Kerberos and IPSec

Kerberos - Scalability

- In an enterprise network with N users, suppose that we want to authenticate each other, Alice, Bob, Charlie, David, Eva…
  - Authentication using public keys (how many key pairs?)
    - N users $\Rightarrow$ N key pairs (but needs PKI)
  - Authentication using symmetric keys (how many keys?)
    - N users requires (on the order of) N^2 keys

When N is large, symmetric key case does not scale

Discussion: what should we do?

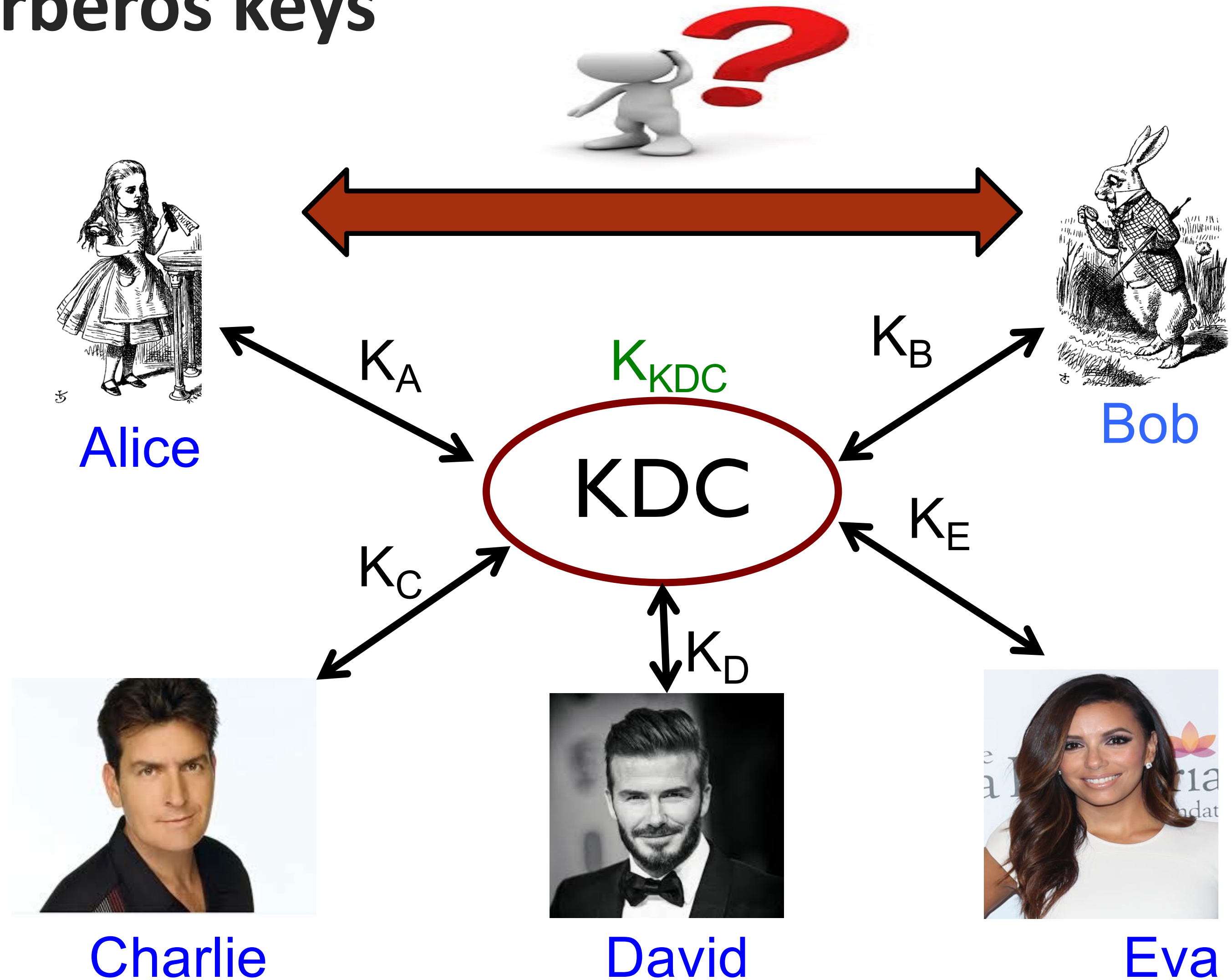
Kerberos in Computer Security

- In security, Kerberos is an authentication protocol based on symmetric key crypto
  - Designed for LANs or corporate networks
  - Originated at MIT
  - Relies on a **Trusted Third Party (TTP)**
    - Security depends on TTP
  - Kerberos based on symmetric keys but only requires N keys for N users
  - 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

