

Cryptography and Network Security



Digital Signature

Xiang-Yang Li





Message Authentication

Digital Signature

Authentication

-  Authentication requirements
-  Authentication functions

Mechanisms

-  MAC: message authentication code
-  Hash functions, security in hash functions
-  Hash and MAC algorithms
 -  MD5, SHA, RIPEMD-160, HMAC

Digital signatures

Message Attacks

✍ Possible attacks

- ✍ Disclosure
- ✍ Traffic analysis
- ✍ Masquerade
- ✍ Content modification
- ✍ Sequence modification
- ✍ Time modification
- ✍ Repudiation
 - ✍ Denial of the receipt of message by the destination or
 - ✍ Denial of the transmitting by the source

Authentication

- ✍ Enables receiver to verify message authenticity
 - ✍ Using some lower level functions as primitive
- ✍ Three types of functions
 - ✍ Message encryption
 - ✍ Message authentication code
 - ✍ Hash function

Message Encryption

✍ Conventional Encryption

- ✍ Authentication provided due to the secret key
- ✍ But the message need to be meaningful
 - ✍ What happened it message is not readable?
 - ✍ How to determine intelligible automatically?

✍ Approach

- ✍ Checksum or frame check sequence(FCS) to message
- ✍ Encrypt the message and the appending FCS
- ✍ Receiver decrypt the ciphertext
- ✍ Computes FCS of message, compare with received one

Public Key Encryption

- ✍ Direct encryption by receiver's public key
 - ✍ Only confidentiality, no authentication
- ✍ For authentication
 - ✍ Encrypt using sender's private key
 - ✍ Assume the message is intelligible
 - ✍ No confidentiality: everyone can decrypt
- ✍ Confidentiality and authentication
 - ✍ Encrypt by sender's, then receiver's public key
 - ✍ But too time-consuming: 4 rounds RSA on large data

Message Authentication Code

- ✍ Assume both uses share secret key k
- ✍ Procedure
 - ✍ Sender computes $MAC=C_k(M)$ for M
 - ✍ Sent M and MAC of it to receiver
 - ✍ Receiver computes the MAC on received M
 - ✍ Compare it with received MAC
 - ✍ If match, then accepts the message
- ✍ MAC is similar to encryption, but not need be reversible!

MAC with Confidentiality

✍ Two options







- ✍ Using another key to encrypt M and MAC
- ✍ Using another key to encrypt M only

✍ Requirements of MAC

- ✍ Size of MAC: n
- ✍ Size of key: k
- ✍ Need 2^n computations of MAC and n/k pairs of M_i and MAC_i

Why not Conventional Encrypt

Possible situations

-  Broadcast a message (one destination can verify)
-  Authentication is done selectively
-  Authentication of computer program
-  Authentication may be important than secrecy
-  Architecture flexibility
-  Authentication lasts longer than secret protection

MAC Requirements

- ✍ Computationally infeasible to construct M' such that $C_k(M') = C_k(M)$
- ✍ $C_k(M)$ uniformly distributed

Data Authentication Algorithm

- ✍ ANSI standard X9.17
- ✍ Based on DES
- ✍ Using Cipher Block Chaining mode
 - ✍ Data is grouped into 64 bits blocks
 - ✍ Padding 0's if necessary
 - ✍ $\text{Output}_i = E_k(D_i \oplus \text{Output}_{i-1})$
 - ✍ $0 < i$, and $\text{Output}_0 = 0$'s
 - ✍ The data authentication code DAC consists of the leftmost m bits of the last output, $m = 16$

Hash Function

✍ Map a message to a smaller value

✍ Requirements

✍ Be applied to a block of data of any size

✍ Produced a fixed length output

✍ $H(x)$ is easy to compute (by hardware, software)

✍ **One-way:** given code h , it is computationally infeasible to find x : $H(x)=h$

✍ **Weak collision resistance:** given x , computationally infeasible to find y so $H(x)=H(y)$

