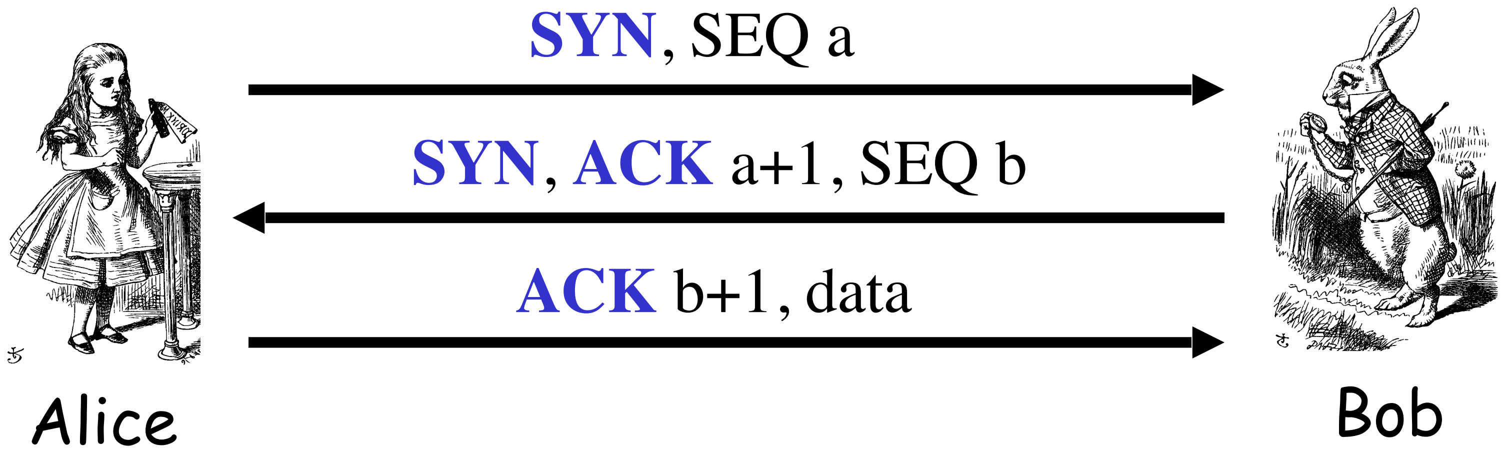
Authentication and TCP

TCP-based Authentication

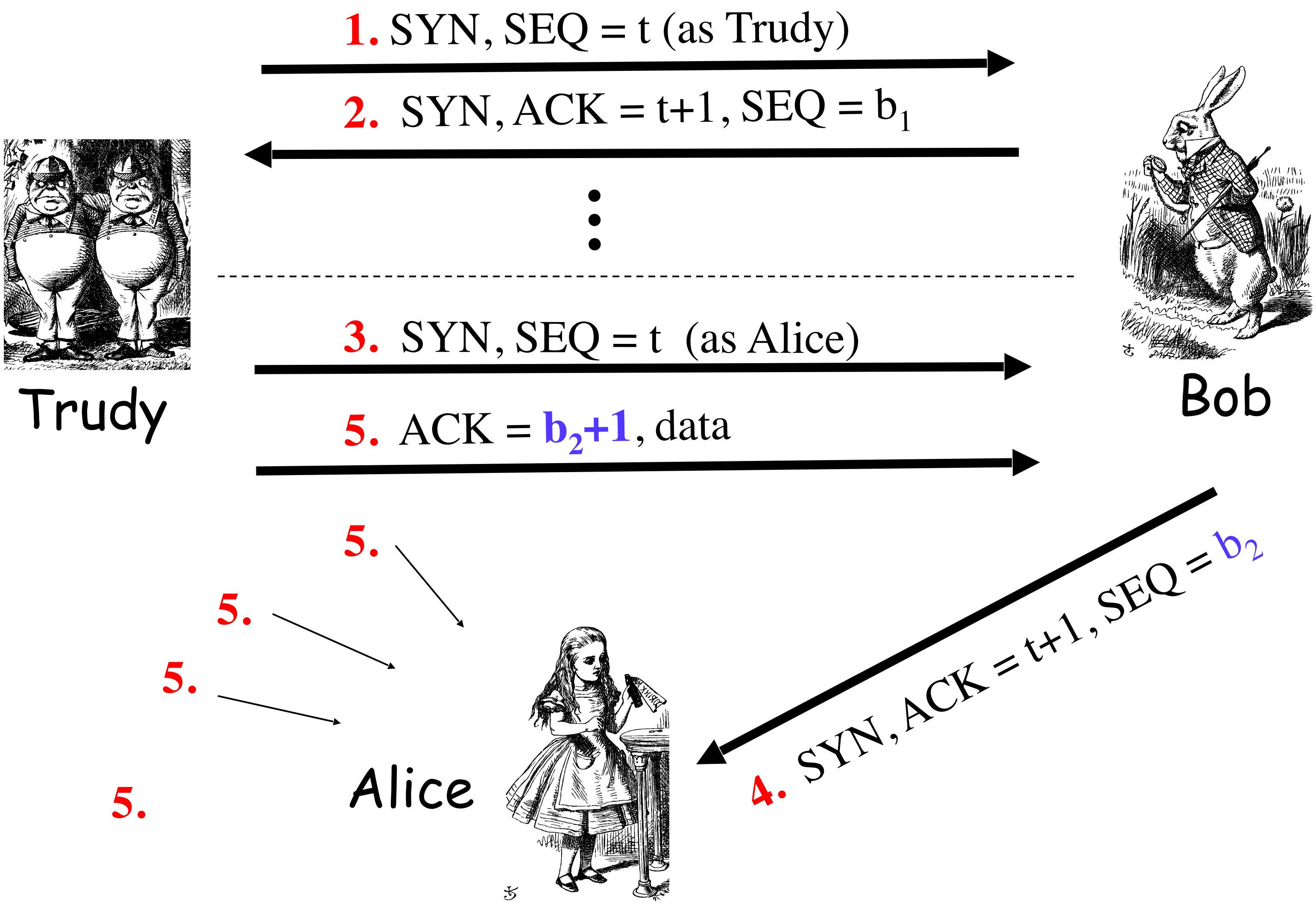
- TCP not intended for use as an authentication protocol
- But IP address in TCP connection may be (mis)used for authentication
- Also, one mode of IPSec relies on IP address for authentication