Kerberos keys



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

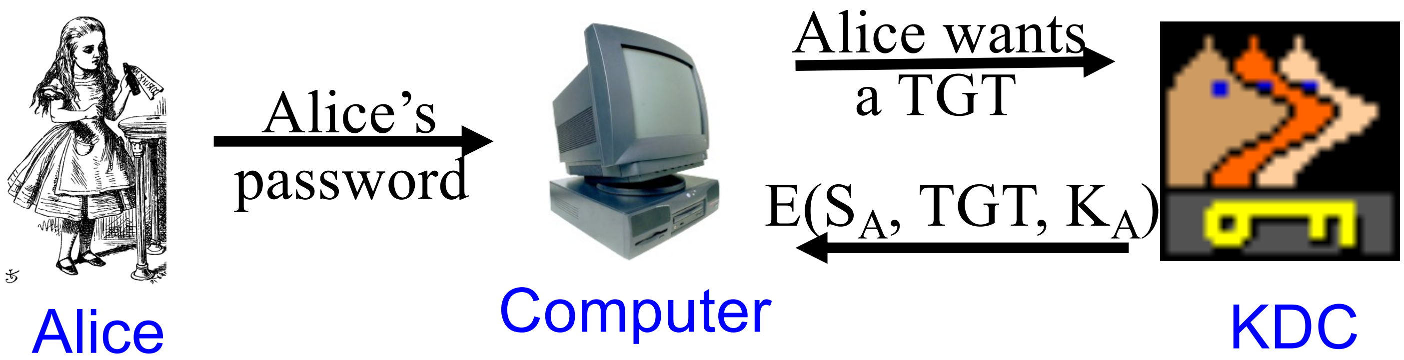
Three phases of Kerberos

- Phase I: Kerberized Login
- Phase II: Alice Requests “Ticket to Bob”
- Phase III: Alice Uses Ticket to Bob

Phase I: Kerberized Login

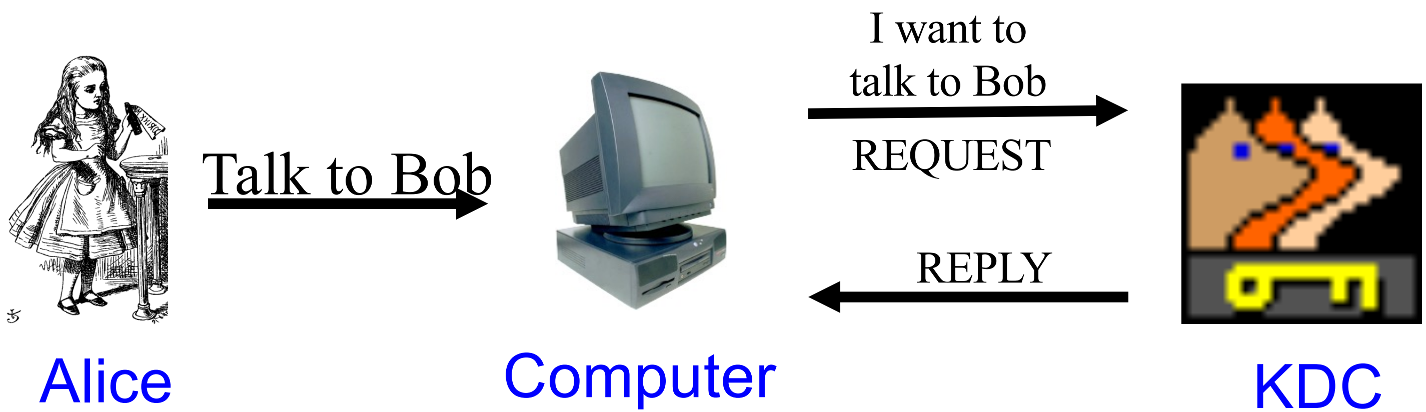
- Alice enters her password
- Then Alice's computer does following:
  - Derives K_A from Alice's password
  - Uses K_A to get TGT for Alice from KDC
- Alice then uses her TGT (credentials) to securely access network resources
- **Plus:** Security is transparent to Alice
- **Minus:** KDC ***must*** be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

Kerberized Login



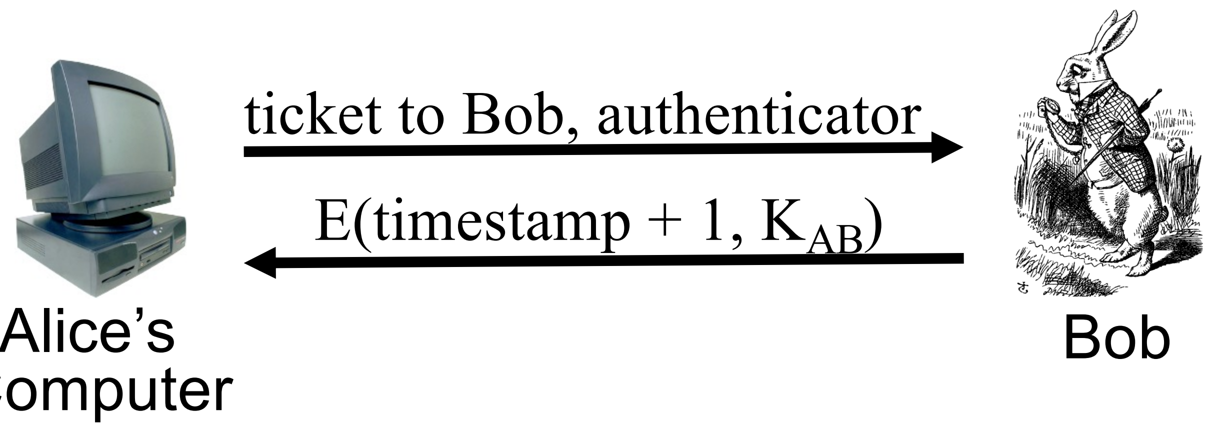
- Key $K_A = h(\text{Alice's password})$
- KDC creates session key S_A
- Alice's computer decrypts S_A and TGT
  - Then it forgets K_A
- $TGT = E(\text{"Alice"}, S_A, K_{KDC})$

