Part III: Protocols

Protocol

- Human protocols   the rules followed in human interactions
  - Example: Asking a question in class
- Networking protocols   rules followed in networked communication systems
  - Examples: HTTP, FTP, etc.
- Security protocol   the (communication) rules followed in a security application
  - Examples: SSL, IPSec, Kerberos, etc.

Protocols

- Protocol flaws can be very ***subtle***
- Several well-known security protocols have significant flaws
  - Including WEP,WPA2/3, GSM, and IPSec
- Implementation errors can also occur
  - Recently, IE implementation of SSL
- Not easy to get protocols right…

Ideal Security Protocol

- Must satisfy security requirements
  - Requirements need to be precise
- Efficient
  - Minimize computational requirement
  - Minimize bandwidth usage, delays…
- Robust
  - Works when attacker tries to break it
  - Works if environment changes (slightly)
- Easy to implement, easy to use, flexible…
- Difficult to satisfy all of these!

Chapter 10:
Real-World Protocols

The wire protocol guys don't worry about security because that's really
a network protocol problem. The network protocol guys don't
worry about it because, really, it's an application problem.
The application guys don't worry about it because, after all,
they can just use the IP address and trust the network.

Marcus J. Ranum

In the real world, nothing happens at the right place at the right time.
It is the job of journalists and historians to correct that.

Mark Twain

Real-World Protocols

- Next, we look at real protocols
  - SSH relatively simple & useful 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)

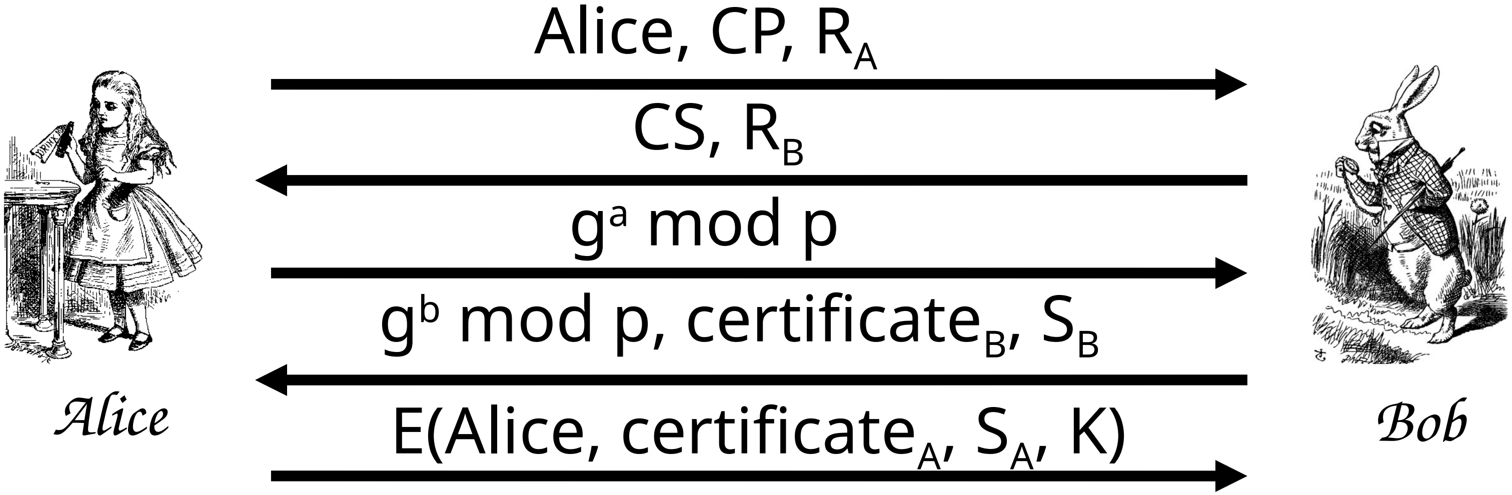
SSH

- 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 in homework problems
- We consider slightly simplified SSH…

Simplified SSH



Alice

Bob

Alice, CP, R_A

CS, R_B

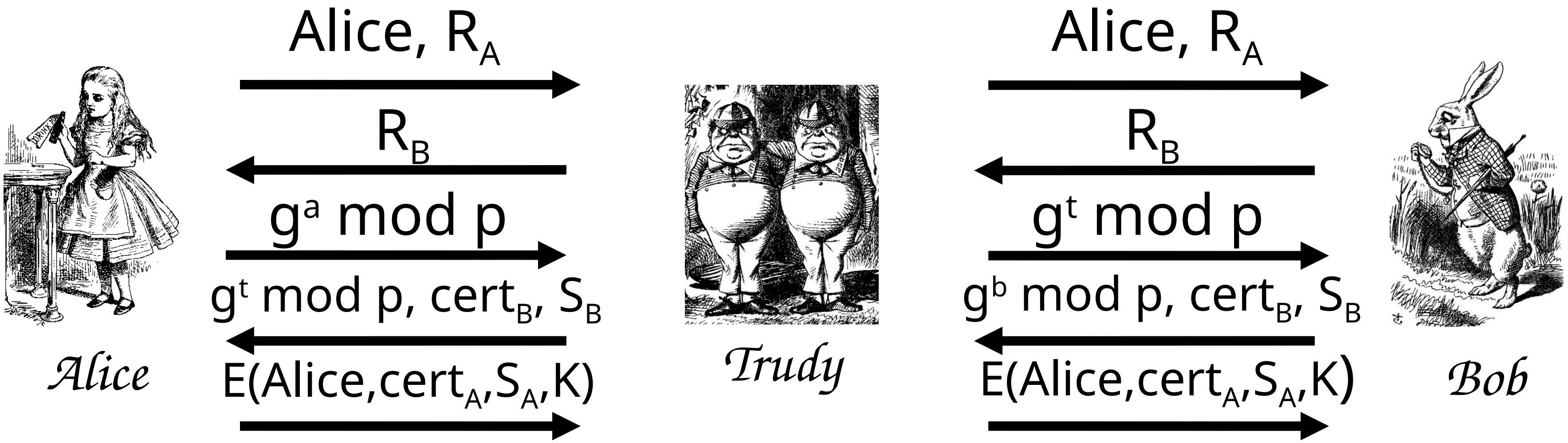
g^a mod p

g^b mod p, certificate_B, S_B

E(Alice, certificate_A, S_A, K)

- 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)
- $S_B = [H]_{\text{Bob}}$
- $S_A = [H, \text{Alice}, \text{certificate}_A]_{\text{Alice}}$
- K = g^{ab} mod p

MiM Attack on SSH?



Alice, R_A
R_B
g^a mod p
g^t mod p, cert_B, S_B
E(Alice,cert_A,S_A,K)
Alice
Trudy
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)

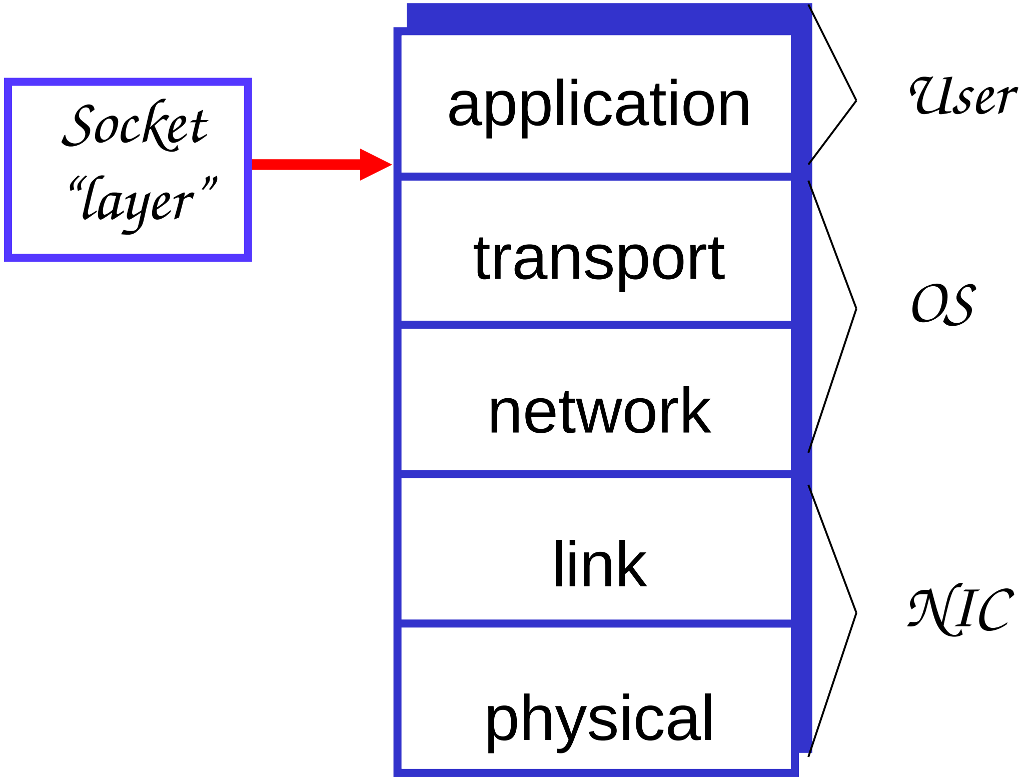
- *But Bob signs*

  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

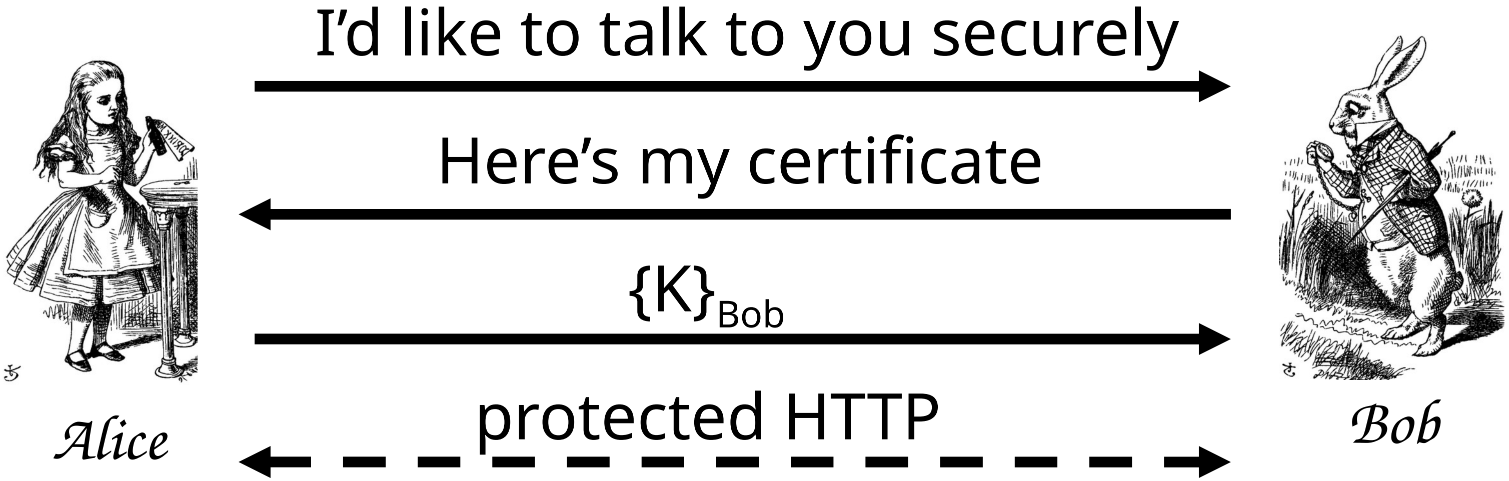
- "Socket layer" lives between application and transport layers
- SSL usually between HTTP and TCP



What is SSL?

