Real-World Protocols

- Next, we look at real 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)

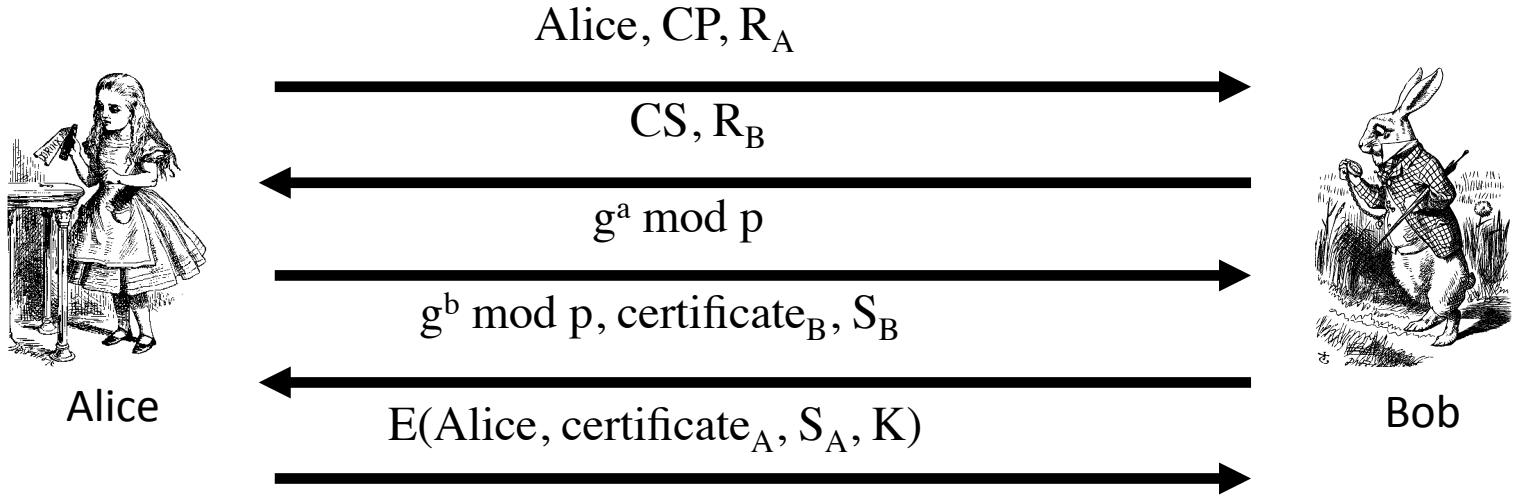
SSH

- 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 a relatively simple protocol

SSH

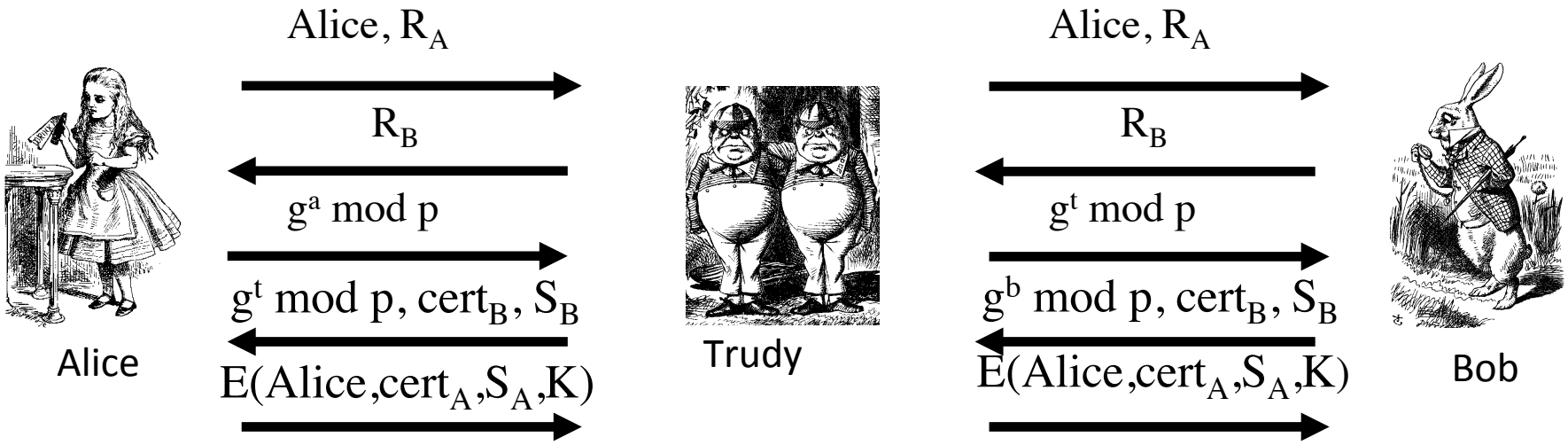
- SSH authentication can be based on:
  - Public keys, or
  - Digital certificates, or
  - Passwords
- Here, we consider ***certificate*** mode
  - Other modes, see homework problems
- We consider slightly simplified SSH…