✍ **Strong collision resistance:** Computationally infeasible to find x, y so $H(x)=H(y)$

Basic Uses of Hash Function

Six basics usages

- ✍ $E_k(M||H(M))$
 - ✍ Confidentiality and authentication
- ✍ $M||E_k(H(M))$
 - ✍ Authentication
- ✍ $M||E_{KR_a}(H(M))$
 - ✍ Authentication and digital signature
- ✍ $E_k(M||E_{KR_a}(H(M)))$
 - ✍ Authentication, digital signature and confidentiality
- ✍ $M||H(M||S)$
 - ✍ Authentication (S shared by both sides)
- ✍ $E_k(M||H(M||S))$
 - ✍ Confidentiality and authentication

Birthday Attacks

- ✍ If 64-bits hash code is used
 - ✍ On average, how many messages need to try to find one match the intercepted hash code?
- ✍ Birthday paradox
 - ✍ A will sign a message appended with m -bits hash code
 - ✍ Attacker generates some variations of fraud message, also variations of good message
 - ✍ Find pair of message each from the two sets messages
 - ✍ Such that they have the same hash code
 - ✍ Give good message to A to get signature
 - ✍ Replace good message with fraud message

Analysis

- ✍ Using birthday attack, given 64-bits hash code
 - ✍ How many message variations needed so the success probability is large, say 90%?

Examples

✍ Simple hash functions

✍ XOR of the input message

$$\text{✍ } H(M) = X_1 \oplus X_2 \oplus \dots \oplus X_{m-1} \oplus X_m$$

✍ But not secure

✍ $Y_m = H(M) \oplus Y_1 \oplus Y_2 \oplus \dots \oplus Y_{m-1}$ has same hash value as $(X_1 X_2 \dots X_{m-1} X_m)$, where Y_i is any value

Cont.

- ✍ Based on DES, block chaining technique
 - ✍ Rabin, 1978
 - ✍ Divide message M into fix-sized blocks M_i
 - ✍ Assume total n data blocks
 - ✍ H_0 =initial value
 - ✍ $H_i = E_{m_i}[H_{i-1}]$
 - ✍ H_n is the hash value
- ✍ Birthday attack still applies
 - ✍ If still 64-bits code used

More Attacks





- ✍ Birthday attack applied if chosen plaintext
- ✍ Meet in the middle attack if known plaintext
 - ✍ Known signed hash code G
 - ✍ Construct $n-2$ desired message block Q_i
 - ✍ Compute $H_i = E_{Q_i}[H_{i-1}]$
 - ✍ Generate $2^{m/2}$ random blocks X
 - ✍ For each X , Compute $H_{n-1} = E_X[H_{n-2}]$
 - ✍ Generate $2^{m/2}$ random blocks Y
 - ✍ For each Y , Compute $H'_{n-1} = D_Y[G]$
 - ✍ Find X, Y such that $H_{n-1} = H'_{n-1}$
 - ✍ Then $Q_1, Q_2, \dots, Q_{n-2}, X, Y$ is a fraud message

Security

- ✍ The size of hash code determines security
 - ✍ 128bits is not secure
 - ✍ Currently, most use 160 bits hash code
- ✍ Attack MAC
 - ✍ Object find valid $(x, C_k(x))$ pair
 - ✍ Attack the key space: roughly 2^k , k =key size
 - ✍ Attack the MAC value

More Hash Algorithms

Algorithms



-  Message Digest:MD5 (was mostly widely used)
-  Secure Hash Algorithm: SHA-1 (from MD4)
-  RIPEMD-160
-  HMAC

Digital Signatures

Authentication

-  Protects two parties from the third party
-  But not protect against each other

Digital signature

-  Verification of the message source
-  Protects the authority from anyone

Requirements

✍ Requirement lists

- ✍ Signature depends on the message
- ✍ Signature uses information unique to the signer
- ✍ Relatively easy to sign
- ✍ Relatively easy to recognize and verify it
- ✍ Computationally infeasible to forge signature
 - ✍ New messages using old signature
 - ✍ Create signature for a given message
- ✍ Practical to retain a copy of the signature for later verification