- SSL is the protocol used for majority of secure Internet transactions today
- For example, if you want to buy a book 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 really care who you are…
  - …so, no need for mutual authentication

Simple SSL-like Protocol



I'd like to talk to you securely

Here's my certificate

$\{K\}_{\text{Bob}}$

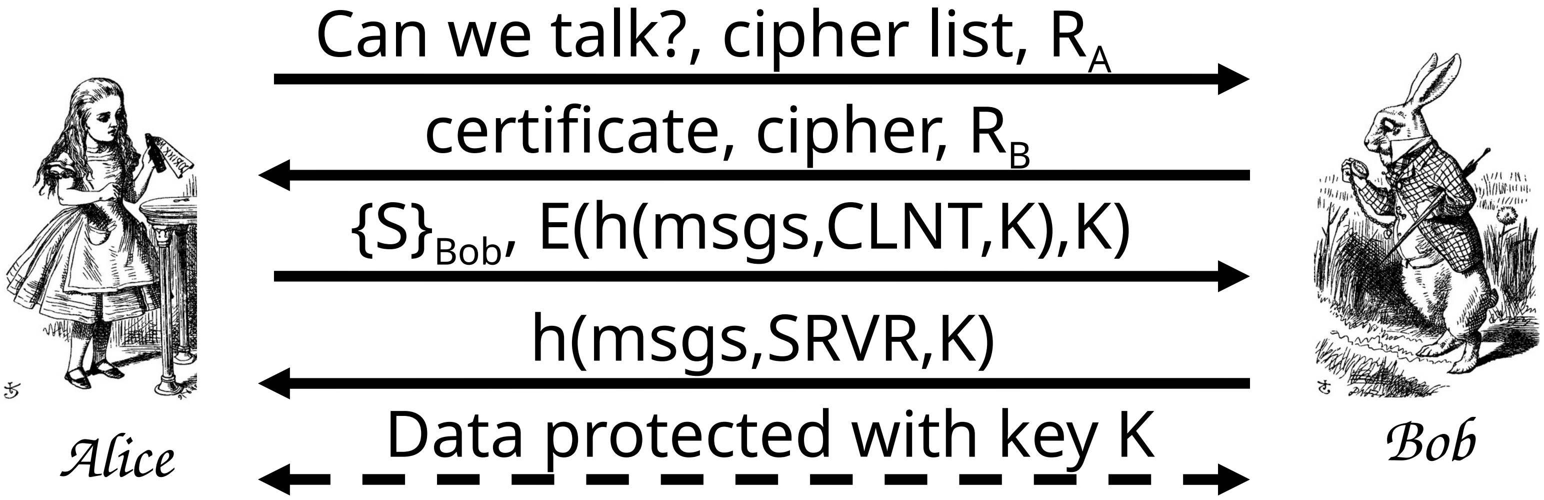
protected HTTP

Alice

Bob

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

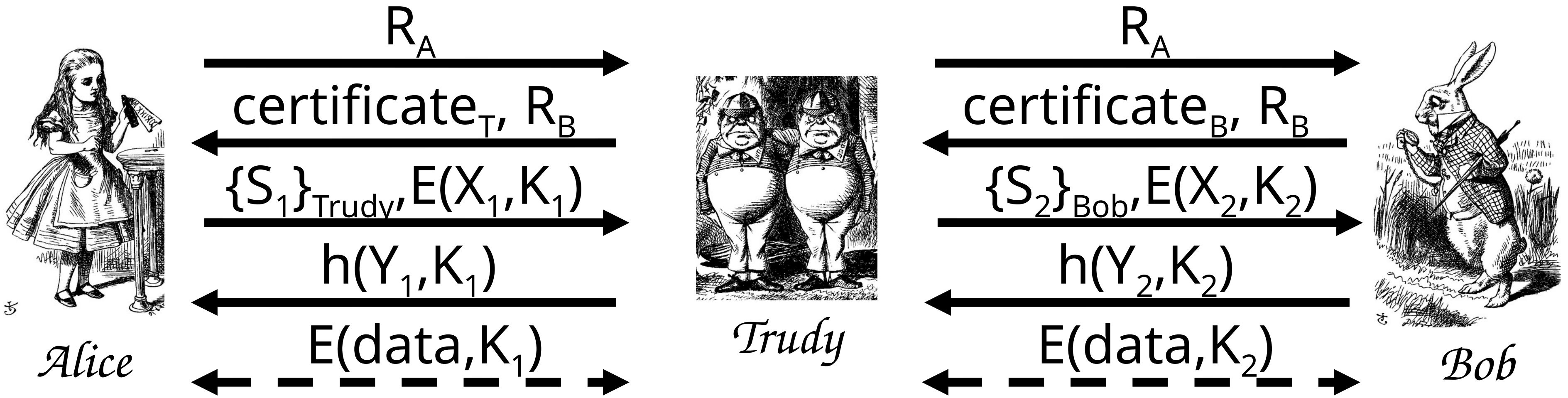
SSL Keys

- 6 "keys" derived from K = $h(S,R_A,R_B)$
  - 2 encryption keys: client and server
  - 2 integrity keys: client and server
  - 2 IVs: client and server
  - 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?



R_A

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

certificate$_B$, R_B

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

$h(Y_2,K_2)$

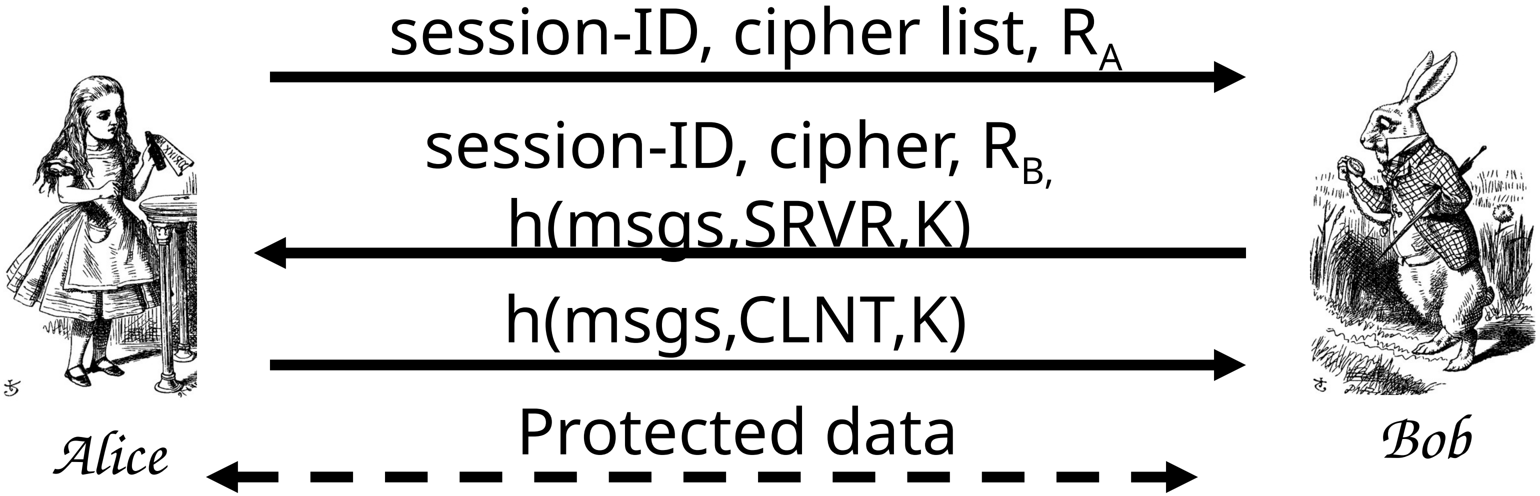
$E(\text{data},K_2)$

- **Q:** What prevents this MiM "attack"?
- **A:** Bob's certificate must be signed by a certificate authority (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



session-ID, cipher list, R_A

session-ID, cipher, $R_{B,}$ h(msgs,SRVR,K)

h(msgs,CLNT,K)

Protected data

Alice

Bob

- 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)

SSL vs IPSec

- IPSec discussed in next section
  - Lives at the network layer (part of the OS)
  - Encryption, integrity, authentication, etc.
  - Is overly complex, has some security “issues”
- SSL (and IEEE standard known as TLS)
  - Lives at socket layer (part of user space)
  - Encryption, integrity, authentication, etc.
  - Relatively simple and elegant specification

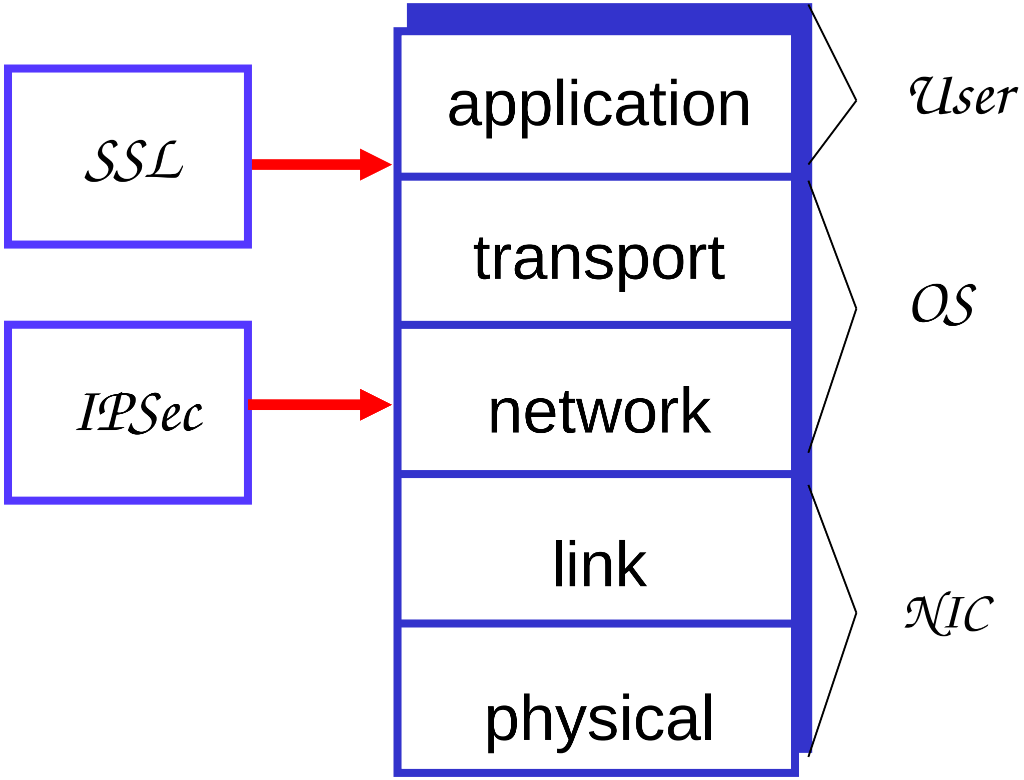
SSL vs IPSec

- IPSec: OS must be aware, but not apps
- SSL: Apps must be aware, but not OS
- SSL built into Web early-on (Netscape)
- IPSec often used in VPNs (secure tunnel)
- Reluctance to retrofit applications for SSL
- IPSec not widely deployed (complexity, etc.)
- The bottom line?
- **Internet less secure than it could be!**

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

- 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?

