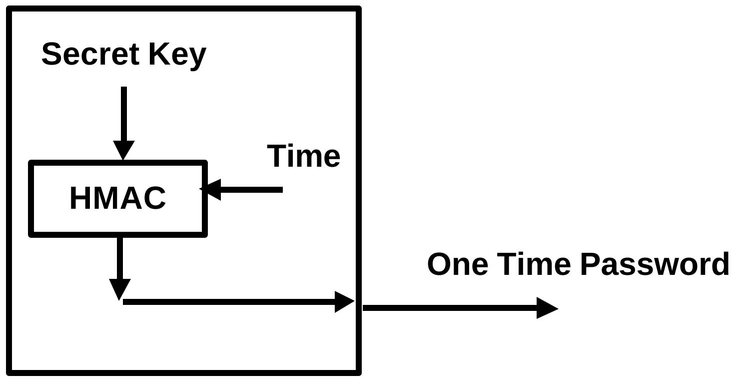




# TIME SYNCHRONIZED

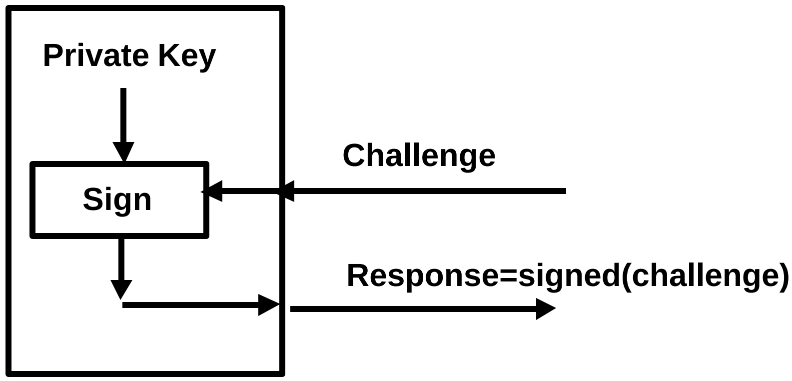


# TIME SYNCHRONIZED

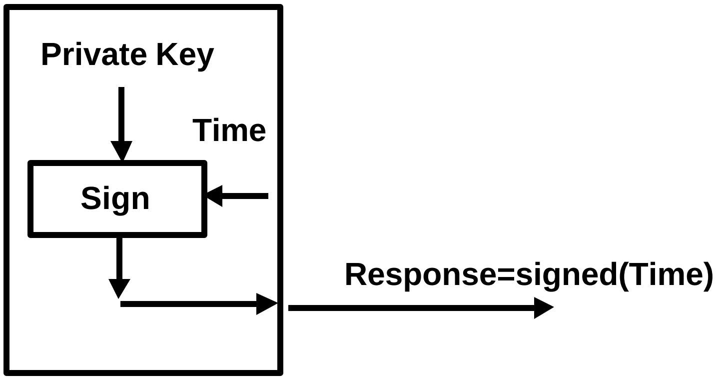





# PUBLIC KEY SIGNATURE BASED



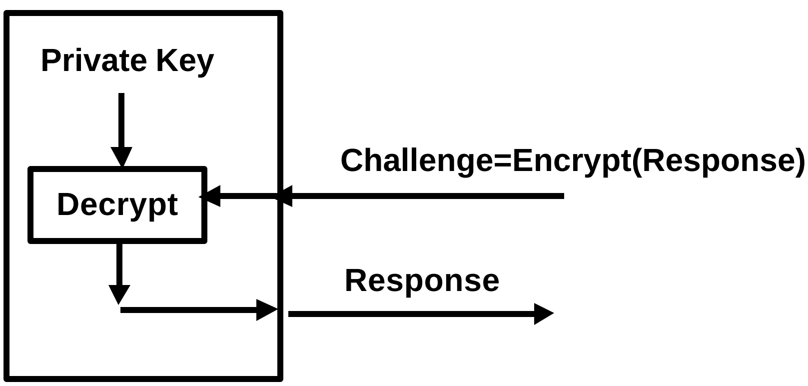
# PUBLIC KEY SIGNATURE BASED



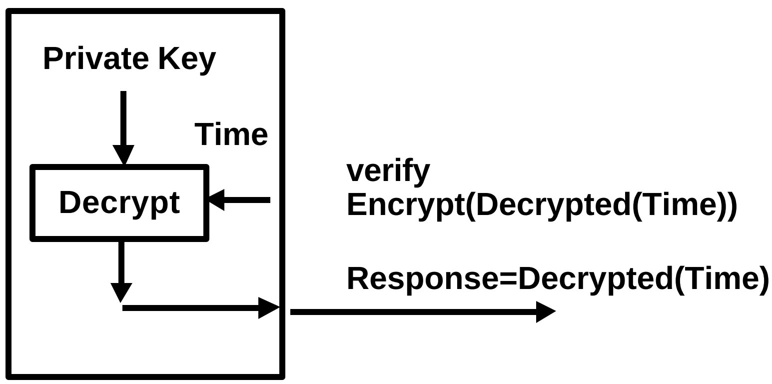




# PUBLIC KEY ENCRYPTION BASED



# PUBLIC KEY ENCRYPT BASED






# PUBLIC-KEY INFRASTRUCTURE

# PUBLIC-KEY CERTIFICATES

- **reliable distribution of public-keys**
- **public-key encryption**
  - **sender needs public key of receiver**
- **public-key digital signatures**
  - **receiver needs public key of sender**
- **public-key key agreement**
  - **both need each other's public keys**




