CHAPTER 10

REAL-WORLD PROTOCOLS

SSH

SECURE SOCKET LAYER

IPSEC

KERBEROS

REAL TIME SECURITY COMM

- Real time protocol
  - The parties negotiate interactively to authenticate each other and establish a session key
- Security Association (SA)
  - The conversation protected with that session key

REAL-WORLD PROTOCOLS

- Next, we'll look at specific protocols
  - SSH – a simple & useful security protocol
  - SSL – practical security on the Web
  - IPSec – security at the IP layer
  - Kerberos – symmetric key, single sign-on
  - WEP – "Swiss cheese" of security protocols
  - GSM – mobile phone (in)security

SECURE SHELL (SSH)

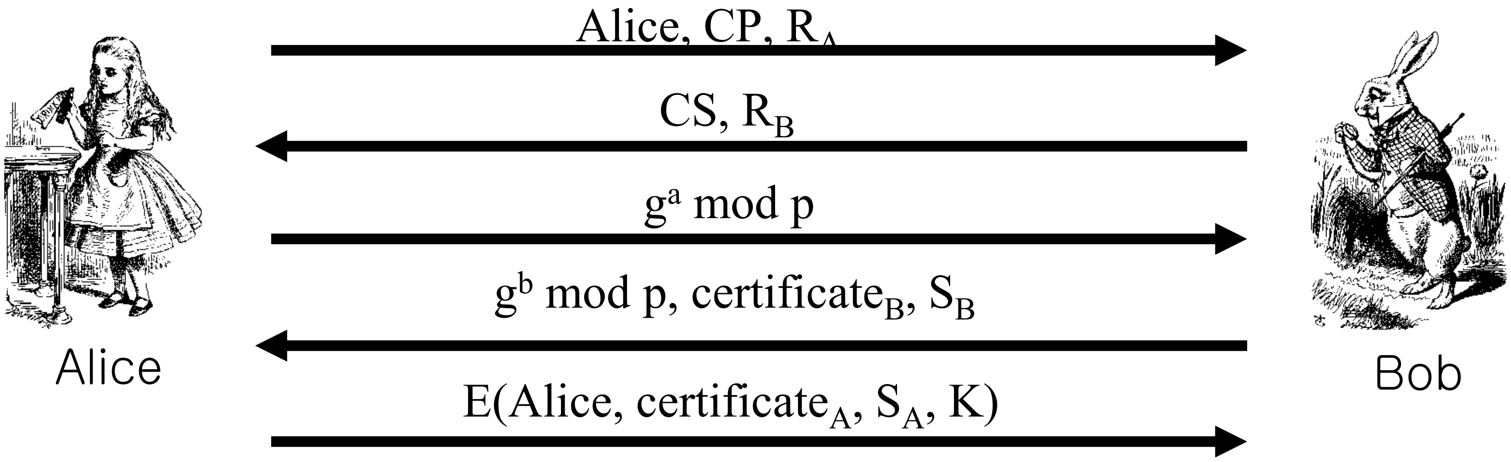
SSH

- Creates a "secure tunnel"
- Insecure command sent thru SSH tunnel are then secure
- SSH used with things like rlogin
  - Why is rlogin insecure without SSH?
  - Why is rlogin secure with SSH?
- SSH is very simple protocol

SSH

- SSH authentication can be based on…
    - Public keys, or
    - Digital certificates, or
    - Passwords
- Here, we consider certificate mode
- Other modes in homework problems
- We consider slightly simplified SSH…