TCP 3-way Handshake

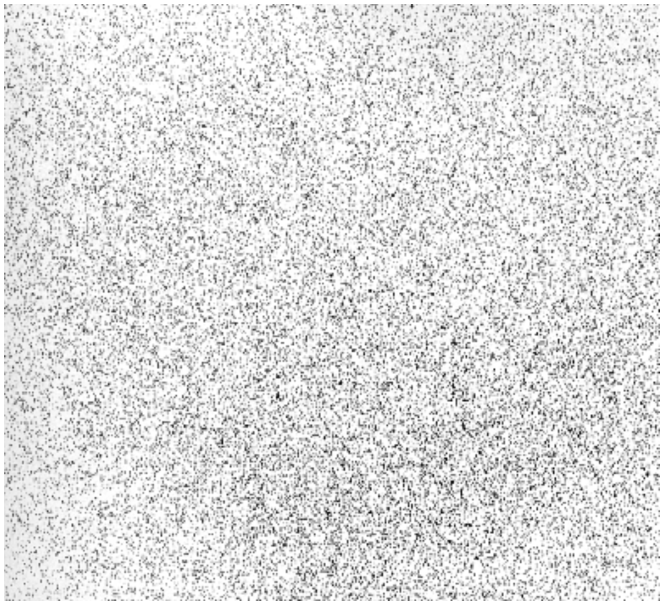


- Initial sequence numbers: SEQ a and SEQ b
  - Supposed to be selected at random
- If not, might have problems…

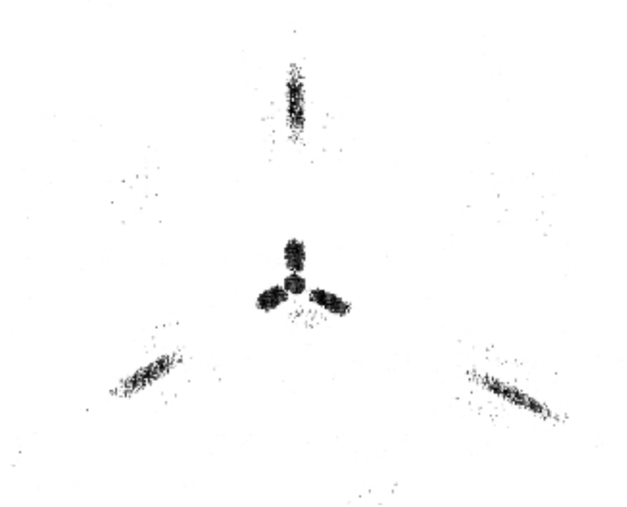
TCP Authentication Attack



TCP Authentication Attack



Random SEQ numbers



Initial SEQ numbers
Mac OS X

- If initial SEQ numbers not very random…
- …possible to guess initial SEQ number…
- …and previous attack will succeed

TCP Authentication Attack

- Trudy cannot see what Bob sends, but she can send packets to Bob, while posing as **Alice**
- Trudy must prevent Alice from receiving Bob's response (or else connection will terminate)
- If **password** (or other authentication) required, this attack fails
- If TCP connection is relied on for authentication, then attack might succeed
- **Bad idea** to rely on TCP for authentication

Best Authentication Protocol?

- It depends on…
  - The sensitivity of the application/data
  - The delay that is tolerable
  - The cost (computation) that is tolerable
  - What crypto is supported (public key, symmetric key, …)
  - Whether mutual authentication is required
  - Whether PFS, anonymity, etc., are concern
- …and possibly other factors

Secure Shell (SSH)

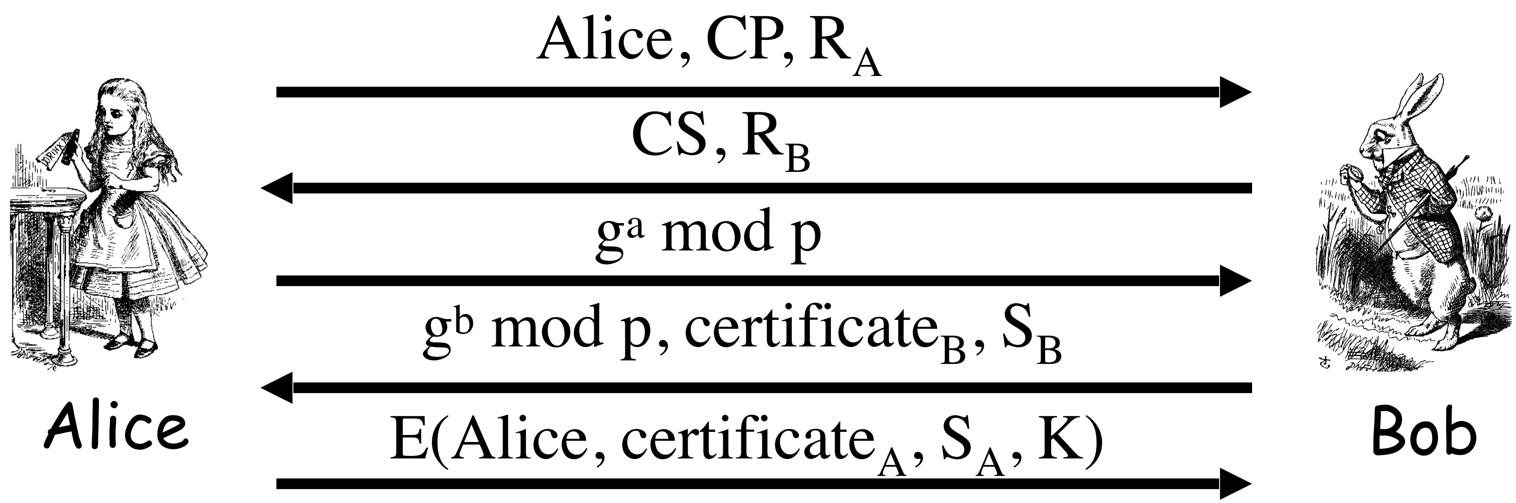
SSH

- 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
- 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 mod p,g^b mod p,g^{ab} mod p)
- $S_B = [H]_{Bob}$
- $S_A = [H, Alice, certificate_A]_{Alice}$
- K = g^{ab} mod p

MiM Attack on SSH?