# X.509 CERTIFICATE

| |
|---|
| **VERSION** |
| **SERIAL NUMBER** |
| **SIGNATURE ALGORITHM** |
| **ISSUER** |
| **VALIDITY** |
| **SUBJECT** |
| **SUBJECT PUBLIC KEY INFO** |
| ***SIGNATURE*** |

# X.509 CERTIFICATE

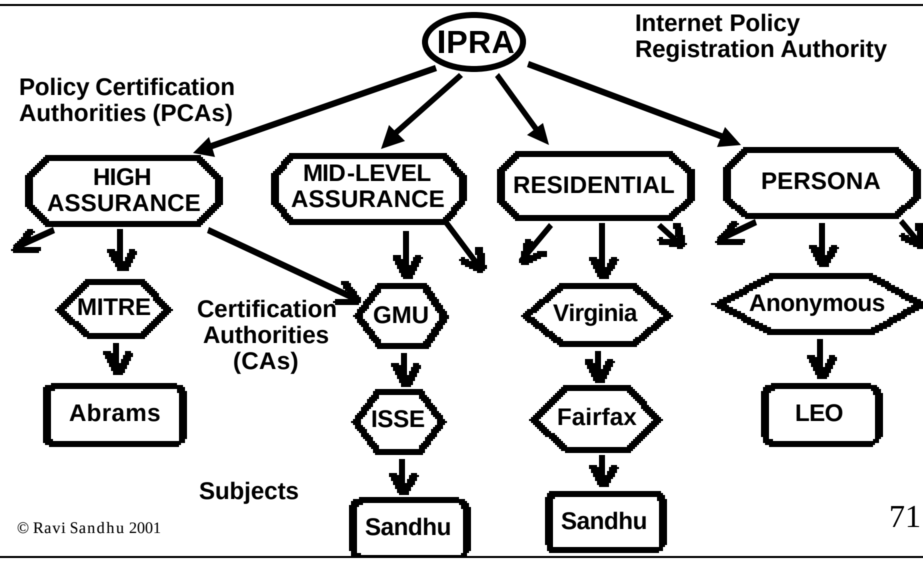
| |
|---|
| 1 |
| 1234567891011121314 |
| RSA+MD5, 512 |
| C=US, S=VA, O=GMU, OU=ISSE |
| 9/9/99-1/1/1 |
| C=US, S=VA, O=GMU, OU=ISSE, CN=Ravi Sandhu |
| RSA, 1024, xxxxxxxxxxxxxxxxxxxxxxxxx |
| ***SIGNATURE*** |




# CERTIFICATE TRUST

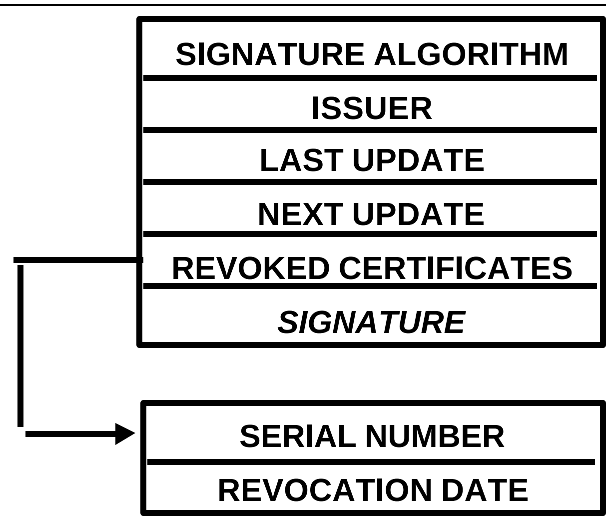
- **how to acquire public key of the issuer to verify signature**
- **whether or not to trust certificates signed by the issuer for this subject**

# PEM CERTIFICATION GRAPH






# CRL FORMAT



# X.509 CERTIFICATES

- **X.509v1**
  - **very basic**
- **X.509v2**
  - **adds unique identifiers to prevent against reuse of X.500 names**
- **X.509v3**
  - **adds many extensions**
  - **can be further extended**



