Module 3 – Key Exchange, Identity and Authentication

Nasir Memon – Polytechnic University
Course Issues

- HW 3 assigned.
- Next class on Monday – Oct 10.
- HW will maintain weekly schedule. Thursday – Thursday (roughly). Will get back into sync a few weeks later.
- Read chapter 9, 11 and 13.
- Any lab or course issues?
- Midterm November 3.
Key Exchange

- Cryptographic primitives seen so far assume
  - Alice and Bob share a secret key which is unknown to Oscar.
  - Alice has a “trusted” copy of Bob’s public key.
- But how does this happen in the first place?!!
- Alice and Bob meet and exchange key.
- Not always practical or possible.
- We need key exchange protocols!
- There are key distribution protocols and key agreement protocols.
Session Key Exchange With KDC.

- Protocol assumes that Alice and Bob share a session key $K_A$ and $K_B$ with a Key Distribution Center (KDC).
  - Alice calls Trent (Trusted KDC) and requests a session key to communicate with Bob.
  - Trent generates random session key $K$ and sends $E_{K_A}(K)$ to Alice and $E_{K_B}(K)$ to Bob.
  - Alice and Bob decrypt with $K_A$ and $K_B$ respectively to get $K$.
- This is a key distribution protocol.
- Susceptible to replay attack!
Session Key Exchange With KDC.

- **A -> KDC**  \( ID_A \ || \ ID_B \ || \ N_1 \)
  
  (Hello, I am Alice, I want to talk to Bob, I need a session Key and here is a random nonce identifying this request)

- **KDC -> A**  \( E_{K_A}(K \ || \ ID_B \ || \ N_1 \ || \ E_{K_B}(K \ || \ ID_A)) \)
  
  Encrypted(Here is a key, for you to talk to Bob as per your request \( N_1 \) and also an envelope to Bob containing the same key)

- **A -> B**  \( E_{K_B}(K \ || \ ID_A) \)
  
  (I would like to talk using key in envelope sent by KDC)

- **B -> A**  \( E_K(N_2) \)
  
  (OK Alice, But can you prove to me that you are indeed Alice and know the key?)

- **A -> A**  \( E_K(f(N_2)) \)
  
  (Sure I can!)

- Last two steps - challenge-response. Commonly used to thwart replay attack.
Session Key Exchange With Public Keys

- Alice gets Bob’s public key from KDC.
- Alice generates a random session key, encrypts with Bob’s public key and sends to Bob.
- Bob decrypts using his private key to get session key.
- Alice and Bob exchange a challenge-response.

Above is still susceptible to man-in-the-middle attack.
Man-in-the-middle Attack

- Alice send request to KDC. Mallory intercepts and sends his own public key.
- Alice generates random session key and encrypts using Mallory’s (she thinks Bob’s) public key and send to Bob.
- Mallory intercepts session key, decrypts, then encrypts with Bob’s public key and send to Bob.
- Bob decrypts session key.
- Alice and Bob use a session key to communicate that Bob knows!
Diffie-Hellman Key Exchange

- Protocol for exchanging secret key over public channel.
  - Select global parameters $n$ and $g$. $n$ is prime and $g$ is a \textit{primitive root} in $\mathbb{Z}_n$. These parameters are public and known to all.
  - Alice privately selects random $a$ and sends to Bob $g^a \mod n$.
  - Bob privately selects random $b$ and sends to Alice $g^b \mod n$.
  - Alice and Bob privately compute $g^{ab}$ which is their shared secret.
  - An observer Oscar can only compute $g^a \mod n \cdot g^b \mod n = g^{a+b} \mod n$. To compute $g^{ab}$ he needs to know either $a$ or $b$ or solve the discrete log problem.
- This is a key agreement protocol.
The Discrete Log Problem

- Given \( y \) and \( a \) in \( \mathbb{Z}_p \) where \( p \) is prime, find the unique \( x \) in \( \mathbb{Z}_p \), such that \( y = a^x \mod p \).

