CS530 Authentication in Applications

Bill Cheng

http://merlot.usc.edu/cs530-s10

Authentication in Applications

- Unix login
- Telnet
- Rsh/rlogin
- Ssh
- HTTP (web browsing)
- FTP
- Windows login
- E-mail (SMTP, POP, IMAP)
- NFS
- Network access services

Unix Login (review)

- One way encryption of password
  - salted as defense against pre-computed dictionary attacks
  - to validate, encrypt and compare with stored encrypted password
  - may use shadow password file

Telnet

- A remote login application
  - normally just an unencrypted channel over which plaintext password is sent
  - supports encryption option and authentication options using protocols like Kerberos
  - early implementation has an implementation vulnerability due to poorly generated random numbers

RSH (Remote Shell/Remote Login)

- Rsh is more efficient than telnet
- Usually IP address and asserted account name
  - reverse DNS lookup, not so easily spoofed (good thing that a two way communication is required)
    - 128.125.5.168 → lookup 168.5.125.128.in-addr.arpa
    - note: it's easier to compromise forward DNS lookup
      - nunki.usc.edu → 128.125.5.168
      - counter measure: do both
  - privileged port (client port number < 1024) means accept asserted identity
    - this is the case where a ~/.rhosts file is used
    - `rsh` must be setuid root
      (makes the client machine more vulnerable)
  - if not trusted (no ~/.rhosts file), Unix password in the clear

RSH (Remote Shell/Remote Login) (Cont...)

- Kerberos based options available
  - Kerberos based authentication and optional encryption
    - using XOR (stream cipher)

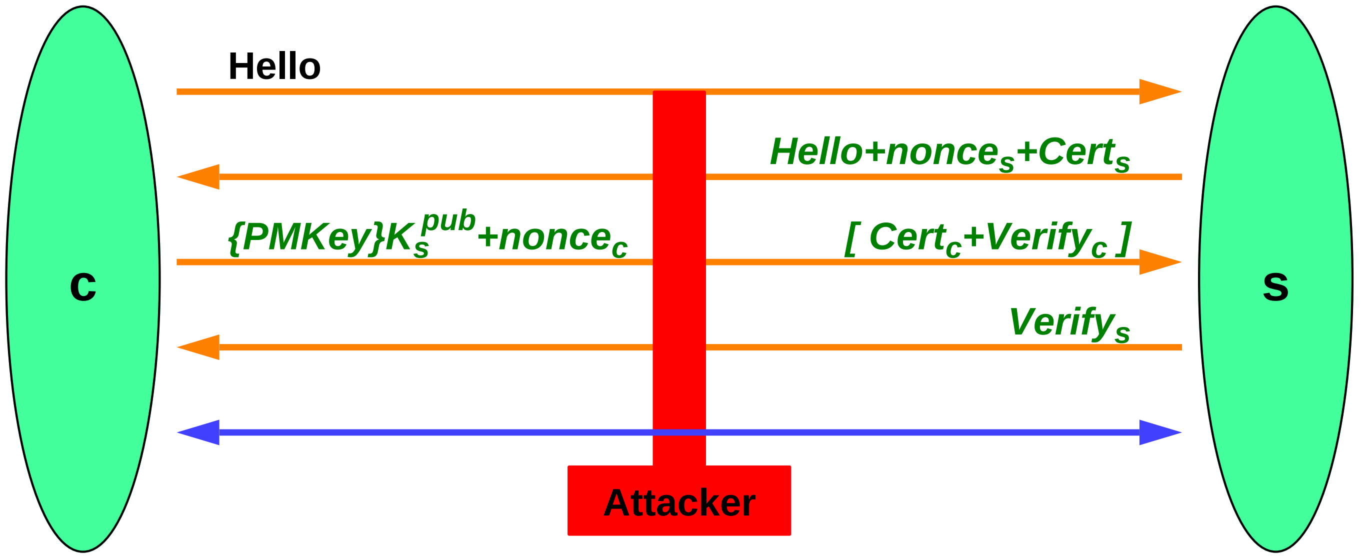
Secure Shell (SSH)

- Encrypted channel with Unix login
  - establish encrypted channel, using public key presented by server
  - send password of user over channel
  - Unix login to validate password
  - vulnerable to man-in-the-middle attack
  - can reply the whole session! (is this a problem?)
- Public key stored on target machine
  - limits where login can come from
  - user generate public/private key pair, and uploads the public key to directory on target host
  - target host validates that corresponding private key is known
  - key distribution without authentication

Web Browsing (HTTP/HTTPS)