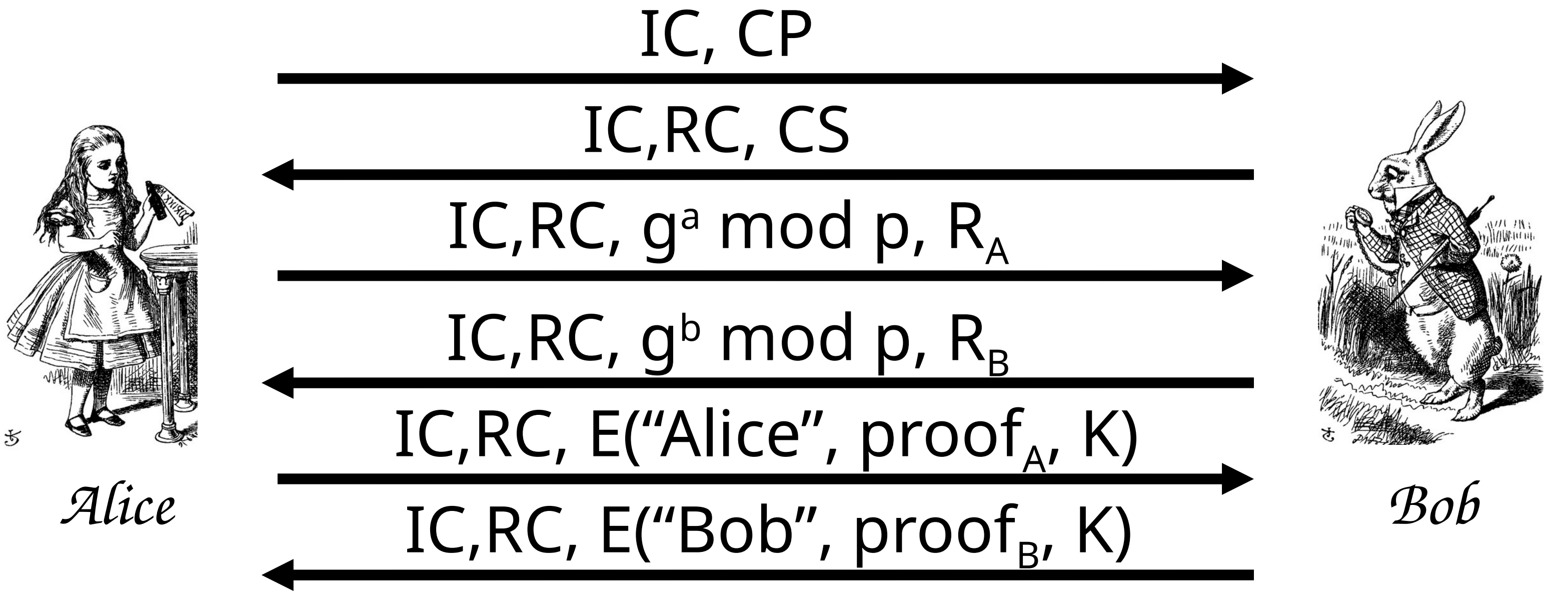
IKE Phase 1

- *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)



IC, CP

IC,RC, CS

IC,RC, g^a mod p, R_A

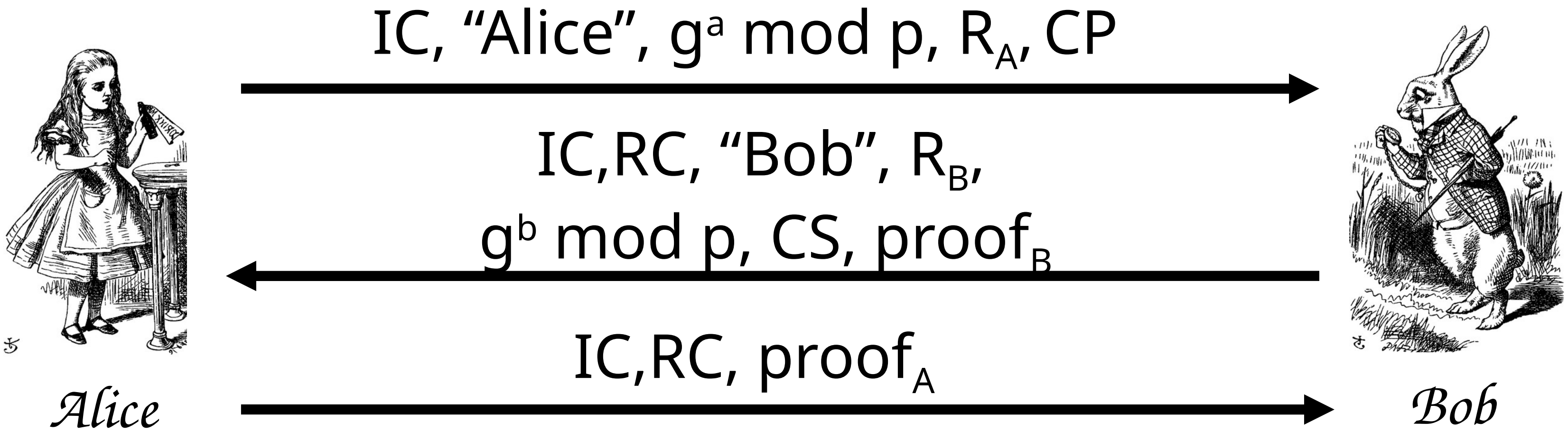
IC,RC, g^b mod p, R_B

IC,RC, E(“Alice”, proof_A, K)

IC,RC, E(“Bob”, proof_B, K)

- 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”)]$_{\text{Alice}}$

IKE Phase 1: Public Key Signature (Aggressive Mode)



IC, “Alice”, $g^a \bmod p$, R_A, CP

IC,RC, “Bob”, R_B,
$g^b \bmod p$, CS, proof_B

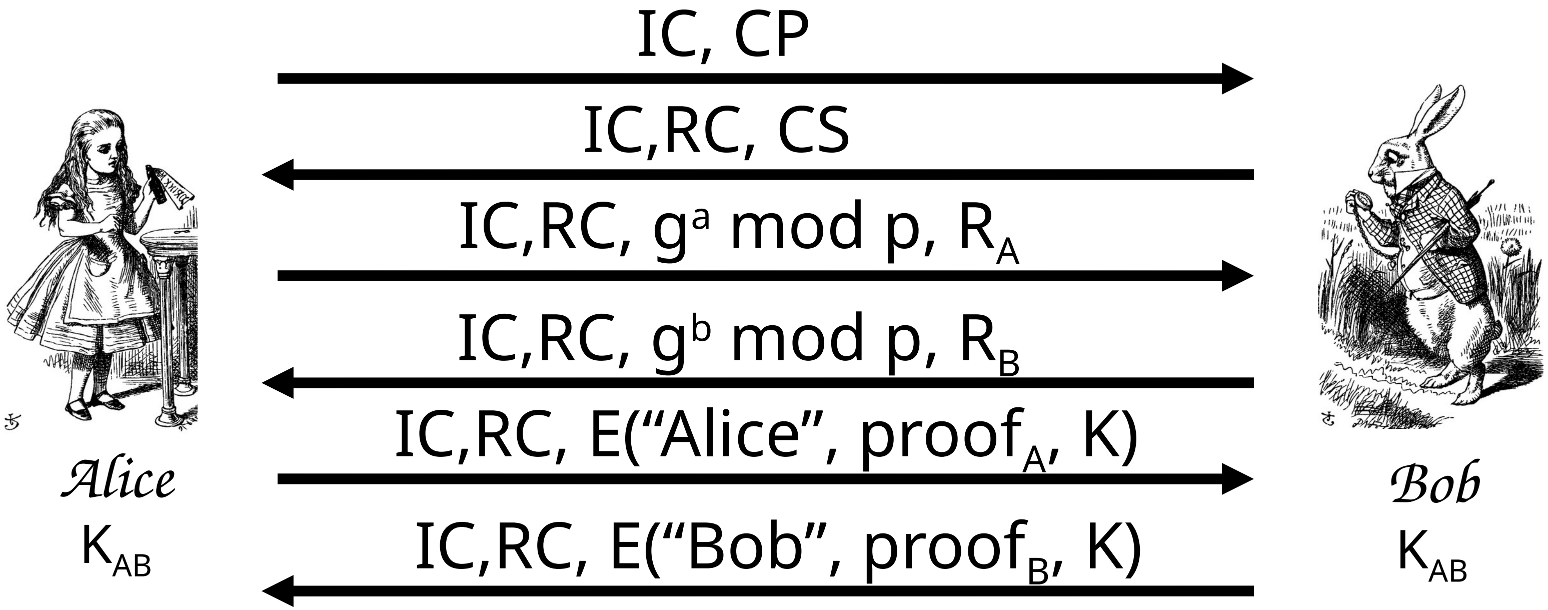
IC,RC, proof_A

- 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)



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)

Alice
K_{AB}

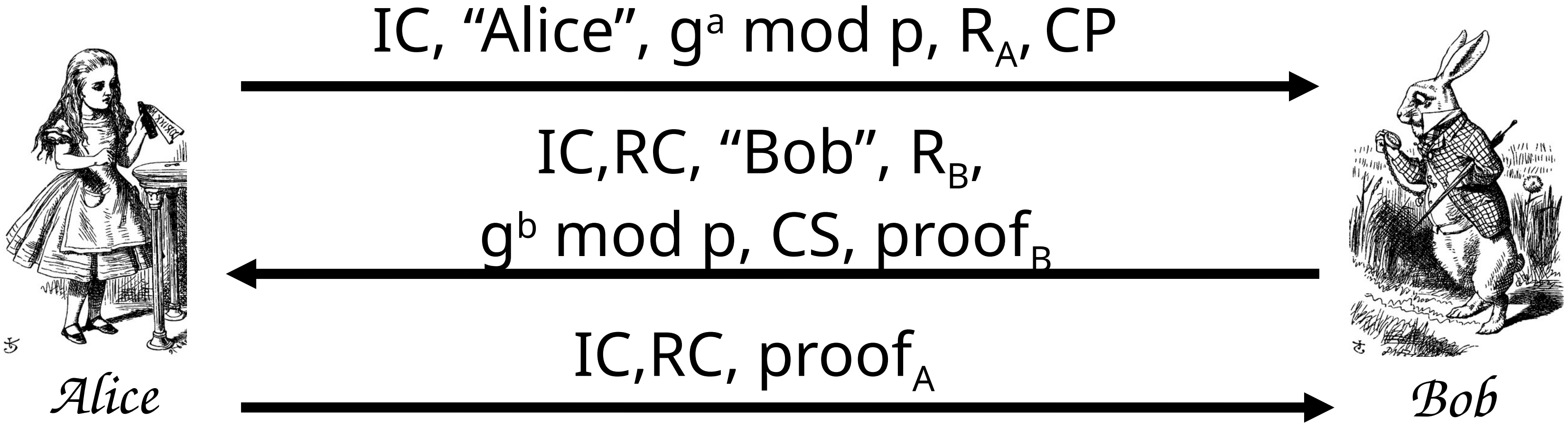
Bob
K_{AB}

- *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)

