ACM Turing Award

- Peter Naur won the 2005 ACM A.M. Turing Award for his work on defining the Algol 60 programming language
- In particular, his role as editor of the influential "Report on the Algorithmic Language Algol 60" with its pioneering use of BNF was recognized
- http://www.naur.com/

Network Security

Application Level Authentication

Why Application Level Security?

- Open Environment
- Clients Access Services
- Restrict Access to Authorized Users
- Workstation Can't Be Trusted
- Impersonate a Workstation (Spoof)
- Eavesdrop and Replay
- Firewalls Don't Always Do It
- Passwords Can Be Sniffed

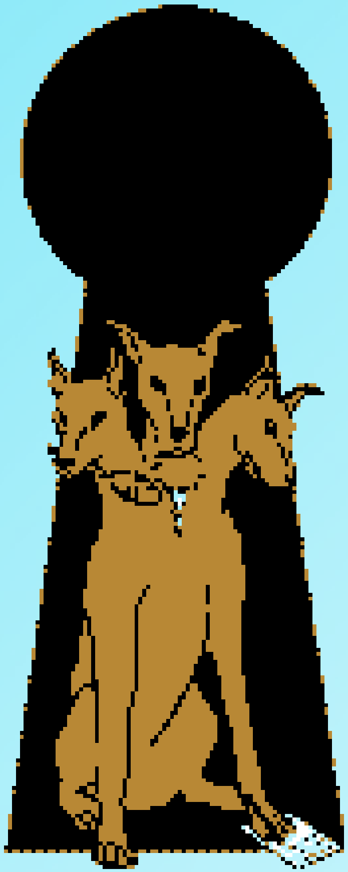
Kerberos

- MIT – 1988 – Project Athena
- Protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection
- Client and server can also encrypt all of their communications to assure privacy and data integrity as they go about their business

Cerberus


Sponte petit rapidos Erebi tirinthius amnes
Ianitor erepta Coniuge caesus obit

Kerberos



- Cerberus was a three-headed hound who patrolled the shore of the river Styx (Hades), devouring both living intruders and fugitive ghosts
- For Hercules' twelfth task, he was to bring Cerberus up from the underworld without any weapons

Pioneering Work of Famous MIT Professor

Kerberos

- Provides a centralized authentication server – authenticate users to servers and servers to users
- Relies exclusively on conventional encryption
- Version 4 & Version 5 (RFC 1510)

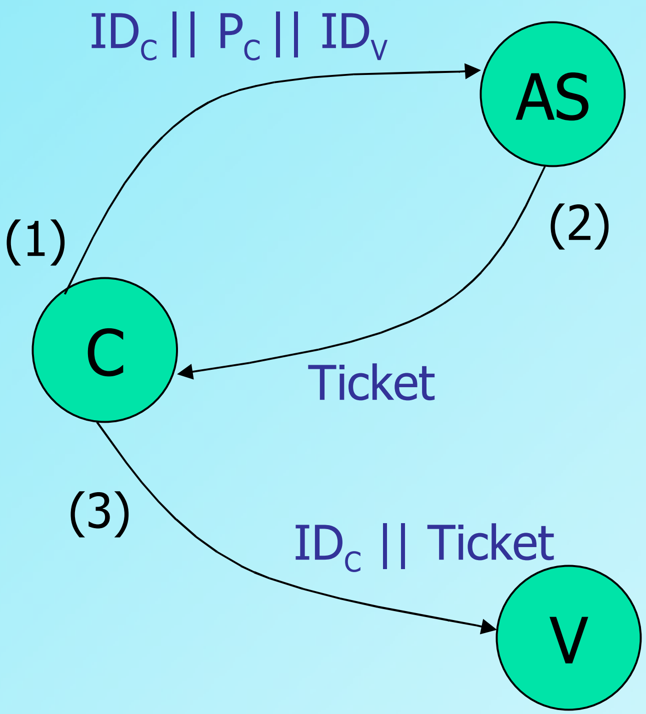
Kerberos Requirements

- Secure – no masquerading
- Reliable – distributed server architecture
- Transparent – user unaware authentication is taking place
- Scalable – support large number of clients and servers

Simple Client Authentication

- Obvious risk: impersonation
- Server needs to confirm identity of each client – NOT scalable
- Use an authentication server (AS)
  - Knows password of all users (database)
  - Shares a secret key with each server

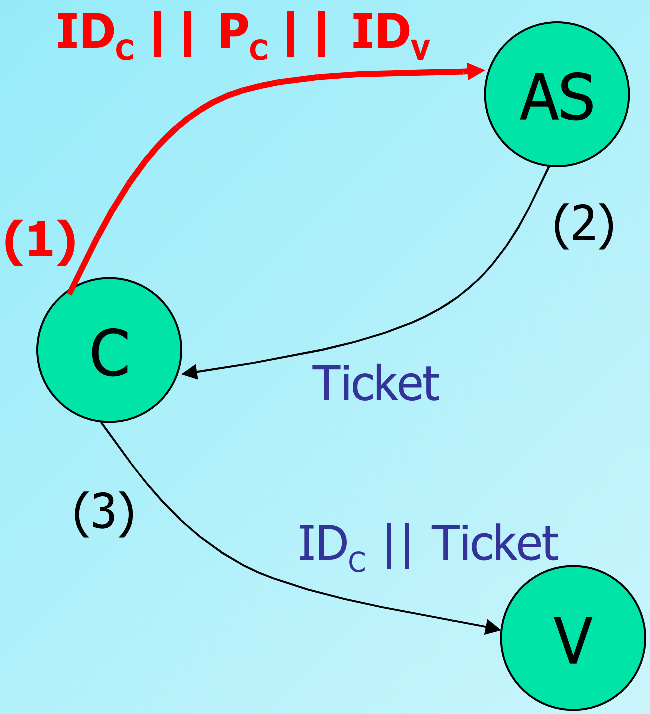
Simple Kerberos



C = client
AS = authentication server
V = server
ID_C = identifier of user on C
ID_V = identifier of V
P_C = password of user on C
AD_C = network address of C
K_V = secret encryption key shared by AS and V
|| = concatenation

Ticket = $E_{KV}[ID_C \,||\, AD_C \,||\, ID_V]$

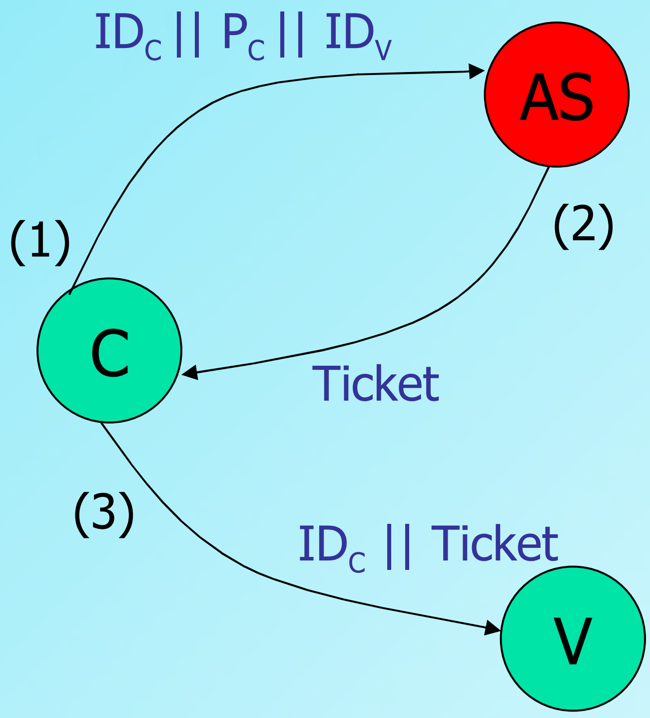
Simple Kerberos



Ticket = $E_{KV}[ID_C || AD_C || ID_V]$

- User logs on and requests access to server V
- Client module requests user password
- Sends message to the AS with user's ID, server's ID and user's password