SIMPLIFIED SSH



Alice, CP, R_A

CS, R_B

g^a mod p

g^b mod p, certificate$_B$, S_B

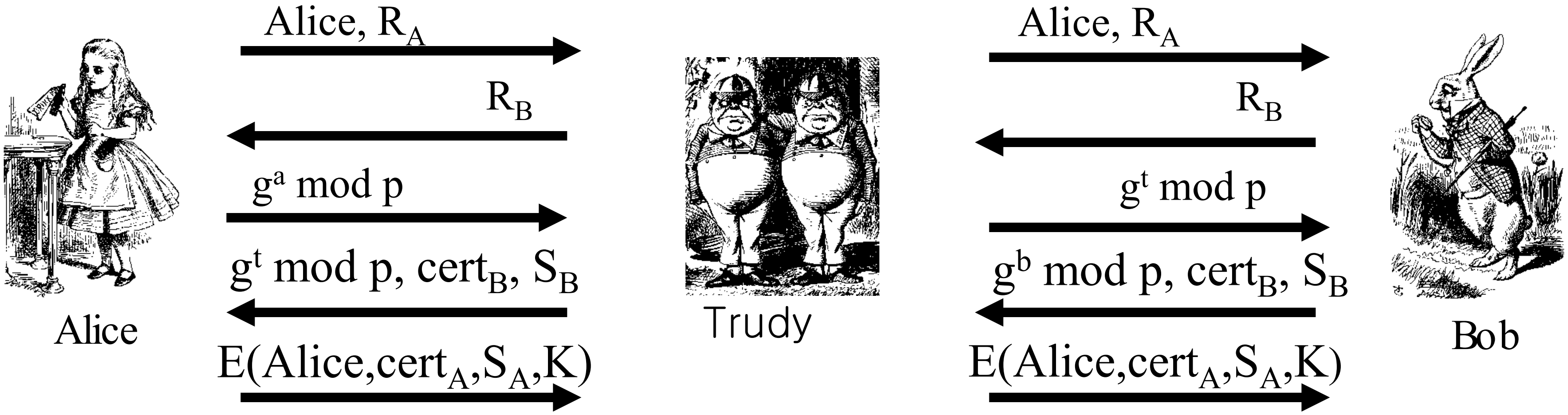
E(Alice, certificate$_A$, S_A, K)

Alice

Bob

- CP = "crypto proposed", and CS = "crypto selected"
- $H = h(\text{Alice},\text{Bob},CP,CS,R_A,R_B,g^a \bmod p,g^b \bmod p,g^{ab} \bmod p)$
- $S_B = [H]_{\text{Bob}}$
- $S_A = [H, \text{Alice}, \text{certificate}_A]_{\text{Alice}}$
- $K = g^{ab} \bmod p$

MIM ATTACK ON SSH?