Simplified SSH



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

MiM Attack on SSH?

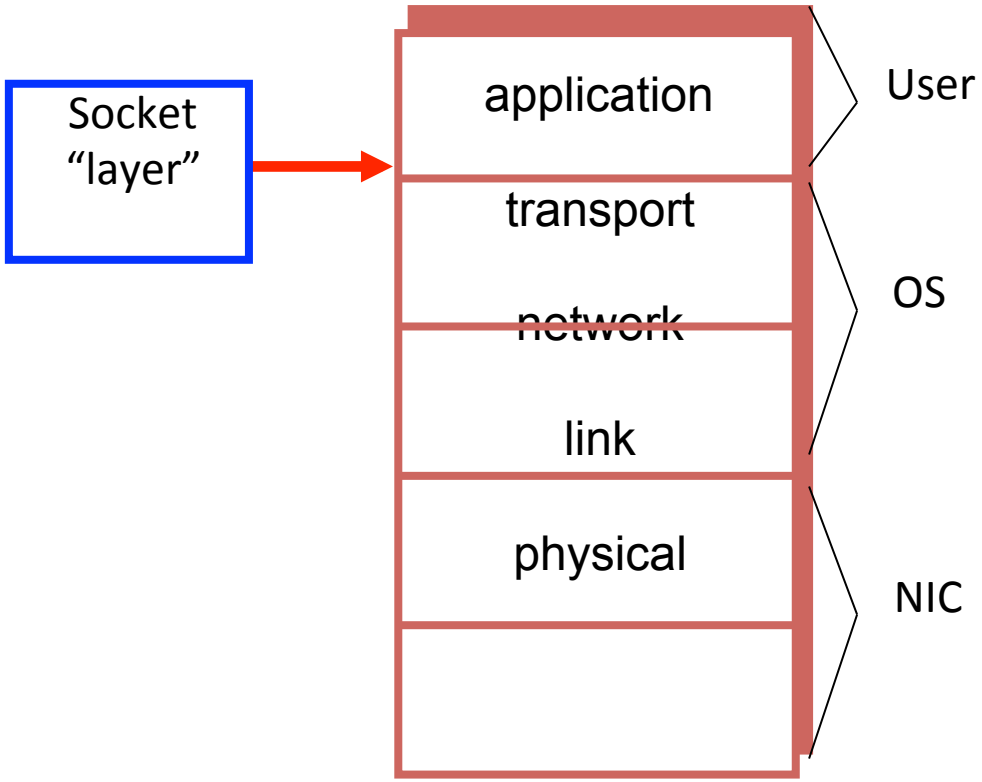


- Where does this attack fail?
- Alice computes:
  - $H_a = h(Alice,Bob,CP,CS,R_A,R_B,g^a \bmod p,g^t \bmod p,g^{at} \bmod p)$
- But Bob signs:
  - $H_b = h(Alice,Bob,CP,CS,R_A,R_B,g^t \bmod p,g^b \bmod p,g^{bt} \bmod p)$