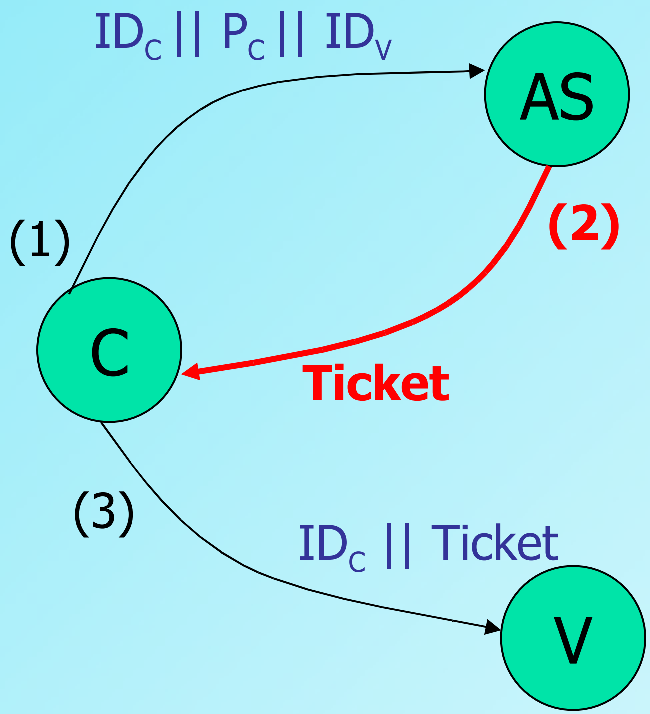
Simple Kerberos



Ticket = $E_{KV}[ID_C \| AD_C \| ID_V]$

- AS checks database to see if user has supplied the proper password and is permitted to access server V
- If authentic, then creates a ticket containing user's ID, network address, asn server's ID

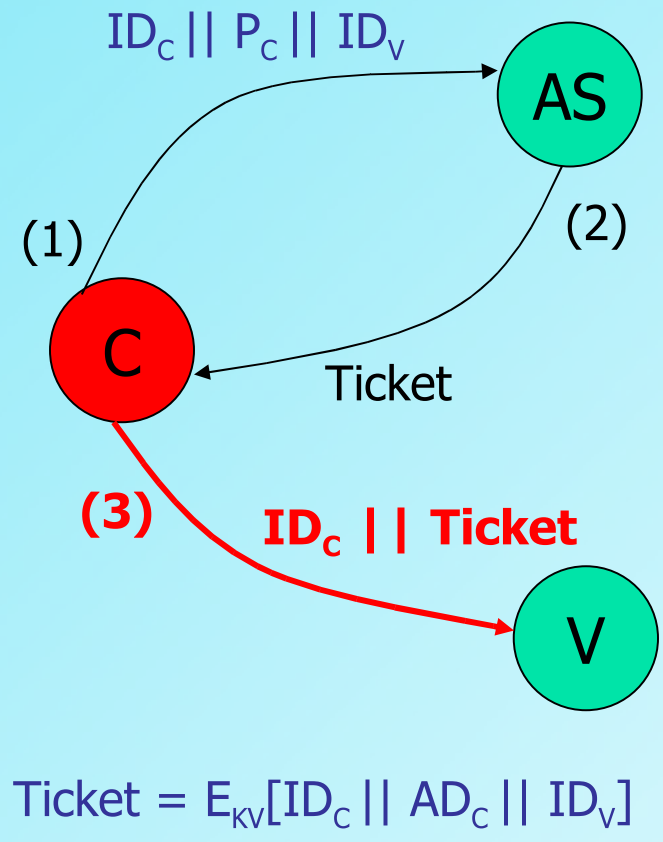
Simple Kerberos



Ticket = $E_{KV}[ID_C || AD_C || ID_V]$

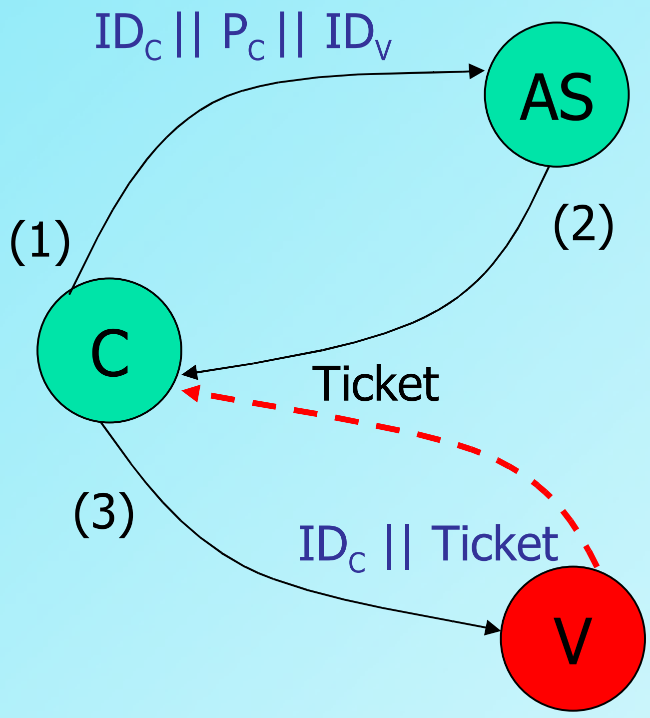
- Ticket is encrypted using the secret key shared by the AS and the server V
- Send ticket back to C
- Because the ticket is encrypted, it cannot be altered by C or an attacker

Simple Kerberos



- C can now apply to V for service
- C sends message to V with user’s ID and the ticket
- Server’s ID_V is included so that the server can verify it has decrypted the ticket properly
- Ticket is encrypted to prevent capture or forgery

Simple Kerberos

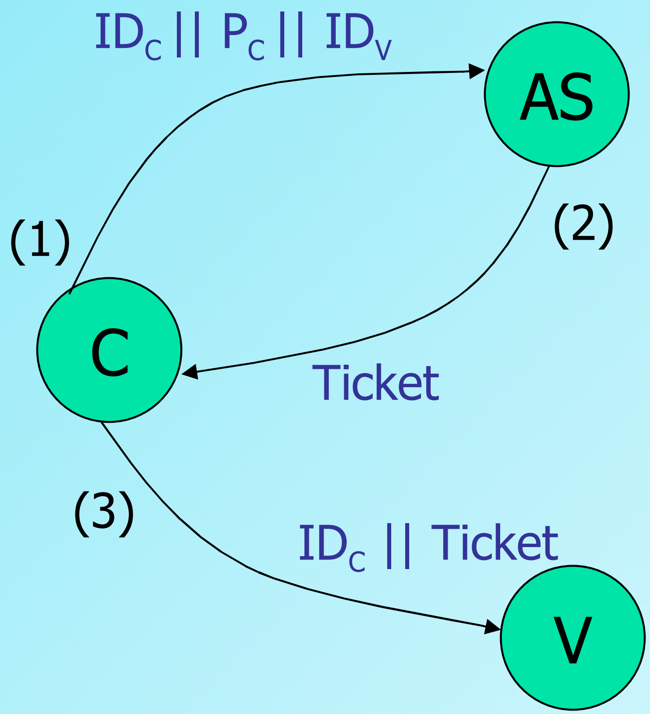


Ticket = $E_{KV}[ID_C || AD_C || ID_V]$

- V decrypts the ticket and verifies that the user ID_C in the ticket is the same as in the message
- AD_C in the message guarantees it came from original requesting workstation
- Finally, V grants the requested service

...But There's A Problem, Jon!