- For example given 2 and 5 in \( \mathbb{Z}_7 \), find the unique \( x \) such that \( 5^x = 2 \mod 7 \).
  - \( 5^1 = 5, \ 5^2 = 4, \ 5^3 = 6, \ 5^4 = 2, \ 5^5 = 3, \ 5^6 = 1 \)

- Now, given 949 and 2 in \( \mathbb{Z}_{2579} \), find the unique \( x \) such that \( 2^x = 949 \mod 2579 \)!!

- \( 2^{765} = 949 \mod 2579 \). Check with bc.

- No efficient algorithm known – NP-Hard.

- Note, \( a \) has to be a *primitive root*.
  - \( 2^1 = 2, \ 2^2 = 4, \ 2^3 = 1, \ 2^4 = 2, \ 2^5 = 4, \ 2^6 = 1 \)
Man-in-the-middle again!

- Diffie-Hellman is susceptible to man-in-the-middle attack.
  - Mallory captures a and b in transmission and replaces with own a’ and b’.
  - Essentially runs two Diffie-Hellman’s. One with Alice and one with Bob.
- Possible as Alice is not able to identify and authenticate who she is talking to.
Identity and Authentication

- What is identity?
  A computer’s representation of an unique entity (principal).

- What is authentication?
  Binding principal to system’s internal representation of identity.

- Why do we need identity?
  - Accountability
  - Access control
Identity for Files and Objects

- Files and other objects identified by “names”
  - File name – humans use.
  - File descriptor or handle – process use.
  - File allocation table entry – kernel use.

- Example
  - Unix – inodes, file descriptors, relative and absolute path names.
  - URL’s – Uniform Resource Locator.
Identity for Users

- User identity (UID)
  - Identity of a single entity
  - System represents user identity in different ways.
  - Not always human/physical entity.

- Example – Unix
  - Login name
  - UID - integer. UID 0 is root.
  - Logging done using login name.

- Same principal may have different identities.
  - Real and effective UID in UNIX. Used by SUID programs.
  - Saved UID – Free BSD and Solaris.
  - Audit or login UID – set at login and never changed. Allows one to track the original UID of a process.
Groups and Roles

- Users may need to share resources.
- Groups allow assignment of rights to multiple principals simultaneously.
- Group identity is static or can change.
- Example – UNIX
  - Each user assigned to one or more groups.
  - Each process has user id and group id.
- *Role* is a type of group that ties membership to function.
  - *Sysadmin role*, *Backup role*, *webmaster role* etc.
  - Allows finer grained control over access rights.
  - Mimics organizational structure of an enterprise.
Identity on the internet

- Host Identity
  - Related to network. Each network “layer” may use a different name. All names point to the same host but within different context.
  - Hostname, IP address, Ethernet (MAC) address.
  - Databases contain mappings between different names.
  - Can be spoofed. Mapping mechanism may not be secure.

- Static and dynamic identifiers.
  - DHCP, NAT. Local identifiers and global identifiers.
Naming and Certificates

- Certification authority’s vouch for the identity of an entity - *Distinguished Names (DN)*.
  
  `/O=Polytechnic University/OU=CS/CN=John Doe`
  
  - Although CN may be same, DN is different.

- Policies of certification
  
  - *Authentication policy*
    What level of authentication is required to identify the principal.
  
  - *Issuance policy*
    Given the identity of principal will the CA issue a certificate?
Types of Certificates

- CA’s vouch at some level the identity of the principal.

- Example - Verisign:
  - Class 1 - Email address
  - Class 2 - Name and address verified through database.
  - Class 3 - Background check.
Public Key Certificate

- Public Key Certificate - Signed messages specifying a name (identity) and the corresponding public key.
- Signed by whom - Certification Authority (CA), an organization that issues public key certificates.
- We assume that everyone is in possession of a trusted copy of the CA’s public key.
- CA could be
  - Internal CA.
  - Outsourced CA.
  - Trusted Third-Party CA.
Public Key Certificate

Unsigned certificate:
contains user ID, user's public key

Generate hash code of unsigned certificate

Encrypt hash code with CA's private key to form signature

Signed certificate:
Recipient can verify signature using CA's public key.