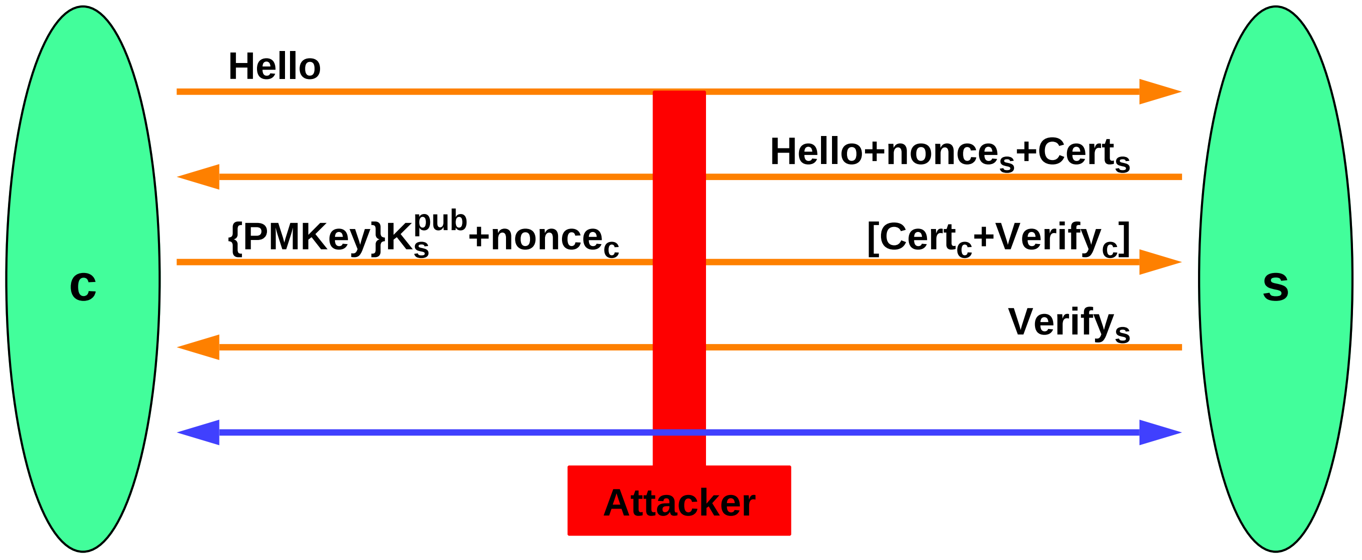
- Basic authentication: connect in the clear, Unix password
  - base64 encoded "UserID:Password"
- Digest authentication (RFC 2617)
  - server sends nonce (to mitigate reply attack)
  - responds is MD5 checksum of:
    - username
    - password
    - nonce URI
- Connect through SSL, Unix password
- User certificate, strong authentication

Secure Sockets Layer (and TLS)



- TLS is Transaction Layer Security (IETF version of SSL)
- *PMKey* is the pre-master key, session key(s) derived from this
- *[Cert$_c$+Verify$_c$]* = optional client authentication
- *Verify$_c$* = *nonce$_s$* encrypted with client private key
- *Verify$_s$* = *nonce$_c$* encrypted with *PMKey*

Secure Sockets Layer (and TLS)



- encryption support provided between browser and web server - below HTTP layer
- client checks server certificate
  - works as long as client starts with the correct URL
- key distribution supported through certification steps
- authentication provided by verify steps

File Transfer Protocol (FTP)

- Password based authentication
  - on UNIX
    - wu-ftpd use to have lots of implementation bugs
      - e.g., buffer overflow
      - e.g., root login, abort password, login as anonymous
    - run as root is necessary
      - server needs to bind priviledged port
      - server needs to su to any user ID
  - on Windows XP, ftp server turned off by default
    - the default ftp mode is the *anonymous* mode
    - warns about password-based authentication?!

- GSS-API based authentication
  - including use of Kerberos
  - authentication occurs and then stream is encrypted

Windows Network Login

- In Win2K and later uses Kerberos
- In Win NT
  - challenge response (NTLM)
  - server generates 8 byte nonce
  - prompts for password and hashes it
  - uses hash to obtain 3 keys and then DES encrypt nonce 3 times

File System Authentication

- Sun's Network File System (NFS)
  - typically address based
    - the remote host is trusted to assert the real UID
  - Athena Kerberized version
    - maps authenticated UID's to addresses
  - NFS built on ONC RPC (ONC is Open Network Computing)
    - ONC RPC has stronger Kerberos/GSS-API support
- Andrew File System (AFS)
  - based on Andrew RPC
  - uses Kerberos 4 authentication
- OSF's DCE File System (DFS)
  - based on DCE RPC and AFS
  - uses Kerberos 5 authenciation

Network Access Servers

- Used by dialups and PPPoE
- Radius
  - problem: not connected to network until connection established
  - need for indirect authentication
    - network access server must validate login with radius server
    - password sent to radius server encrypted using key between agent and radius server

Email

- SMTP - to send mail
  - usually network address based (or no authentication -- incoming mail is relayed)
    - can use inverse IP address lookup, but spoofed easily
    - HELO hostname (any hostname!)
    - FROM (anything you want!)
    - open for spamming
  - can use password
  - can be SSL protected (not really done)
  - SMTP after POP

Email (Cont...)