Simple Kerberos



Ticket = $E_{KV}[ID_C \| AD_C \| ID_V]$

Two problems:

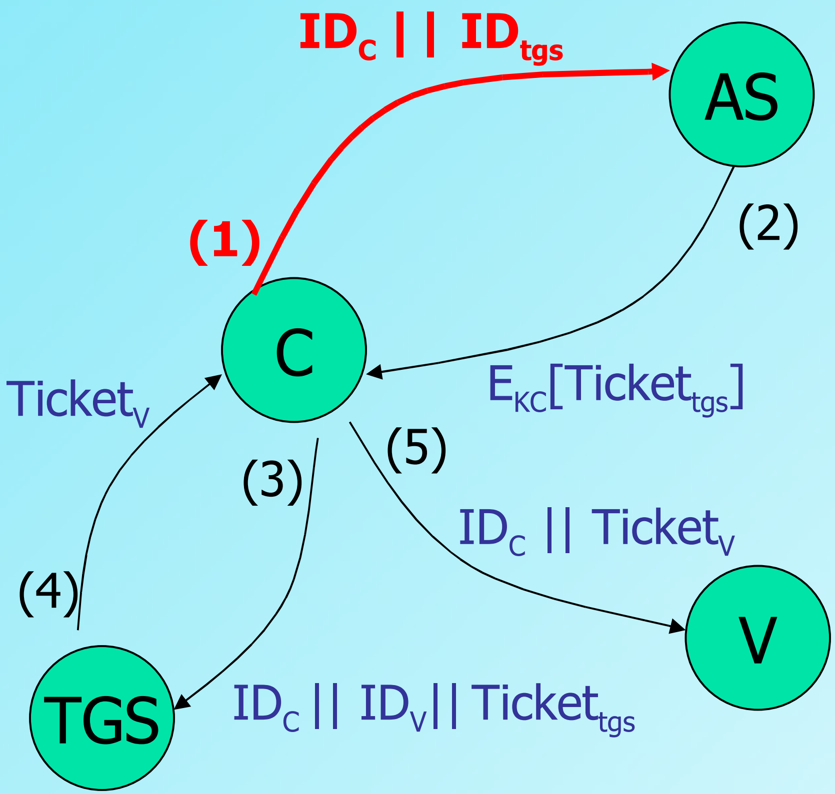
1) We would like to minimize the number of times that a user has to enter a password – reuse password

2) Password is in the clear – Ticket Granting Server

Ticket Granting Server (TGS)

- *A TGS issues tickets to users who have been authenticated to the AS*
- User first requests a ticket granting ticket, $Ticket_{tgs}$, from the AS and saves it in the client's workstation
- A client requesting services applies to the TGS using the ticket to authenticate itself
- TGS then grants a ticket, $Ticket_{v}$, for the particular service
- Client saves this and uses it each time a service is requested

Simple Kerberos w/TGS

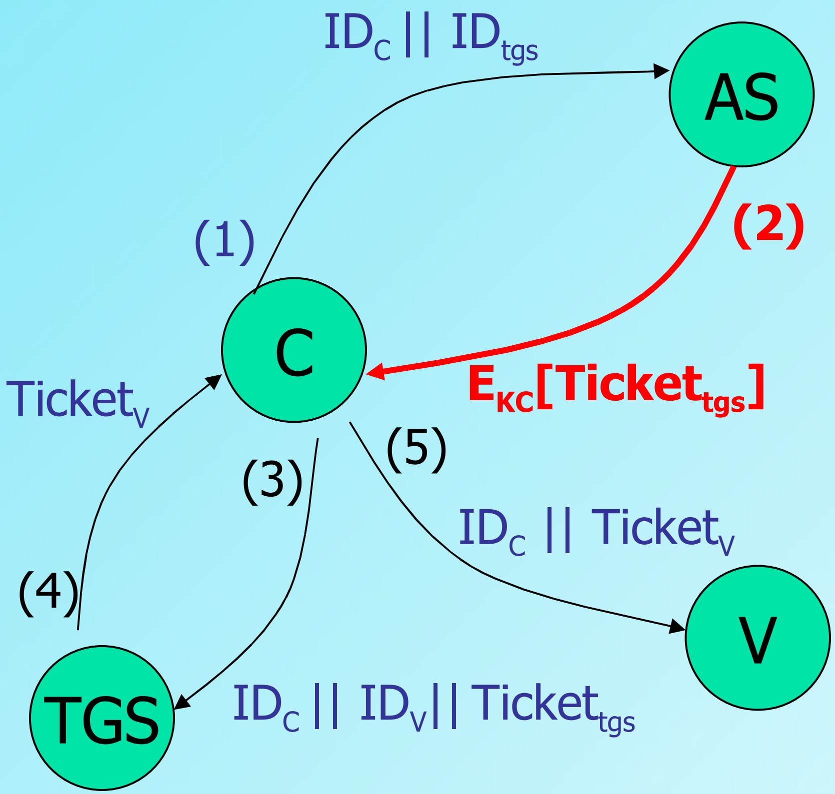


$Ticket_{tgs}=E_{Ktgs}[ID_C||AD_C||ID_{tgs}||TS_1||Lifetime_1]$

$Ticket_V=E_{KV}[ID_C||AD_C||ID_V||TS_2||Lifetime_2]$

- Client requests a ticket granting ticket on behalf of user
- Sends user's ID and the ID of the TGS
- Indicates request for TGS service