Secure Socket Layer

Socket layer

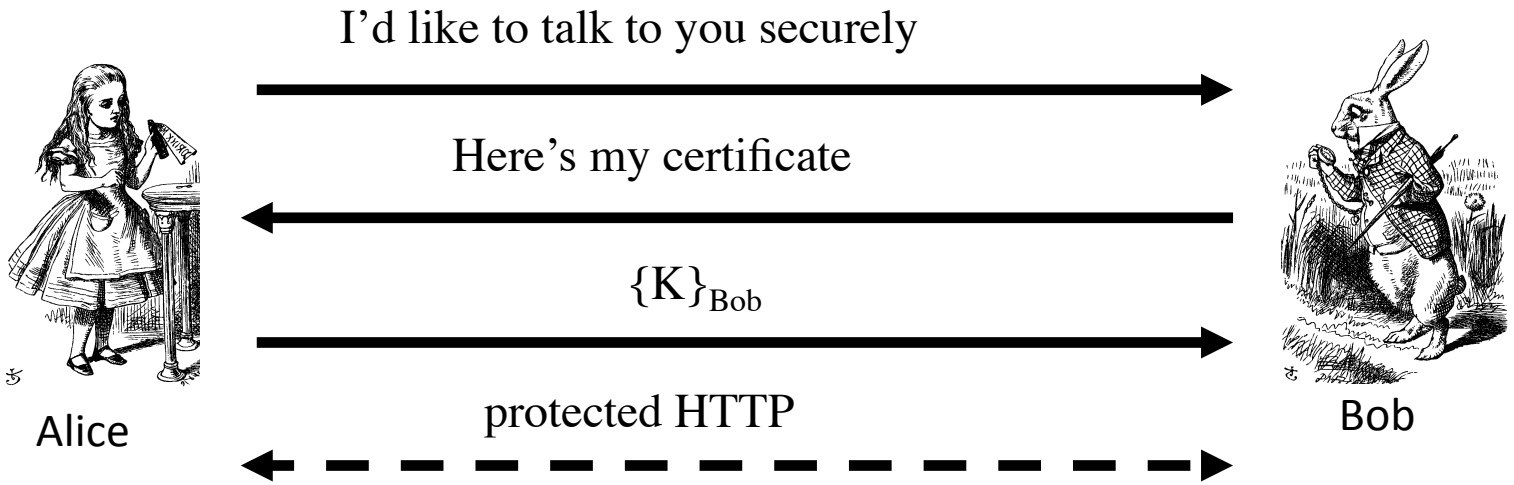
- "Socket layer" lives between application and transport layers
- SSL usually between HTTP and TCP



What is SSL?

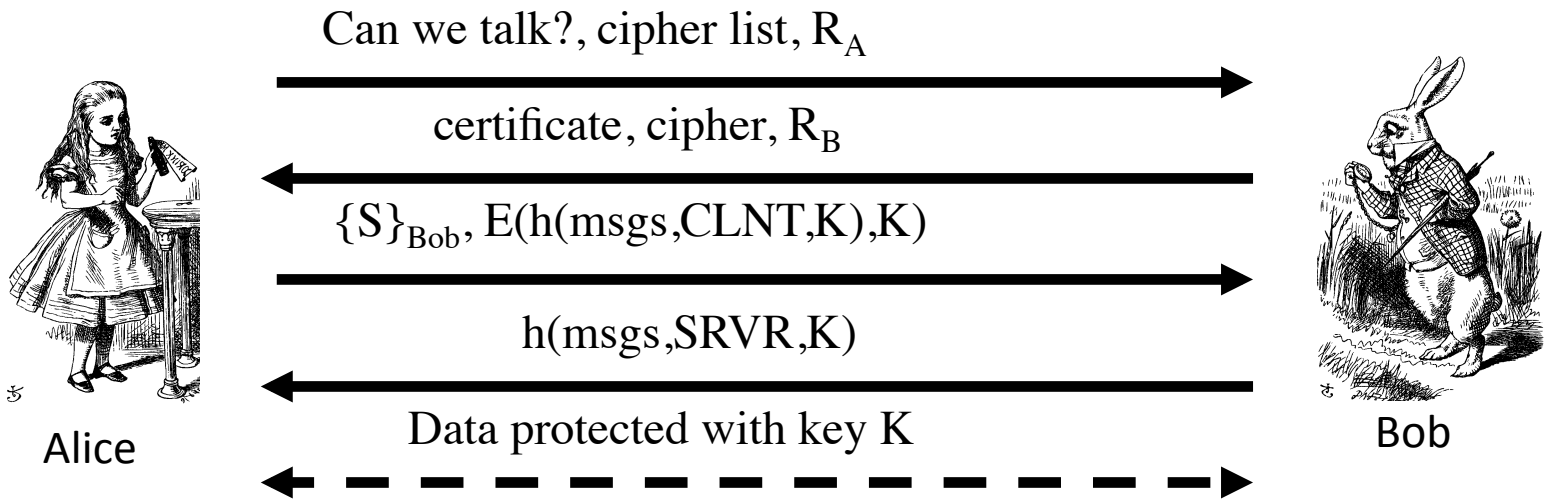
- SSL is the protocol used for majority of secure transactions on the Internet
- For example, if you want to buy a book at amazon.com…
  - You want to be sure you are dealing with Amazon (**authentication**)
  - Your credit card information must be protected in transit (**confidentiality** and/or **integrity**)
  - As long as you have money, Amazon does not care who you are
  - So, no need for mutual authentication

