CS530 Authentication
Bill Cheng

http://merlot.usc.edu/cs530-s10
Identification vs. Authentication

Identification
- associating an identity (or a claimed identity) with an individual, process, or request

Authentication
- verifying a claimed identity

Ex: user ID is identification, password is authentication
Basis for Authentication

Ideally
- who you are

Practically
- something you know
  - e.g., password
- something you have
  - e.g., smartcard, magnetic stripe card, passport, driver’s license
- something about you
  - e.g., face, hand, voice, fingerprint (i.e., biometrics)
  - sometimes mistakenly called things you are

Note: policy determines how and what to do
Something You Know

- **Password**
- **Algorithm**
  - e.g., encryption key derived from password
- **Issues**
  - someone else may learn it
    - find it, sniff it, trick you into providing it
    - Ex: e-mail from eBay or Paypal asking you to validate your password
  - other party must know how to check
    - keep in table
      - once this table is obtained, the attacker may use it to login to other systems
  - you must remember it (tend to use same password)
  - how stored and checked by verifier
Examples of Password Systems

Verifier knows password
- can one crack password one letter at a time (as often seen in movies)?
  - timing attacks (look at power consumptions, time between successive guesses)

Encrypted Password
- one way encryption
- Ex: UNIX
  - login name, UID, GID, encrypted password all stores in /etc/passwd
  - old systems make /etc/passwd globally readable
  - new systems move encrypted passwords to /etc/shadow
  - salt the password (12-bit salt) to protect against pre-computed dictionary attack
Examples of Password Systems (Cont...)

- Third Party Validation
  - Ex: Liberty Alliance
    - Microsoft Passport
    - Kerberos
  - Public key systems with Directory Services
Attacks on Password

- Brute force
- Dictionary
- Pre-computed Dictionary
- Guessing
  - what’s your pet’s name? (favorite city, birth place, ...)
- Finding elsewhere
  - sitting in Windows’ Registry
  - sitting on USB harddrive
Something You Have

Cards
- mag stripe (= password?)
- smart card, USB key
  - something your device knows!
  - verifier knows that the device is present!
- time varying password
  - secure ID card
  - challenge/response card
  - smartcard requires special reader, this does not
    the user is the device!
  - limited data length to reduce human mistakes

Issues
- how to validate
- how to read (i.e. infrastructure)
Something About You

Biometrics
- measures some physical attribute
  - iris scan (can’t really scan the retina)
  - fingerprint
  - picture
  - hand scan (geometry of hand)
  - voice
  - keystroke patterns?

Issues
- how to prevent spoofing
  - suited when biometric device is trusted/secure, not suited otherwise
- fingerprint reading device at home, is that a good idea?
  - must be connected to a tamper-proof device
Other Forms of Authentication

- IP address, MAC address
  - e.g., NFS, DHCP

- Caller ID (or call back)
  - also works with e-mail

- Past transaction information
  - e.g., what’s the amount of your last bill?
"Enrollment" (for Something You Know)

- How to initially exchange the secret
  - in-person enrollment
  - information known in advance
    - e.g., what’s the amount of your last bill?
  - third party verification
    - e.g., a notary public
  - mail or email verification
    - e.g., activation code in e-mail, click here to activate
Multi-factor Authentication

Require at least two of the three classes above
- e.g. Smart card plus PIN
- e.g. credit card plus zip code of billing address
- e.g. biometric and password

Issues
- better than one factor
- be careful about how the second factor is validated
  - E.g., on card, or on remote system
  - PIN goes to remote system (or goes through smartcard and then remote system)
General Problems with Password

- Space from which passwords are chosen
- Too many passwords
  - and what it leads to
  - solution is "single sign on"?
Single Sign On

"Users should log in once and have access to everything"

Many systems store password lists
- which are easily stolen

Better is encryption based credentials
- usable with multiple verifiers
- interoperability is complicating factor

Liberty Alliance
- communicating information about authentication using a markup language (Security Association Markup Language)

Microsoft Passport
- original version based on cookies and hotmail passwords
- next version based on Kerberos (cross realm authentication)
Encryption Based Authentication

- Proving knowledge of encryption key
  - nonce = non repeating value

\[ \text{Nonce/timestamp}\cdot K_{cs} \]
Authentication with Conventional Cryptography

Kerberos

C

KDC

S

\{K_{c,s}\}K_c
\{K_{c,s}\}K_s

{\text{data}}K_{c,s}, \{t\}K_{c,s}, \{K_{c,s}\}K_s

S

K_s

Copyright © William C. Cheng
Authentication with Conventional Cryptography

- Kerberos or Needham-Schroeder
  - includes challenge/response
  - optional pre-authenticator in original message

Diagram:
- KDC
- C
- S
- \(K_TGS\)
- \(s, \{t\}K_c\)
- \({K_c,s}K_c\)
- \({K_c,s}K_s\)
Kerberos

Third-party authentication service
- distributes session keys for authentication, confidentiality, and integrity
  - KDC & TGS is usually combined
  - KDC can generate cross realm TGT (pre-arranged)