Simple Kerberos w/TGS

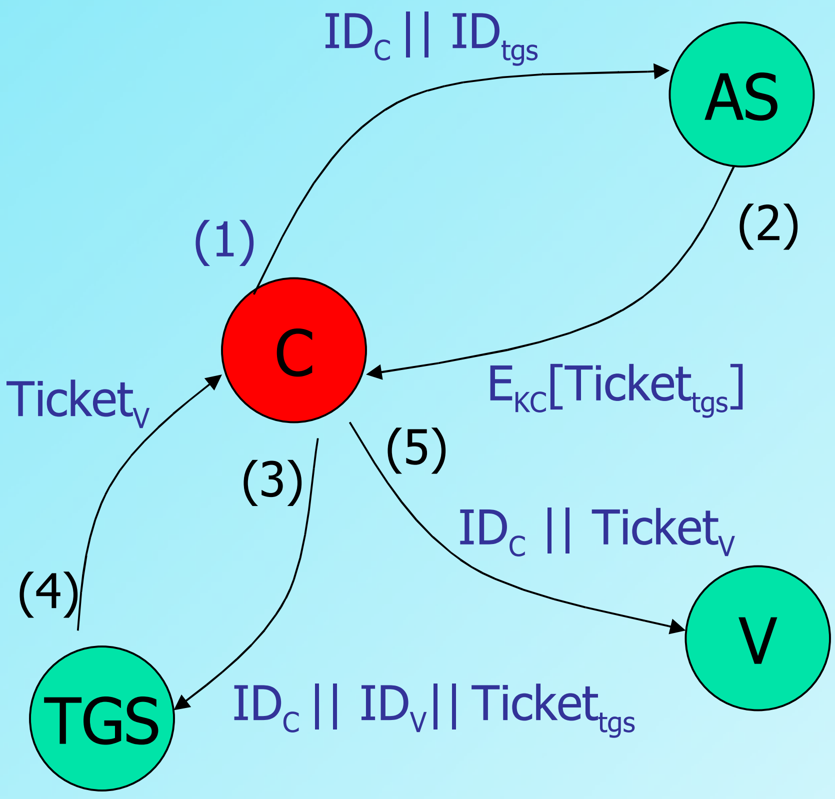


$Ticket_{tgs} = E_{Ktgs}[ID_C || AD_C || ID_{tgs} || TS_1 || Lifetime_1]$

$Ticket_V = E_{KV}[ID_C || AD_C || ID_V || TS_2 || Lifetime_2]$

- AS responds with a ticket that is encrypted with a key from user's password

Simple Kerberos w/TGS

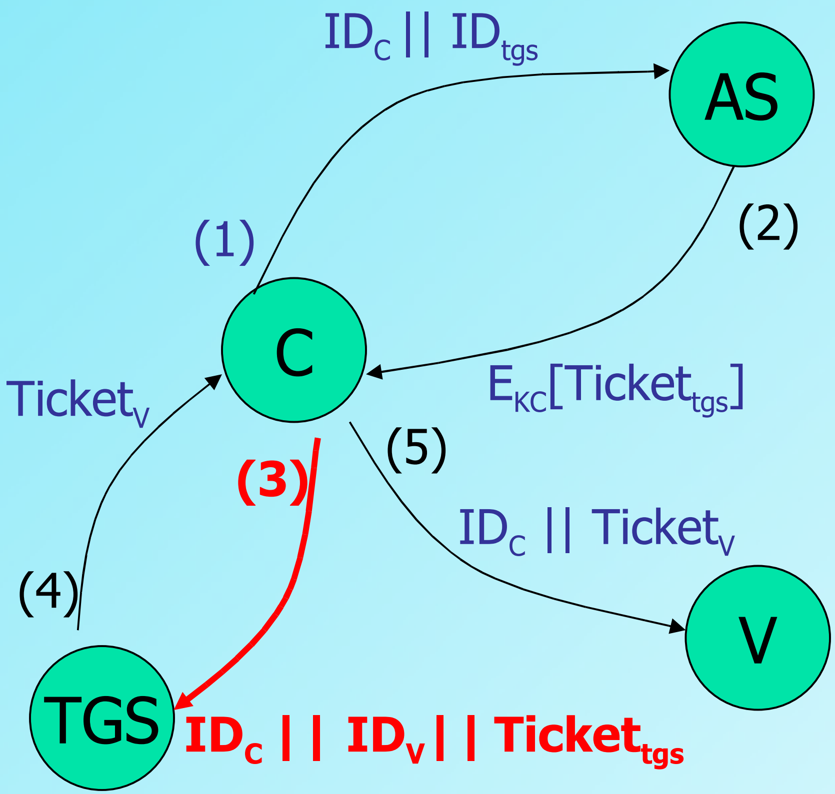


- Client prompts user for password, generates key and decrypts message
- Ticket is recovered!
- No need to transmit password in plaintext
- Ticket(tgs) is reusable

$Ticket_{tgs}=E_{Ktgs}[ID_C||AD_C||ID_{tgs}||TS_1||Lifetime_1]$

$Ticket_V=E_{KV}[ID_C||AD_C||ID_V||TS_2||Lifetime_2]$

Simple Kerberos w/TGS

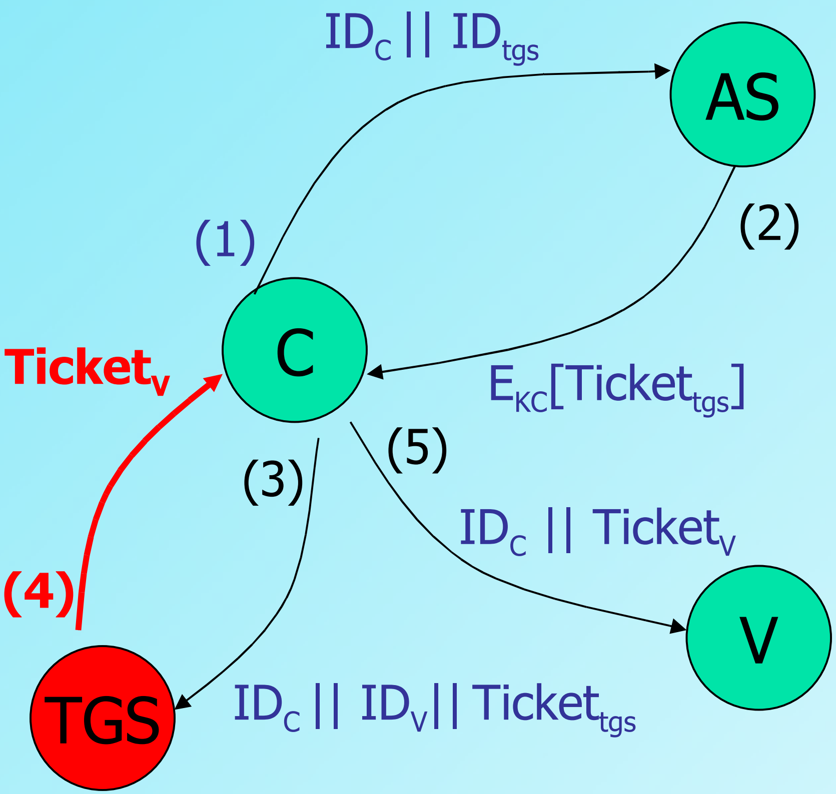


$Ticket_{tgs}=E_{Ktgs}[ID_C||AD_C||ID_{tgs}||TS_1||Lifetime_1]$

$Ticket_V=E_{KV}[ID_C||AD_C||ID_V||TS_2||Lifetime_2]$

- Client requests a service granting ticket
- Sends message to TGS containing user's ID, ID of the desired service and the ticket granting ticket

Simple Kerberos w/TGS

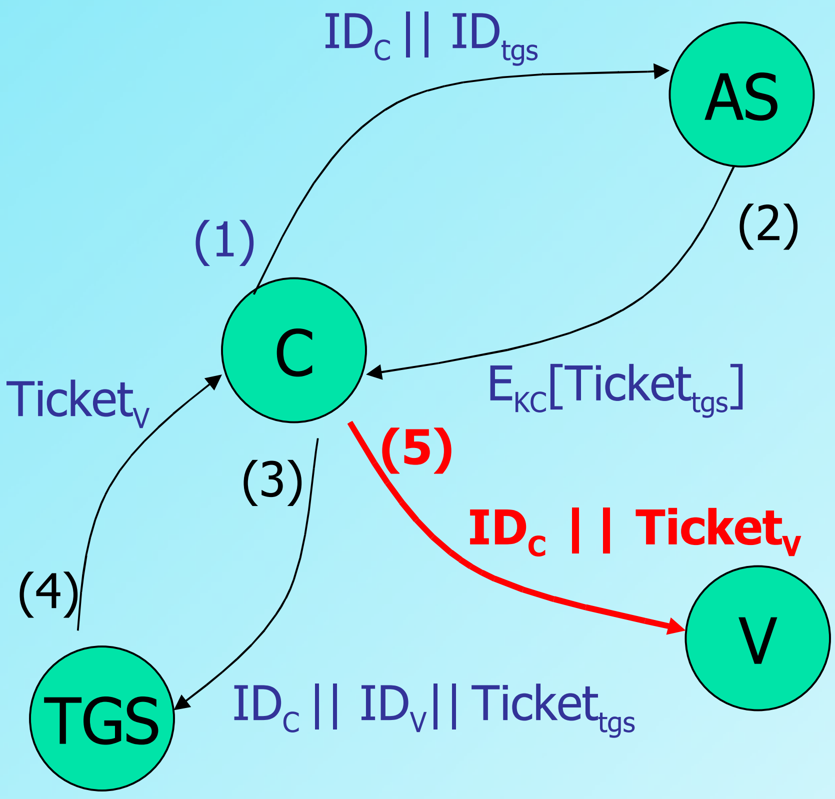


$Ticket_{tgs}=E_{Ktgs}[ID_C||AD_C||ID_{tgs}||TS_1||Lifetime_1]$

$Ticket_V=E_{KV}[ID_C||AD_C||ID_V||TS_2||Lifetime_2]$

- TGS decrypts the incoming ticket and looks for presence of its ID
- Checks lifetime and authenticates the user
- If user permitted access, sends service granting ticket