Alice

Alice → Trudy:

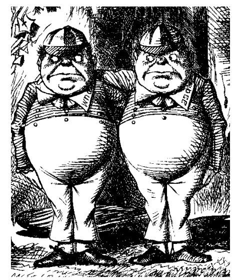
Alice, R_A →

← R_B

g^a mod p →

← g^t mod p, $cert_B$, S_B

E(Alice,$cert_A$,S_A,K) →



Trudy

Trudy → Bob:

Alice, R_A →

← R_B

g^t mod p →

← g^b mod p, $cert_B$, S_B

E(Alice,$cert_A$,S_A,K) →

Bob

- Where does this attack fail?
- Alice computes

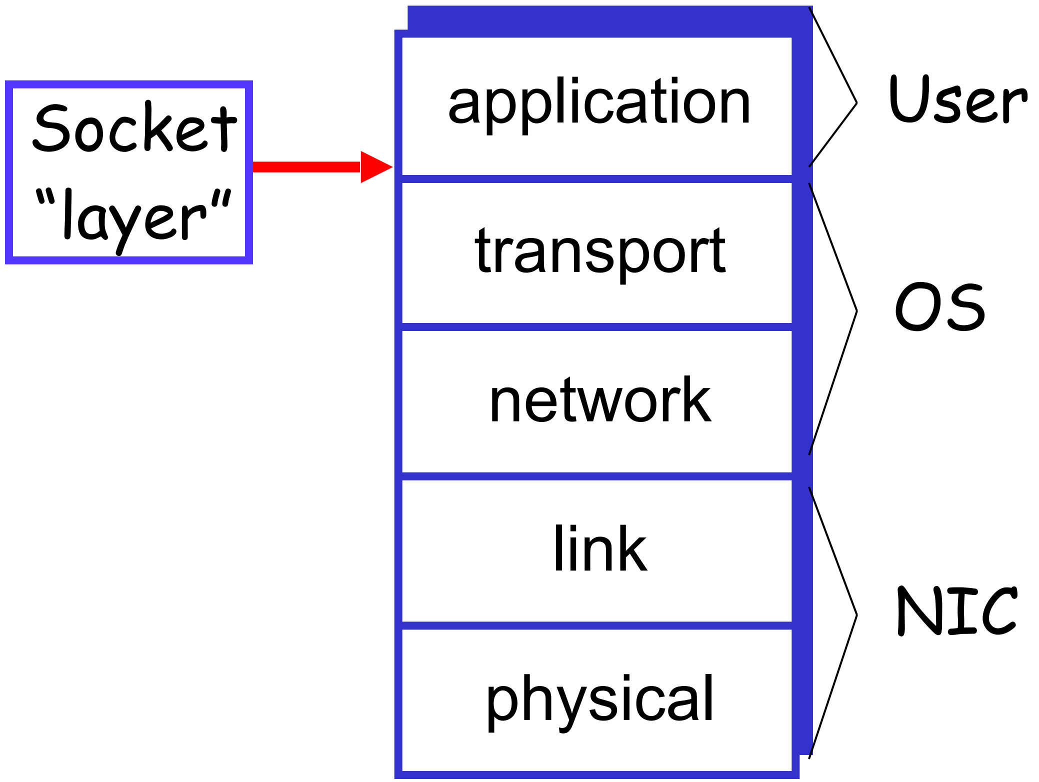
  H_a = h(Alice,Bob,CP,CS,R_A,R_B,g^a mod p,g^t mod p,g^{at} mod p)
- But Bob signs

  H_b = h(Alice,Bob,CP,CS,R_A,R_B,g^t mod p,g^b mod p,g^{bt} mod 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 Internet transactions today
- For example, if you want to buy a book from 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 really care who you are…
  - …so, no need for mutual authentication

Simple SSL-like Protocol

Alice

I'd like to talk to you securely $\rightarrow$

$\leftarrow$ Here's my certificate

$\{K\}_{Bob}$ $\rightarrow$

$\leftarrow$ protected HTTP $\rightarrow$

Bob

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

Simplified SSL Protocol

Alice

Can we talk?, cipher list, R_A →

← certificate, cipher, R_B

$\{S\}_{Bob}$, E(h(msgs,CLNT,K),K) →

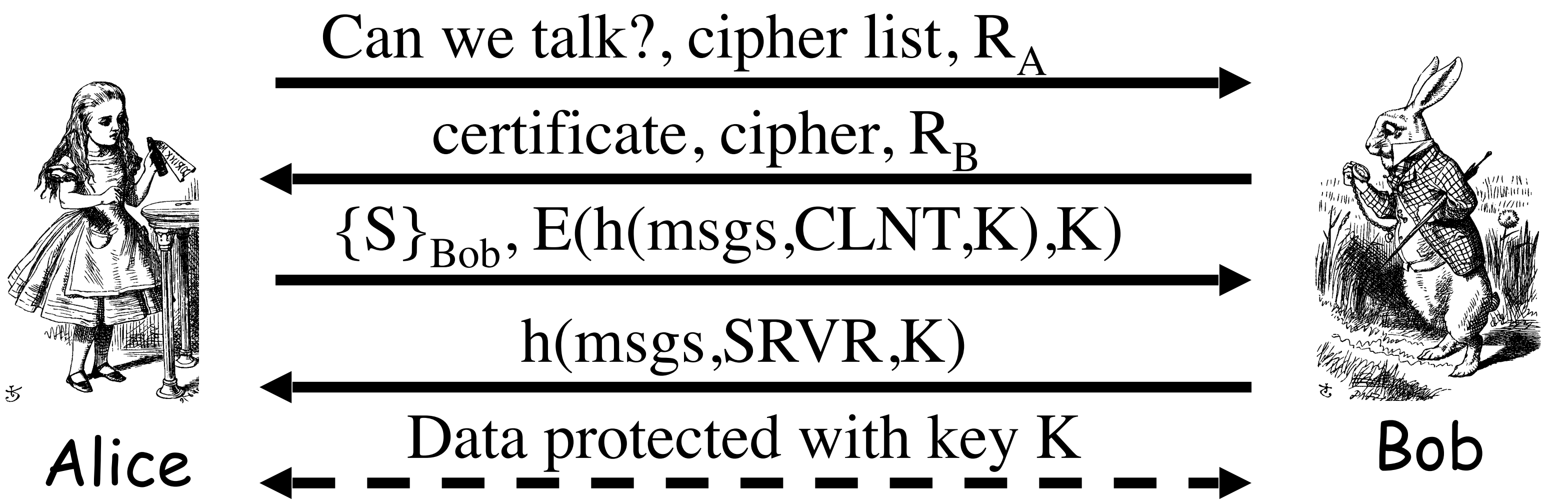
← h(msgs,SRVR,K)