Alice, R_A

R_B

$g^a \bmod p$

$g^t \bmod p$, cert_B, S_B

$E(\text{Alice},\text{cert}_A,S_A,K)$

Alice

Trudy

Alice, R_A

R_B

$g^t \bmod p$

$g^b \bmod p$, cert_B, S_B

$E(\text{Alice},\text{cert}_A,S_A,K)$

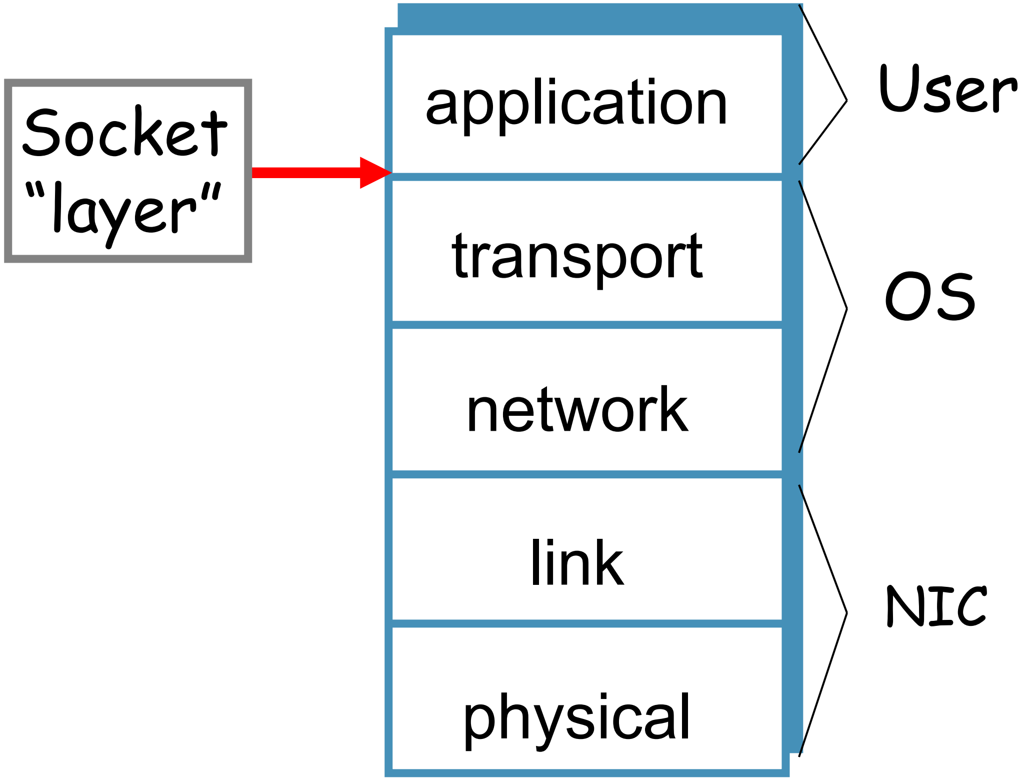
Bob

- Where does this attack fail?
- Alice computes:
  - $H_a = h(\text{Alice},\text{Bob},\text{CP},\text{CS},R_A,R_B,g^a \bmod p,g^t \bmod p,g^{at} \bmod p)$
- But Bob signs:
  - $H_b = h(\text{Alice},\text{Bob},\text{CP},\text{CS},R_A,R_B,g^t \bmod p,g^b \bmod p,g^{bt} \bmod p)$

SECURE SOCKET LAYER

SOCKET LAYER

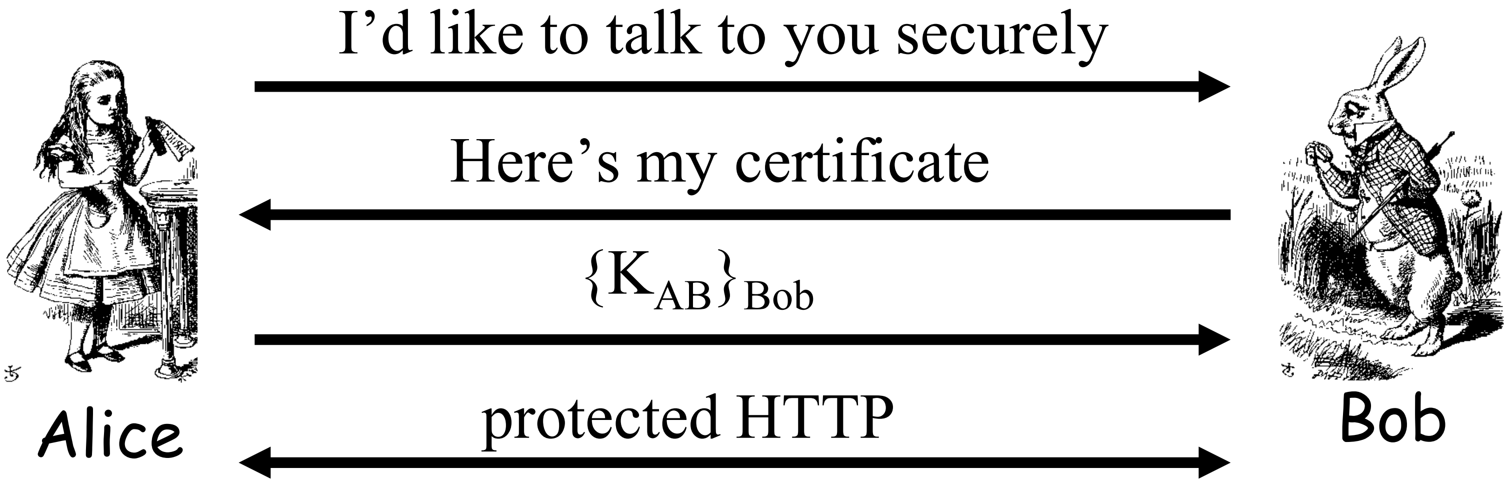
- "Socket layer" lives between application and transport layers
- SSL usually lies between HTTP (application) and TCP (transport)