Simple SSL-like Protocol



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

Simplified SSL Protocol



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

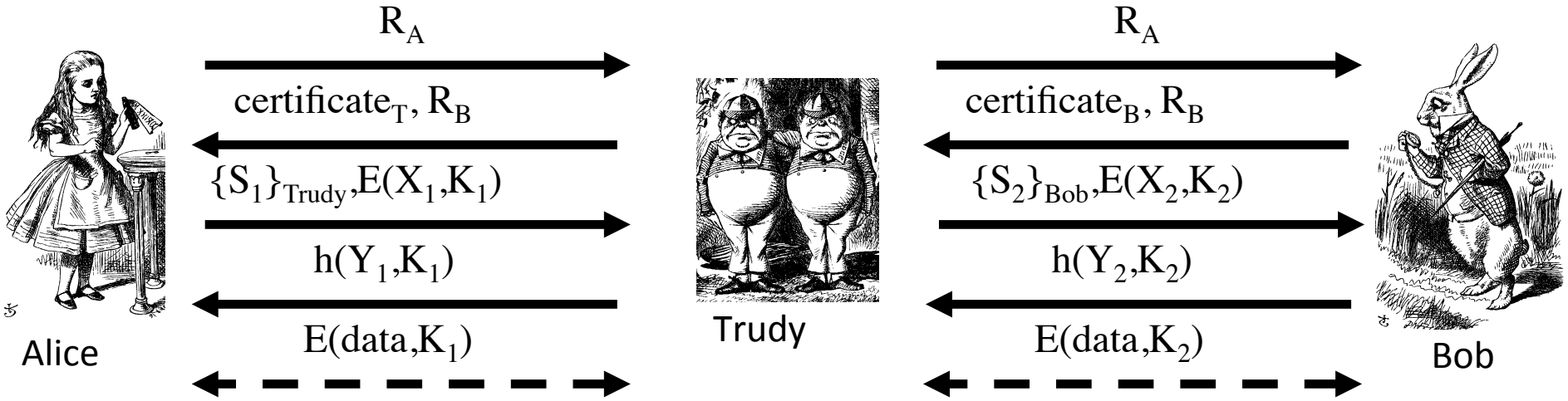
SSL Keys

- 6 "keys" derived from $K = h(S, R_A, R_B)$
  - 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?
- **A:** Apparently, it adds no security…

SSL Authentication

- Alice authenticates Bob, not vice-versa
  - How does client authenticate server?
  - Why would server not authenticate client?
- Mutual authentication is possible: Bob sends **certificate request** in message 2
  - Then client must have a valid certificate
  - But, if server wants to authenticate client, server could instead require password

SSL MiM Attack?