- Post Office Protocol (POP)
  - download e-mail
  - plaintext password
  - can be SSL protected
  - mail client such as Outlook Express and Netscape Mail checks for incoming mail about every 10 minutes (this is how often it communicate password information)
  - Eudora supports Kerberos authentication
- IMAP
  - e-mail stay on server
  - password authentication
  - can also support Kerberos

CS530
Stopping SPAM

Bill Cheng

http://merlot.usc.edu/cs530-s10

Stopping SPAM

- We will discuss two papers
  - Freitas and Levene, *Spam on the internet: Is it here to stay or can it be eradicated?* [Freitas04a]
  - Walfish et al., *Distributed Quota Enforcement for Spam Control?* [Walfish06a]

Stopping SPAM [Freitas04a]

- Block listing
  - list of IP addresses of known sources of spam
  - 90% of all spm received in North America and Europe can be traced to a group of 200 spam outfits (all operating illegally)
  - e.g., SBL (Spamhaus Block List), RBL (Realtime Blackhole List), MAPS (Mail Abuse Prevention System)
  - avoids false positive

Stopping SPAM (Cont...)

- Protocol changes
  - by Anti-Spam research Group, subgroup of IETF
  - provide a method of tracking e-mail source in SMTP
    - so that one cannot forge/spoof e-mail addresses
  - various proposals:
    - modify DNS to identify the actual machines acting as mail servers
    - domain keys - use PKI
      - verify the sender of an e-mail (use digital signatures)
    - Sender Policy Framework
      - safe listing system, i.e., requiring domain owners to publish IP addresses from where e-mails are sent
    - none of these proposals will "stop spam"
      - help anti-spam technologies to identify origin of spam
      - force offenders to move to new domains

Stopping SPAM (Cont...)

- *Economic* solutions - make spammers pay for sending spam
  - make spamming *financially* unviable
  - various proposals:
    - allocate quota of e-mails a user is allowed to send
      - charge fees on all e-mails above the quota
      - need to solve the problem of how to pay and how to collect these micropayments
    - the receiver of an e-mail charges the sender a fee for each e-mail (which could be $0 for some senders)
      - if sender not on the list, payment details must be agreed prior to accepting the message
      - need to transfor money to the recipient's e-mail account (e.g., use e-stamps)
      - recipient may not collect if the e-mail is not spam

Stopping SPAM (Cont...)

- *Computational* solutions
  - make spammers pay, via computation, for sending spam
    - make spamming *computationally* unviable
  - proposals:
    - Microsoft Penny Black Project
      - an e-mail header contains a "computational stamp", as proof that the sender has spent enough effort
      - the effort could be measures in number of CPU cycles, memory cycles, or CAPTCHA
    - cryptographic challenges
      - client sends *n1* to mail server
      - server sends *N+n2* back to client (*N* may depend on current system load)
      - client must reply with *n3* such that *SHA1(n1+n2+n3)* has *N* bits of leading (or trailing) zeroes

Stopping SPAM (Cont...)

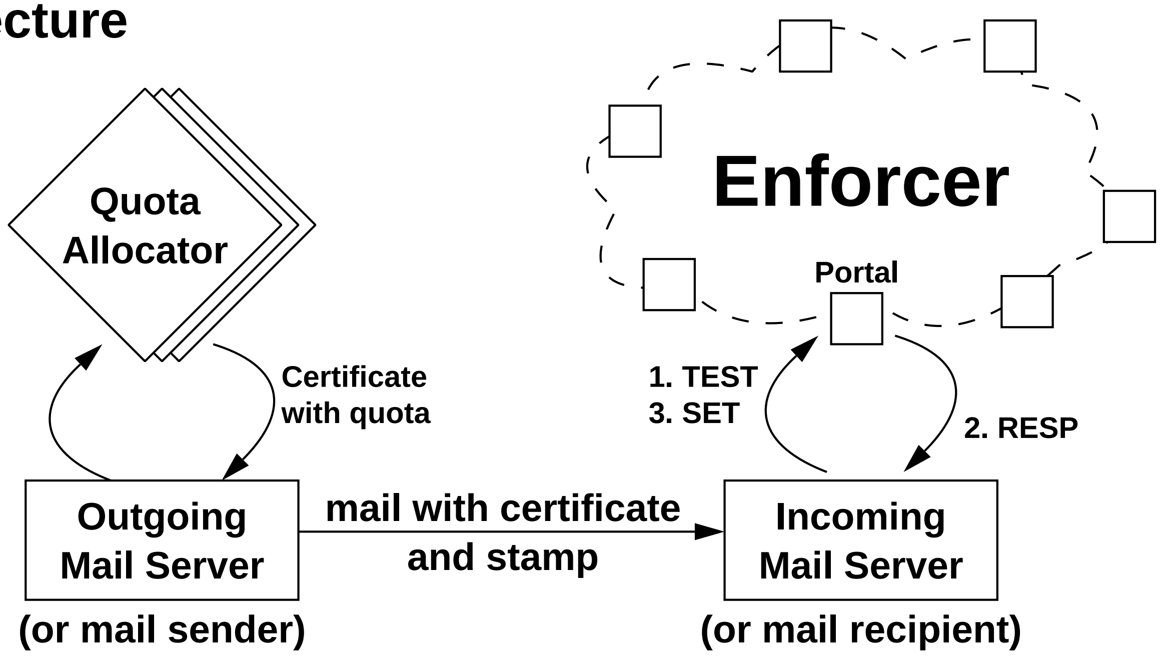
- Other solutions
  - e-mail aliasing
    - restrict senders to verified ones only by auto-replying with Reply-to being an alias
  - sender warranted e-mail
    - only accept e-mail with patented material in header
  - try to classify if e-mail is spam or ham
    - colaborative filtering
    - rule-based solutions
    - statistical solutions
  - legislative solutions