Data protected with key K ↔

Bob

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

Simplified SSL Protocol

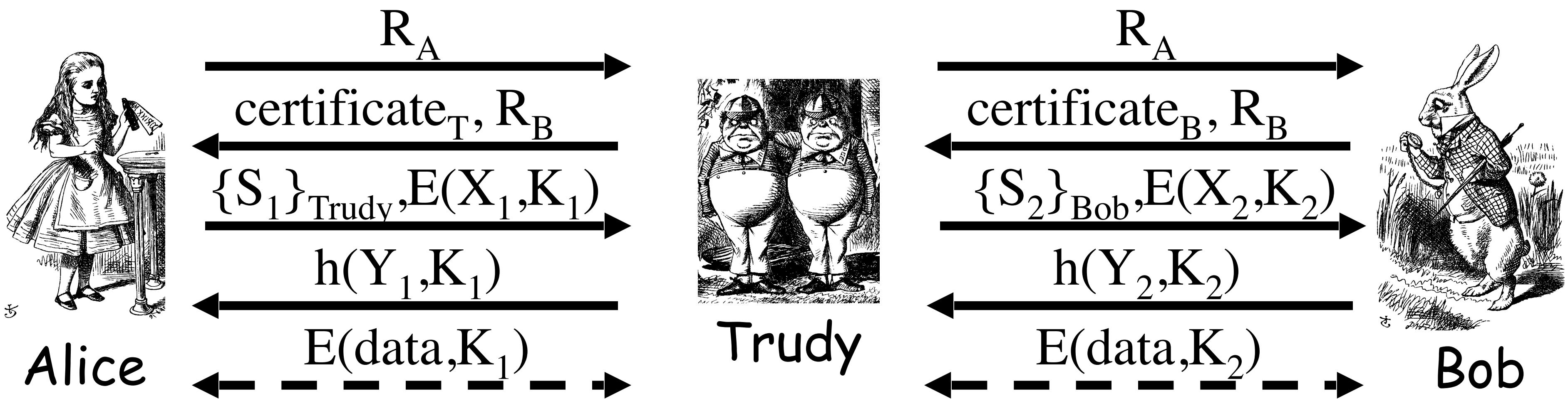


- **Q:** Why is h(msgs,CLNT,K) encrypted?
- **A:** Apparently, it adds no security…

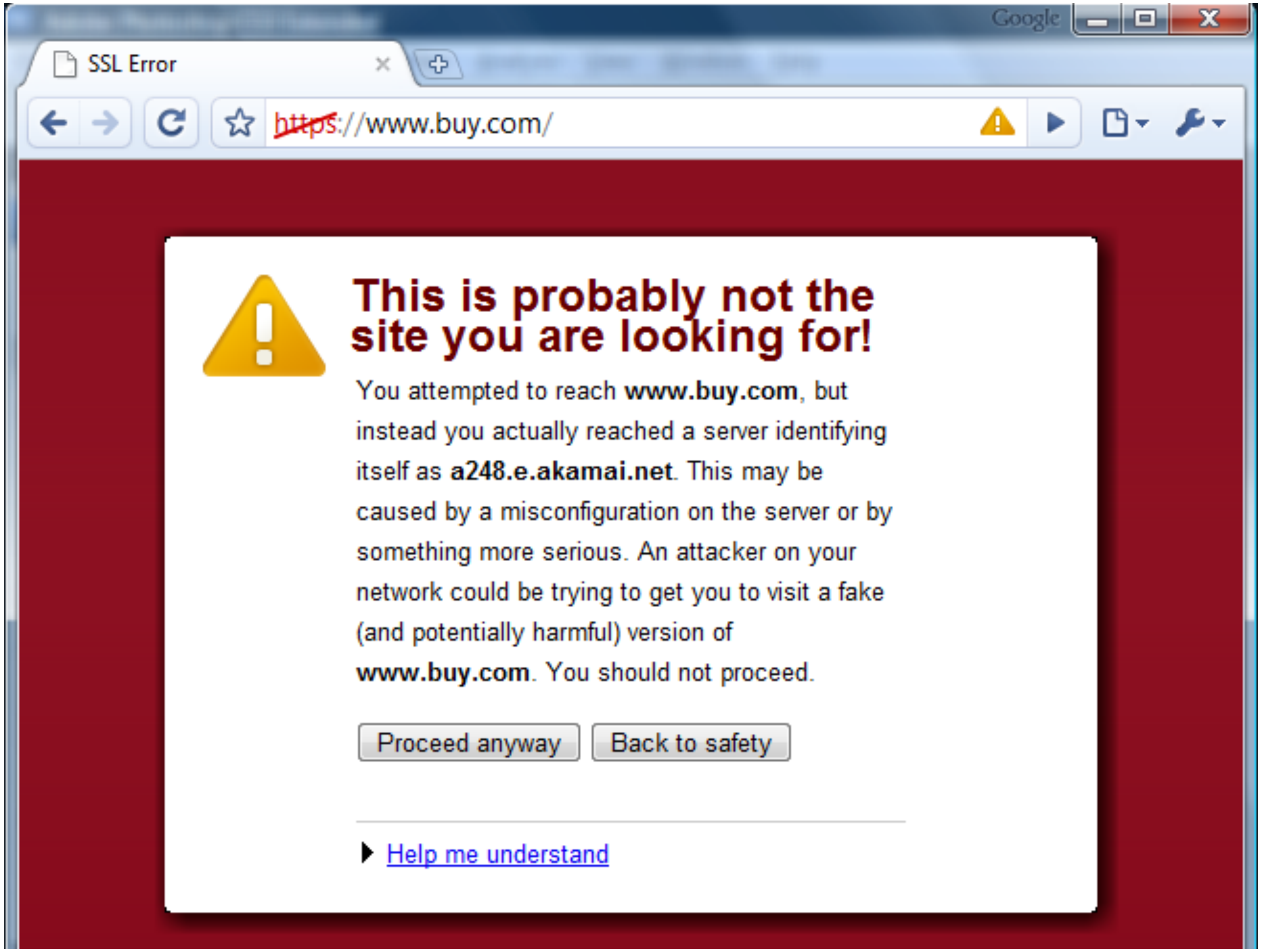
SSL Authentication

- 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?