Simple Kerberos w/TGS



$Ticket_{tgs}=E_{Ktgs}[ID_C||AD_C||ID_{tgs}||TS_1||Lifetime_1]$

$Ticket_V=E_{KV}[ID_C||AD_C||ID_V||TS_2||Lifetime_2]$

- Client requests access to service on behalf of the user
- Sends user's ID and service granting ticket
- This can happen repeatedly without prompting for password

Things Are Looking Better

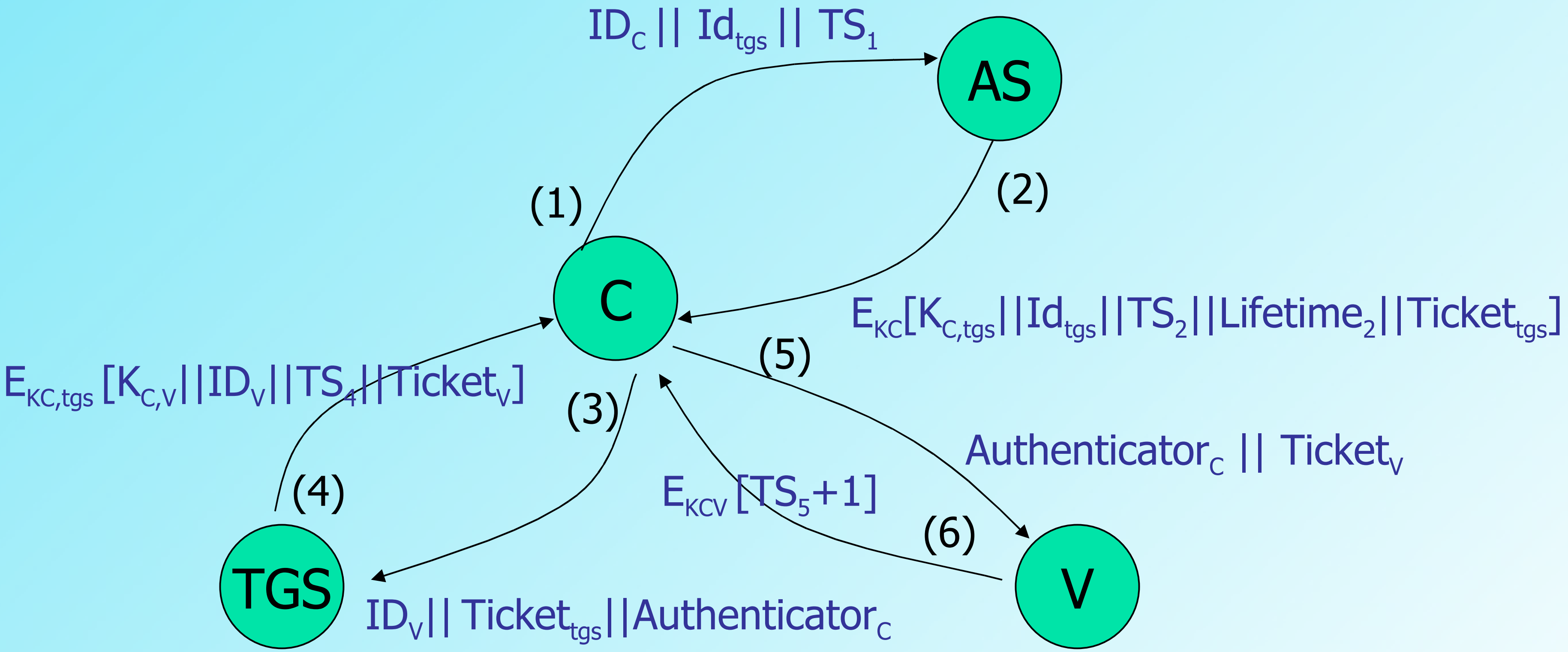
Version 4 Authentication

- Problems:
  - Lifetime associated with the ticket granting ticket – too short, repeated password prompting; too long, vulnerable to capture
  - Server authentication to user – false server could act as a real server

Version 4 Authentication

- Session Key – this is included in the encrypted message, $K_{C,tgs}$ and $K_{C,V}$
- Authenticator – encrypted with the session key it includes the user ID and address of the client and a timestamp. It is used only once – short lifetime