Note: Mechanism of certification and content of certificate, will vary but at the minimum we have email verification and contains ID and Public Key.
Certificate Revocation

- CA also needs some mechanism to *revoke* certificates
  - Private key compromised.
  - CA mistake in issuing certificate.
  - Particular service the certificate grants access to may no longer exist.
  - CA compromised.

- Expiration time solves the problems only partially.
- Certification Revocation Lists (CRL) – a list of every certificate that has been revoked but not expired.
  - CRL’s quickly grow large!
  - CRL’s distributed periodically.
  - What about time period between revocation and distribution of CRL?
Advantages of CA Over KDC

- CA does not need to be on-line!
- CA can be very simple computing device.
- If CA crashes, life goes on (except CRL).
- Certificates can be stored in an insecure manner!!
- Compromised CA cannot decrypt messages.
- Scales well.
How does Alice talk to Bob?

- She obtains Bob’s certificate signed by KGB-KDC.
- She obtains KGB-KDC’s certificate signed by CIA-KDC.
- Concept can be generalized to multiple CA’s.
- Helps if they are organized in a hierarchy.
X.509

- Clearly, there is a need for standardization – X.509.
- Originally 1988, revised 93 and 95.
- X.509 is part of X.500 series that defines a directory service.
- Defines a framework for authentication services by X.500 directory to its users.
- Used in S/MIME, IPSEC, SSL, SET etc.
- Does not dictate use of specific algorithm (recommends RSA).
X.509 Certificate

(a) X.509 Certificate

(b) Certificate Revocation List
X.509 CA Hierarchy - Example.

Y<<X>> means the certificate of user X issued by CA Y.

To talk to B, A obtains the following chain:
X<<W>>
W<<V>>
V<<Y>>
Y<<Z>>
Z<<B>>

Simpler if X has
X<<Z>>
X.509 Authentication – One-way.

1. $A\{t_A, r_A, B, \text{sgnData}, E_{KUb}[K_{ab}]\}$

- Establishes the following
  - Identity of $A$ and message was generated by $A$
  - Message was intended for $B$
  - Integrity and originality of message.
X.509 Authentication – Two-way.

- One-Way plus the above which establishes the following:
  - Identity of B and message was generated by B
  - Message was intended for A
  - Integrity and originality of message.
X.509 Authentication – Three-way.

1. $A\{t_A, r_A, B, \text{sgnData}, E_{KUb}[K_{ab}]\}$

2. $B\{t_B, r_B, A, r_B, \text{sgnData}, E_{KUa}[K_{ba}]\}$

3. $A\{r_B\}$

- Nonce’s echoed back and forth to prevent replay attacks.
- Needed when synchronized clock is not available.
Internet Certificate Hierarchy

- Internet Policy Registration Authority
  - Policy Certification Authorities
    - Certification Authority
      - Individuals/roles/orgs.
Types of certificates

- **Organizational Certificates**
  Principal’s affiliation with an organization

- **Residential certificates**
  Principal’s affiliation with an address

- **Persona Certificates**
  Principal’s Identity

- Principal need not be a person. It could be a role.

- Is *anonymous certificate* an oxymoron?
  - They do get issued!
Public-key Infrastructure (PKI)

- Combination of digital certificates, public-key cryptography, and certificate authorities.
- A typical enterprise's PKI encompasses
  - issuance of digital certificates to users and servers
  - end-user enrollment software
  - integration with corporate certificate directories
  - tools for managing, renewing, and revoking certificates; and related services and support
- Verisign, Thawte and Entrust – PKI providers.
- Your own PKI using Netscape/Microsoft certificate servers
Problems with PKI – Private Key

- Where and how is private key stored?
  - Host – encrypted with pass phrase
  - Host – encrypted by OS or application
  - Smart Card
  - ...