- 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)



IC, “Alice”, g^a mod p, R_A, CP

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

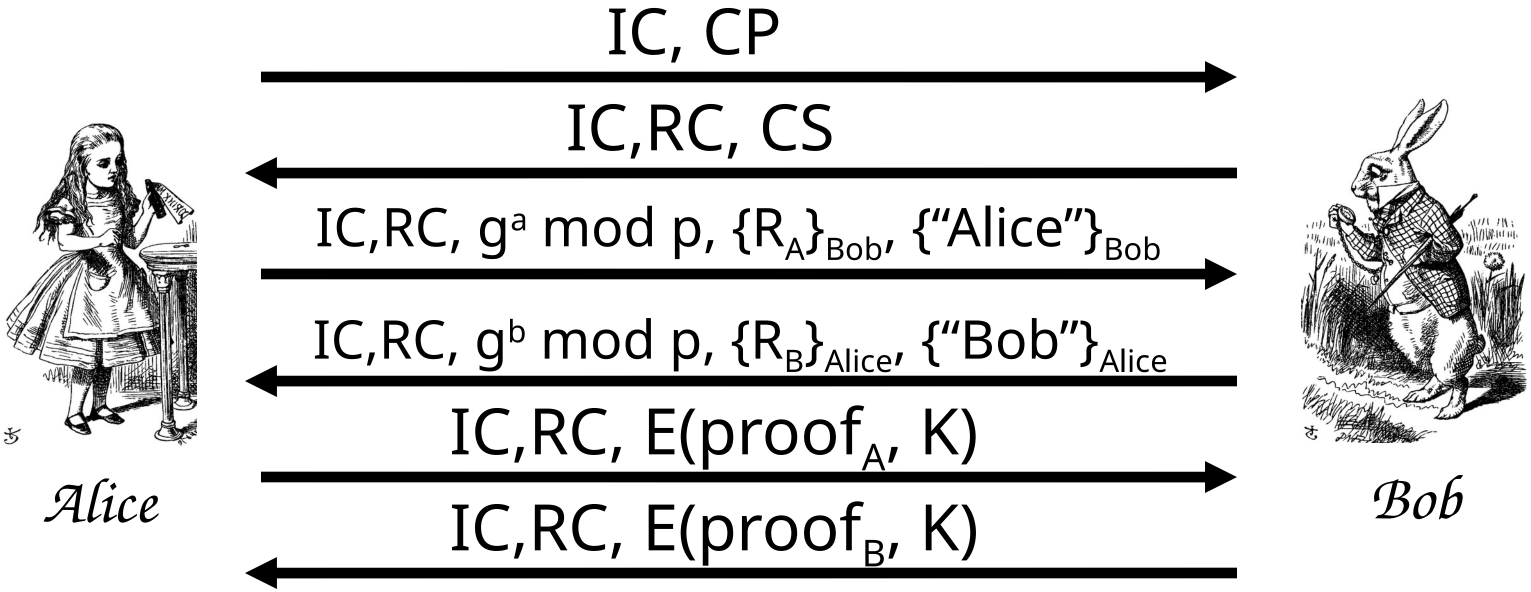
IC,RC, proof_A

Alice

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)



IC, CP

IC,RC, CS

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

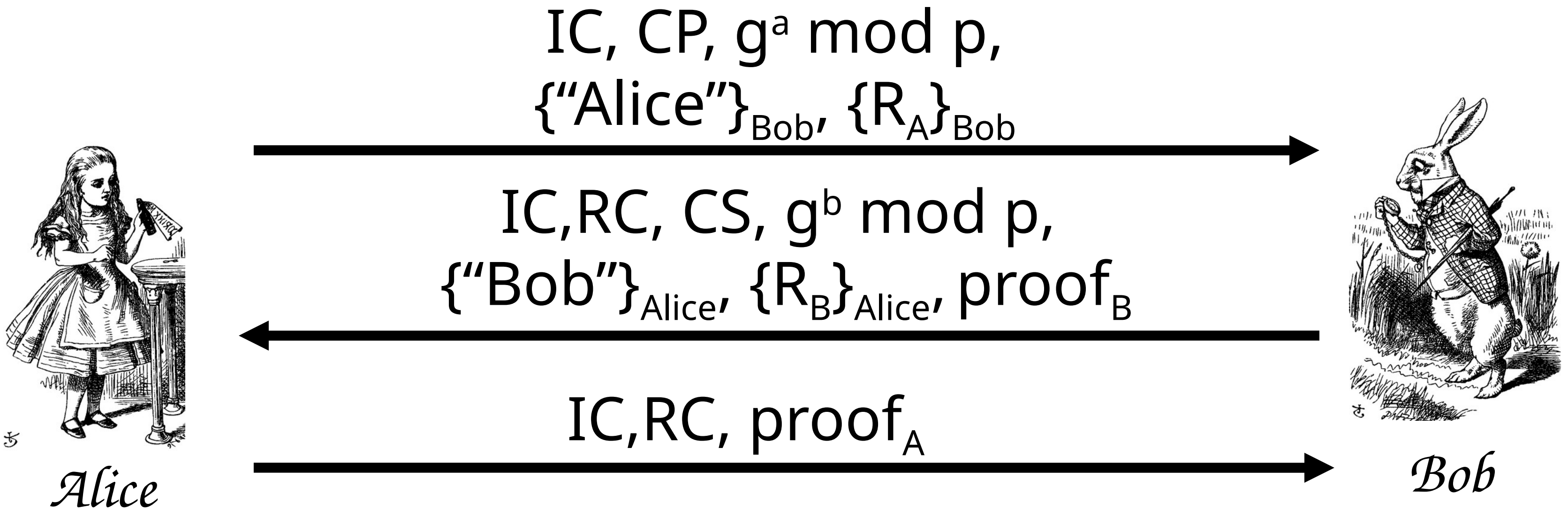
IC,RC, g^b mod p, $\{R_B\}_{Alice}$, $\{\text{“Bob”}\}_{Alice}$

IC,RC, E(proof_A, K)

IC,RC, E(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”)

IKE Phase 1: Public Key Encryption (Aggressive Mode)



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

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

IC,RC, proof_A

Alice

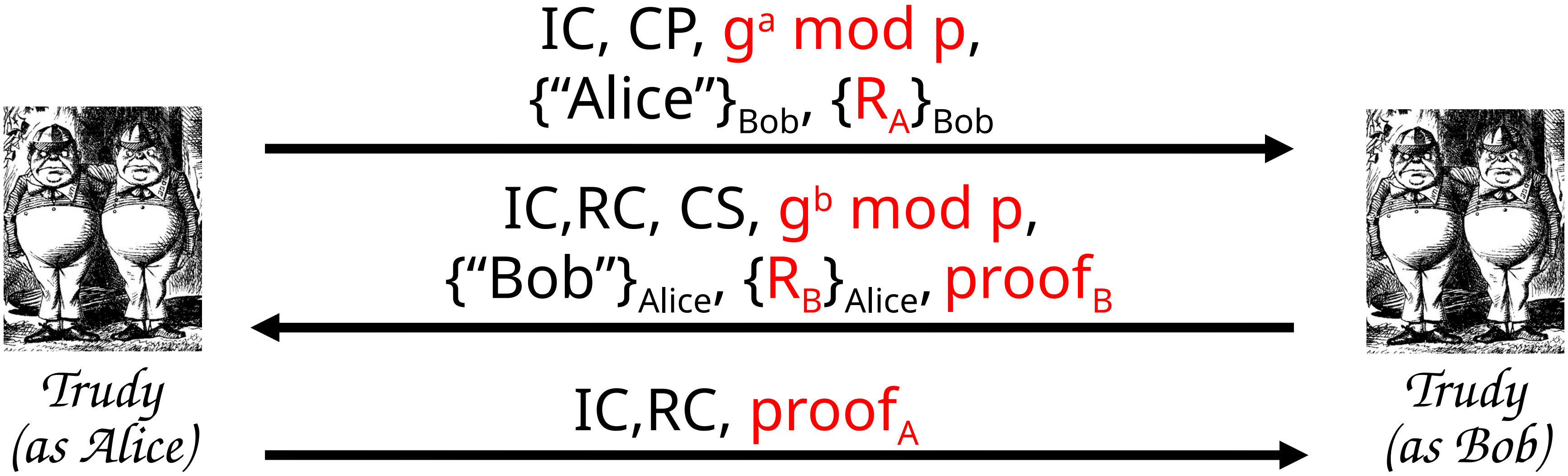
Bob

- 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?



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 Alice)

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

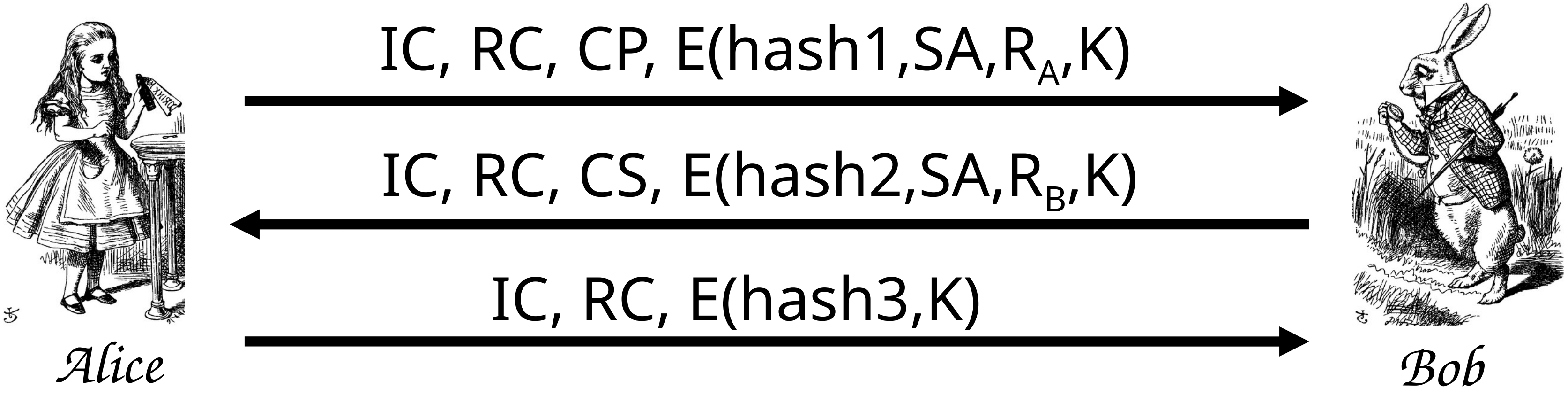
IKE Phase 1 Summary

- Result of IKE phase 1 is
  - Mutual authentication
  - Shared symmetric key
  - IKE **Security Association (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