Distributed Quota Enforcement for SPAM Control [Walfish06a]

- Quota for making digital stamps
- Main issues
  - how to create digital stamps that cannot be forged?
  - how to check if a digital stamp has been reused?

Distributed Quota Enforcement (Cont...)

- Basic architecture



- Quota Allocator (QA) issue certificates
  - contains mail server public key, quota and expiration time
  - digitally signed with QA's private key
  - (cont...)

Distributed Quota Enforcement (Cont...)

- digitally signed with QA’s private key (cont...)

$$C_S = DS_{QA}\,[\,S_{pub},\ \text{expiration date},\ \text{quota}\,]$$

  - C_S is the certificate for mail server S
  - S_{pub} is the public key of server S
- anyone can verify QA’s signature

S, the mail sender buys a "stamp machine" from QA to "manufacture" digital stamps

- the stamp machine is basically an algorithm with parameters written in a digitally signed certificate
- to create a stamp, S signs $\{i,t\}$ (using S_{priv}) where i is a sequence number ($1 \le i \le quota$) and t is a date ($t \le$ expiration date)

$$\text{stamp} = C_S,\ DS_S\,[\,i,\ t\,]$$

- anyone can verify the stamp

Distributed Quota Enforcement (Cont...)

- Is it a reuse?
  - the mail recipient asks the Enforcer to check the stamp
  - if the Enforcer says the stamp is new, the mail recipient asks the Enforcer to remember the stamp
- Enforcer is a peer-to-peer (p2p) network which implements a DHT (Distributed Hash Table) system
  - DHT
    - *put(key, data)* stores a data item with the specified key
    - *get(key)* retrieves data item(s) corresponding to key
    - key is usually a *hash of data contents*
  - key here is SHA1(stamp), no data
    - actually, key is *SHA1(SHA1(stamp))* and data is *SHA1(stamp)* so that Enforcer cannot cheat
  - Enforcer implementation is resilient to node failures

Distributed Quota Enforcement (Cont...)

- SHA1(stamp) must be unique
  - recall that

    stamp = C_S, $DS_S[\ i, t\ ]$

  - if the digital signature scheme contains randomness, a spammer can generate many stamps with the same *{i,t}*
  - encrypting a hash with S_{priv} is weak
    - hash is very small relative to S_{priv}
      - RSA multiplicative property
    - often, a nonce is added to pad data before signing
    - cannot be used here
- How to create unique signatures?
  - use Full Domain Hash (FDH)

Full Domain Hash (Review)

- Using RSA as an example
  - FDH maps a message of arbitrary bit-length to a value uniformly distributed between *0* and *n-1*
    - and has all the properties of a good cryptographic hash function
  - appropriate for encrypting with private key since there is no randomness
- Realizing FDH using SHA1 for a 1024-bit RSA private key
  - output of SHA1 is 160 bits
    - needs 7 of these, $FDH(m) = (B_1,B_2,\ldots,B_7) \bmod n$

      B_1 = SHA1("SHA1/FDH:1/7", m)

      B_2 = SHA1("SHA1/FDH:2/7", m)

      ...