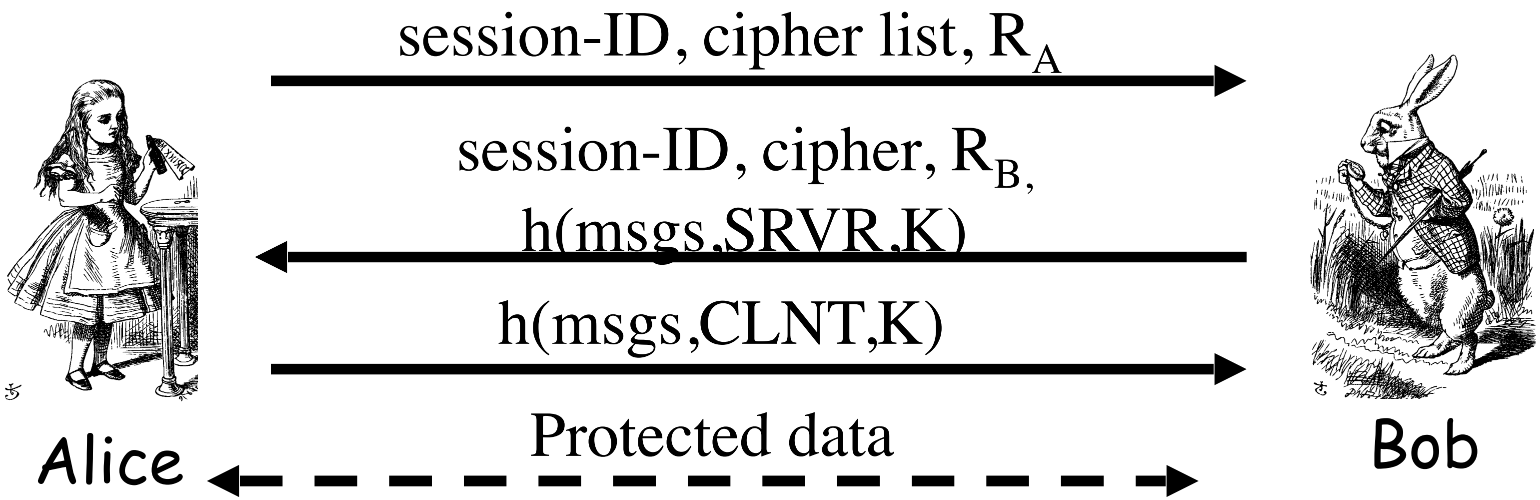
- **Q:** What prevents this MiM "attack"?
- **A:** Bob's certificate must be signed by a certificate authority (CA)
- What does browser do if signature not valid?
- What does user do when browser complains?


Google
SSL Error
https://www.buy.com/
This is probably not the site you are looking for!
You attempted to reach www.buy.com, but instead you actually reached a server identifying itself as a248.e.akamai.net. This may be caused by a misconfiguration on the server or by something more serious. An attacker on your network could be trying to get you to visit a fake (and potentially harmful) version of www.buy.com. You should not proceed.
Proceed anyway
Back to safety
Help me understand

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

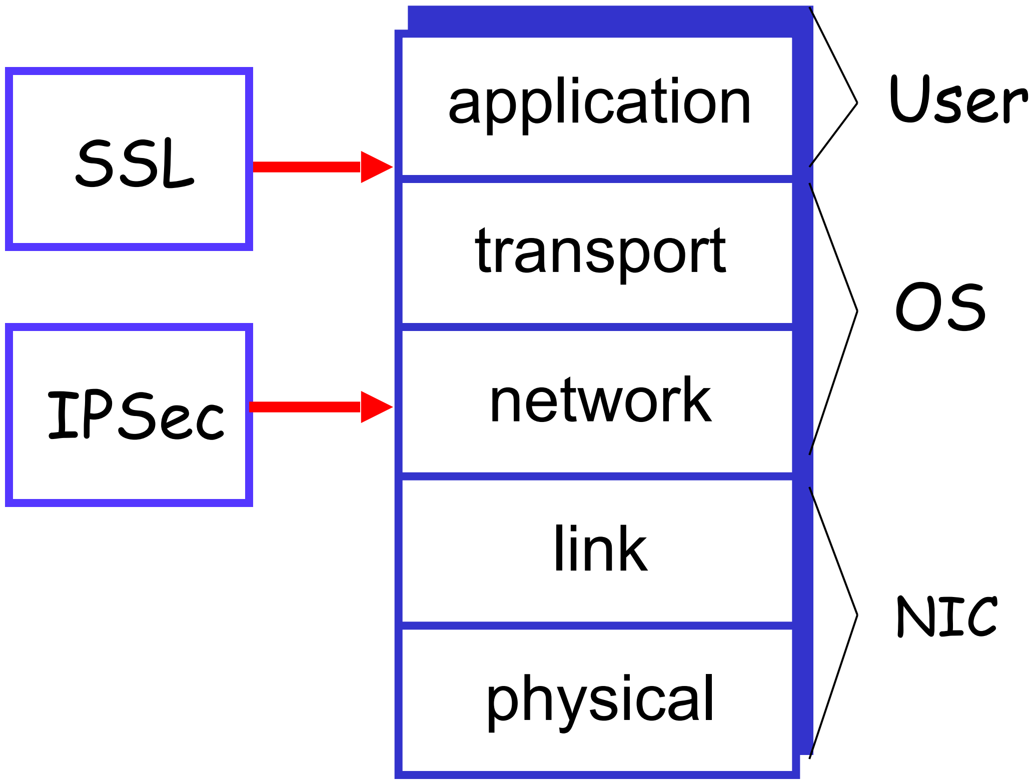


- Assuming SSL **session** exists
- So, S is already known to Alice and Bob
- Both sides must remember session-ID
- Again, $K = h(S,R_A,R_B)$
- **No public key operations!** (relies on known S)

IPSec

IPSec and SSL

- IPSec lives at the network layer
- IPSec is transparent to applications



IPSec and Complexity

- IPSec is a complex protocol
- **Over-engineered**
  - Lots of (generally useless) features
- Flawed — Some significant security issues
- Interoperability is serious challenge
  - Defeats the purpose of having a standard!
- Complex
- And, did I mention, it's complex?