Digital Signature Standard

- ✍ FIPS PUB 186 by NIST
- ✍ It uses
 - ✍ Secure Hashing Algorithm (SHA) for hashing
 - ✍ Digital Signature Algorithm (DSA) for signature
 - ✍ The hash code is set as input of DSA
 - ✍ The signature consists of two numbers
- ✍ DSA
 - ✍ Based on the difficulty of discrete logarithm
 - ✍ Based on Elgamal and Schnorr system

DSA

✍ Global public components

✍ Prime number p with 512-1024 bits

✍ Prime divisor q of $(p-1)$ with 160 bits

✍ Integer $g = h^{(p-1)/q} \bmod p$

✍ Users private key





✍ Random integer x less than q

✍ Users public key




✍ Integer $y = g^x \bmod p$

DSA

Signature

-  For each message M , generates random k
-  Computes $r = (g^k \bmod p) \bmod q$
-  Computes $s = k^{-1}(H(M) + xr) \bmod q$
-  Signature is (r, s)

Verifying

-  Computes $w = s^{-1} \bmod q$, $u_1 = H(M)w \bmod q$
-  Computes $u_2 = rw \bmod q$, $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$
-  Test if $v = r$

Proof of Correctness

✍ Notice that $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$

$$\begin{aligned} \cancel{\text{✍}} &= (g^{H(M)w} \bmod q \cdot y^{rw} \bmod q \bmod p) \bmod q \\ \cancel{\text{✍}} &= (g^{H(M)w} \bmod q \cdot g^{xrw} \bmod q \bmod p) \bmod q \\ \cancel{\text{✍}} &= (g^{H(M)w + xrw} \bmod q \bmod p) \bmod q \\ \cancel{\text{✍}} &= (g^{(H(M)+xr)w} \bmod q \bmod p) \bmod q \\ \cancel{\text{✍}} &= (g^{(H(M)+xr)k} (H(M)+xr)^{-1} \bmod q \bmod p) \bmod q \\ \cancel{\text{✍}} &= (g^k \bmod p) \bmod q \\ \cancel{\text{✍}} &= r \end{aligned}$$

ElGamal Signature

- ✍ Global public components

- ✍ Prime number p with 512-1024 bits

- ✍ Primitive element g in Z_p

- ✍ Users private key





- ✍ Random integer x less than p

- ✍ Users public key




- ✍ Integer $y = g^x \bmod p$

Elgamal

Signature

-  For each message M , generates random k
-  Computes $r = g^k \bmod p$
-  Computes $s = k^{-1}(H(M) - xr) \bmod (p-1)$
-  Signature is (r, s)

Verifying

-  Computes $v_1 = g^{H(M)} \bmod p$
-  Computes $v_2 = y^r r^s \bmod p$
-  Test if $v_1 = v_2$

Proof of Correctness

✍ Computes $v_2 = y^r r^s \bmod q$

✍ So $v_2 = y^r r^s \bmod q = g^{xr} g^{ks} \bmod p$

✍ $= g^{xr + k^{-1}(H(M) - xr) \bmod (p-1)} \bmod p$

✍ $= g^{H(M)} \bmod p = v_1$

✍ Notice that here it uses Fermat theorem to show

✍ That $g^{(H(M) - xr) \bmod (p-1)} \bmod p = g^{(H(M) - xr)} \bmod p$

Non-deterministic

✍ Non-determined signatures

✍ For each message, many valid signatures exist

✍ DSA, Elgamal








✍ Deterministic signatures

✍ For each message, one valid signature exists

✍ RSA

Comparisons

Speed


-  DSS has faster signing than verifying
-  RSA could have faster verifying than signing
-  Message be signed once, but verified many times
 -  This prefers the faster verification
-  But the signer may have limited computing power
 -  Example: smart card
 -  This prefers the faster signing

Authentication Protocols

Central issues

-  Confidentiality: prevent masqueraded and compromised

-  Timeliness: prevent replay attacks





 -  Simple replay, repetition within timestamp, replay arrives but not the true messages, backward replay attack to the sender

Mutual authentication

One-way authentication

Coping with Replay

Time stamps

-  Party A accepts a message only if has valid timestamp within a valid time
-  Need synchronized clock
-  How to set the synchronized clock?
 -  Network delay consideration?