WHAT IS SSL?

- SSL is the protocol used for majority of secure transactions over the Internet
- For example, if you want to buy a book at amazon.com…
  - You want to be sure you are dealing with Amazon (**authentication**)
  - Your credit card information must be protected in transit (**confidentiality** and/or **integrity**)
  - As long as you have money, Amazon doesn’t care who you are (authentication need not be mutual)

SIMPLE SSL-LIKE PROTOCOL


I'd like to talk to you securely

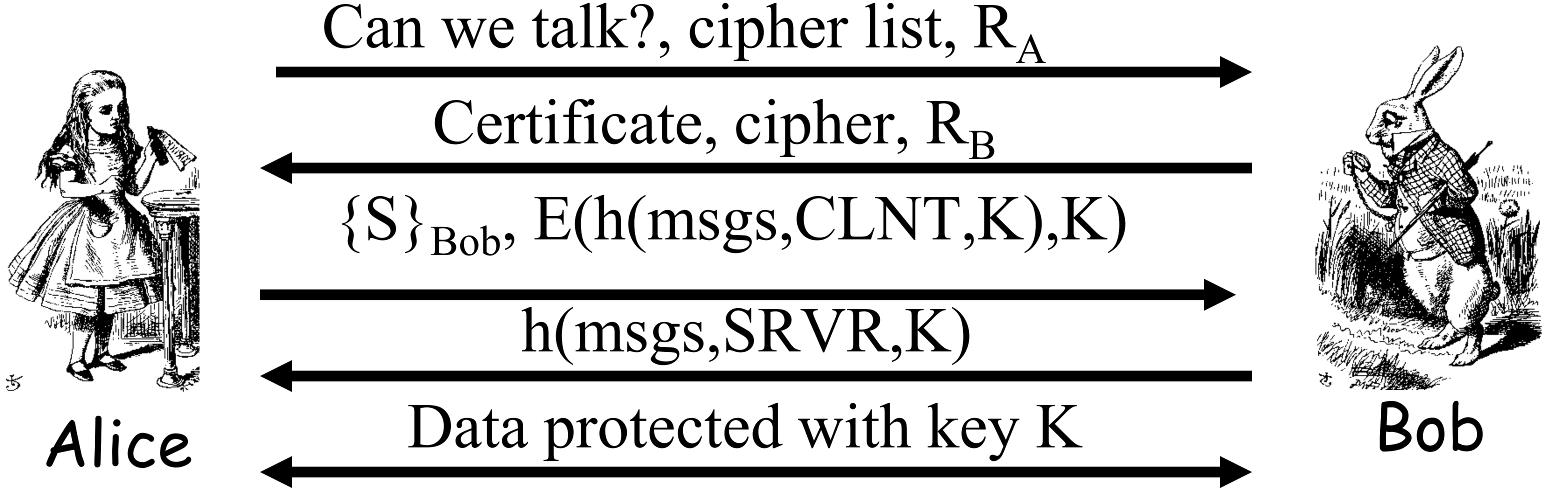
Here's my certificate

$\{K_{AB}\}_{Bob}$

protected HTTP

- Is Alice sure she's talking to Bob?
- Is Bob sure he's talking to Alice?