IKE and ESP/AH

- Two parts to IPSec…
- **IKE:** Internet Key Exchange
  - Mutual authentication
  - Establish session key
  - Two "phases" — like SSL session/connection
- **ESP/AH**
  - **ESP**: Encapsulating Security Payload — for confidentiality and/or integrity
  - **AH**: Authentication Header — integrity only

IKE

- IKE has 2 phases
  - Phase 1 — IKE security association (SA)
  - Phase 2 — AH/ESP security association
- Phase 1 is comparable to SSL **session**
- Phase 2 is comparable to SSL **connection**
- Not an obvious need for two phases in IKE
  - In the context of IPSec, that is
- If multiple Phase 2's do not occur, then it is **more** costly to have two phases!

IKE Phase 1

- 4 different "key options"
  - Public key encryption (original version)
  - Public key encryption (improved version)
  - Public key signature
  - Symmetric key
- For each of these, 2 different "modes"
  - Main mode and aggressive mode
- **There are 8 versions of IKE Phase 1!**
- Need more evidence it's over-engineered?

IKE Phase 1

- We discuss 6 of the 8 Phase 1 variants
  - Public key signatures (main & aggressive modes)
  - Symmetric key (main and aggressive modes)
  - Public key encryption (main and aggressive)
- Why public key encryption and public key signatures?
  - Always know your own private key
  - **May not** (initially) know other side's public key

IKE Phase 1

- Uses ephemeral Diffie-Hellman to establish session key
  - Provides perfect forward secrecy (PFS)
- Let a be Alice's Diffie-Hellman exponent
- Let b be Bob's Diffie-Hellman exponent
- Let g be generator and p prime
- Recall that p and g are public

IKE Phase 1: Digital Signature (Main Mode)

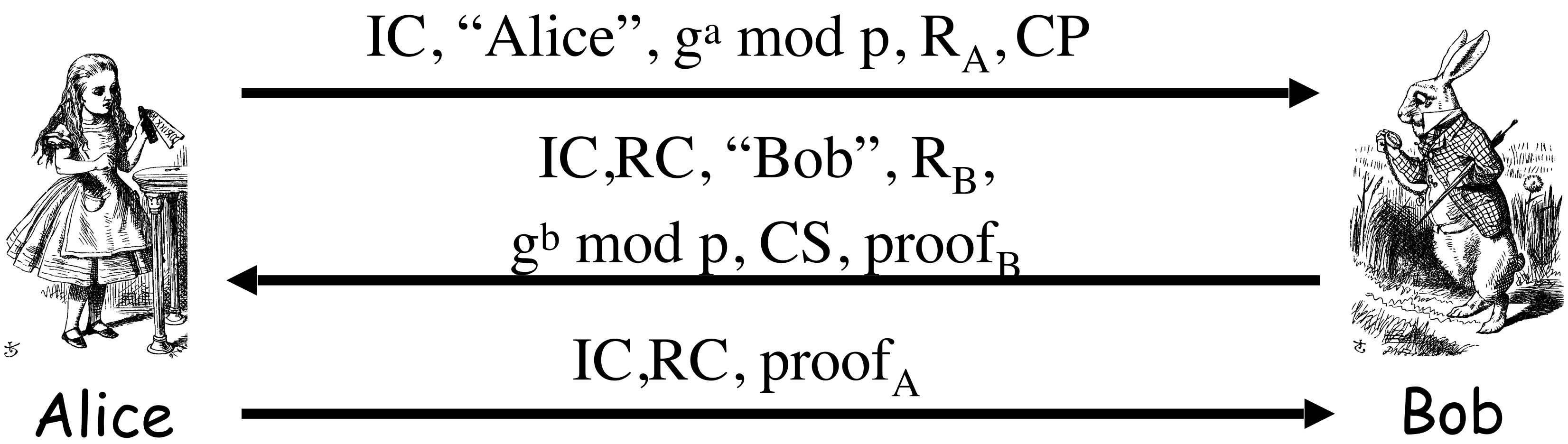
Alice

Bob

- Alice → Bob: IC, CP
- Bob → Alice: IC,RC, CS
- Alice → Bob: IC,RC, g^a mod p, R_A
- Bob → Alice: IC,RC, g^b mod p, R_B
- Alice → Bob: IC,RC, E("Alice", $proof_A$, K)
- Bob → Alice: IC,RC, E("Bob", $proof_B$, K)

- CP = crypto proposed, CS = crypto selected
- IC = initiator "cookie", RC = responder "cookie"
- K = h(IC,RC,g^{ab} mod p,R_A,R_B)
- SKEYID = h(R_A, R_B, g^{ab} mod p)
- $proof_A$ = [h(SKEYID,g^a mod p,g^b mod p, IC, RC, CP,
- "Alice") $]_{Alice}$

IKE Phase 1: Public Key Signature (Aggressive Mode)



- Main differences from main mode
  - Not trying to hide identities
  - Cannot negotiate g or p

Main vs Aggressive Modes

- Main mode **MUST** be implemented
- Aggressive mode **SHOULD** be implemented
  - So, if aggressive mode is not implemented, "you should feel guilty about it"
- Might create interoperability issues
- For public key signature authentication
  - **Passive attacker** knows identities of Alice and Bob in aggressive mode, but not in main mode
  - **Active attacker** can determine Alice's and Bob's identity in main mode

IKE Phase 1: Symmetric Key (Main Mode)

Alice

K_{AB}

IC, CP

IC,RC, CS

IC,RC, g^a mod p, R_A

IC,RC, g^b mod p, R_B

IC,RC, E("Alice", proof_A, K)

IC,RC, E("Bob", proof_B, K)