Version 4 Authentication

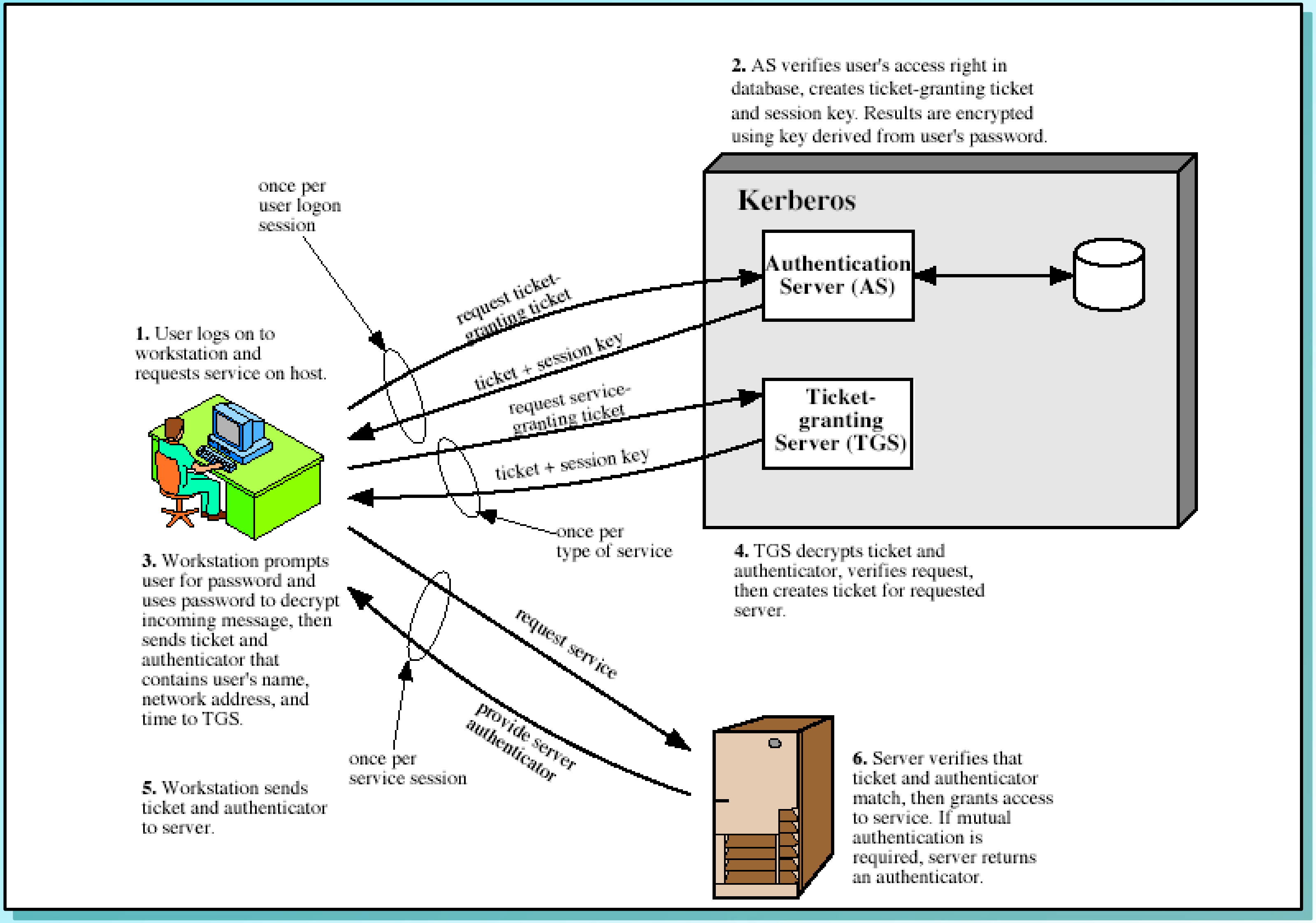


$Ticket_{tgs}=E_{Ktgs}[K_{C,tgs}\ ||ID_C||AD_C||ID_{tgs}||TS_2||Lifetime_2]$

$Ticket_V=E_{KV}[K_{C,V}||ID_C||AD_C||ID_V||TS_4||Lifetime_4]$

$Authenticator_C= E_{KC,tgs}\,[ID_C||AD_C||TS_3]$

Overview of Kerberos



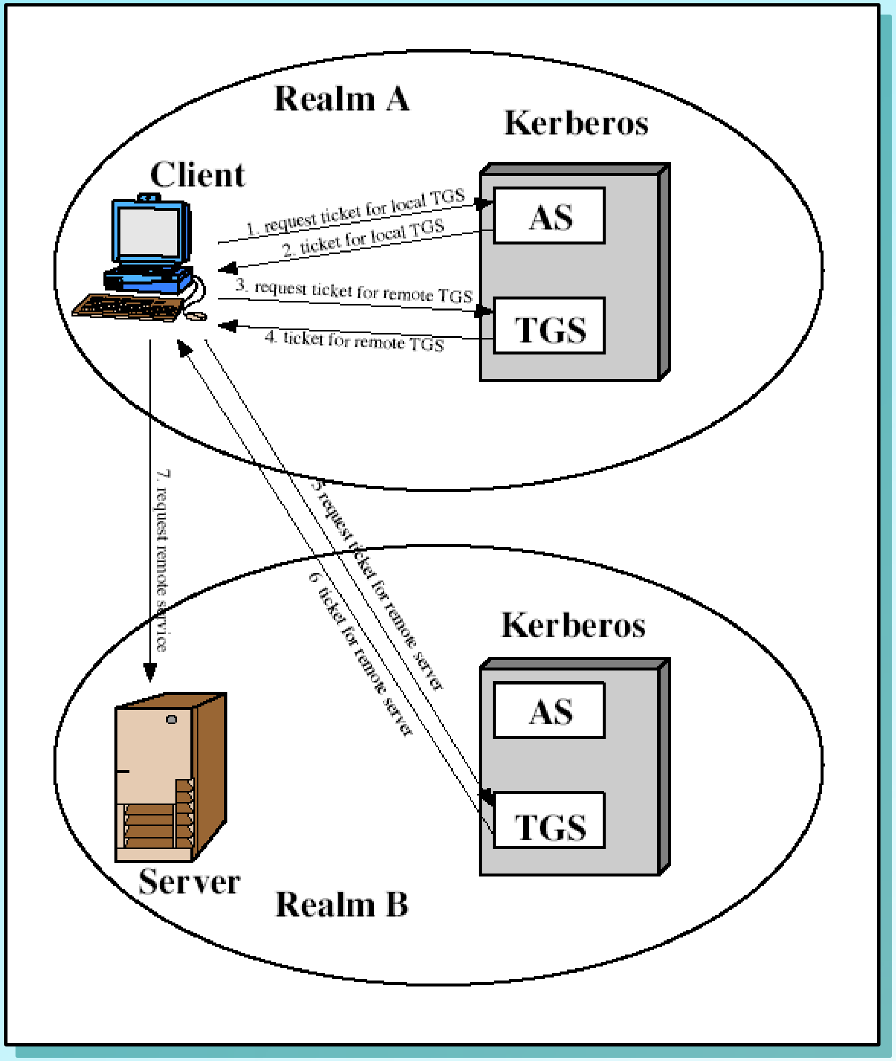
Kerberos Realms

- A realm is a collect of clients and servers under single administration such that
  - Kerberos server has the user ID and hashed password of all participating users in its database *(all users registered with Kerberos)*
  - Kerberos server shares a secret key with each server *(all servers registered with Kerberos)*

Kerberos Realms

- Users in one realm may need access to servers in another realm
- Kerberos server in each interoperating realm shares a secret key with the server in the other realm *(Kerberos servers are registered with each other)*
- The Kerberos server in one realm must trust the Kerberos server in the other realm to authenticate its users

Requesting Service In Another Realm



Kerberos Realms

- Doesn't scale well to many realms
- Given N realms, there must be

  $$N(N-1)/2$$

  secure key exchanges between each of the Kerberos servers

Kerberos Version 5

- Specified in RFC 1510 – 1993
- Does not depend on DES - can use any encryption technique
- Arbitrary ticket lifetime – start and end time
- Authentication forwarding
- Interrealm authentication – eliminates N^2 order of K-to-K relationships

Kerberos Version 5

Number of new improvements:

- Session keys – client and server can negotiate a subsession key, used only for one connection
- Password attacks – preauthentication mechanism
- Ticket flags – expanded functionality

Not Too Shabby, Huh!