SIMPLIFIED SSL PROTOCOL



Can we talk?, cipher list, R_A

Certificate, cipher, R_B

$\{S\}_{Bob}$, E(h(msgs,CLNT,K),K)

h(msgs,SRVR,K)

Data protected with key K

Alice

Bob

- S is **pre-master secret**
- $K = h(S,R_A,R_B)$
- msgs = all previous messages
- CLNT and SRVR are constants

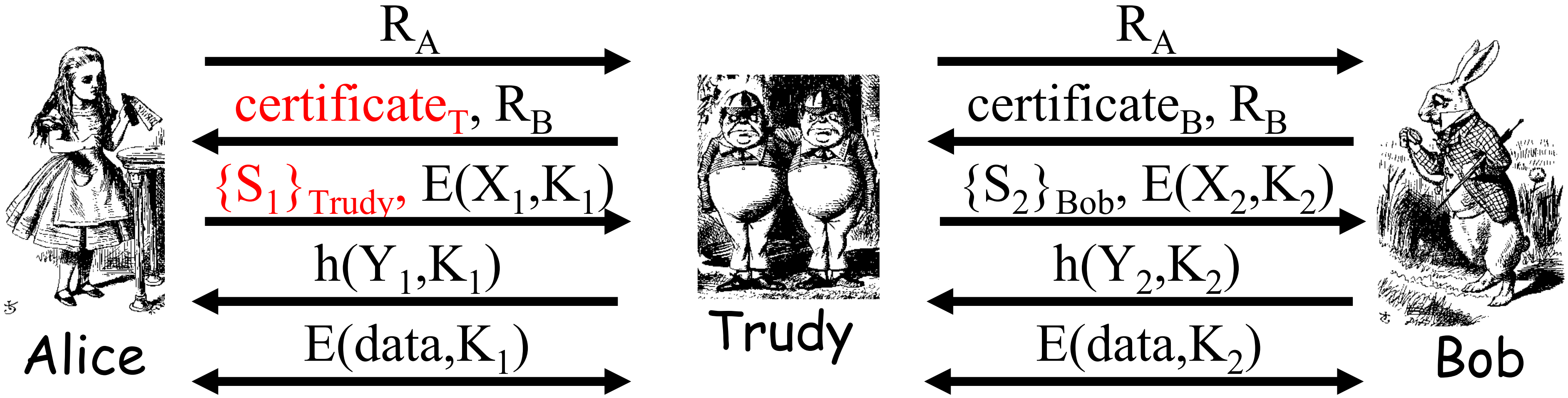
SSL KEYS

- 6 “keys” derived from K = hash(S,R_A,R_B)
  - 2 encryption keys: send and receive
  - 2 integrity keys: send and receive
  - 2 IVs: send and receive
  - Why different keys in each direction?
- **Q:** Why is h(msgs,CLNT,K) encrypted (and integrity protected)?
- **A:** Apparently, it adds no security…

SSL AUTHENTICATION

- Alice authenticates Bob, not vice-versa
  - How does client authenticate server?
  - Why does server not authenticate client?
- Mutual authentication is possible: Bob sends **certificate request** in message 2
  - This requires client to have certificate
  - If server wants to authenticate client, server could instead require (encrypted) password