Phase II: Alice Requests "Ticket to Bob"



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

Phase III: Alice Uses Ticket to Bob



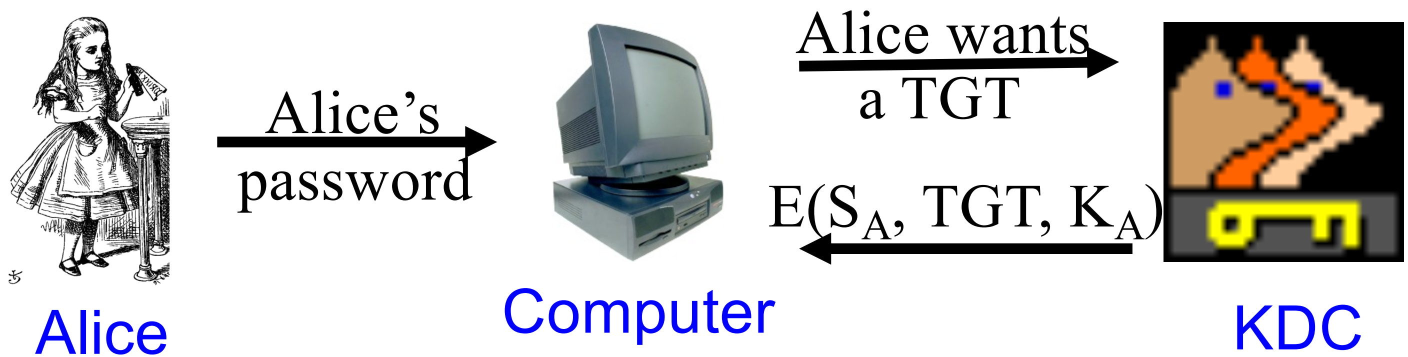
- ticket to Bob = E("Alice", K_{AB}, K_B)
- authenticator = E(timestamp, K_{AB})
- Bob decrypts "ticket to Bob" to get K_{AB} which he then uses to verify timestamp

Kerberos Question 1

- 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!



Kerberos Question 2

- In Alice's "Kerberized" login to Bob, can Alice remain anonymous?
  - Alice remain anonymous in REQUEST
  - REQUEST = (TGT, authenticator)
    - authenticator = E(timestamp, S_A)

Talk to Bob

I want to talk to Bob

REQUEST

REPLY

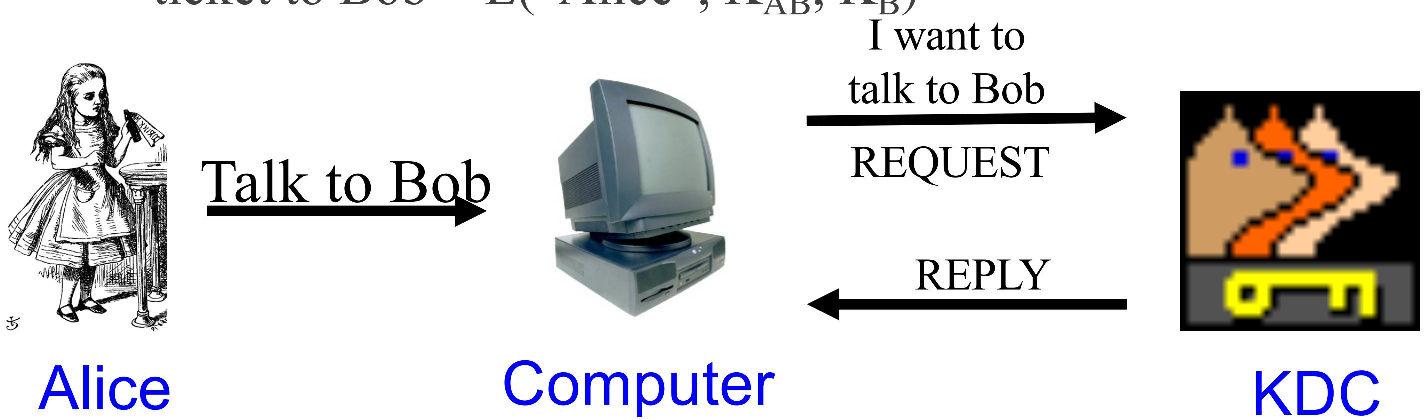
Alice

Computer

KDC

Kerberos Question 3

- Why is "ticket to Bob" sent to Alice?
  - Why doesn't KDC send it directly to Bob?
    - Bob needs to remember K_{AB} until Alice initiate communication, make it **stateful** which against the design of Kerberos
- REPLY = E("Bob", K_{AB}, **ticket to Bob**, S_A)
  - ticket to Bob = E("Alice", K_{AB}, K_B)