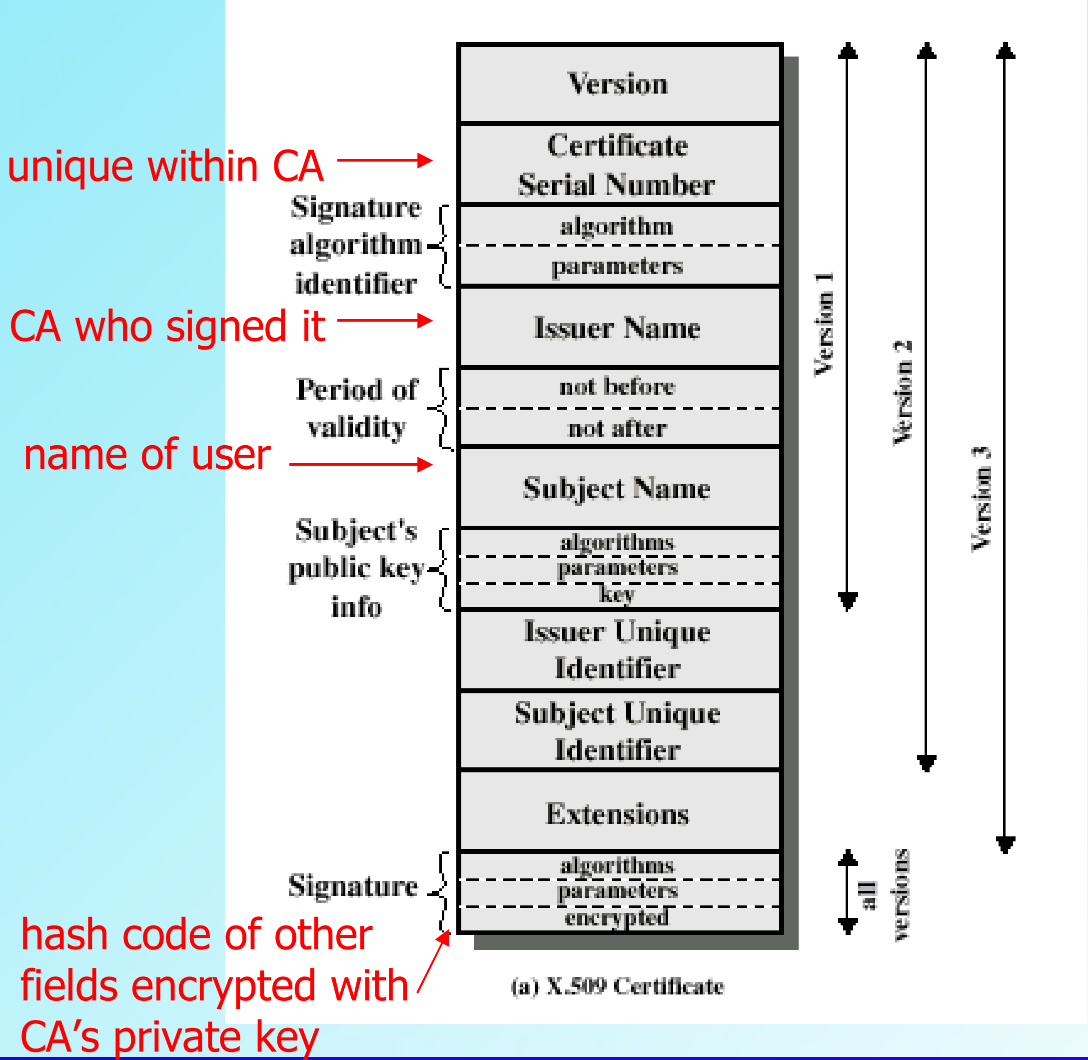
X.509 Authentication Service

- X.509 is part of X.500 series which defines a directory service
- 1988, V2-1993, V3-1995
- Based on public-key cryptography and digital signatures
- Defines a framework for the provision of authentication services
- Repository of public key certificates
- Used in S/MIME, IPSec, SSL and SET

Certificates

- Each certificate contains the *public key of a user* and is signed with the *private key of a trusted certification authority*
- A certificate is associated with each user
- It's the heart of the X.509 scheme

X.509 Formats



(a) X.509 Certificate

Certificate Notation

Y{I} = the signing of I by Y

$$CA<<A>> = CA\ \{V, SN, AI, CA, T_A, A, A_P\}$$

certificate of user A issued by certification authority CA

encrypted hash code

Certificate Characteristics

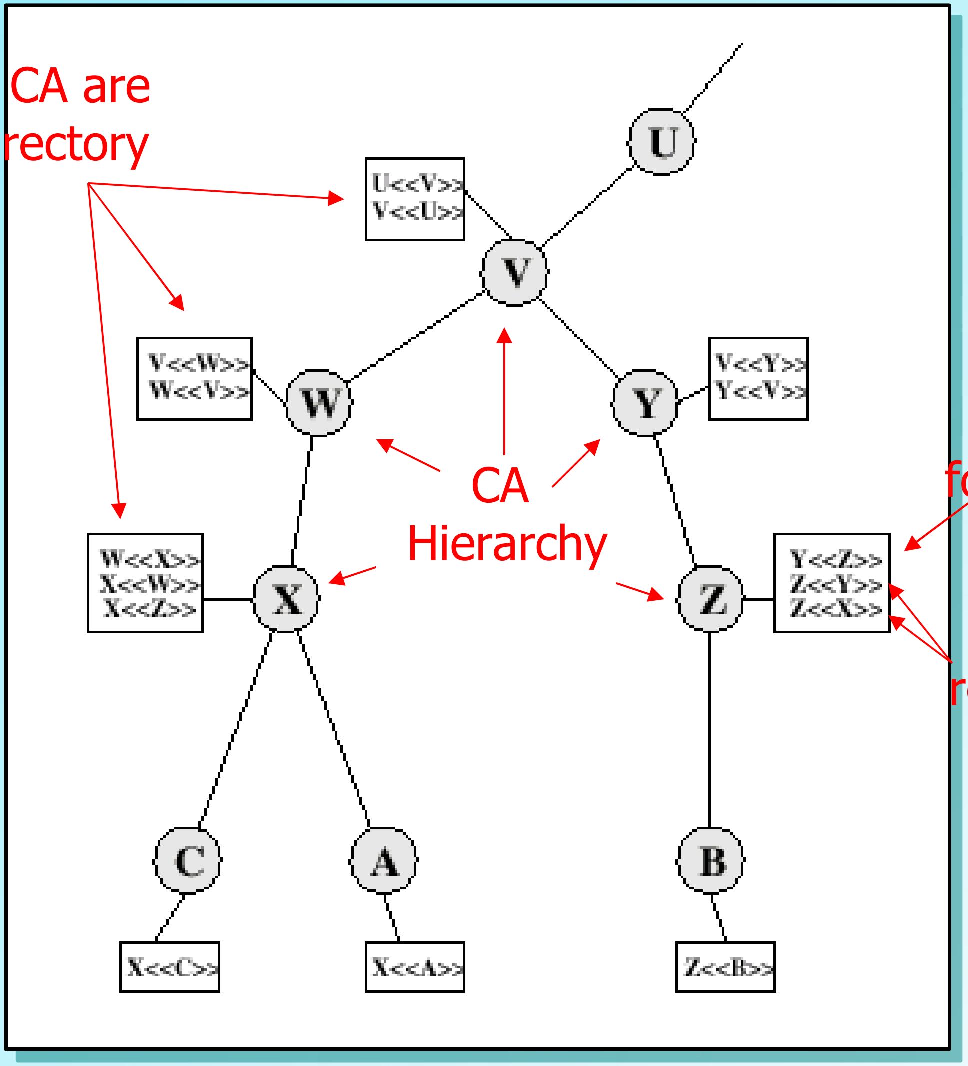
- If you have the public key of the CA, you can recover the user public key that was certified
- Only the certificate authority can modify the certificate
- Placed in a directory without special protection

Certificate Characteristics

- If all users subscribe to the same CA, then there is common trust of that CA
- User can transmit his certificate directly to others
- Assured messages are secure from eavesdropping and unforgeable
- Not all users can subscribe to the same CA

Chain of Certificates

certificates for each CA are maintained in the directory



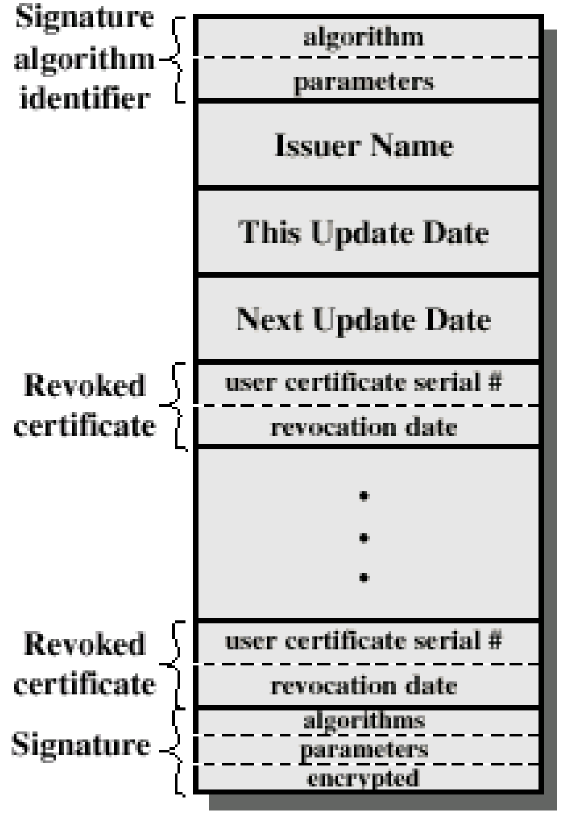
forward certificate

reverse certificates

Revocation of Certificates

- Certificates have a period of validity
- Certificates can also be revoked because:
  - user's key is compromised
  - user no longer certified by CA
  - CA's certificate is assumed to be compromised
- CA maintains a list of revoked certificates and post it on the directory

