Kerberos
One more strong password

- Ford-Kaliski two-server protocol
- Store, for Alice, $k_1$ on server $S_1$ and $k_2$ on server $S_2$, where $k_1$ and $k_2$ are random #s
- $W = h(\text{pwd})$, $p$ is known prime
- $K = h(W^{k_1}, W^{k_2})$
- can store user’s private key encrypted with $K$, a strong secret
Ford-Kaliski

computes $W$, knows $p$ and $k_1$
chooses $z$, and $z^{-1}$, exponentiative inverses mod $p$

Alice

$W_z \mod p$

raise to $z^{-1}$

get $W_{k_1}$

$W_z^{*k_1} \mod p$

knows $p$, $k_1$
Kerberos Authentication (Basic)

Alice  KDC  Bob

Alice wants Bob

{“Bob”, Kab, {“Alice”,Kab}Kb} Ka

{“Alice”, Kab}Kb, {timestamp}Kab

{timestamp+1}Kab
Getting rid of long term secrets

• It is dangerous for the workstation to hold Alice’s secret for her entire login session
• How can the long-term secret be turned into a session secret?
Ticket Granting Tickets

• TGT looks just like ticket but encrypted with KDC’s key
• WS keeps $TGT = \{“Alice”,S\}K_{kdc}$ and $S$
• Kerberos documentation refers to 3 things:
  – AS (authentication server)
  – TGS (ticket granting server)
  – KDC (the box that provides the 2 services)
• should just be “KDC”
Kerberos Authentication
(with TGT={"Alice", S}K_{K_{dc}})

Alice wants Bob, TGT

{"Bob", K_{ab}, {"Alice", K_{ab}}K_{b}}S

"Alice", K_{ab}K_{b}, {timestamp}K_{ab}

{{timestamp+1}K_{ab}
Tickets

- Alice’s name, instance, realm
- Alice’s layer 3 (IPv4) address
- session key $K_{\text{Alice-Bob}}$
- ticket lifetime, units of 5 minutes (1 byte)
- time when ticket created
- Bob’s name, instance
Comments

- Why is there no realm for Bob?
- Why is there name-instance for Bob?
- Why is there an IP address for Alice?
- How might delegation be safely allowed?
- Potential problem with NATs
Network Address in Ticket

- Prevent theft of authenticator and ticket within 5 minutes (and unencrypted session to Bob)
- Prevent delegation…but might be desirable
- Problems with NAT
- Compare with address in authenticator
Why Bob name/instance

- Assumes colocated services sharing a key
- scenario in book: gameplaying and nuclear missile launching service
  - someone only has authority for games, can’t use ticket for wrong service by changing port #
  - someone that steals ticket and authenticator within clock skew and replaying game-playing data (which to missile-launching service does bad things)
Authenticator

• Alice’s name, instance, realm
• checksum (incredibly badly named, and mostly useless field)
  – some apps set it to Alice’s process ID (why?)
  – KDC replication process sets it to size of DB
  – sample app says “type in a value”
• 4-byte time in seconds
• 1 byte “5 millisecond timestamp” (could be sequence #)
Credentials (ticket+session key)

- session key
- Bob’s name, instance, realm
- ticket lifetime (expires lifetime+timestamp)
- Bob’s key version number
- ticket length, ticket
- timestamp
AS_REQ (ask for ticket using master key)

- Kerberos version (4)
- message type (=1) (plus bit, byte order flag)
- Alice name, instance, realm
- Alice’s “timestamp” (to match request with reply)
- desired ticket lifetime
- Bob’s name, instance (why not realm?)
TGS_REQ (ask for ticket using TGT)

- differences from AS_REQ
  - message type=3
  - KDC’s key version number
  - KDC’s realm (for interrealm-discuss)
  - TGT
  - authenticator (not useful, makes it look like AP-REQ (“application request”, i.e., contact with Bob)
AS_REP and TGS_REP

- msg type 2, response to TG or AS request
- Alice’s name, instance, realm
- Alice’s timestamp (to match with req)
- # of tickets (unused)
- ticket expiration time
- Alice’s key version number
- credential length, credential
Error reply (to AS or TGS req)

- Kerberos version
- message type (32)
- Alice’s name, instance, realm
- Alice’s “timestamp”
- error code
- error text
AP_REQ (connect to Bob)

- Kerberos version, message type (8)
- Bob’s key version number
- Bob’s realm
- ticket length, ticket
- authenticator length, authenticator
Encrypted Stuff

- Kerberos version, message type (8)
- message type (6)
- length of encrypted stuff, encrypted stuff
  - length of data, data
  - 5-millisecond timestamp
  - sender’s IP address
  - direction flag, timestamp
Kerberos v4’s integrity check on encrypted data

- Modified CBC to “PCBC”:
  - XOR not just $c_i$ and $m_{i+1}$, but also $m_i$
- This isn’t self-stabilizing like CBC, so can use recognizable plaintext at end.
- But there are ways of modifying the ciphertext so the end will decrypt properly
- What happens if blocks get rearranged?
Integrity-protected Stuff