- Assumes secure host or tamper proof smartcard.
Problems with PKI - Conflicts

- X.509, PGP and IPRA remain silent on conflicts.
- They assume CA’s and PCA’s will ensure that no conflicts arise.
- But in practice conflicts may exist –
  - John A. Smith and John B. Smith may live at the same address.
Trustworthiness of Issuer

- A certificate is the binding of an external identity to a cryptographic key and a distinguished name. If the issuer can be fooled, all who rely upon the certificate can be fooled 😞

- How do you trust CA from country XYZ (your favorite prejudice).
Ten Risks of PKI – Ellison and Schneier

- Who do we trust, and for what?
- Who is using my key?
- How secure is the verifying computer?
- Which John Robinson is he?
- Is the CA an authority?
- Is the user part of the security design?
- Was it CA or CA plus Registration Authority?
- How did the CA identify the certificate holder?
- How secure are the certificate practices?
- Why are we using the CA process, anyway?
Subject Authentication

- “Authentication is binding of an identity to a subject.”

- What information can be used to establish this binding
  - What the subject knows (secrets)
  - What the entity has (smart cards)
  - What the entity is (retinal prints)
  - Where the entity is (current location)

- Our focus is on subjects’ secrets!
Password based authentication

- Simple and natural mechanism. Widely used.
- How do we store passwords?
  - Encrypted password file
  - Fixed messages encrypted with passwords
  - Hash of password
- First is not a good idea
- Second and third widely used.
Unix login and user accounts

- Users identified by *usernames* and authenticated by *passwords*
- Passwords used as key to encrypt the all zero word using *crypt* which uses a modified version of DES repeated 25 times. Result stored in `/etc/passwd` file
- All authorized users have an entry in the password file
- User name usually 8 characters and represented internally by 16 bit number – UID
The /etc/passwd file

- An entry looks as follows:
  jdoe:987hggw2fd435:103:1001:John Doe: /home/jdoe:/bin/csh
- The fields are (in order): Username, User’s “encrypted password”, UID, GID, User information (GECOS) field, User’s home directory and User’s shell.
- “Invalid Entry” (usually ‘*’, in password field indicates non-login account.
- In a distributed environment, passwd file entry may be somewhere on the “network”.

A sample /etc/passwd file

root:3rztyHOkz3ZQc:0:0:root:/root:/bin/bash (if no shadow)
root:*:0:0:root:/root:/bin/bash  (if passwords shadowed)

bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
amd:*:3:4:adm:/var/adm:
mail:*:8:12:mail:/var/spool/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
ftp:*:14:50:FTP User:/home/ftp:
memon:*:500:500:Nasir Memon:/home/memon:/bin/bash
yuhong:*:502:502:yuhong yu:/home/yuhong:/bin/bash
sachin:*:503:503:Sachin Tendulkar:/home/sachin:/bin/bash
The “password encryption” algorithm

- **Salt**: 12 bits
- **Password**: 56 bits
- **User ID**: 11 characters

Password File:
- User ID
- Salt
- $E_{pwd}[salt, 0]$

Diagram:
- Salt (12 bits) and password (56 bits) are loaded into a crypt (3) function.
- The output is stored in the Password File.
First eight characters password used as the DES key to encrypt constant 64-bit block (consisting of all zero bits) via DES 25 times.

The result of each encryption is used to feed the next round.

The resultant 64-bits is converted into a string of 11 printable ASCII characters by encoding every six bits into a printable ASCII character and zero padding the 11th character.
The role of Salt

- The salt is used to perturb the $E$ expansion block.
- When a password is first selected, the password encryption program selects a random 12-bit number as the salt.
- Salt and the result is stored in password file.
- Later on, when the user attempts to login the salt is extracted from the password file and is used.
- The effect of salting is to allow for 4096 possible encryptions of same password string.
- With Salt two users may have the same password but the “encrypted” versions would look different.
Dictionary attack

- To find a user’s password from the encrypted password -
  - Searching through all possible keys? No!
  - Search through all passwords is better!

- Password can be found by encrypting dictionary of possible passwords with all possible salts and comparing the result.