Challenge/response

-  Party A, (receiver), sends B a nonce (challenge) and requires the subsequent message contains it

Challenge-Response

- ✍ To ensure a password is never sent in the clear.
Given a client and a server share a key
 - ✍ server sends a random challenge vector
 - ✍ client encrypts it with private key and returns this
 - ✍ server verifies response with copy of private key
 - ✍ can repeat protocol in other direction to authenticate server to client (2-way authentication)
- ✍ Secret key management
 - ✍ physically distributed before secure communications
 - ✍ keys are stored in a central trusted key server

Conventional Encryption App.

- ✍ Each user shares a secret master key with KDC (Key Distribution Center)
 - ✍ Kerberos is an example
 - ✍ Needham-Schroeder protocol
 - ✍ Party A ✍ KDC $Ida|Idb|Na$
 - ✍ KDC ✍ A $E_{ka}(Ks|Idb|Na|E_{kb}(Ks|Ida))$
 - ✍ A ✍ B $E_{kb}(Ks|Ida)$
 - ✍ B ✍ A $E_{ks}(Nb)$
 - ✍ A ✍ B $E_{ks}(f(Nb))$

Weakness

- ✍ Step 4 and 5 prevent the replay of step 3
 - ✍ Assume that K_s is not compromised
- ✍ If K_s is compromised
 - ✍ Vulnerable to replay attack
 - ✍ Attacker can replay step 3
 - ✍ Unless B remembers all previous session keys with A, it can not tell that it is a replay!

Denning Protocol

✍ Denning Protocol

✍ Party A ✍ KDC $Ida|Idb$

✍ KDC ✍ A $E_{ka}(Ks|Idb|T|E_{kb}(Ks|Ida|T))$

✍ A ✍ B $E_{kb}(Ks|Ida|T)$

✍ B ✍ A $E_{ks}(Nb)$

✍ A ✍ B $E_{ks}(f(Nb))$

✍ Here T is timestamp assures the freshness of the key Ks

✍ Rely on synchronized clock

Public-key Encryption App.

✍ The simple one proposed by Denning

✍ AS: authentication server

✍ A ✍ AS $I_{da}|I_{db}$

✍ AS ✍ A $E_{kr_{as}}(KU_a|I_{da}|T)|E_{kr_{as}}(K_{ub}|I_{db}|T)$

✍ A ✍ B $E_{kr_{as}}(KU_a|I_{da}|T)|E_{kr_{as}}(K_{ub}|I_{db}|T)|$

✍ $E_{k_{ub}}(E_{k_{ra}}(K_s|T))$

✍ It needs clock synchronization

Cont.