SSL MIM ATTACK



R_A

$\text{certificate}_T, R_B$

$\{S_1\}_{\text{Trudy}}, E(X_1,K_1)$

$h(Y_1,K_1)$

$E(\text{data},K_1)$

R_A

$\text{certificate}_B, R_B$

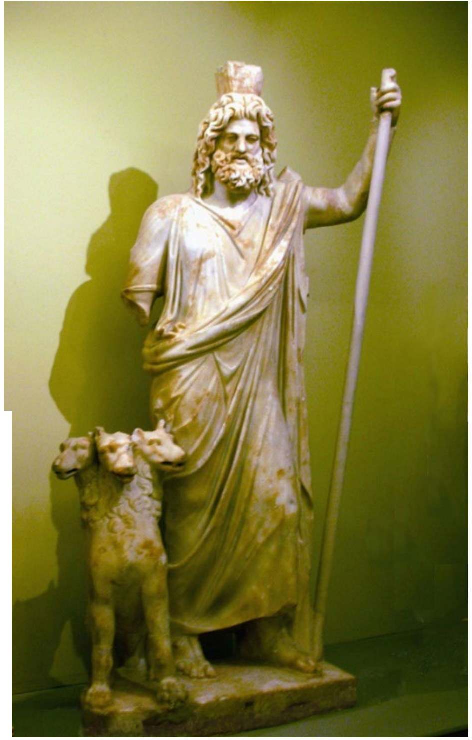
$\{S_2\}_{\text{Bob}}, E(X_2,K_2)$

$h(Y_2,K_2)$

$E(\text{data},K_2)$

- **Q:** What prevents this MiM attack?
- **A:** Bob's certificate must be signed by a certificate authority (such as Verisign)
- What does Web browser do if sig. not valid?
- What does user do if signature is not valid?

KERBEROS



Kerberos

In Greek mythology, a many headed dog, the guardian of the entrance of Hades

KERBEROS

- In Greek mythology, Kerberos is 3-headed dog that guards entrance to Hades
  - “Wouldn’t it make more sense to guard the exit?”
- In security, Kerberos is an authentication system based on symmetric key crypto
  - Originated at MIT
  - Based on work by Needham and Schroeder
  - Relies on a **Trusted Third Party (TTP)**

MOTIVATION FOR KERBEROS

- Authentication using public keys
  - N users ⇒ N key pairs
- Authentication using symmetric keys
  - N users requires about N^2 keys
- Symmetric key case **does not scale!**
- Kerberos based on symmetric keys but only requires N keys for N users
  - But must rely on TTP
  - Advantage is that no PKI is required

KERBEROS KDC

- Kerberos Key Distribution Center or **KDC**
  - Acts as a TTP
  - TTP must not be compromised!
  - KDC shares symmetric key K_A with Alice, key K_B with Bob, key K_C with Carol, etc.
  - Master key K_{KDC} known only to KDC
  - KDC enables authentication and session keys
  - Keys for confidentiality and integrity
  - In practice, the crypto algorithm used is DES