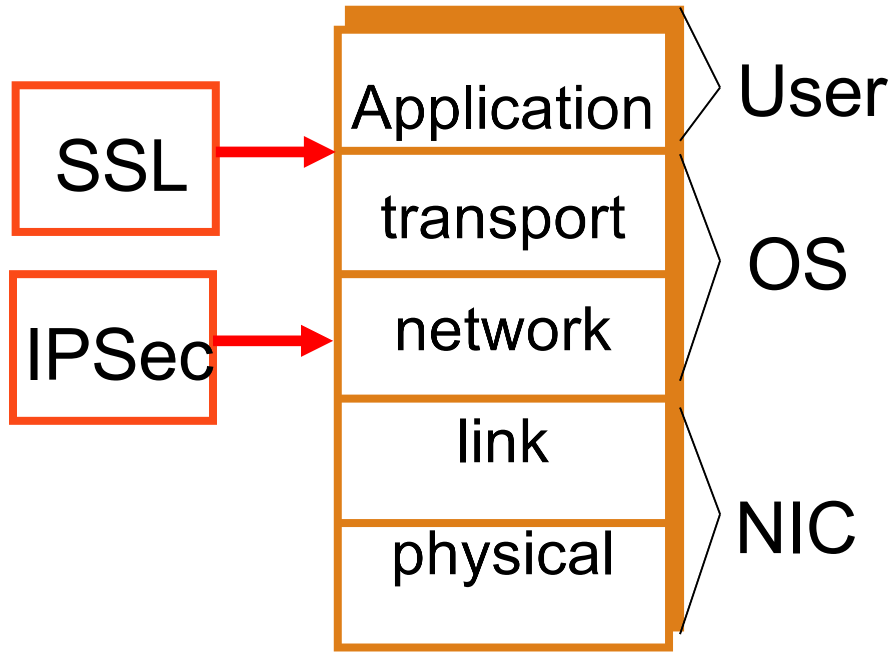
Kerberos Question 4

- Why not have KDC remember session key instead of putting it in a TGT? If so, there is no need for TGT!
  - TGT = E("Alice", S_A, K_{KDC})

  - **Answer: Stateless KDC is a major feature of Kerberos**

IPSec

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



Overview

- IPSec comprises **a suite of protocols** to ensure the **integrity**, **confidentiality**, and **authentication** of data communications over an IP network

- Designed with the goal to achieve security for all IP-related protocols

- But predominantly used in VPNs at this moment

IKE and ESP/AH

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

IKE

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

IKE Phase 1

- Four different "key" options
  - Public key encryption (original version)
  - Public key encryption (improved version)
  - Public key signature
  - Symmetric key
- For each of these, two different "modes"
  - Main mode and aggressive mode
- **There are 8 versions of IKE Phase 1!**
- Over-engineered!

IKE Phase 1

- We discuss 6 of 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 Diffie-Hellman to establish session key
- 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

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

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

IKE Phase 1: Digital Signature (Aggressive Mode)

Alice

IC, “Alice”, g^a mod p, R_A, CP →

← IC, RC, “Bob”, R_B, g^b mod p, CS, proof_B

IC, RC, proof_A →

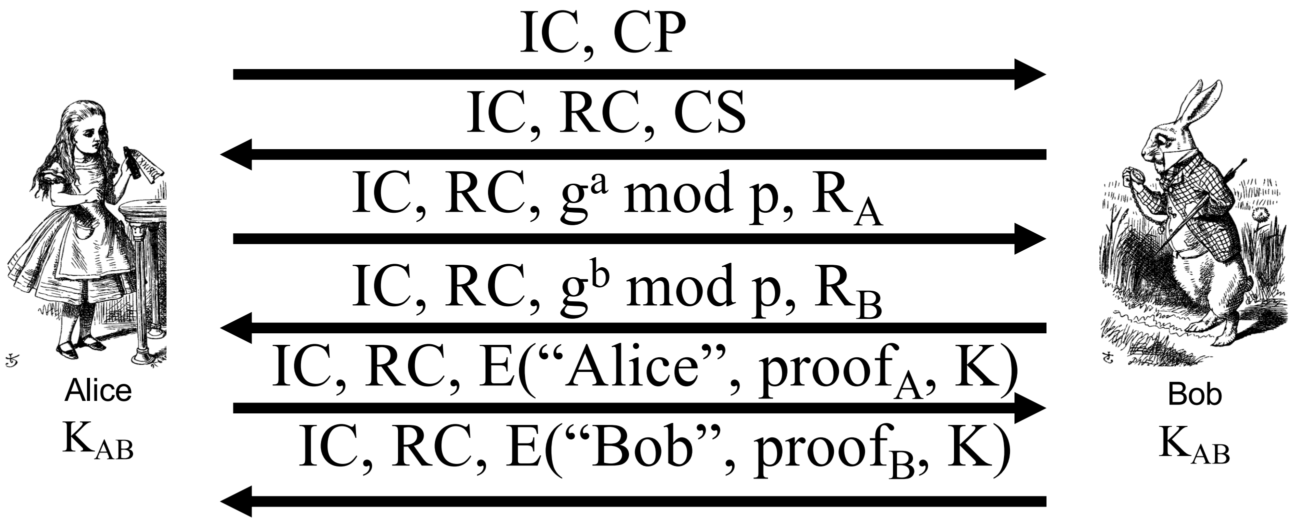
Bob

- Main difference from main mode
  - Not trying to protect 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