✍ Protocol by Woo and Lam, using nonce

✍ A ✍ KDC $Ida|Idb$

✍ KDC ✍ A $E_{KRau}(Idb|KU_b)$

✍ A ✍ B $E_{KU_b}(Na|Ida)$

✍ B ✍ KDC $Idb|Ida|E_{KUau}(Na)$

✍ KDC ✍ B $E_{KRau}(Ida|KU_a)|E_{KU_b}(E_{kRau}(Na|Ks|Ida|Idb))$

✍ B ✍ A $E_{KU_a}(E_{kRau}(Na|Ks|Ida|Idb) | Nb)$

✍ A ✍ B $E_{ks}(Nb)$

One-way Authentication

✍ Using Public Key approach

✍ If confidentiality is main concern

$$\text{✍ A} \rightarrow \text{✍ B: } E_{K_{Ub}}(K_s) \mid E_{K_s}(M)$$

✍ If authentication is main concern

$$\text{✍ A} \rightarrow \text{✍ B: } M \mid E_{K_{Ra}}(H(M))$$

✍ This can not avoid the interception and replay attack

✍ Sign the message then

$$\text{✍ } E_{K_{Ub}}(M \mid E_{K_{Ra}}(H(M)))$$

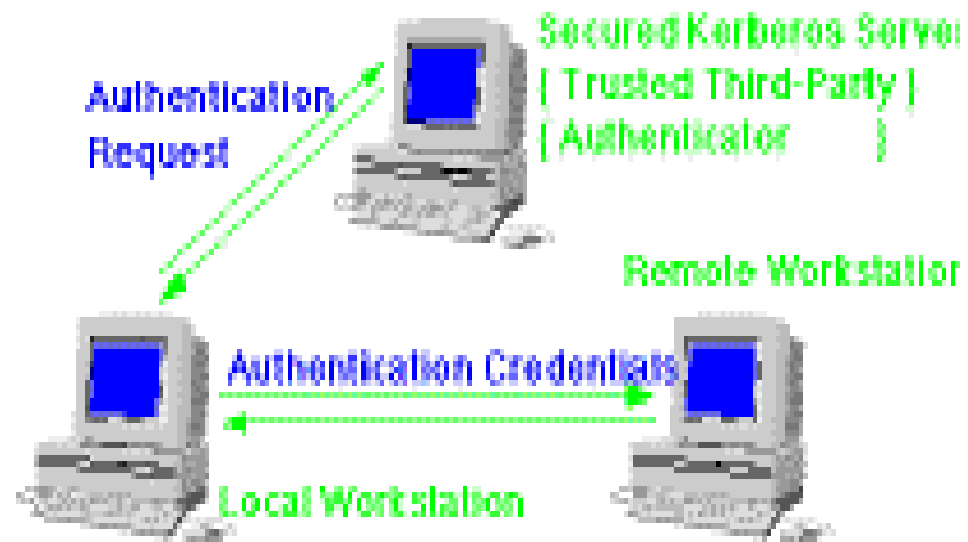
$$\text{✍ Or } E_{K_{Ub}}(K_s) \mid E_{K_s}(M \mid E_{K_{Ra}}(H(M)))$$

✍ Also A can send the digital certificate $E_{K_{Rau}}(T \mid I_d \mid K_{Ua})$

Kerberos

- ✍ Trusted key server system developed by MIT
 - ✍ Provides centralized third-party authentication in a distributed network
 - ✍ access control may be provided for
 - ✍ each computing resource
 - ✍ in either a local or remote network (realm)
 - ✍ A Key Distribution Centre (KDC), containing database:
 - ✍ principles (customers and services)
 - ✍ encryption keys
 - ✍ KDC provides non-corruptible authentication credentials (tickets or tokens)

Basic Model



Kerberos

✍ Initial User Authentication

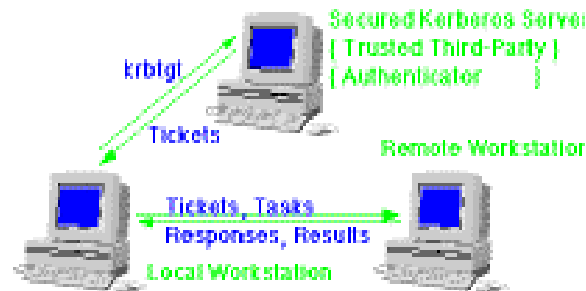
- ✍ User requests an initial ticket from KDC used as basis for all remote access requests



Kerberos




✍ Request for a Remote Service

- ✍ User requests access to a remote service
- ✍ Obtains a ticket from KDC protected with remote key
- ✍ Sends ticket with request to remote server





Kerberos



Two Kerberos versions

-  4 : restricted to a single realm
-  5 : allows inter-realm authentication, in beta test
-  Kerberos v5 is an Internet standard specified in RFC1510