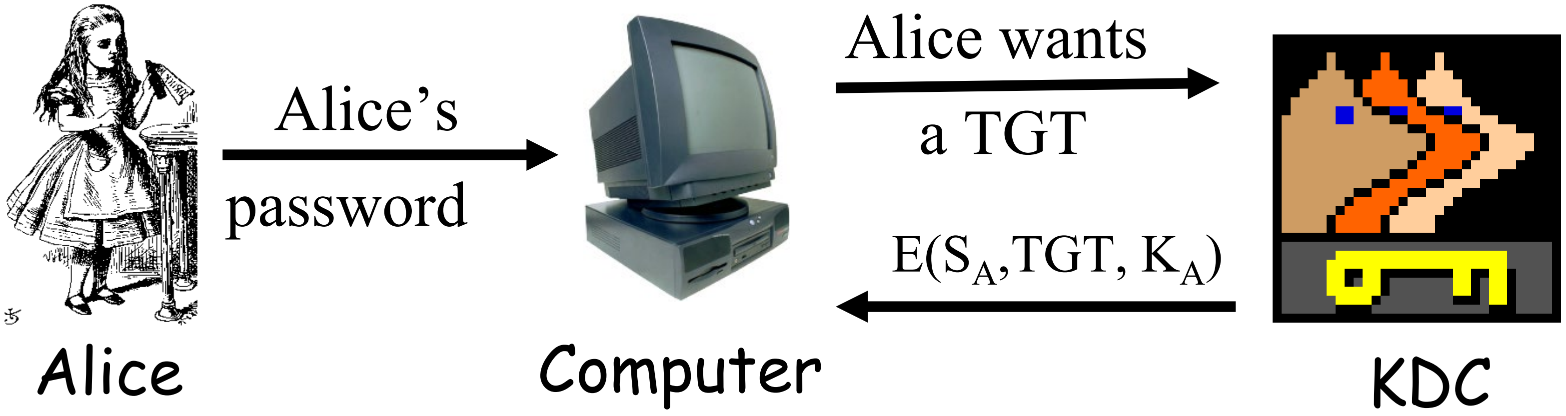
KERBEROS TICKETS

- KDC issues a ticket containing info needed to access a network resource
- KDC also issues ticket-granting tickets or **TGTs** that are used to obtain tickets
- Each TGT contains
  - Session key
  - User's ID
  - Expiration time
- Every TGT is encrypted with K_{KDC}
  - TGT can only be read by the KDC

KERBERIZED LOGIN

- Alice enters her password
- Alice's workstation
  - Derives K_A from Alice's password
  - Uses K_A to get TGT for Alice from the KDC
- Alice can then use her TGT (credentials) to securely access network resources
- **Plus:** Security is transparent to Alice
- **Minus:** KDC must be secure --- it's trusted!

KERBERIZED LOGIN



Alice’s password

Alice wants a TGT

E(S_A,TGT, K_A)

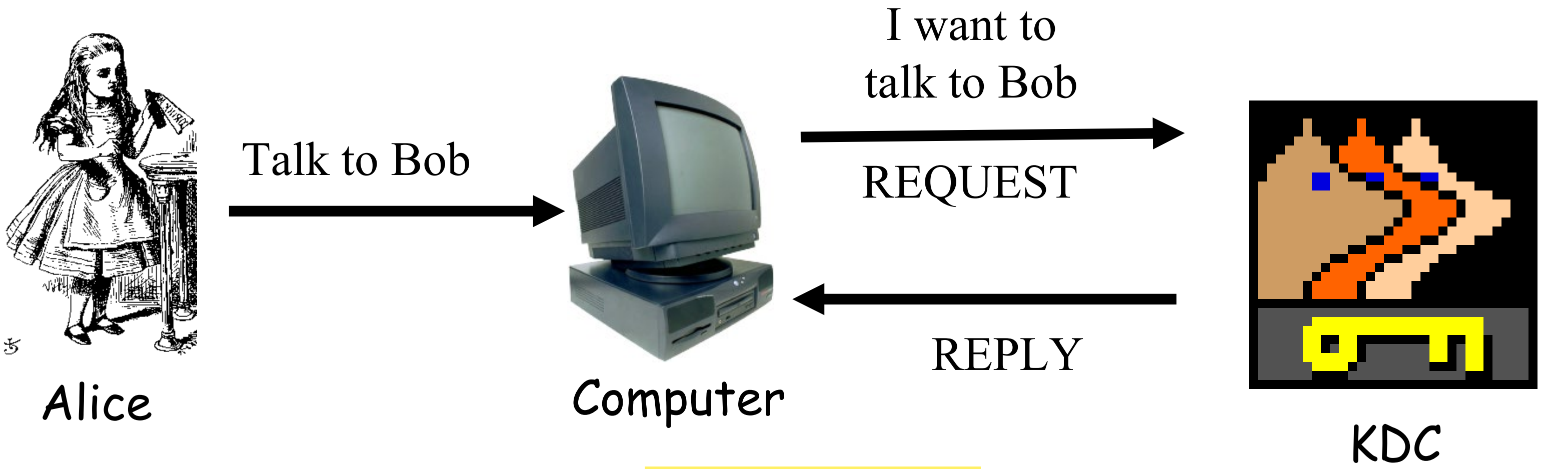
Alice

Computer

KDC

- Kerberos used for authentication
  - Key K_A derived from Alice’s password
  - KDC creates session key S_A
  - Workstation decrypts S_A, TGT, forgets K_A
  - TGT = E(“Alice”, S_A, K_{KDC})