IKE Phase 1: Symmetric Key (Main Mode)



- Same as signature mode except
  - K_{AB} = symmetric key shared in advance
  - K = h(IC, RC, g^{ab} mod p, R_A, R_B, K_{AB})
  - SKEYID = h(K, g^{ab} mod p)
  - proof_A = h(SKEYID,g^a mod p,g^b mod p,IC,RC,CP,"Alice")

Problems with 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}

- Catch-22(dilemma)
  - Alice sends her ID in message 5
  - Alice's ID encrypted with K
  - To find K Bob must know $K_{AB:}$ $K = h(IC, RC, g^{ab} \bmod p, R_A, R_B, K_{AB})$
  - To get K_{AB} Bob must know he's talking to Alice!
- Result: **Alice's ID must be IP address!**
- Useless mode
  - Alice must use a static IP
  - Failed to hide identity

Quiz

- **What is SSH primarily used for?**
  - A) Transferring files between machines
  - B) Securing remote computer connections
  - C) Encrypting email communications
  - D) Browsing the web anonymously

Quiz

- **What is the purpose of the SSH key pair in SSH connections?**
  - A) To encrypt the connection
  - B) To verify the identity of the client to the server
  - C) To increase the connection speed
  - D) To monitor the data transfer

Quiz

- **What is a primary reason for a server not to authenticate clients in SSL/TLS communications?**
  - A) To simplify the server configuration
  - B) To reduce the computational load on the server
  - C) To increase the security of the server
  - D) Both A and B are correct

Quiz

- **What is the primary purpose of SSL?**
  - A) Encrypting data transfers between a client and a server
  - B) Speeding up website performance
  - C) Providing stronger passwords
  - D) Filtering spam emails

Quiz

- **SSL certificates are issued by entities known as:**
  - A) Internet service providers
  - B) Certificate authorities
  - C) Domain registrars
  - D) Web hosts

Quiz

- **Which protocol has largely replaced SSL for security purposes?**
  - A) HTTPS
  - B) SFTP
  - C) TLS
  - D) SSH

Quiz

- **Which of the following measures can mitigate the impact of the Heartbleed?**
  - A) Changing user passwords
  - B) Updating to a patched version of OpenSSL
  - C) Installing antivirus software
  - D) Reducing the number of network connections

Quiz

- **How does the Heartbleed bug expose sensitive data?**
  - A) By intercepting data during transmission
  - B) By allowing unauthorized access to databases
  - C) By causing buffer over-reads in memory
  - D) By corrupting data encryption

IKE Phase 1: Symmetric Key (Aggressive Mode)

Alice

IC, "Alice", $g^a \bmod p$, R_A, CP $\rightarrow$

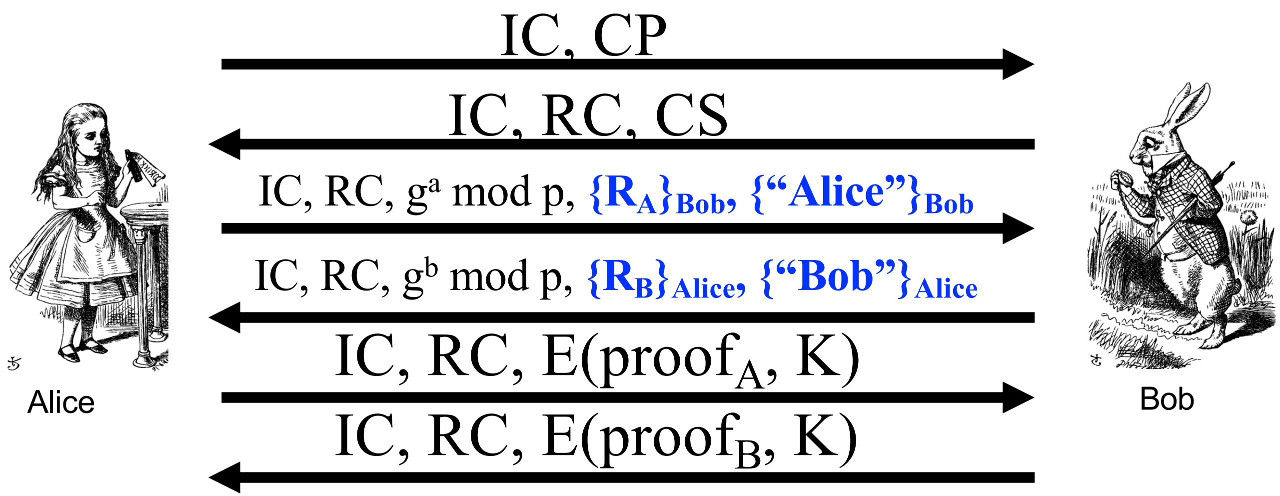
$\leftarrow$ IC, RC, "Bob", R_B, $g^b \bmod p$, CS, proof_B

IC, RC, proof_A $\rightarrow$

Bob

- 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)



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

IKE Phase 1: Public Key Encryption (Aggressive Mode)

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

Alice

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

IC,RC, proof_A →

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?
  - Negotiate g and p in main mode

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

IPSec cookies

Alice

IC, CP

IC, **RC**, CS

IC, **RC**, $g^a \bmod p$, R_A

IC, **RC**, $g^b \bmod p$, R_B

IC, **RC**, E(“Alice”, proof_A, K)

IC, **RC**, E(“Bob”, proof_B, K)

Bob

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

Stateless Cookie Protocol

Alice

Bob, I want to talk!