      B_7 = SHA1("SHA1/FDH:7/7", m)

Delegated Authentication

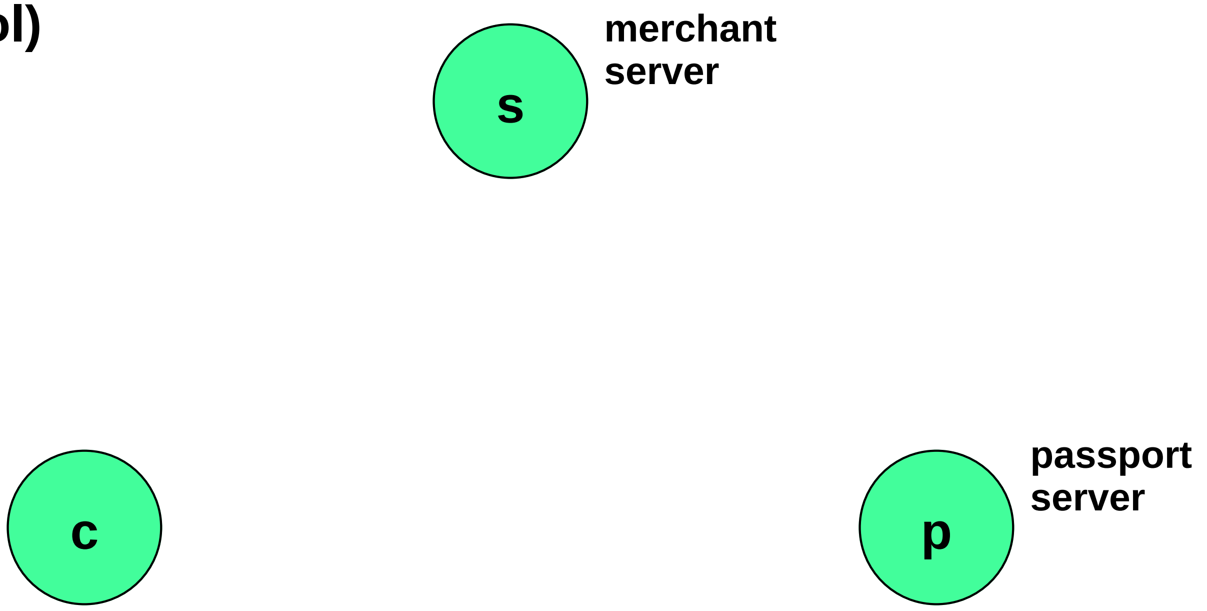
- Usually an authorization problem
- How to allow an intermediary to perform operations on your behalf
  - pass credentials needed to authenticate yourself
  - apply restrictions on what they may be used for
- Systems
  - Microsoft Passport
  - Liberty Alliance
  - Kerberos restricted proxies (later)

Microsoft Passport vs. Liberty Alliance

- Two versions of Microsoft Passport
  - current deployed version has lots of weaknesses and is centralized
    - A. Rubin, *Risks of the Passport Single Signon Protocol* http://avirubin.com/passport.html
  - version under development is "federated" and based on Kerberos
- Liberty Alliance
  - loosely federated with framework to describe authentication provided by others
  - also to address problems with MS Passport v1

Microsoft Passport v1

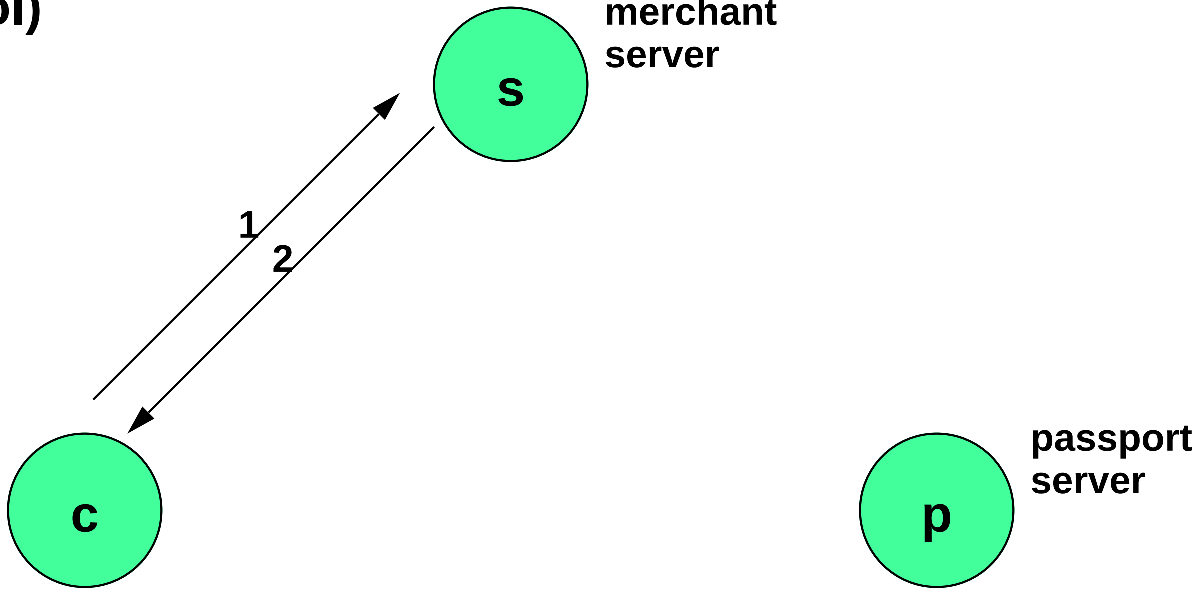
- Goal is single sign on
- Implemented via redirections (must work with the web protocol)



- client enrollment - need to setup what type of information (e.g., address, creditcard number, etc.) is okay to send to what type of merchants

Microsoft Passport v1 (Cont...)