Certificate Revocation List (CRL)

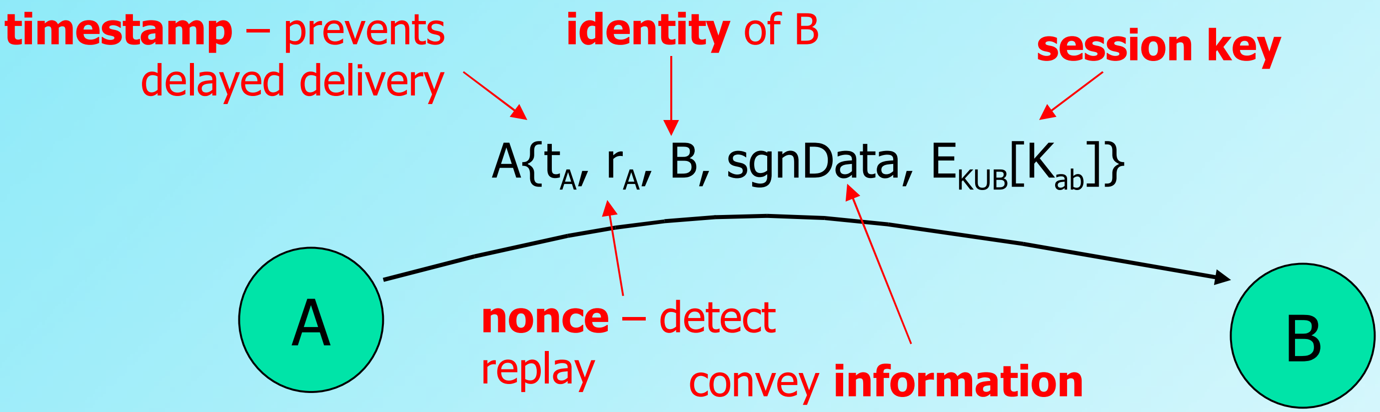


(b) Certificate Revocation List

Authentication Procedures

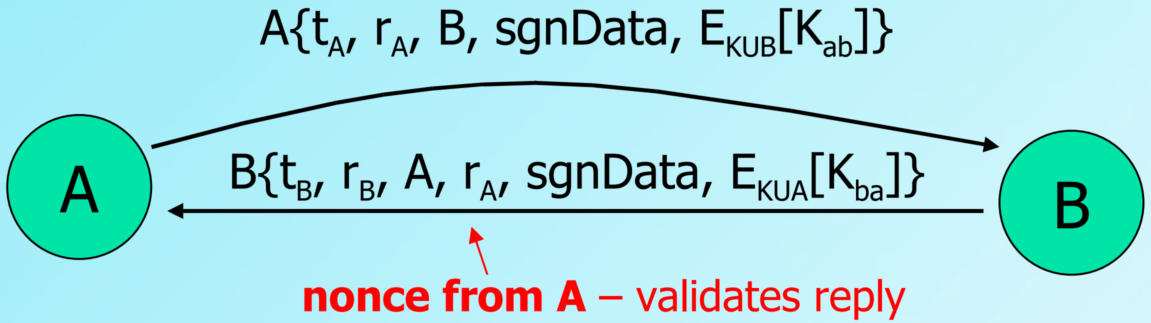
- X.509 includes three authentication procedures making use of public key signatures
- Intended for a variety of applications
- Assumes two parties know each other’s public key

One Way Authentication



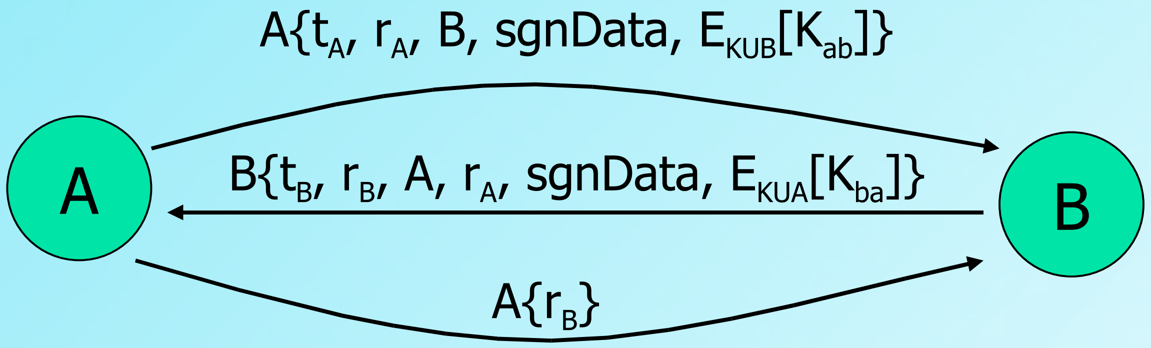
- Establishes the identity (and only the identity) of A and that the message was generated by A
- The message was intended for B
- Establishes the integrity and originality of the message; presents credentials

Two Way Authentication



- Establishes the identity of B and that the reply message was generated by B
- The message was intended for A
- Establishes the integrity and originality of the reply
- Both parties verify the identity of the other

Three Way Authentication



- Final message from A to B is included, with a signed copy of the nonce r_B
- No need for timestamps; each sides echoes back a nonce to prevent replay
- Used when no synchronized clocks available

X.509 Version 3 Requirements

- Subject field needs to convey more information about the key owner
- Subject field needs more info for applications: IP address, URL
- Indicate security policy information (IPSec)
- Set constraints on certificate applicability – limit damage from faulty CA
- Identify separately different keys used by the same owner at different times – key life cycle management

X.509 Version 3 Extensions

- Added optional extensions rather than fixed fields
- `{extension id, criticality indicator, extension value}`
- Three main categories:
  - Key and policy information – *EDI only*
  - Certificate subject and issuer attributes – *alternative names*
  - Certification Path Constraints - *restrictions*

Important URLs

- http://web.mit.edu/kerberos/www/
  Information about Kerberos, including the FAQs, papers and documents and pointers to commercial product sites
- http://www.ietf.org/html.charters/pkix-charter.html
  Information from the IETF about X.509
- http://www.verisign.com/
  One of the leading commercial vendors of X.509
- http://csrc.nist.gov/pki/
  Good source of info on PKI and other crypto subjects

Important URLs

- http://http://primes.utm.edu/
  Prime Number research, records, and resources. Checkout “Prime Curios!” - a collection of curiosities, wonders and trivia related to prime numbers.
- http://www.certicom.com/
  Lots of material on elliptic curve cryptography.

Homework

- Read Chapter Four

No Class Next Week!!!

- I'll be out of town
- Limited access to email
- Next Class is March 20th
- But in the meantime…

Term Paper

- Due Monday, May 1
- Should be about 6-8 pages (9 or 10 font, single space)
- Suggested template: http://www.acm.org/sigs/pubs/proceed/
- This should be an opportunity to explore a selected area
- Send me your topic by March 20th

Term Paper

Possible topics:

- Elliptic Curve Cryptography
- Cyber Forensics
- Digital Rights Management
- Security In Software Development
- Virtualization & Security
- Legal, Ethical Issues Around Security & Privacy
- Wireless/Mobile Security
- Phishing/Identity Theft
- Distributed DoS Attacks
- Electronic Cash
- Anti-Virus Software
- Any Topic Discussed In Class
- Programming Project Can Be Substituted If You Want

Assignment 1

- Pick sun.com and one other site. Using whois and ARIN, get as much information as possible about the IP addressing, the DNS and the site (location, owner, etc.)
- Problems (p83): 3.5,c and 3.6
- Due next class March 6 **(TODAY!)**

See You In Two Weeks

Happy St. Patrick's Day!