ALICE REQUESTS TICKET TO BOB

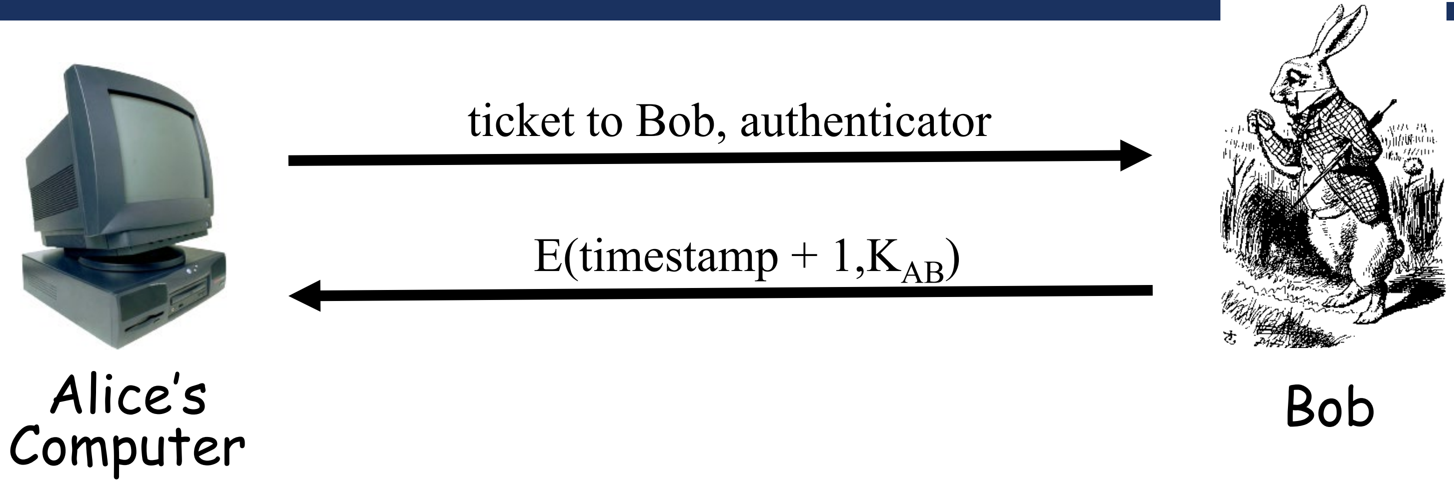


- REQUEST = (TGT, authenticator) where

  authenticator = E(timestamp, S_A)

- REPLY = E("Bob", K_{AB}, ticket to Bob, S_A)
- ticket to Bob = E("Alice", K_{AB}, K_B)
- KDC gets S_A from TGT to verify timestamp

ALICE USES TICKET TO BOB



ticket to Bob, authenticator

E(timestamp + 1,K_{AB})

Alice's Computer

Bob

- ticket to Bob = E("Alice", K_{AB}, K_B)
- authenticator = E(timestamp, K_{AB})
- Bob decrypts "ticket to Bob" to get K_{AB} which he then uses to verify timestamp

KERBEROS

- Session key S_A used for
  - authentication
  - Can also be used for confidentiality/integrity
- Timestamps used for
  - mutual authentication
- Recall that timestamps reduce number of messages
  - Acts like a nonce that is known to both sides
  - Note: **time** is a security-critical parameter!

KERBEROS KEYS

- In Kerberos, K_A = h(Alice's password)
- Could instead generate random K_A and
  - Compute K_h = h(Alice's password)
  - And workstation stores $E(K_A, K_h)$
- Then K_A need not change (on workstation or KDC) when Alice changes her password
- This alternative approach is often used in applications (but not in Kerberos)