IC, RC, CP, E(hash1,SA,R_A,K)

IC, RC, CS, E(hash2,SA,R_B,K)

IC, RC, E(hash3,K)

Alice

Bob

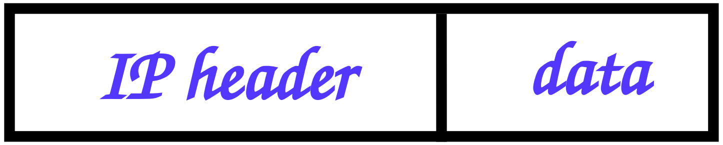
- *Key* K, IC, RC *and* SA *known from Phase 1*
- *Proposal* CP *includes ESP and/or AH*
- *Hashes 1,2,3 depend on* SKEYID, SA, R_A *and* R_B
- *Keys derived from* KEYMAT = h(SKEYID,R_A,R_B,junk)
- *Recall* SKEYID *depends on phase 1 key method*
- *Optional PFS (ephemeral Diffie-Hellman exchange)*

IPSec

- After IKE Phase 1, we have an IKE SA
- After IKE Phase 2, we have an IPSec SA
- Authentication completed and have a shared symmetric key (session key)
- Now what?
  - We want to protect **IP datagrams**
  - But what is an IP datagram?
  - From the perspective of IPSec…

IP Review

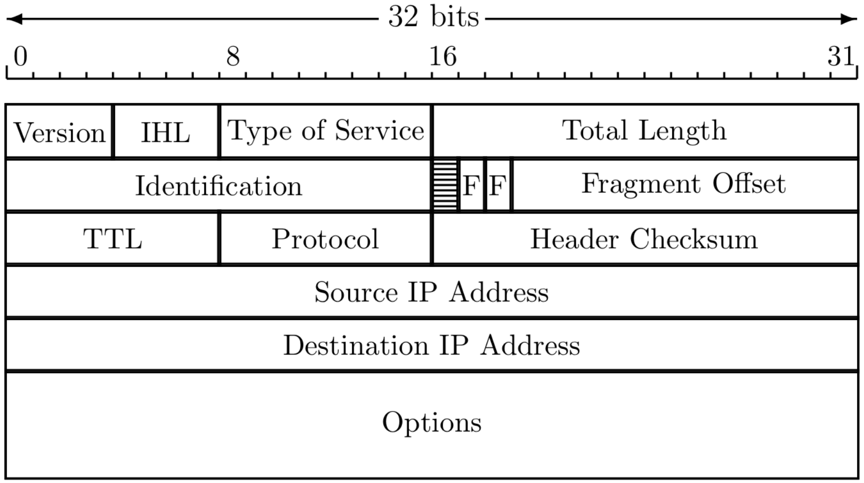
- IP datagram is of the form



IP header

data

- Where IP header is



32 bits

0 8 16 31

Version | IHL | Type of Service | Total Length

Identification | F | F | Fragment Offset

TTL | Protocol | Header Checksum

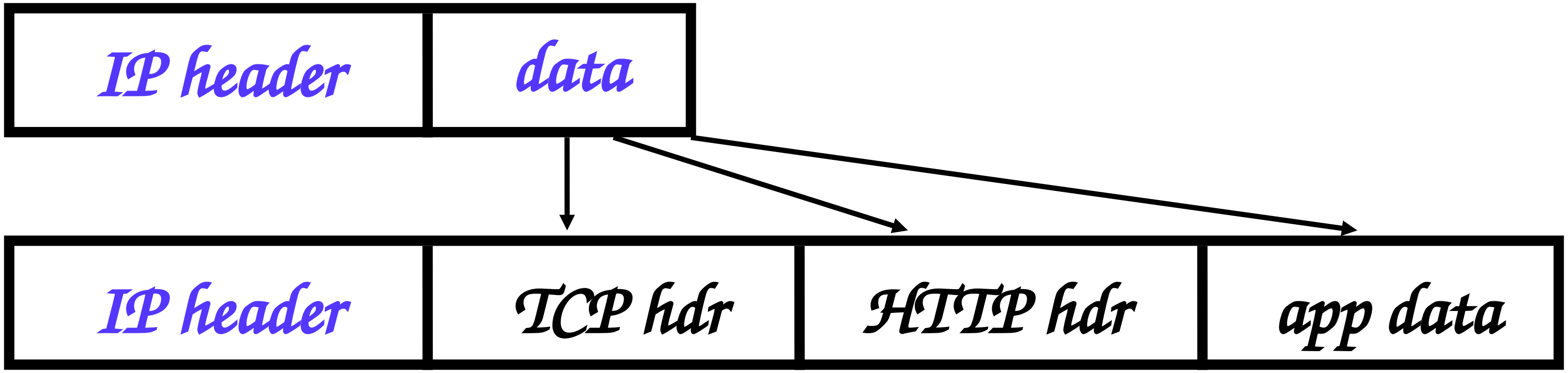
Source IP Address

Destination IP Address

Options

IP and TCP

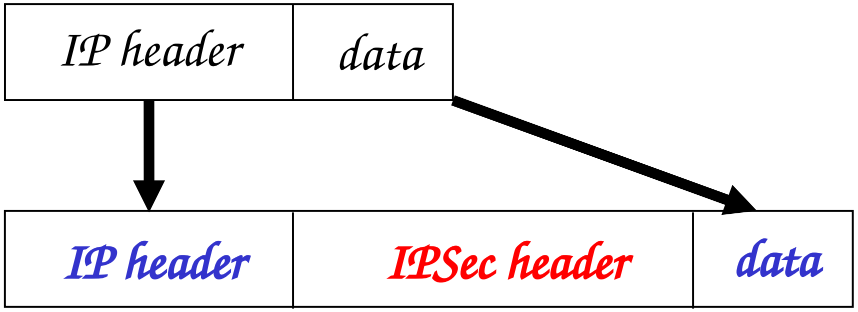
- Consider Web traffic, for example
  - IP encapsulates TCP and…
  - …TCP encapsulates HTTP



- IP **data** includes TCP header, etc.

IPSec Transport Mode

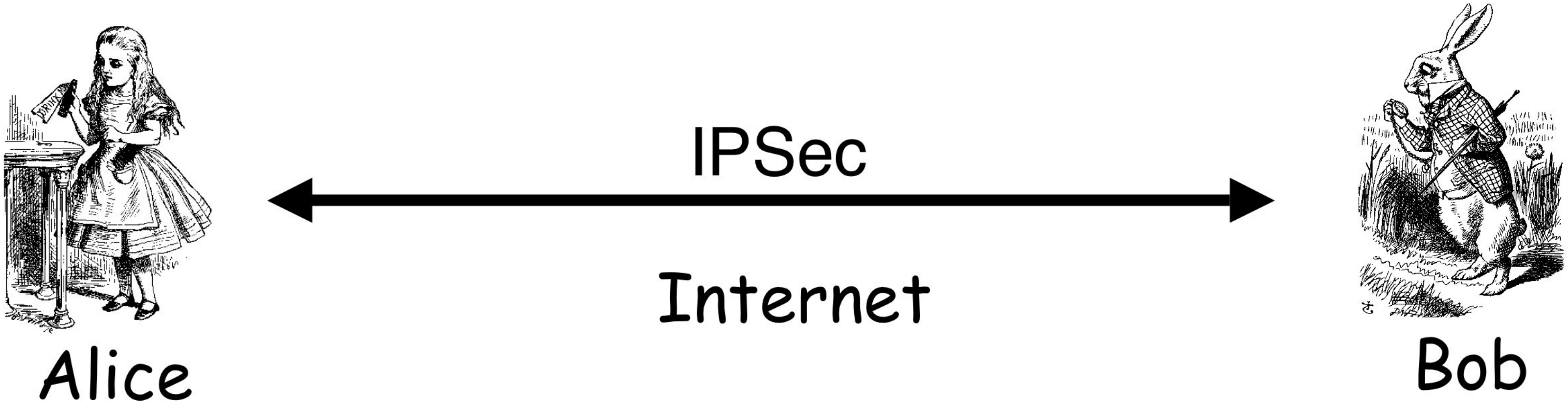
- IPSec **Transport Mode**



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

IPSec: Host-to-Host

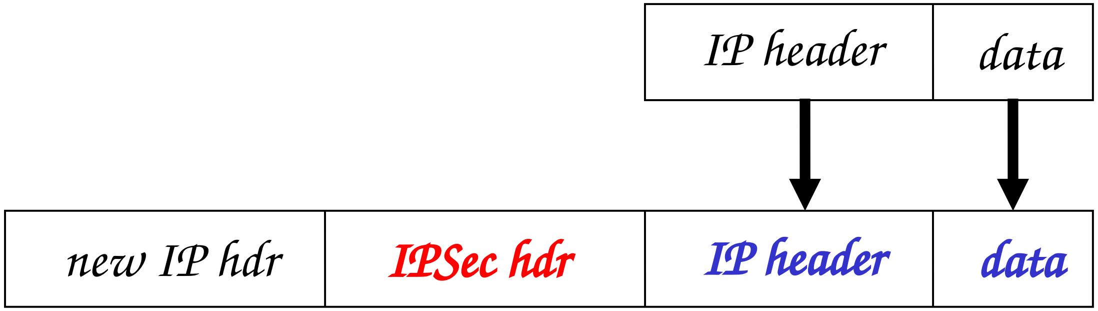
- IPSec transport mode used here



- There may be firewalls in between
  - If so, is that a problem?