Cookie c

Cookie c, rest of protocol

Bob

Does c = hash(IP address, local secret)?
If so, continue protocol

Cookie c = hash(IP address, local secret)

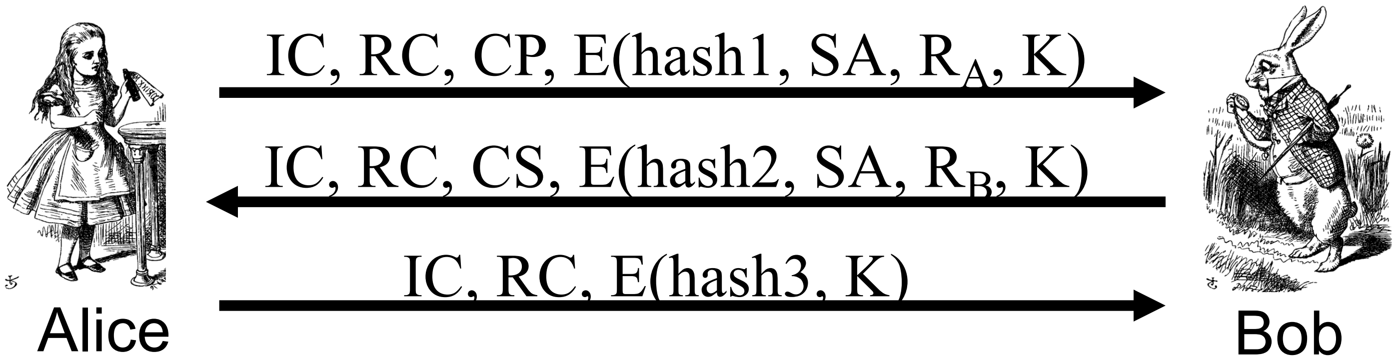
IKE Phase 1 Summary

- Result of IKE phase 1 is
  - Mutual authentication
  - Shared symmetric key
  - IKE **Security Association (IKE SA)**
- But phase 1 is expensive
  - Especially in public key and/or main mode
- Developers of IKE thought it would be used for lots of things — not just IPSec
  - Partly explains the over-engineering…

IKE Phase 2

- Phase 1 establishes **IKE SA**
- Phase 2 establishes **IPSec SA**
- Comparison to SSL
  - SSL session is comparable to IKE Phase 1
  - SSL connections are like IKE Phase 2
- IKE **could** be used for lots of things…
- …but in practice, it's **not!**

IKE Phase 2



- Key K, IC, RC and SA known from Phase 1 (SA: id), R_A & R_B new
- Proposal **CP** includes **ESP** and/or **AH**
- Hashes 1,2,3 depend on SKEYID, SA, R_A and R_B
- Keys derived from KEYMAT = h(SKEYID, R_A, R_B, junk)
- Recall SKEYID depends on phase 1 key method

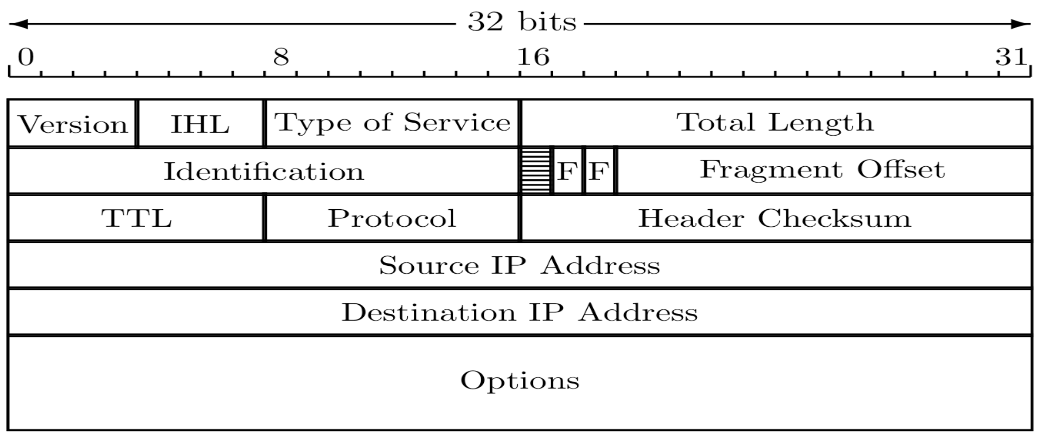
IPSec

- After IKE Phase 1, we have an IKE SA
- After IKE Phase 2, we have an IPSec SA
- Both sides have a shared symmetric key
- Now what?
  - We want to protect **IP datagrams**
- But what is an IP datagram?
  - Considered from the perspective of IPSec…