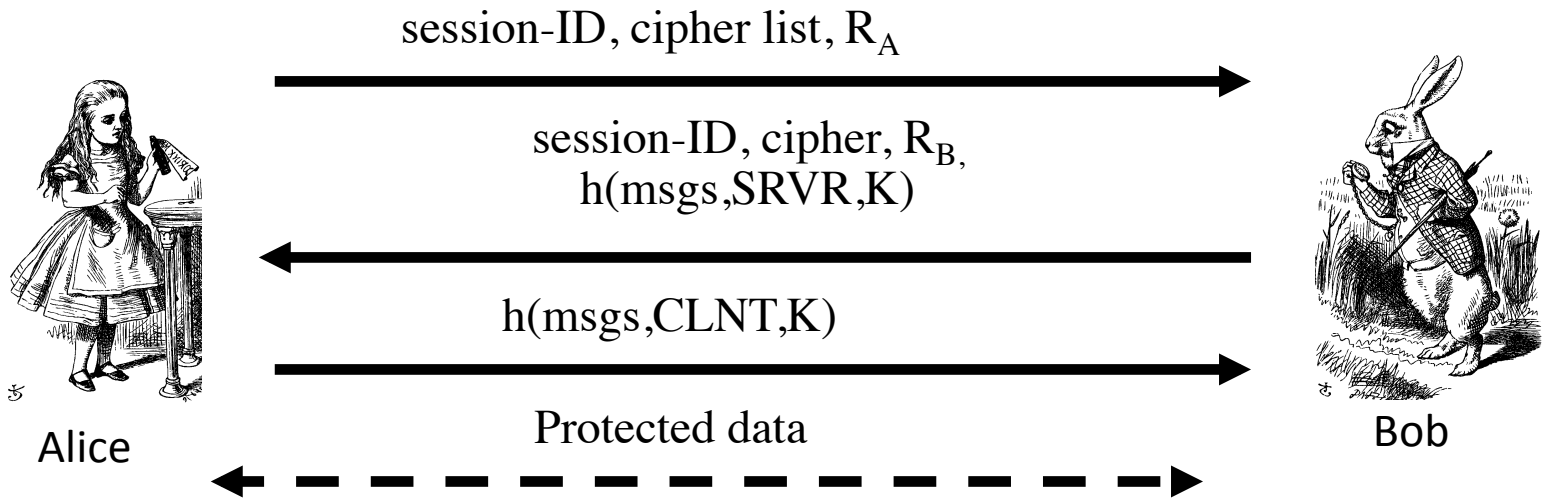


- **Q:** What prevents this MiM "attack"?
- **A:** Bob's certificate must be signed by a certificate authority (CA)
- What does browser do if signature not valid?
- What does user do when browser complains?

SSL Sessions vs Connections

- SSL **session** is established as shown on previous slides
- SSL designed for use with HTTP 1.0
- HTTP 1.0 often opens multiple simultaneous (parallel) **connections**
  - Multiple connections per session
- SSL session is costly, public key operations
- SSL has an efficient protocol for opening new connections ***given an existing session***

SSL Connection



- Assuming SSL **session** exists
- So, S is already known to Alice and Bob
- Both sides must remember session-ID
- Again, $K = h(S,R_A,R_B)$
  - No public key operations! (relies on known S)

Kerberos

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 protocol 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 $\Rightarrow$ N key pairs
- Authentication using symmetric keys
  - N users requires (on the order of) N^2 keys
- Symmetric key case **does not scale**
- Kerberos based on symmetric keys but only requires N keys for N users
  - \- Security depends on TTP
  - \+ No PKI is needed