# X.509v3 CERTIFICATE INNOVATIONS

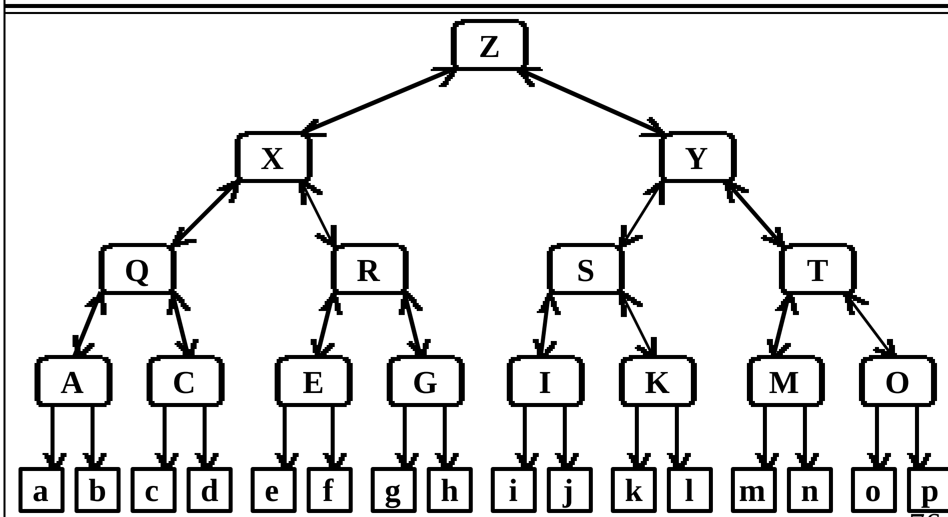
- **distinguish various certificates**
  - **signature, encryption, key-agreement**
- **identification info in addition to X.500 name**
  - **internet names: email addresses, host names, URLs**
- **issuer can state policy and usage**
  - **good enough for casual email but not for signing checks**
- **limits on use of signature keys for further certification**
- **extensible**
  - **proprietary extensions can be defined and registered**
- **attribute certificates**
  - **ongoing work**

# X.509v2 CRL INNOVATIONS

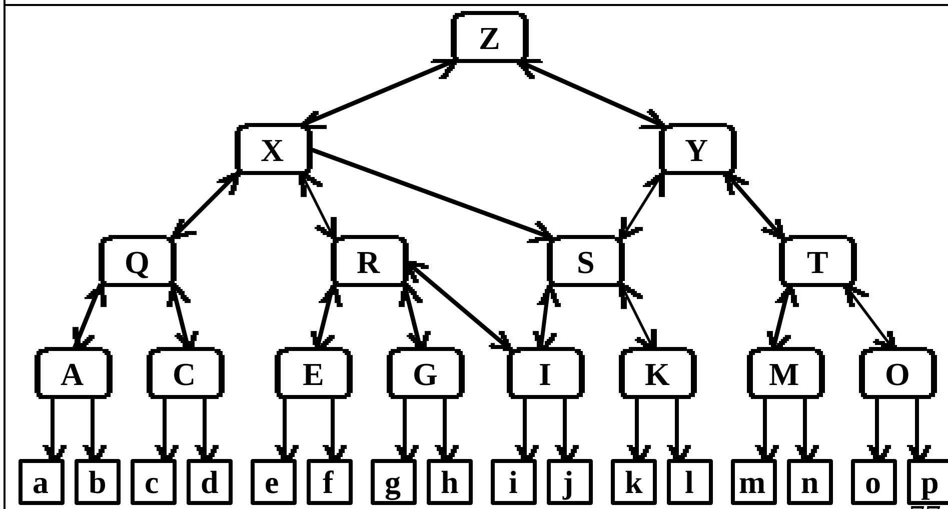
- **CRL distribution points**
- **indirect CRLs**
- **delta CRLs**
- **revocation reason**
- **push CRLs**




# GENERAL HIERARCHICAL STRUCTURE



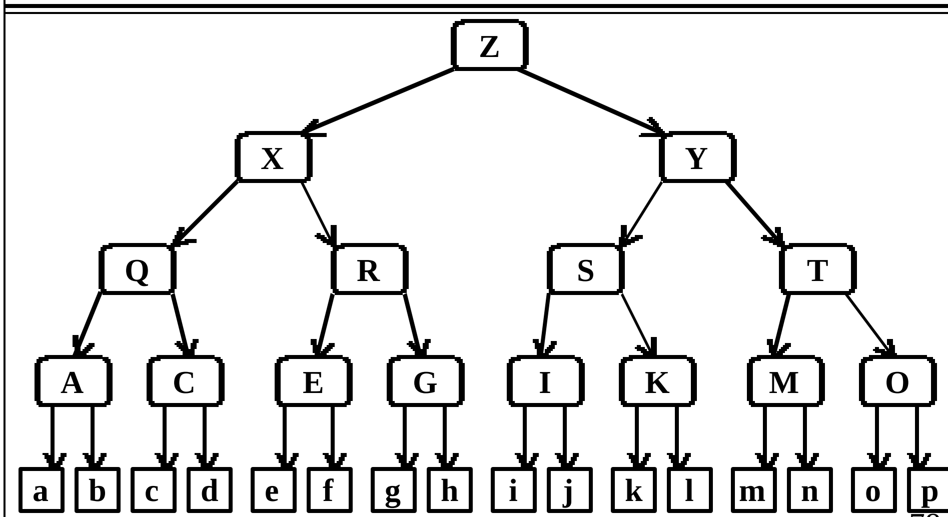
# GENERAL HIERARCHICAL STRUCTURE WITH ADDED LINKS







# TOP-DOWN HIERARCHICAL STRUCTURE



# FOREST OF HIERARCHIES