IP Review

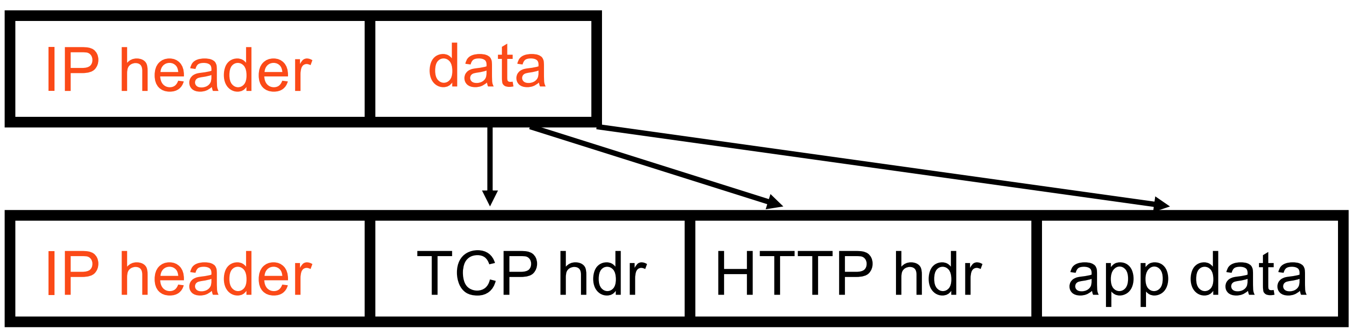
- IP datagram is of the form

- Where IP header is



IP and TCP

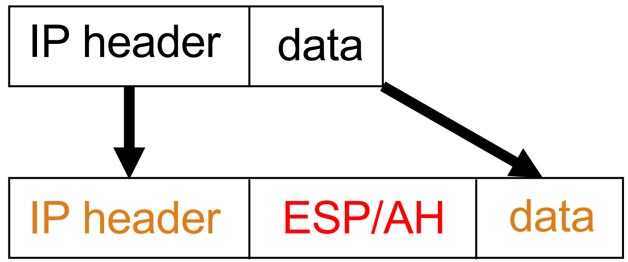
- Consider Web traffic
  - IP encapsulates TCP and…
  - …TCP encapsulates HTTP



- IP data includes TCP header, etc.

IPSec Transport Mode

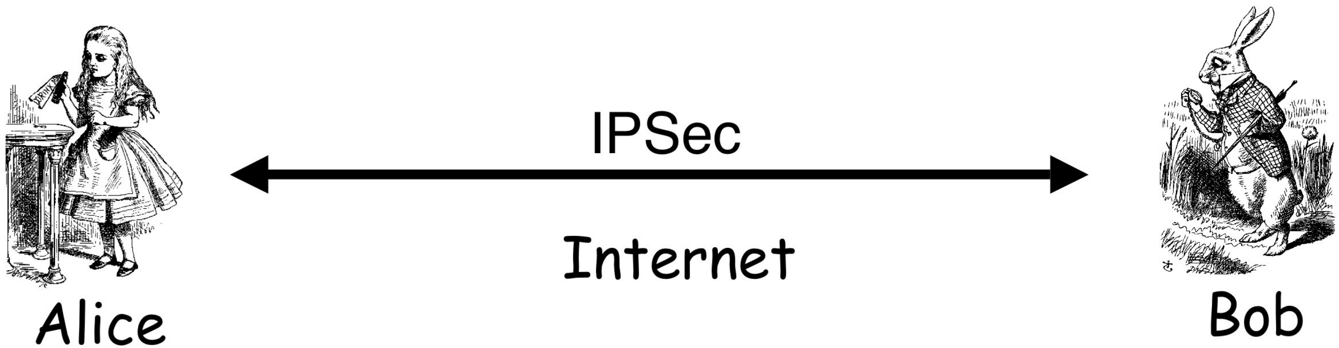
- IPSec **Transport Mode**



- Transport mode designed for *host-to-host*
- Transport mode is efficient
  - Adds minimal amount of extra header
- The original header remains
  - Passive attacker can observe