Kerberos KDC

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

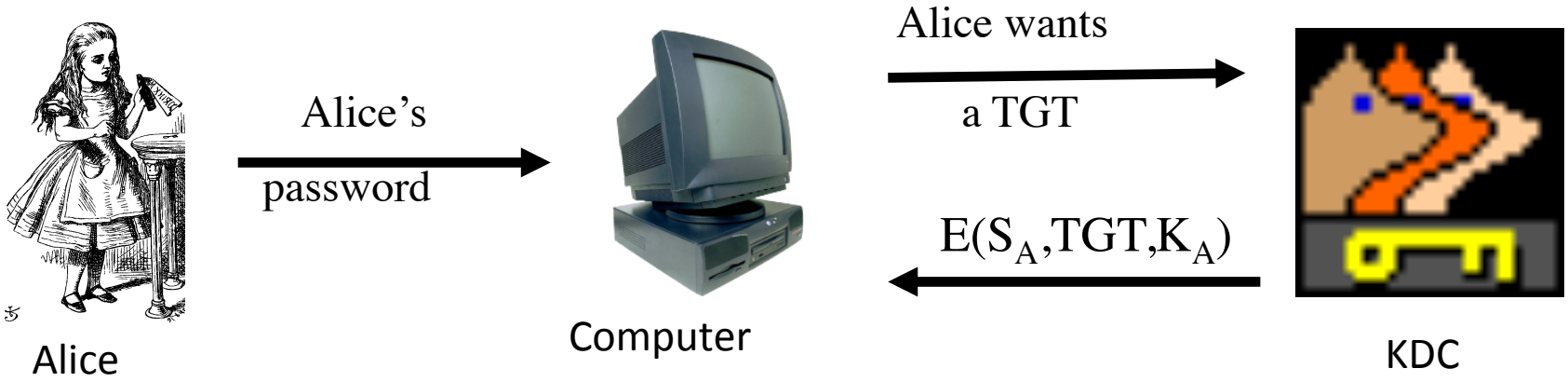
Kerberos Tickets

- KDC issue **tickets** containing info needed to access network resources
- 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}
  - So, TGT can only be read by the KDC

Kerberized Login

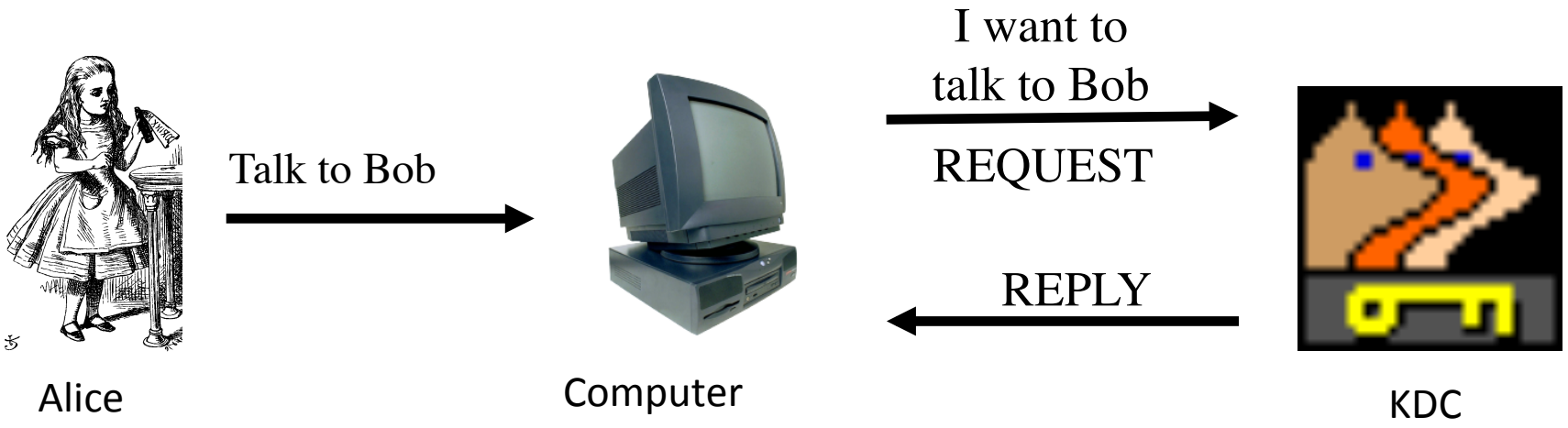
- Alice enters her password
- Then Alice's computer does following:
  - Derives K_A from Alice's password
  - Uses K_A to get TGT for Alice from KDC
- Alice then uses her TGT (credentials) to securely access network resources
- **Plus:** Security is transparent to Alice
- **Minus:** KDC ***must*** be secure — it's trusted!