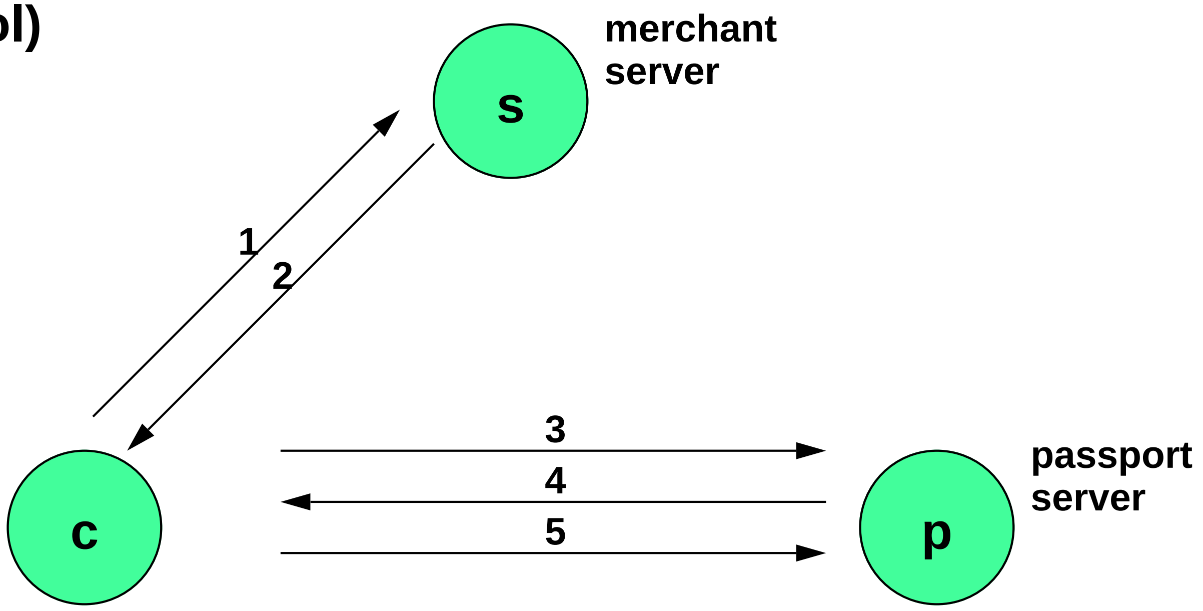
- Goal is single sign on
- Implemented via redirections (must work with the web protocol)



- msg 2: HTTP redirect
  - information about the merchant, what the merchant needs (e.g., user’s address, creditcard number)

Microsoft Passport v1 (Cont...)

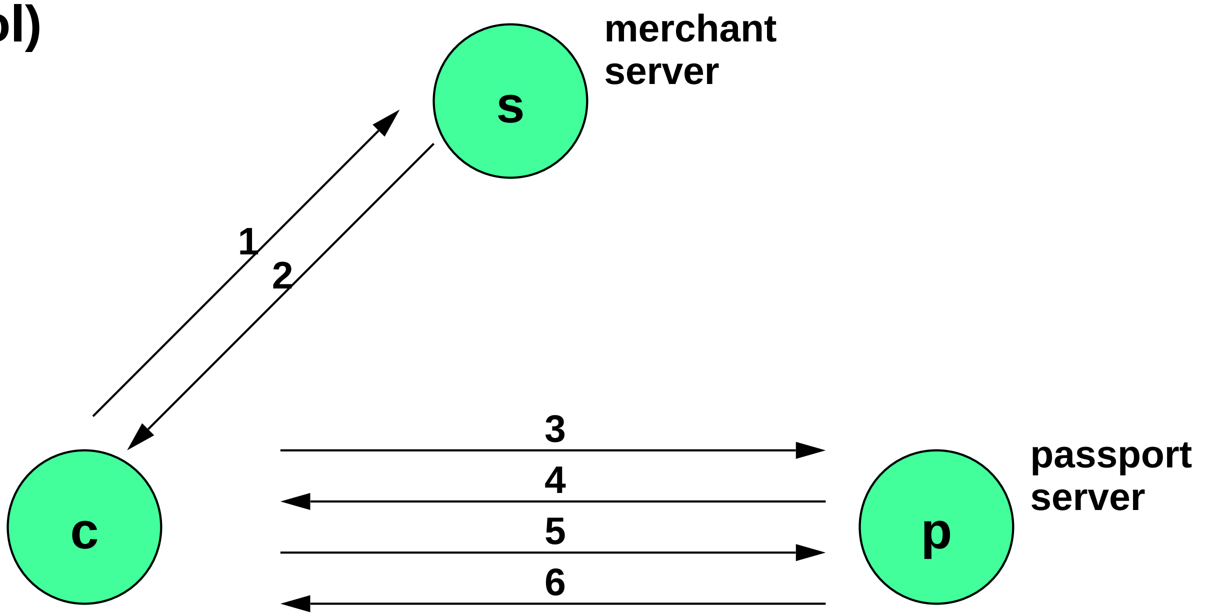
- Goal is single sign on
- Implemented via redirections (must work with the web protocol)