KDC
{K_{c,TGS}}K_c
{K_{c,TGS}}K_{TGS}

TGS
K_{TGS}

C
K_c

S
K_s

{data}K_{c,s}, \{K_{c,s}\}K_s

Copyright © William C. Cheng
Authentication with Public Key Cryptography

- Based on public key certificates
  - DS = Directory Server
  - Client can include public key certificate in the first message
  - Contact DS mainly to check to see if the public key certificate has been revoked and to obtain other certificates

\[
\text{DS} = \{\text{Nonce/timestamp}\}K_{\text{ses}}, \{K_{\text{ses}}K_{\text{pub}}\}
\]
Public Key Cryptography Summary

Key distribution
- confidentiality not needed for public key
- solves $n^2$ problem

Performance
- slower than conventional cryptography
- implementations use for key distribution, then use conventional crypto for data encryption

Trusted third party still needed
- to issue public key certificates
- to obtain other public key certificates
- to manage revocation
- in some cases, third party may be off-line
Certificate-Based Authentication Summary

- Certification authorities issue signed certificates
  - banks, companies, & organizations like Verisign act as CA’s
  - certificates bind a public key to the name of a user
  - public key of CA certified by higher-level CA’s
  - root CA public keys configured in browsers & other software
  - certificates provide key distribution

- Authentication steps
  - verifier provides nonce, or a timestamp is used instead
  - principal selects session key and sends it to verifier with nonce, encrypted with principal’s private key and verifier’s public key, and possibly with principal’s certificate
  - verifier checks signature on nonce, and validates certificate
Authentication with Hash Chains

Based on the one-wayness of cryptographic hash functions

- generate secret $s$, send $h(s)$ to server
- to prove identity, present $s$ to server
- but now $s$ is exposed
Authentication with Hash Chains (Cont...)

Use Lamport’s hash (or hash chain)

- \( h^{100}(s) \leftarrow h^{99}(s) \leftarrow h^{98}(s) \leftarrow \ldots \leftarrow h^{2}(s) \leftarrow h(s) \leftarrow s \)
- client generate \( s \) (seed) and \( N \) and compute \( h^N(s) \)
  - sends \( N \) and \( h^N(s) \) to server
  - seed can be derived from a passphrase
- server keeps a state, start with \([N=100, h^N(s)]\)
- client sends name to server and server responds with \( N \)
  - client computes and sends \( x = h^{N-1}(s) \)
  - server computes \( h(x) \) and compare with current state
  - if succeed, new state is \([N-1, x] \)
- an attacker who has the server’s state cannot login
- this is one of the one-time password schemes
Authentication with Hash Chains (Cont...)

Man-in-the-middle *small N attack*
- man-in-the-middle attack intercepts $N$ from server and forward $N-10$ to client
- client sends $h^{N-11}(s)$ which the attacker will intercept
  - use this to compute $h^{N-1}(s)$
- attacker can login 10 times without knowing $s$

Mitigating the small N attack
- the client needs to remember the last $N$ received from this server
Authentication with Hash Chains (Cont...)

- Other weakness in Lamport’s hash
  - short lifetime of key
    - when \( N \) reaches 1, must generate new seed
    - can use a salt so that the seed can stay the same
      - client generate \( s \) (seed) and \( t \) (salt) and \( N \) and compute \( h^N(s+t) \)
      - sends \( N \) and \( t \) and \( h^N(s+t) \) to server
      - client can discard the salt
      - on client login, server responds with \( N \) and \( t \)

- problem with multiple servers
  - need different seeds
  - 3rd party authentication may not be desirable
  - salt also helps with logging to multiple servers with the same seed or passphrase
  - use a different salt per server
Trust Models for Certification

- X.509 hierarchical
  - OSI model:
    - X.400 - e-mail
    - X.500 - naming (DNS equivalent)
    - X.509 - authentication standard
  - single root (original plan) - UN is the root CA
  - multi-root (better accepted)
  - SET (Secured Electronic Transaction) has banks as CA’s and common SET root
  - private key of the SET root CA is split and spread among child CA’s
Trust Models for Certification (Cont...)

PGP Model
- "Friends and Family approach" - S. Kent
  - put more trust on more paranoid people
    as a result, look like a hierarchy!

Other representations for certifications
- X.509 (popular)

No certificates at all
- out of band key distribution
- SSH
  - ~/.ssh/authorized_keys
Global Authentication Service

- From DEC
- Pair-wise trust in hierarchy
  - name is derived from path followed
  - shortcuts allowed, but changes name
  - exposure of path is important for security
- Compared to Kerberos
  - transited field in Kerberos - doesn’t change name
- Compared with X.509
  - X.509 has single path from root
  - X.509 is for public key systems
- Compared with PGP
  - PGP evaluates path at end, but may have name conflicts
Generic Security Services API (GSS-API)

- Standard interface for choosing among authentication methods
  - once an application uses GSS-API, it can be changed to use a different authentication method easily
    - difficulty lies in the fact that different methods of authentication use different models of interaction
      - e.g., one way vs. challenge/response (requires, at a minimum, 2 messages), with zero knowledge proof, can have hundreds of messages
  - API calls
    - acquire and release credentials
    - manage security context
      - init, accept (on server side), and process tokens
    - wrap (confidentiality and/or integrity) and unwrap