Bob

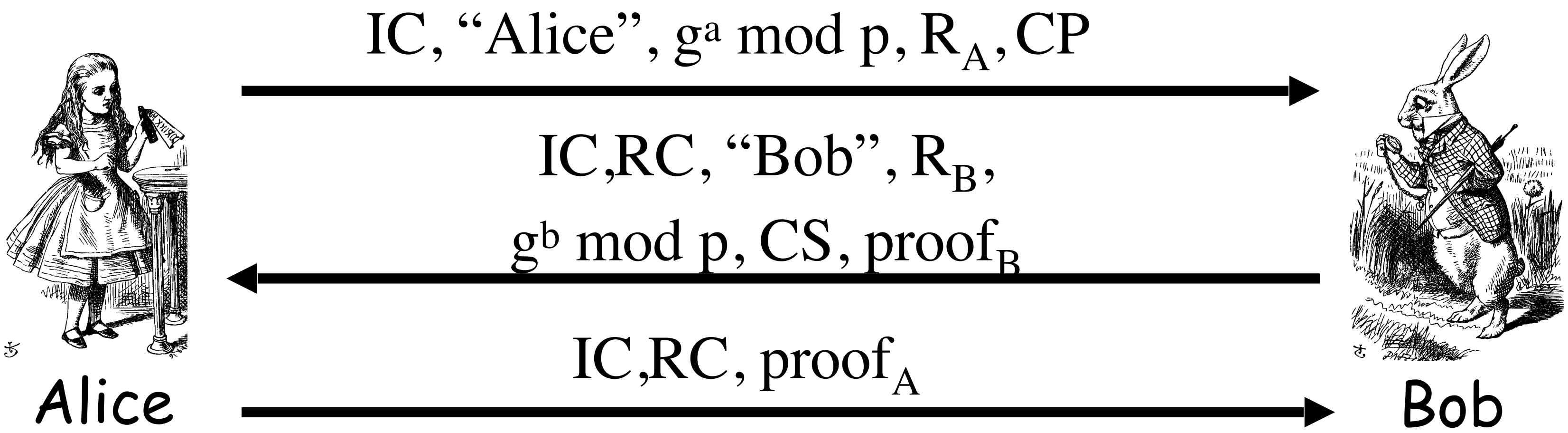
K_{AB}

- Same as signature mode except
  - K_{AB} = symmetric key shared in advance
  - $K = h(IC,RC,g^{ab} \bmod p,R_A,R_B,K_{AB})$
  - $SKEYID = h(K, g^{ab} \bmod p)$
  - $\text{proof}_A = h(SKEYID,g^a \bmod p,g^b \bmod p,IC,RC,CP,\text{“Alice”})$

Problems with Symmetric Key (Main Mode)

- Catch-22
  - Alice sends her ID in message 5
  - Alice's ID encrypted with K
  - To find K Bob must know K_{AB}
  - To get K_{AB} Bob must know he's talking to Alice!
- Result: **Alice's IP address used as ID!**
- Useless mode for the "road warrior"
- Why go to all of the trouble of trying to hide identities in 6 message protocol?

IKE Phase 1: Symmetric Key (Aggressive Mode)



- Same format as digital signature aggressive mode
- Not trying to hide identities…
- As a result, does **not** have problems of main mode
- But does not (pretend to) hide identities

IKE Phase 1: Public Key Encryption (Main Mode)

Alice

Bob

$\text{IC, CP} \longrightarrow$

$\longleftarrow \text{IC,RC, CS}$

$\text{IC,RC, } g^a \text{ mod p, } \{R_A\}_{Bob}, \{\text{“Alice”}\}_{Bob} \longrightarrow$

$\longleftarrow \text{IC,RC, } g^b \text{ mod p, } \{R_B\}_{Alice}, \{\text{“Bob”}\}_{Alice}$

$\text{IC,RC, E(proof}_A\text{, K)} \longrightarrow$

$\longleftarrow \text{IC,RC, E(proof}_B\text{, K)}$

- CP = crypto proposed, CS = crypto selected
- IC = initiator “cookie”, RC = responder “cookie”
- $K = h(IC,RC,g^{ab} \bmod p,R_A,R_B)$
- $SKEYID = h(R_A, R_B, g^{ab} \bmod p)$
- $proof_A = h(SKEYID,g^a \bmod p,g^b \bmod p,IC,RC,CP,\text{“Alice”})$

IKE Phase 1: Public Key Encryption (Aggressive Mode)

$IC, CP, g^a \bmod p, \{\text{“Alice”}\}_{Bob}, \{R_A\}_{Bob}$ →

← $IC, RC, CS, g^b \bmod p, \{\text{“Bob”}\}_{Alice}, \{R_B\}_{Alice}, proof_B$

$IC, RC, proof_A$ →

Alice

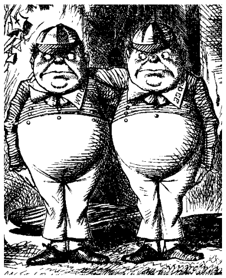
Bob

- $K, proof_A, proof_B$ computed as in main mode
- Note that identities are hidden
  - The only aggressive mode to hide identities
  - So, why have a main mode?

Public Key Encryption Issue?

- In public key encryption, aggressive mode…
- Suppose **Trudy** generates
  - Exponents **a** and **b**
  - Nonces $\mathbf{R_A}$ and $\mathbf{R_B}$
- Trudy can compute "valid" keys and proofs: $\mathbf{g^{ab} \bmod p}$, **K**, **SKEYID**, $\mathbf{proof_A}$ and $\mathbf{proof_B}$
- All of this also works in main mode

Public Key Encryption Issue?