- HTTPS session with the passport server
- msg 4: request credentials (login screen)
- msg 5: user ID and password

Microsoft Passport v1 (Cont...)

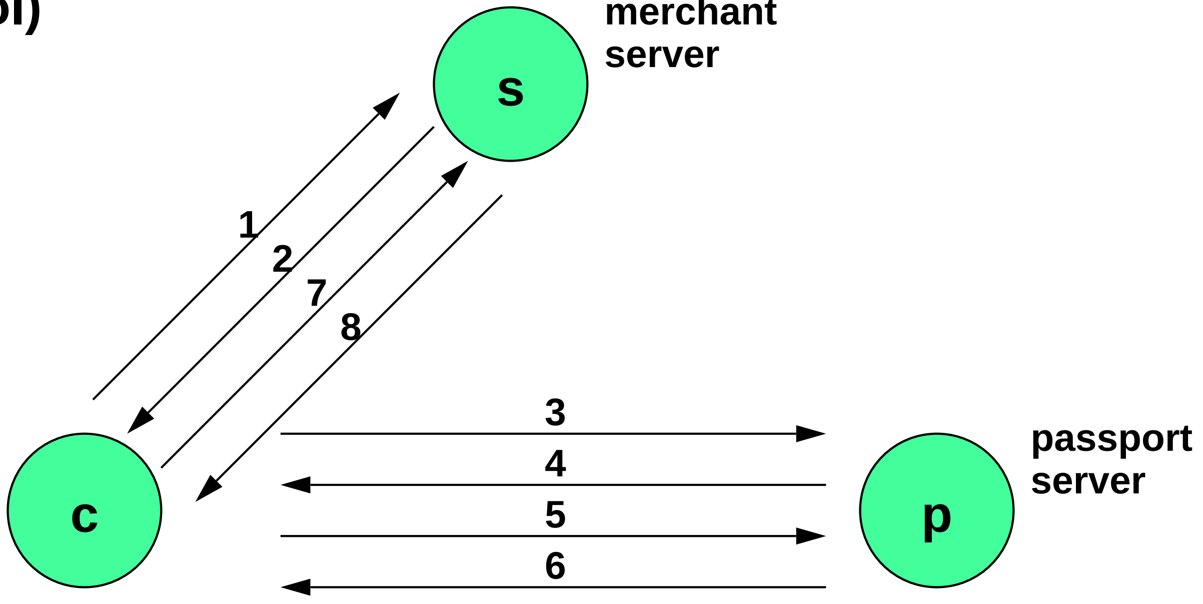
- Goal is single sign on
- Implemented via redirections (must work with the web protocol)



- msg 6: HTTP redirect
  - information encrypted using long term secret key between the merchant server and the passport server
  - token in header, stored as cookie on the client

Microsoft Passport v1 (Cont...)

- Goal is single sign on
- Implemented via redirections (must work with the web protocol)



  - msg 7: credentials (3DES encrypted with key previously agreed upon between passport server and merchant)
  - msg 8: set cookie in browser (a future visit to merchant, cookie is sent to merchant server, no need to visit passport server)

Risks of Passport Single Signon Protocol [Kormann00a]

- By Dave Kormann and Aviel Rubin <http://avirubin.com/passport.html>
  - various risks of Passport
- User interface
  - passport signout suppose to remove cookies
  - is hotmail logout the same thing as passport signout?
    - hotmail logout suppose to mean remove hotmail credentials
    - passport signout suppose to mean remove passport credentials to all services
    - not clear to an average user
  - netscape interaction with passport server bug
    - most browsers now by default only return cookies to the server from which they came, so it's no longer an issue
    - problems with cookies in general in early days

Risks of Passport Single Signon Protocol (Cont...)

- Key management
  - passport protocol requires passport server share triple DES keys with each merchant
    - ideally these triple DES keys should be transported out-of-band (via physical mail or over the phone)
    - can these triple DES keys be generated randomly and safely sent over SSL?
      - potential risks since it requires autentication of the merchant in some way
  - (cont...)