Kerberized Login



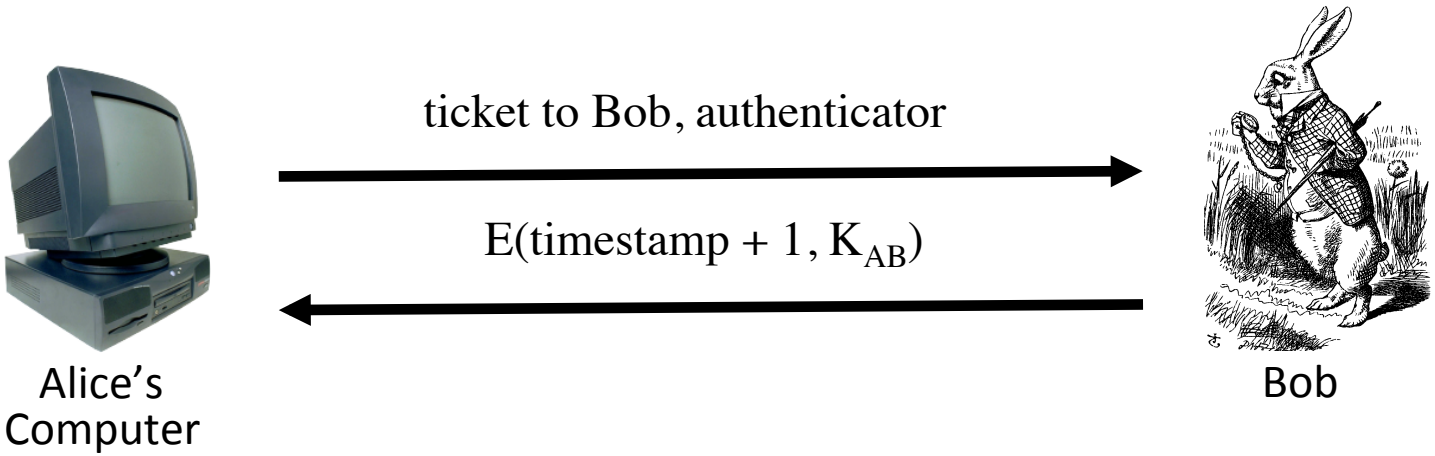
- Key $K_A = h(\text{Alice's password})$
- KDC creates session key S_A
- Alice's computer decrypts S_A and TGT
  - Then it forgets K_A
- $TGT = E(\text{“Alice”}, S_A, K_{KDC})$

Alice Requests "Ticket to Bob"



- REQUEST = (TGT, authenticator)
  - 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 = 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

- Key S_A used in authentication
  - For confidentiality/integrity
- Timestamps for authentication and replay protection
- Recall, that timestamps…
  - Reduce the number of messages—like a nonce that is known in advance
  - But, "time" is a security-critical parameter

Kerberos Questions

- When Alice logs in, KDC sends $E(S_A, TGT, K_A)$ where $TGT = E(\text{“Alice”}, S_A, K_{KDC})$

  Q: Why is TGT encrypted with K_A?

  A: Extra work for no added security!

- In Alice’s “Kerberized” login to Bob, why can Alice remain anonymous?

- Why is “ticket to Bob” sent to Alice?

  – Why doesn’t KDC send it directly to Bob?

Kerberos Alternatives

- Could have Alice's computer remember password and use that for authentication
  - Then no KDC required
  - But hard to protect passwords
  - Also, does not scale
- Could have KDC remember session key instead of putting it in a TGT
  - Then no need for TGT
  - But **stateless** KDC is major feature of Kerberos

Kerberos Keys

- In Kerberos, $K_A = h(\text{Alice's password})$
- Could instead generate random K_A
  - Compute $K_h = h(\text{Alice's password})$
  - And Alice's computer stores $E(K_A, K_h)$
- Then K_A need not change when Alice changes her password
  - But $E(K_A, K_h)$ must be stored on computer
- This alternative approach is often used
  - But not in Kerberos