• Kerberos version, message type (7)
• message type (6)
• length of encrypted stuff, encrypted stuff
  – length of data, data
  – 5-millisecond timestamp
  – sender’s IP address
  – direction flag, timestamp
  – modified Jueneman checksum
Replicated KDCs

- One master
- User secrets individually encrypted with KDC’s secret (shared by all replicas)
- Update: create, change, or delete entry entry=(username, key)
- Can replication be done without cryptographic protection (since keys are encrypted)?
Variants

- Authentication with token, or S/KEY
- Could client create its own TGT and session key?
- What if non-synchronized clocks?
- What if no clocks?
Kerberos v5

• spec now 133 pages long
• specs are not fun to read
  “AuthorizationData is always used as an OPTIONAL field and should not be empty”
Kerberos v5

- Traded “B” bit for ASN.1
  - Instead of 4 bytes for address, so only IPv4
  - In Kerberos v5:
    
    ```
    SEQUENCE (2 octets: T=seq, L)
    { addr-type[0] INTEGER,
      - T=0, length, (for addr-type) = 2
      - T=“integer”, length, value = 3
      address[1] OCTET STRING } (4 octets+address)
    2+5+8 = 15 bytes instead of 4
    ```
If ASN.1 guru...

- security people don’t tend to think about ASN.1
- Using “IMPLICIT” as in:
  - addr-type[0] IMPLICIT INTEGER
- reduces overhead by 2 bytes
- With “CHOICE”, can be just 2 bytes of overhead per type of address
Names

- v4: name, instance, realm
  - not long enough sometimes (<40)
  - some necessary characters illegal
  - realm is DNS name

- v5: just name and realm, but name is a type and a varying number of arbitrary strings. Realm can be DNS or X.500
Delegation

- Could give Bob master key
- Or give Bob session key plus TGT
- Or know in advance all resources necessary, get tickets, and give tickets and keys to Bob
- But tickets contain addresses
- V5 allows requesting a different address, allows auditing
Limited Delegation

- Give Bob specific tickets rather than TGT
- Use “authorization-data” field, app-specific
- K5 makes delegation optional
- Two flags: proxiable (TGT can be used to request tickets for other address), forwardable (TGT can be exchanged)
  - could configure options into user’s entry, or ask
Interrealm

- V4-direct relationships
- Work through steps
PreAuthentication

- Anyone can request a ticket on behalf of Alice, and the response will be encrypted under her password. Dictionary attack.
- Kerberos V5 requires an encrypted timestamp on the request
  - Only an eavesdropper can guess passwords
- Unless get a ticket to a human...
Key in ticket

- Alice can carry on multiple conversations with Bob using same ticket
- But might want different session key for each session
- So Alice can choose a session key and put it into the ticket
Getting a ticket to a human

• Example: XWINDOWS
• Human’s client node no longer remembers Bob’s master key
• How could this work?
Timestamp issues

- V4: max 21 hours (4 octet start-time, 1 octet lifetime, units of 5 minutes)
- v5: virtually unlimited, but units of seconds
  - so also need “microsecond” (up to 5 bytes)
- Also: renewable, postdated
- in tickets (and TGTs), each 17 bytes:
  - start-time, end-time, auth-time, renew-till
Cleanups

- Ticket no longer doubly-encrypted
- Remove Bob’s name from ticket
- More flexible cryptographic algorithms
- Fix integrity protection (not PCBC or Jueneman)
- Make master keys in different realms different (even if same pwd)
Hierarchy of Realms

- “Transited” field lists KDCs
- Left up to Alice to find path
- Left up to Bob to decide if he likes the path
- Could a bad KDC avoid being in the ticket?
PKINIT: public keys for users. How would you do this? How was it done?

If chain of certs, checks first is trust anchor, and the others are listed in Transited

How might public keys of servers be used?
Changing Password

• How would you do this?
KDC database

- “Alice”
- {key, key version number}..KDC’s key #
- max-life (of tickets to this principal)
- max-renewable life
- expiration (of this database entry)
- last modification date, and by whom
- flags (require preauth, allow proxiable, forwardable, etc.)
- password expiration, time last changed
- last successful TGT request (with pre-auth)
Ticket

- msg type, Kerb version, Bob’s name, realm
- encrypted:
  - flags
  - key
  - Alice’s name and realm
  - transited
  - timestamps
  - set of addresses for which this ticket valid
  - authorization data
Authenticator

- Encrypted
  - version #
  - Alice’s name, realm
  - “checksum”
  - time at Alice (seconds, microseconds)
  - (optional) subkey
  - initial sequence number for data
  - authorization data
Kerberos Cousin: OSF DCE

- Built on top of Kerberos V5
- Adds UID and group membership info into Kerberos tickets in “authorization” field
- Client learns server name. Server learns client UID and group memberships
Another Kerberos Cousin: Windows 2000 Kerberos

- Also shares syntax with Kerberos V5
- Also adds UID and GID to authorization field (but incompatibly with OSF DCE)
- Implements KDC “referrals” to centralize cross-realm configuration at KDCs and hence simplifies client configuration