Risks of Passport Single Signon Protocol (Cont...)

- Key management (cont...)
  - passport encrypts information for itself and store it in a cookie on client machine (msg 6)
    - a single key is used to encrypt all cookies for all clients
    - compromising this key compromises all cookies stored on all clients
    - a better solution is to use a master key to generate a unique key per client
      - let *MK* be the master key, *$CLIENT_n$* be the IP address (or hostname) of client *n*
      - *K_n = 3DES(MK,$CLIENT_n$)* be the key to encrypt the cookie stored on client *n*
      - if a single key is compromised, it does not compromise cookies stored on other client machines

Risks of Passport Single Signon Protocol (Cont...)

- Central point of attack
  - denial of service to the passport server
    - usual solution to DoS attacks is replication of service
    - but it’s not easy to make database consistent and there are key distribution problems
  - more problem with cookies
    - an active attacker can impersonate the passport server and delete cookies on client machines
    - the Cookie Monster bug for new domain names could easily overwrite merchant cookies on any client
- Cookies and Javascript
  - passport requires these 2 technologies
  - cookies and Javascript have been shown to compromise user privacy
    - doesn’t give users a sense of trust

Risks of Passport Single Signon Protocol (Cont...)

- Persistent cookies option
  - passport leaves *authenticators*, in the form of browser cookies on the client machine
    - these authenticators do not get deleted until expired, even if the machine is turned off
  - these are not the same as Kerberos authenticators
    - in Kerberos, an authenticator is a timestamp encrypted by a shared key, if decrypted successfully, the client must have the correct key
      - this prevents theft and misuse of a ticket found lying on a machine
    - in passport, where cookies stand in for tickets, not real authenticators since cookies have much longer lifetime
    - breach is undetected and an attacker gets unlimited use of the victim’s authentication information

Risks of Passport Single Signon Protocol (Cont...)

- Automatic credential assignment
  - to demonstrate the ability of passport, all of hotmail account were automatically moved on top of passport by Microsoft
    - hotmail user ID and password became passport credentials
    - when user logs into hotmail, they actually run the passport protocol, with hotmail server acting as the merchant server
    - hotmail has lots of problems and is a very weak link
      - e.g., one compromise allowed an attacker to log into any hotmail account without knowing the password
    - a compromised hotmail account can go shop online with other passport merchants

Risks of Passport Single Signon Protocol (Cont...)

- Bogus merchant attack
  - an attacker sets up shop, convinces a legitimate CA to issue a certificate for passsport.com
  - when user logs on, the user ID and password can then be used to authenticate to passport on behalf of the user, user wallet services, etc.
  - can also do this in man-in-the-middle fashion or through DNS spoofing

Risks of Passport Single Signon Protocol (Cont...)

- Conclusions
  - passport must run over the web with unmodified browsers
    - retrofitting the complex process of single signon over the web technology created risks
  - the bulk of passport’s flaw arise directly from its reliance on systems that are either not trustworthy (such as HTTP referrals and the DNS) or assume too much about user awareness (such as SSL)
  - (cont...)

Risks of Passport Single Signon Protocol (Cont...)

- Conclusions (cont...)
  - suggested improvements
    - rotating keys used to encrypt cookies
    - using a master key to generate encryption keys
    - requiring SSL for all transactions would eliminate forged redirects (at the cost of slowed down merchant servers)
    - replacing password-based authentication with a challenge-response scheme (such as HTTP digest authentication) would make it unlikely for an attacker to reuse passwords to impersonate a user
  - given the requirements, passport risks may be inevitable

Federated Microsoft Passport

- Announced September 2001
  - to address the major concern that Microsoft becomes part of every signon under Passport v1
- Multiple registrars
  - e.g., ISPs register own users
- Kerberos credentials
  - embedded authorization data to pass other information to merchants
- Federated Microsoft Passport is predominantly vaporware today, but .net authentication may be where their federated model went
  - major problem is that this does not fit within the Web model of interaction
  - can't really use cookies as tickets

Liberty Alliance (SUN, AOL, Netscape, etc.)

- Answer to MS Federated Passport
- Design criteria was most of the issues addressed by Federated Passport, e.g., no central authority
- Got off to slow start, but to date has produced more than passport has
- Use SAML (Security Association Markup Language) to describe trust across authorities, and what assertions means from particular authorities
- These are hard problems, and comes to the core of what has kept PKI from being as dominant as orginally envisioned
- Phased approach: Single sign on, Web service, Federated Services Infrastrcture

Liberty Alliance (SUN, AOL, Netscape, etc.) (Cont...)

- No hierarchy
  - n^2 problem, does not scale
  - PGP model, sort of
- "The market will decide" (who to trust most)