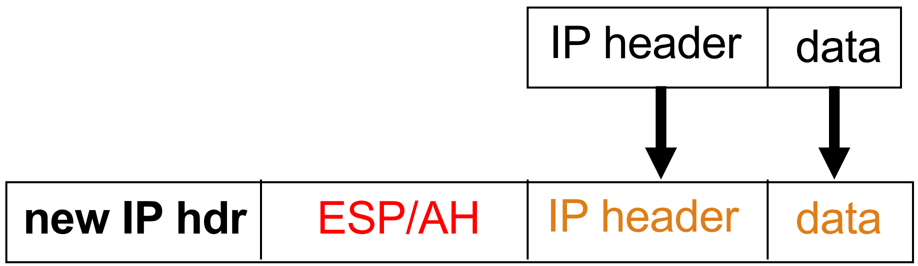
IPSec: Host-to-Host

- **IPSec transport mode**



IPSec Tunnel Mode

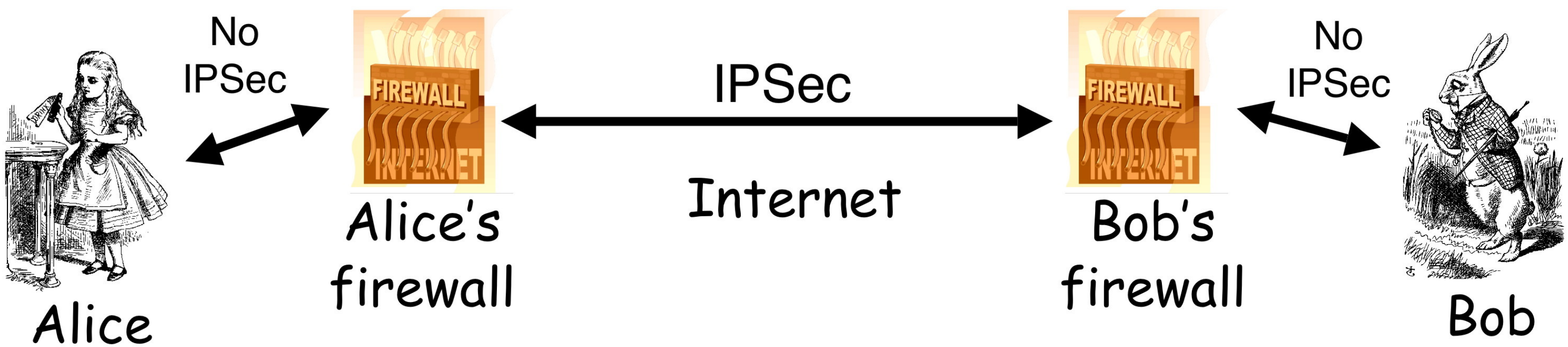
- IPSec Tunnel Mode



- Tunnel mode for *firewall-to-firewall* traffic
- Original IP packet encapsulated in IPSec
- Original IP header not visible to attacker
    - New IP header from firewall to firewall
    - Attacker does not know which hosts are talking

IPSec: Firewall-to-Firewall

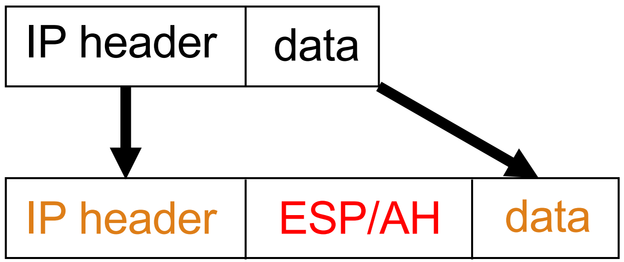
- **IPSec tunnel mode**



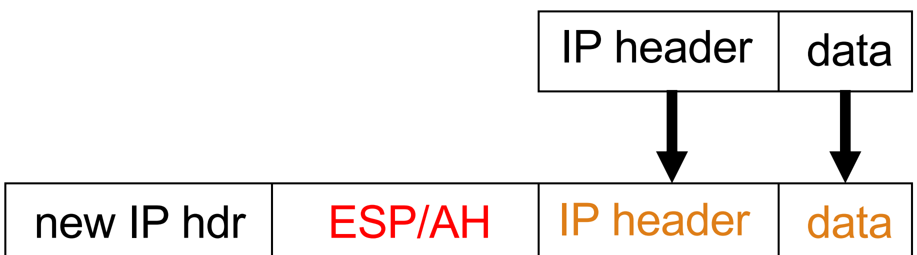
- Local networks not protected

Comparison of IPSec Modes

- Transport Mode



- Tunnel Mode



- Transport Mode
  - Host-to-host
- Tunnel Mode
  - Firewall-to-firewall
- Transport Mode not necessary…
- …but it's more efficient

IPSec Security

- What kind of protection?
  - Confidentiality?
  - Integrity?
  - Both?
- What to protect?
  - Data?
  - Header?
  - Both?
- ESP/AH do some combinations of these

AH vs ESP

- AH —— Authentication Header
  - **Integrity only** (no confidentiality, no encryption)
  - Integrity-protect everything beyond IP header and some fields of header
- ESP —— Encapsulating Security Payload
  - **Integrity and confidentiality** both **required**
  - Protects everything beyond IP header(data in IP)
  - Integrity-only by using NULL encryption

ESP’s NULL Encryption

- According to RFC 2410
  - NULL encryption “is a block cipher the origins of which appear to be lost in antiquity”
  - “Despite rumors”, there is no evidence that NSA “suppressed publication of this algorithm”
  - Evidence suggests it was developed in Roman times as exportable version of Caesar’s cipher
  - Can make use of keys of varying length
  - No IV is required
  - **Null(P,K) = P for any P and any key K**
- Bottom line: Security people can be strange

Why Does AH Exist?

- Cannot encrypt IP header
  - Routers must look at the IP header
  - IP addresses, TTL, etc.
  - IP header exists to route packets!
- AH protects **immutable fields** in IP header
  - Cannot protect all header fields
  - TTL, for example, will change
- Why not use ESP with NULL encryption?
  - ESP provides no protection to the header

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!