IPSec Tunnel Mode

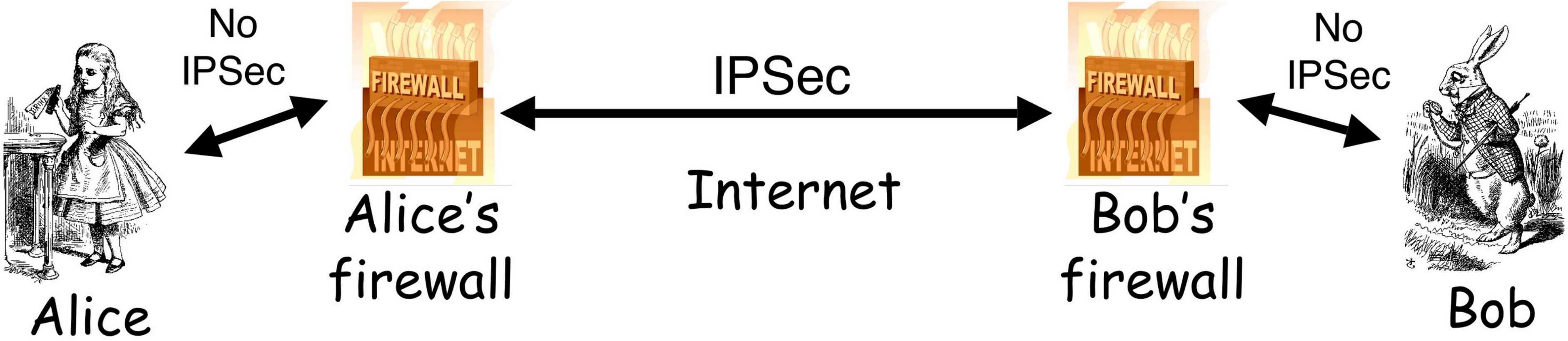
- IPSec **Tunnel Mode**



- 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

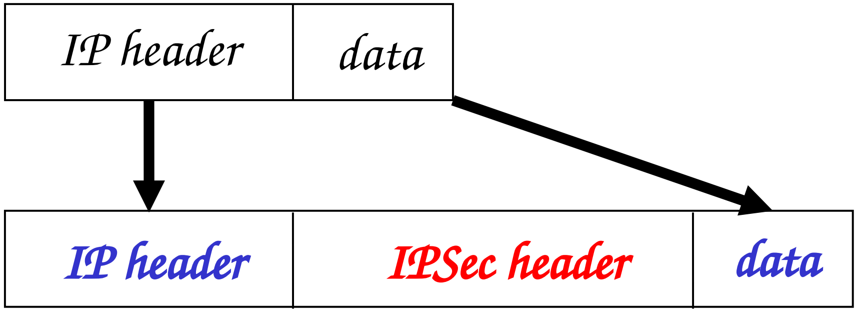
- IPSec tunnel mode used here



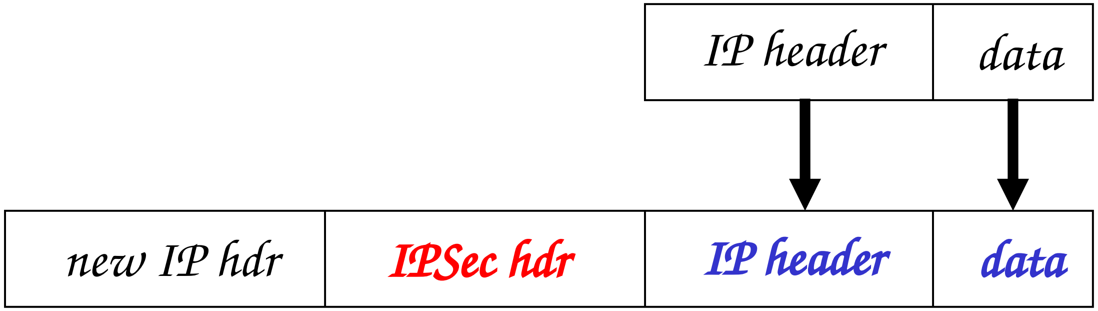
- Note: Local networks not protected
- Is there any advantage here?

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 allow some combinations of these

AH vs ESP

- AH Authentication Header
  - **Integrity only** (no confidentiality)
  - Integrity-protect everything beyond IP header and some fields of header (why not all fields?)
- ESP Encapsulating Security Payload
  - Integrity and **confidentiality** both **required**
  - Protects everything beyond IP header
  - Integrity-only by using NULL encryption

ESP 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
- Is ESP with NULL encryption same as AH ?

Why Does AH Exist? (1)

- 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 integrity protect all header fields
  - TTL, for example, will change
- ESP does not protect IP header at all

Why Does AH Exist? (2)

- ESP encrypts everything beyond the IP header (if non-null encryption)
- If ESP-encrypted, firewall cannot look at TCP header in host-to-host case
- Why not use ESP with NULL encryption?
  - Firewall sees ESP header, but does not know whether null encryption is used
  - End systems know, but **not** the firewalls

Why Does AH Exist? (3)

- The real reason why AH exists:
  - At one IETF meeting "someone from Microsoft gave an impassioned speech about how AH was useless…"
  - "…everyone in the room looked around and said `Hmm. He's right, and we hate AH also, but if it annoys Microsoft let's leave it in since we hate Microsoft more than we hate AH.' "

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 Needham and Schroeder protocol
  - 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 (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

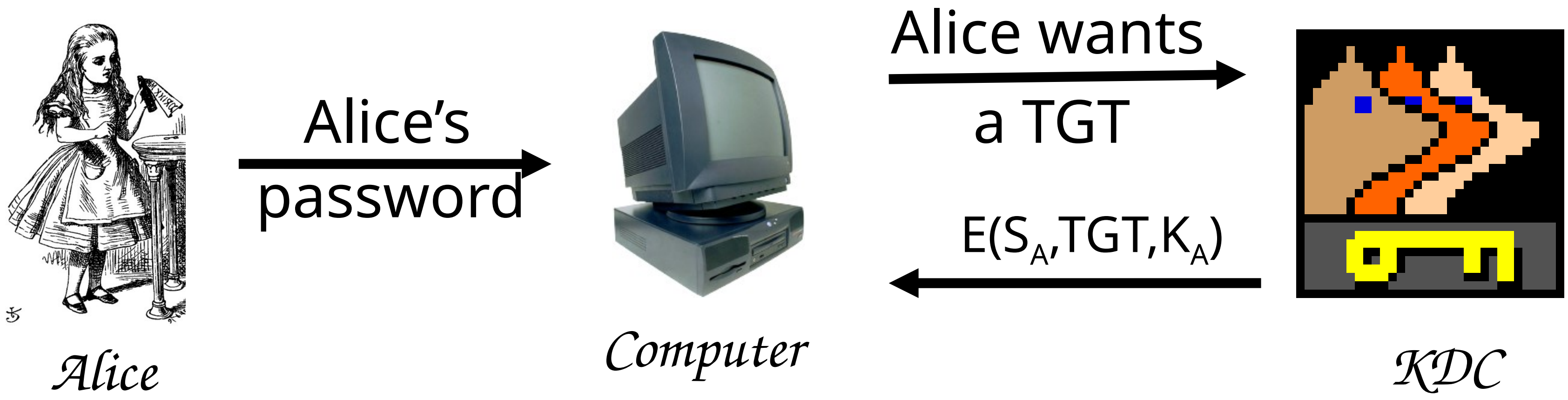
Kerberos Tickets

- 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



Alice's password

Alice wants a TGT

$E(S_A,TGT,K_A)$

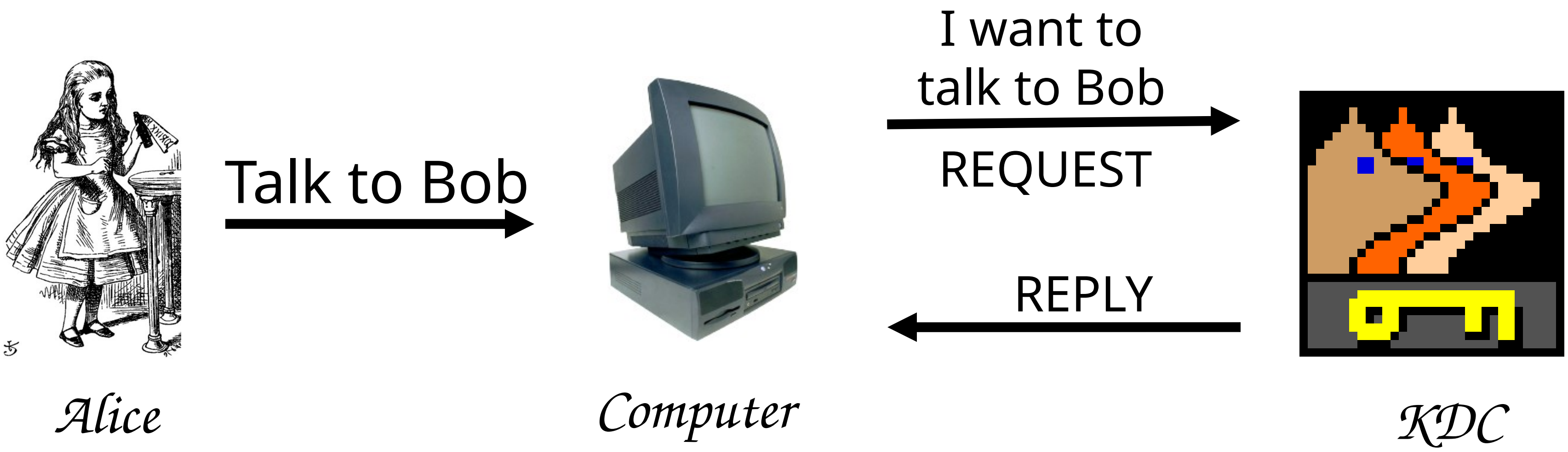
Alice

Computer

KDC

- 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"



Talk to Bob

I want to talk to Bob

REQUEST

REPLY

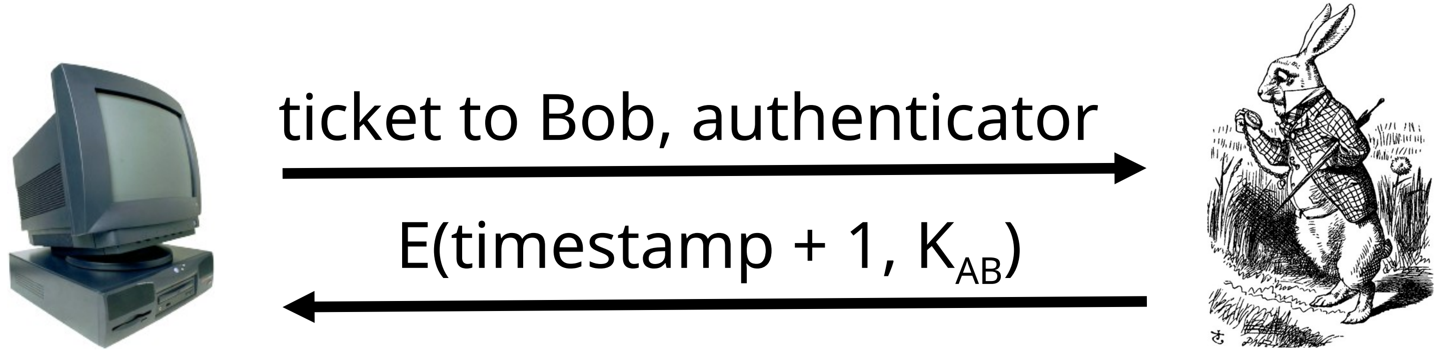
Alice

Computer

KDC

- 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, 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

- 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("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