- Total number of passwords can be large - $70^8$
- Set of typical passwords is surprisingly small!
Improvements

- Keep encryption algorithm secret
  - Security by obscurity is always a bad idea.
- Keep password file hidden from non-privileged users
  - Widely used. Shadow file in Unix.
- Slow down password encryption algorithm
  - Unix also runs on 386 based Linux box.
- Enforce better passwords.
- Other mechanisms?
  - One-time passwords, Biometrics, Smart cards, etc.
Rules for good passwords

- Have both upper case and lower case letters
- Have digits and punctuation characters as well as letters
- May include some control characters and/or spaces
- Are easy to remember so that they do not have to be written down
- Are close to the maximum allowed length (at least seven or eight characters long).
NT/W2K/WXP Passwords

- Passwords stored in encrypted form in SAM database which is part of the registry.
- SAM database is world readable but always in use, so effectively locked.
- There are actually two encrypted passwords
  - LAN Manager password
  - NT password
LAN Manager Hash

- Developed by Microsoft for OS/2

- Algorithm:
  - Truncate/Pad password to 14 characters
  - Upcase password
  - Split the password into two 7-byte halves
  - using DES, build \( \{c\}k1 + \{c\}k2 \), \( c \) a constant
Cracking NT passwords

- LANMAN (LM) Hash is weak and can be easily brute forced!
  - Breaking up into two pieces and encrypting them separately is a dumb idea!! You just have to crack one piece at a time.
  - Passwords are upper cased!!
  - In fact, there is nothing much gained by having a password longer than 7 characters!
  - LM hash can be disabled but Win95/98 only uses LM hash, so often not disabled.
- Make sure you use special characters in NT passwords.
NT hash

- Later systems (NT and 95) calculate $\text{MD4(unicode password)}$
- … but there’s still no salt
- and the password space is still small
- and most applications need to calculate both NT hash and LMHash for backwards-compatibility
How to get Password hashes?

- Registry
  - If you have administrator rights you can dump password hashes from your local machine or over the network if the remote machine allows network registry access.

- SAM File
  - Since OS holds a lock on the SAM file it is not possible to just read them while the OS is running. Sometimes a backup of this file is made on tape or on an Emergency Repair Disk or in the repair directory of the system hard drive. Also, another OS such as DOS can be booted from a floppy and the password hashes can be read directly from the file system!!

- SMB Packet Capture
  - You can capture encrypted hashes over the network!
Authenticating over a Network

- Previous techniques are for console/workstation login. What about logging in via network?
  - FTP, Telnet, Rlogin, PAP, send passwords to the remote machine. Can easily be sniffed. Use ssh!!
  - For NT the procedure is more involved but has a flawed part (LM hash!)

- What we need is a challenge response protocol that is resistant to brute force and also can resist active (For example, replay and man-in-the-middle) attacks.
NT Remote Login – Challenge-Response Mechanism

- User enters password, client calculates its 16 byte hash (LM and NT) and remembers for future use.
- When client connects to server, the server generates an 8-byte random value which it sends to the client.
- Client pads 16-byte hash with 5 null bytes to get 3 56-bit DES keys. 8-byte random challenge DES encrypted with each DES key, generating a 24-byte response.
- Server compares response to that generated by user's hash values from it's password database.
- Someone sniffing only sees 8-byte challenge and 24-byte response. To obtain original hash attacker must perform three brute force DES attacks Not!
- If LM hash sent, it can be easily brute forced! – How??
Defeating NT protocol with modified client.

- Hashed password (uncracked) is password equivalent!
- Client does not actually need to know the password. It only needs to know the hashed password! The regular client software has the end user input his/her password and calculates the hashed password from the manually entered original password.
- If you have a modified client which has access to a stolen copy of uncracked Windows/NT password database, client can look up hash value to use to calculate response. Client can get authenticated without knowing the original (clear text) password!
- This means that even a good password (dictionary attack resistant) can be misused!
One-Time Passwords

- Password is invalidated as soon as it’s used
- Two properties of one-time password scheme
  - Passwords should be randomly generated
  - Client and server must be synchronized
- Lamport suggested following scheme
  Generate: $h(k_0) = k_1$, $h(k_1) = k_2$, $h(k_2) = k_3$ ... $h(k_{n-1}) = k_n$
  Register $k_n$ with server.
  Use in this order: $p_1 = k_{n-1}$, $p_2 = k_{n-2}$ ... $p_{n-1} = k_1$
- S/Key and OPIE (One Time Passwords In Everything)
Kerberos

- An authentication service for distributed systems.
  - Client Workstations – Public places, untrusted software.
  - Server Machines – Moderately secure rooms, potentially untrusted software.
  - Key distribution machines (KDC’s) – Secure areas, trusted software.
- Developed at MIT – Project Athena.
- Current Version 5.
Kerberos - Goals

- Secure
  - Next slide.