To use Kerberos

-  need to have a KDC on your network
-  need to have Kerberised applications running on all participating systems

US export restrictions

-  Cannot be directly distributed outside US in source format
-  Crypto libraries must be re-implemented locally

X.509 Authentication Service




- ✍ Public key certificate associated with user
 - ✍ The certificates are created by Trusted Authority
 - ✍ Then placed in the directory by TA or user
 - ✍ Itself is not responsible for creating certificate
 - ✍ It includes
 - ✍ Version, serial number, signature algorithm identifier, Issuer name, issuer identifier, validity period, the user, user identifier, user's public key, extensions, signature by TA
 - ✍ The signature by TA guarantees the authority
 - ✍ Certificates can be used to certify other TAs
 - ✍ $Y\langle\langle X \rangle\rangle$: certificate of user X issued by TA Y

Certificate Revocation




- ✍ Need the private key together with the certificate to revoke it
- ✍ The revocation is recorded at the directory
- ✍ Each time a certificate is arrived, check the directory to see if it is revoked

Identification

Identification: user authentication

-  convince system of your identity
-  before it can act on your behalf
-  sometimes also require that the computer verify its identity with the user

Based on three methods

-  what you know
-  what you have
-  what you are

Verification

-  Validation of information supplied against a table of possible values based on users claimed identity

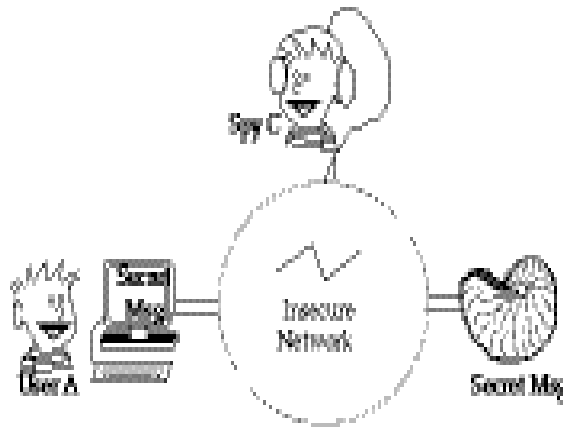
What you Know

✍ Passwords or Pass-phrases

- ✍ prompt user for a login name and password
- ✍ verify identity by checking that password is correct
- ✍ on some (older) systems, password was stored clear
- ✍ more often use a one-way function, whose output cannot easily be used to find the input value
- ✍ either takes a fixed sized input (eg 8 chars)
- ✍ or based on a hash function to accept a variable sized input to create the value
- ✍ important that passwords are selected with care to reduce risk of exhaustive search

Weakness

- ✍ Traditional password scheme is vulnerable to eavesdropping over an insecure network



Solutions?

✍ One-time password

- ✍ these are passwords used once only
- ✍ future values cannot be predicted from older values

✍ Password generation

- ✍ either generate a printed list, and keep matching list on system to be accessed
- ✍ or use an algorithm based on a one-way function f (eg MD5) to generate previous values in series (eg SKey)
 - ✍ start with a secret password s , and number N , $p_0 = f^N(s)$
 - ✍ i th password in series is $p_i = f^{N-i}(s)$
- ✍ must reset password after N uses

What you Have

- ✍ Magnetic Card, Magnetic Key
 - ✍ possess item with required code value encoded
- ✍ Smart Card or Calculator
 - ✍ may interact with system
 - ✍ may require information from user
 - ✍ could be used to actively calculate:
 - ✍ a time dependent password
 - ✍ a one-shot password
 - ✍ a challenge-response verification
 - ✍ public-key based verification

What you Are

- ✍ Verify identity based on your physical characteristics, known as biometrics
- ✍ Characteristics used include:
 - ✍ Signature (usually dynamic)
 - ✍ Fingerprint, hand geometry
 - ✍ face or body profile
 - ✍ Speech, retina pattern
- ✍ Tradeoff between
 - ✍ false rejection (type I error)
 - ✍ false acceptance (type II error)