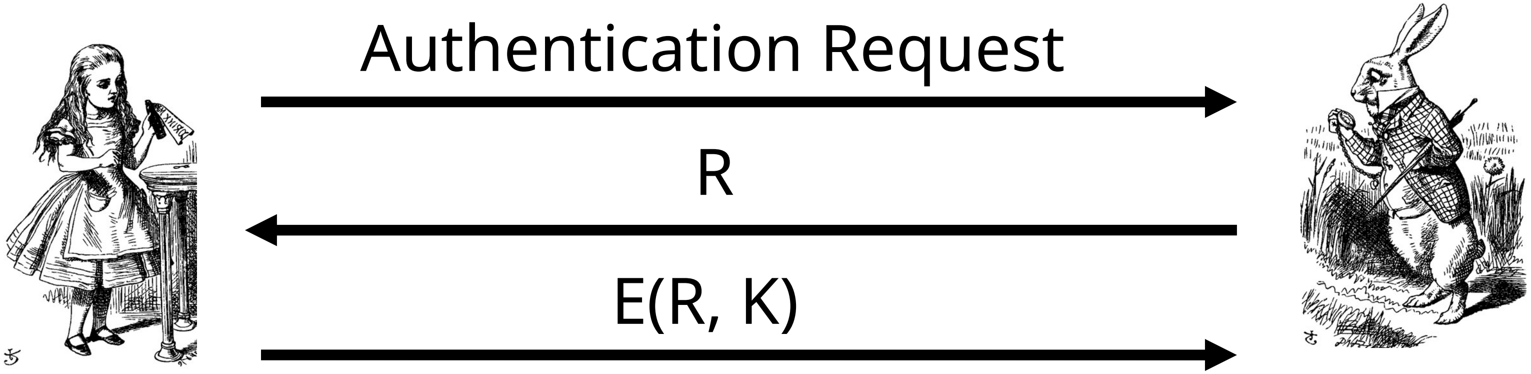
- In Kerberos, K_A = h(Alice's password)
- Could instead generate random K_A
  - Compute K_h = h(Alice's password)
  - And Alice's computer stores $E(K_A, K_h)$
- Then K_A need not change when Alice changes her password
  - But $E(K_A, K_h)$ must be stored on computer
- This alternative approach is often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

WEP

WEP

- WEP   Wired Equivalent Privacy
- The stated goal of WEP is to **make wireless LAN as secure as a wired LAN**
- According to Tanenbaum:
  - "The 802.11 standard prescribes a data link-level security protocol called WEP (Wired Equivalent Privacy), which is designed to make the security of a wireless LAN as good as that of a wired LAN. Since the default for a wired LAN is no security at all, this goal is easy to achieve, and WEP achieves it as we shall see."

WEP Authentication



Authentication Request

R

E(R, K)

Alice, K

Bob, K

- Bob is **wireless access point**
- Key K shared by access point and **all users**
  - Key K seldom (if ever) changes
- WEP has many, many, many security flaws

WEP Issues

- WEP uses RC4 cipher for confidentiality
  - RC4 can be a strong cipher
  - But WEP introduces a subtle flaw…
  - …making cryptanalytic attacks feasible
- WEP uses CRC for "integrity"
  - Should have used a MAC, HMAC, or similar
  - CRC is for error detection, not crypto integrity
  - **Everyone** should know **NOT** to use CRC here…

WEP Integrity Problems

- WEP "integrity" gives no crypto integrity
  - CRC is linear, so is stream cipher (XOR)
  - Trudy can change **ciphertext and CRC** so that checksum on **plaintext** remains valid
  - Then Trudy's introduced changes go undetected
  - Requires no knowledge of the plaintext!
- CRC does **not** provide a cryptographic integrity check
  - CRC designed to detect random errors
  - Not to detect intelligent changes

More WEP Integrity Issues

- *Suppose Trudy knows destination IP*
- *Then Trudy also knows keystream used to encrypt IP address, since*

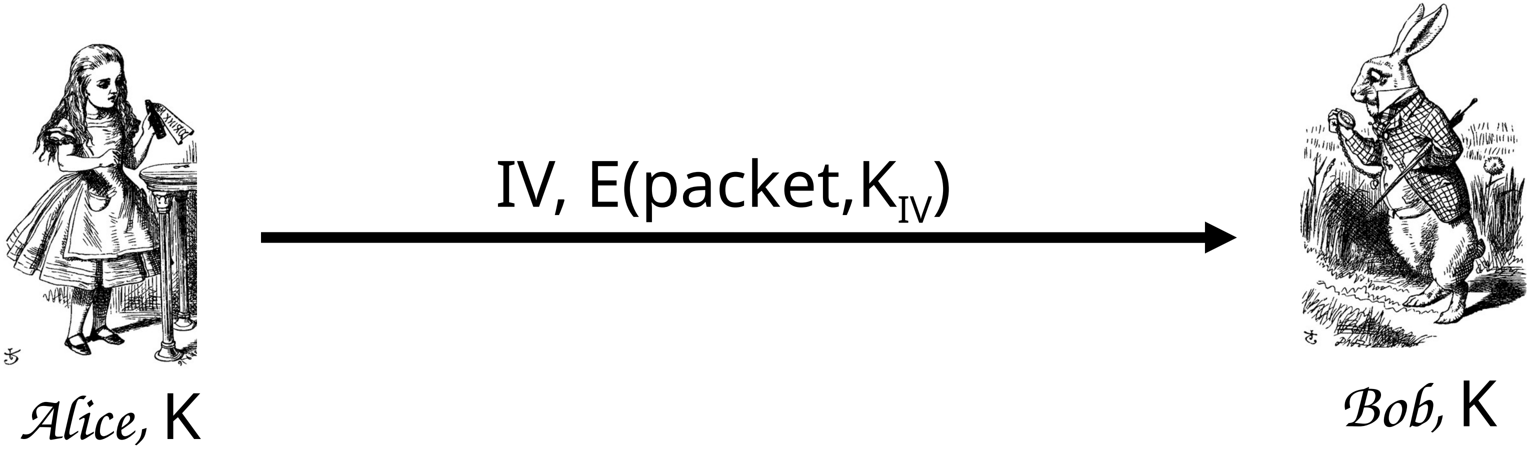
  C = destination IP address 🗐 **keystream**
- *Then Trudy can replace* **C** *with*

  C🔔= Trudy’s IP address 🗐 **keystream**
- *And change the CRC so no error detected*
  - *Then what happens??*
- *Moral: Big problems when integrity fails*

WEP Key

- Recall WEP uses a long-term key K
- RC4 is a stream cipher, so each packet must be encrypted using a different key
  - Initialization Vector (IV) sent with packet
  - Sent in the clear, that is, IV is **not** secret
  - Note: IV similar to MI in WWII ciphers
- Actual RC4 key for packet is (IV,K)
  - That is, IV is ***pre-pended*** to long-term key K

WEP Encryption



IV, E(packet,K_{IV})

Alice, K

Bob, K

- K_{IV} = (IV,K)
  - *That is, RC4 key is* K *with 3-byte* IV *pre-pended*
- *The* IV *is known to Trudy*

WEP IV Issues

- WEP uses 24-bit (3 byte) IV
  - Each packet gets its own IV
  - Key: IV pre-pended to long-term key, K
- Long term key K seldom changes
- If long-term key and IV are same, then same keystream is used
  - This is bad, bad, really really bad!
  - Why?

WEP IV Issues

- Assume 1500 byte packets, 11 Mbps link
- Suppose IVs generated in sequence
  - Since $1500 \blacktriangledown 8/(11 \blacktriangledown 10^{6}) \blacktriangledown 2^{24} = 18{,}000$ seconds, an IV repeat in about 5 hours of traffic
- Suppose IVs generated at random
  - By birthday problem, some IV repeats in seconds
- Again, repeated IV (with same K) is ***bad***

Another Active Attack

- Suppose Trudy can insert traffic and observe corresponding ciphertext
  - Then she knows the keystream for some IV
  - She can decrypt any packet that uses that IV
- If Trudy does this many times, she can then decrypt data for lots of IVs
  - Remember, IV is sent in the clear
- Is such an attack feasible?

Cryptanalytic Attack

- WEP data encrypted using RC4
  - Packet key is IV with long-term key K
  - 3-byte IV is pre-pended to K
  - Packet key is (IV,K)
- Recall IV is sent in the clear (not secret)
  - New IV sent with every packet
  - Long-term key K seldom changes (maybe never)
- So Trudy always knows IV and ciphertext
  - Trudy wants to find the key K

Cryptanalytic Attack

- 3-byte IV pre-pended to key
- Denote the RC4 key **bytes** …
  - … as $K_0, K_1, K_2, K_3, K_4, K_5$, …
  - Where IV = (K_0, K_1, K_2) , which Trudy knows
  - Trudy wants to find K = $(K_3, K_4, K_5, \ldots)$
- Given enough IVs, Trudy can easily find key K
  - Regardless of the length of the key
  - Provided Trudy knows first keystream byte
  - **Known plaintext** attack (1st byte of each packet)
  - Prevent by discarding first 256 keystream bytes

WEP Conclusions

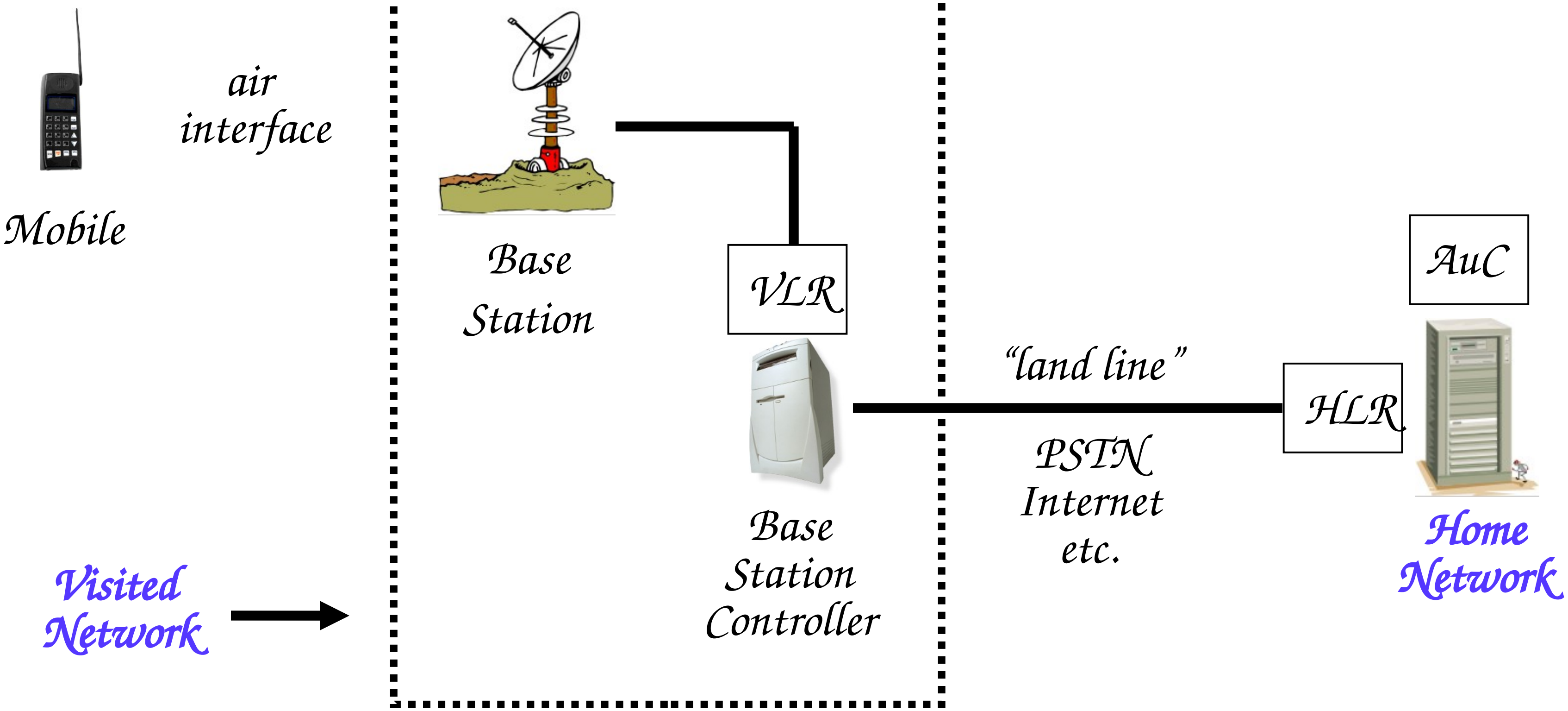
- Many attacks are practical
- Attacks have been used to recover keys and break real WEP traffic
- How to prevent these attacks?
  - Don't use WEP
  - Good alternatives: WPA, WPA2, etc.
- How to make WEP a little better?
  - Restrict MAC addresses, don't broadcast ID, …