- Reliable
  - Lack of availability of kerberos service means lack of availability of supported service.

- Transparent
  - Minimum modification to existing network applications.

- Scalable
  - Modular distributed architecture.
Kerberos – Security Goals

- No cleartext passwords over network.
- No cleartext passwords stored on servers.
- Minimum exposure of client and server keys.
- Compromises should only affect current session.
- Limited authentication time but reusable within that time.
- Require password only at login.
Kerberos - Assumptions

- Public key crypto is too expensive. DES is good enough.
- Global clock.
- There is a way to distribute authorization data.
  - Kerberos provides authentication and not authorization.
Kerberos Authentication (1)

Step 1
Joe to KDC

I would like to Talk to the File Server

Step 2
KDC

Session key for User

Session key for service
Kerberos Authentication (2)

Step 3
KDC

Box 1
Session Key for Joe
Dear Joe,
This key for File server
Locked With Joe’s key

Box 2
Session Key for File server
Dear File server,
This key for Use with Joe
Locked With File Server’s key

Step 4
KDC to Joe
Kerberos Authentication (3)

Step 5
Joe

Opened
Box 1

Dear Joe,
This key for
File server

Box 2

Session Key for
File server

Dear File server,
This key for
Use with Joe

Locked With File Server’s key

Step 6
Joe

Box 3

Dear File server,
The time is
3:40 pm

Locked With Session key

Box 2

Session Key for
File server

Dear File server,
This key for
Use with Joe

Locked With File Server’s key
Kerberos Authentication (4)

Step 7
Joe to File server

Step 8
File server

Unlocked
Box 3

Dear File server,
The time is 3:40 pm

Unlocked
Box 2

Dear File server,
This key for Use with Joe
Kerberos Authentication (5)

- For mutual authentication, file server can create box 4 with time stamp and encrypt with session key and send to Joe.
- Box 2 is called *ticket*.
- KDC issues ticket only after authenticating password.
- To avoid entering passwords every time access needed, KDC split into two – authenticating server and ticket granting server.
Kerberos– One Slide Overview

1. User logs on to workstation and requests service on host.

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.

3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.

4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.

5. Workstation sends ticket and authenticator to server.

6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.
Kerberos – Protecting Application Data.

- A by-product of the Kerberos authentication protocol is the exchange of the session key between the client and the server – this can be used for encryption.

- Kerberos defines two message types - *safe message* and *private message* to encapsulate data that must be protected, but the application is free to use another method.
Kerberos – Additional features and extensions

- **Features present in Version 5 include**
  - renewable and forwardable tickets
  - support for higher level authorization mechanisms.
  - support for multi-hop cross-realm authentication.

- **Extensions include**
  - One time passwords
  - Public Key Cryptography
  - Change password protocol
  - Realm mixed in with password prior to hash, enabling same password in multiple realms in more secure manner.
  - Pre-authentication
Kerberos - Limitations

- Every network service must be individually modified for use with Kerberos.
- Does not work well in time-sharing environment (Keeps keys in /tmp!!). One user per workstation.
- Requires secure Kerberos server.
- Requires continuously available server.
- Stores all passwords with single key.
- Does not protect against Trojan horses on clients.
Further Reading

- Unix password security
  [http://www.ja.net/CERT/Belgers/UNIX-password-security.html](http://www.ja.net/CERT/Belgers/UNIX-password-security.html)

- L0pht Heavy Industries [http://www.l0pht.com/](http://www.l0pht.com/) (NT password security and cracker)


Further Reading

- X.509 page
- Ten Risks of PKI -