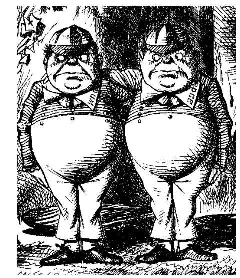


Trudy
(as Alice)

IC, CP, $g^a \bmod p$,
$\{\text{“Alice”}\}_{\text{Bob}}$, $\{R_A\}_{\text{Bob}}$ →

← IC,RC, CS, $g^b \bmod p$,
$\{\text{“Bob”}\}_{\text{Alice}}$, $\{R_B\}_{\text{Alice}}$, proof_B

IC,RC, proof_A →



Trudy
(as Bob)

- Trudy can create messages that appears to be between Alice and Bob
- Appears valid to any observer, **including Alice and Bob!**

Plausible Deniability

- Trudy can create fake "conversation" that appears to be between Alice and Bob
  - Appears valid, even to Alice and Bob!
- A security **failure**?
- In IPSec public key option, it is a **feature**…
  - **Plausible deniability:** Alice and Bob can deny that any conversation took place!
- In some cases it might create a problem
  - E.g., if Alice makes a purchase from Bob, she could later repudiate it (unless she had signed)

IKE Phase 1 "Cookies"

- IC and RC — cookies (or "anti-clogging tokens") supposed to prevent DoS attacks
  - No relation to Web cookies
- To reduce DoS threats, Bob wants to remain **stateless** as long as possible
- But Bob must remember CP from message 1 (required for proof of identity in message 6)
- Bob must keep state from 1st message on
  - So, these "cookies" offer little DoS protection

Kerberos

Cerbero, fiera crudele e diversa,
con tre gola caninamente latra
sovra la gente che quivi è sommersa...

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 Needham and Schroeder protocol
  - 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

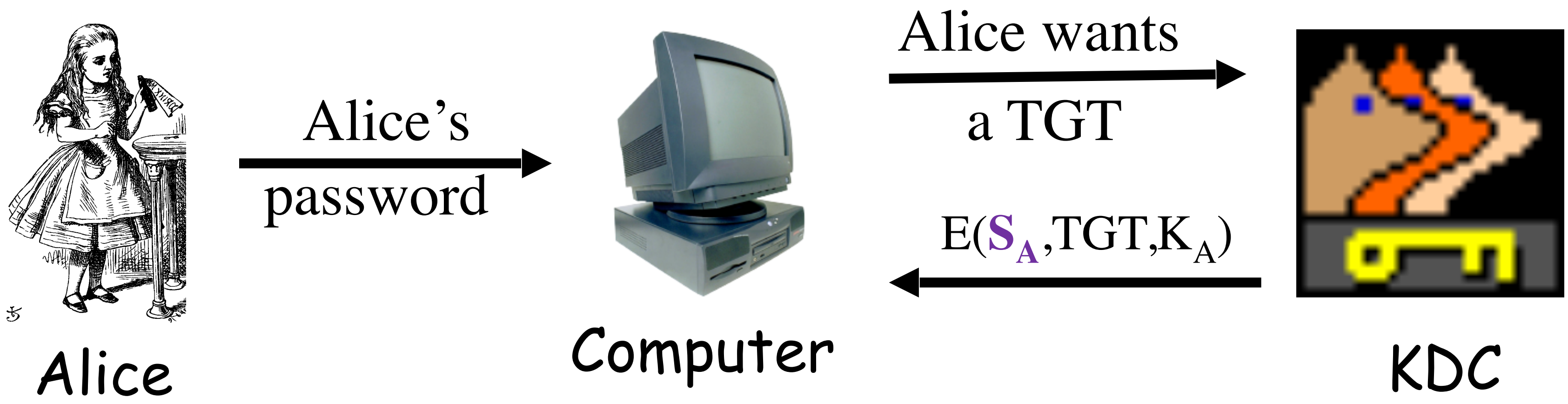
Kerberos Tickets

- 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

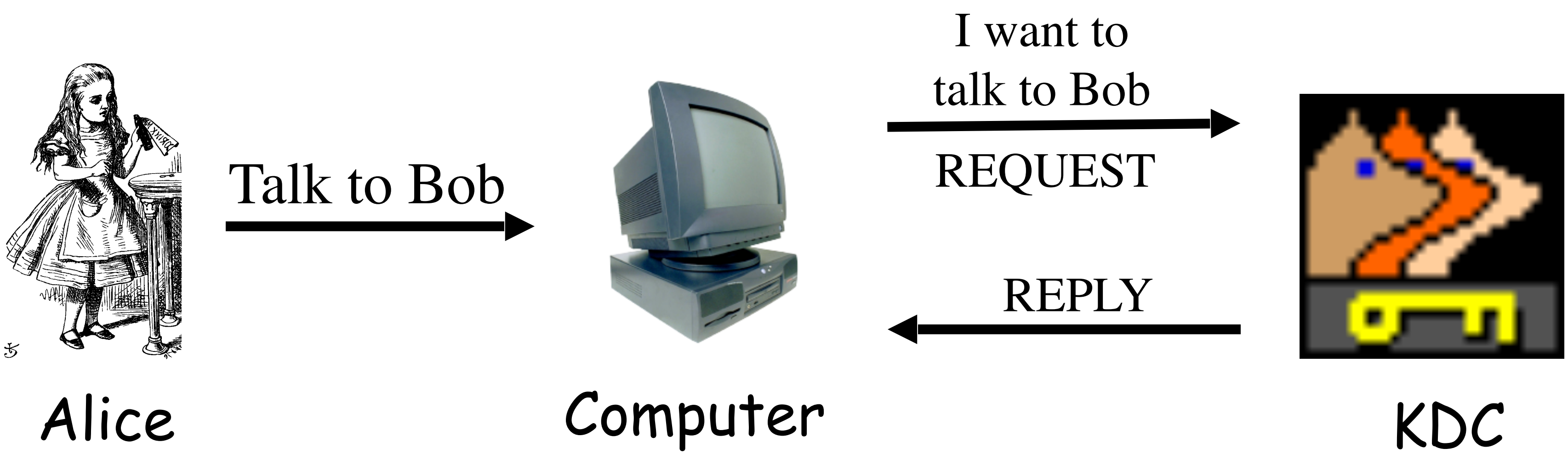
- Alice enters her password
- Then Alice's computer does following:
  - Derives K_A from Alice's password
  - Uses K_A to get TGT for Alice from KDC
- Alice then uses her TGT (credentials) to securely access network resources
- **Plus:** Security is transparent to Alice
- **Minus:** KDC **must** be secure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

Kerberized Login



- Key K_A = h(Alice's password)
- KDC creates session key S_A
- Alice's computer decrypts S_A and TGT
  - Then it forgets K_A
- TGT = E("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

Alice's Computer

ticket to Bob, authenticator $\longrightarrow$

$\longleftarrow$ E(timestamp + 1, K_{AB})

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

Questions about Kerberos

- 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:** Enables Alice to be anonymous when she later uses her TGT to request a ticket
- 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