GSM (In)Security

Cell Phones

- First generation cell phones
  - Brick-sized, analog, few standards
  - Little or **no** security
  - Susceptible to ***cloning***
- Second generation cell phones: ***GSM***
  - Began in 1982 as "Groupe Speciale Mobile"
  - Now, Global System for Mobile Communications
- Third generation?
  - 3rd Generation Partnership Project (3GPP)

GSM System Overview



Mobile

air interface

Base Station

VLR

Base Station Controller

"land line"

PSTN Internet etc.

HLR

AuC

Home Network

Visited Network

GSM System Components

- Mobile phone
  - Contains SIM (Subscriber Identity Module)
- SIM is the **security module**
  - IMSI (International Mobile Subscriber ID)
  - User key: Ki (128 bits)
  - Tamper resistant (smart card)
  - PIN activated (often not used)

GSM System Components

- **Visited network** network where mobile is currently located
  - Base station one "cell"
  - Base station controller manages many cells
  - VLR (Visitor Location Register) info on all visiting mobiles currently in the network
- **Home network** "home" of the mobile
  - HLR (Home Location Register) keeps track of most recent location of mobile
  - AuC (Authentication Center) has IMSI and Ki

GSM Security Goals

- Primary design goals
  - **Make GSM as secure as ordinary telephone**
  - **Prevent phone cloning**
- **Not** designed to resist an active attacks
  - At the time this seemed infeasible
  - Today such an attacks are clearly feasible…
- Designers considered biggest threats to be
  - Insecure billing
  - Corruption
  - Other low-tech attacks

GSM Security Features

- **Anonymity**
  - Intercepted traffic does not identify user
  - Not so important to phone company
- **Authentication**
  - Necessary for proper billing
  - Very, very important to phone company!
- **Confidentiality**
  - Confidentiality of calls over the air interface
  - Not important to phone company…
  - …except for marketing

GSM: Anonymity

- IMSI used to initially identify caller
- Then TMSI (Temporary Mobile Subscriber ID) used
  - TMSI changed frequently
  - TMSI's encrypted when sent
- Not a strong form of anonymity
- But probably useful in many cases

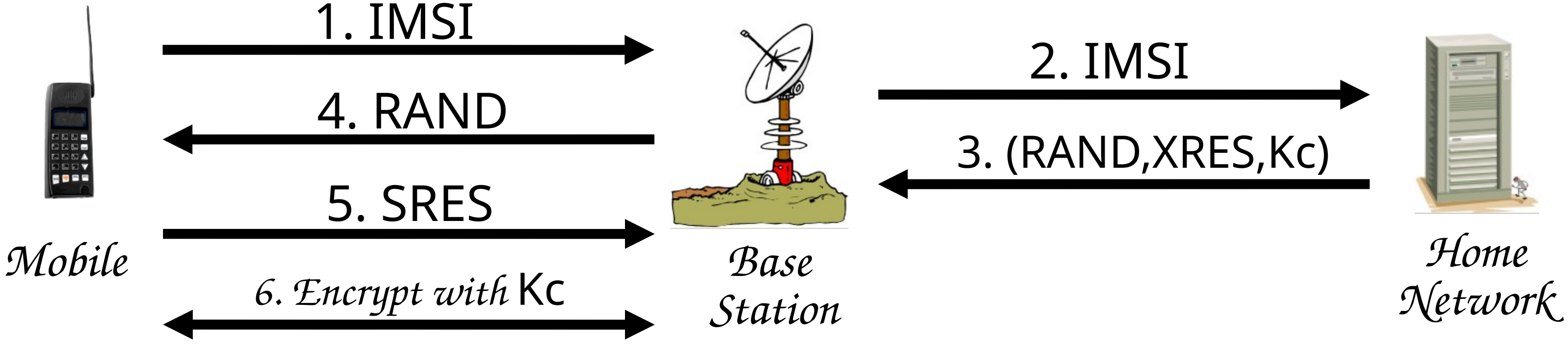
GSM: Authentication

- Caller is authenticated to base station
- Authentication is **not** mutual
- Authentication via **challenge-response**
  - Home network generates RAND and computes XRES = A3(RAND, Ki) where A3 is a hash
  - Then (RAND,XRES) sent to base station
  - Base station sends **challenge** RAND to mobile
  - Mobile's **response** is SRES = A3(RAND, Ki)
  - Base station verifies SRES = XRES
- **Note:** Ki never leaves home network

GSM: Confidentiality

- Data encrypted with stream cipher
- Error rate estimated at about 1/1000
  - Error rate is high for a block cipher
- Encryption key Kc
  - Home network computes Kc = A8(RAND, Ki) where A8 is a hash
  - Then Kc sent to base station with (RAND,XRES)
  - Mobile computes Kc = A8(RAND, Ki)
  - Keystream generated from A5(Kc)
- **Note:** Ki never leaves home network

GSM Security



1. IMSI

4. RAND

5. SRES

6. Encrypt with Kc

2. IMSI

3. (RAND,XRES,Kc)

Mobile

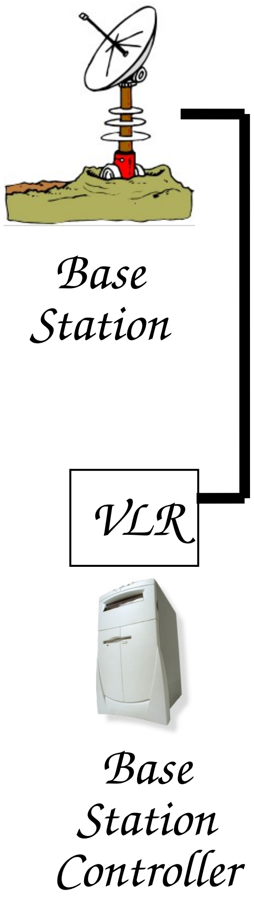
Base Station

Home Network

- SRES and Kc must be uncorrelated
  - Even though both are derived from RAND and Ki
- Must not be possible to deduce Ki from known RAND/SRES pairs (known plaintext attack)
- Must not be possible to deduce Ki from chosen RAND/SRES pairs (chosen plaintext attack)
  - With possession of SIM, attacker can choose RAND 's

GSM Insecurity (1)

- Hash used for A3/A8 is COMP128
  - Broken by 160,000 chosen plaintexts
  - With SIM, can get Ki in 2 to 10 hours
- Encryption between mobile and base station but **no encryption** from base station to base station controller
  - Often transmitted over microwave link
- Encryption algorithm A5/1
  - Broken with 2 seconds of known plaintext



Base Station

VLR

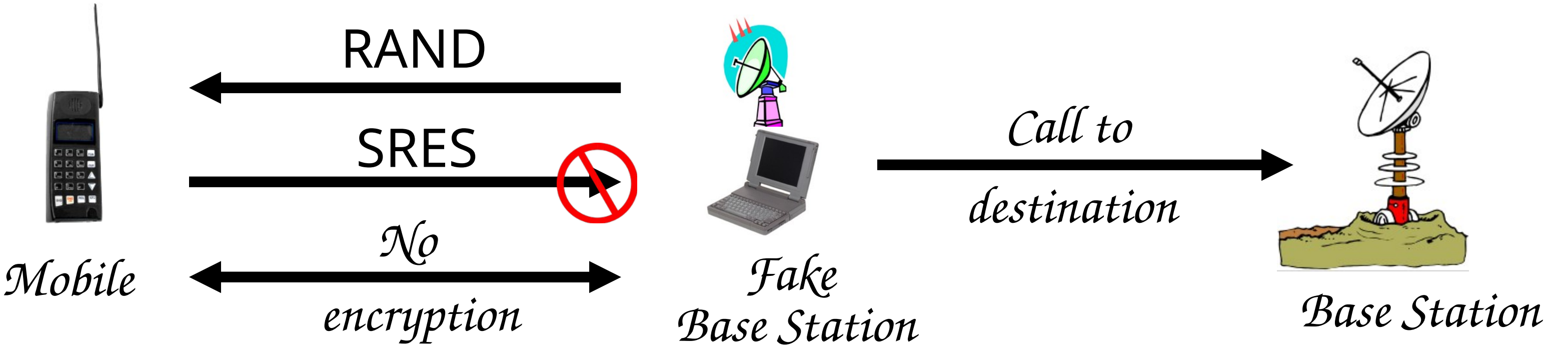
Base Station Controller

GSM Insecurity (2)

- Attacks on SIM card
  - **Optical Fault Induction** could attack SIM with a flashbulb to recover Ki
  - **Partitioning Attacks** using timing and power consumption, could recover Ki with only 8 adaptively chosen "plaintexts"
- With possession of SIM, attacker could recover Ki in seconds

GSM Insecurity (3)

- **Fake base station** exploits two flaws
  1. Encryption not automatic
  2. Base station not authenticated



- **Note:** GSM bill goes to fake base station!

GSM Insecurity (4)

- Denial of service is possible
  - Jamming (always an issue in wireless)
- Can replay triple: (RAND,XRES,Kc)
  - One compromised triple gives attacker a key Kc that is valid forever
  - No replay protection here

GSM Conclusion

- Did GSM achieve its goals?
  - Eliminate cloning? **Yes, as a practical matter**
  - Make air interface as secure as PSTN? **Perhaps…**
- But design goals were clearly too limited
- GSM insecurities   weak crypto, SIM issues, fake base station, replay, etc.
- PSTN insecurities   tapping, active attack, passive attack (e.g., cordless phones), etc.
- GSM a (modest) security success?

3rd Generation Partnership Project (3GPP)

- 3G security built on GSM (in)security
- 3G fixed known GSM security problems
  - Mutual authentication
  - Integrity-protect signaling (such as "start encryption" command)
  - Keys (encryption/integrity) cannot be reused
  - Triples cannot be replayed
  - Strong encryption algorithm (KASUMI)
  - Encryption extended to base station controller

Protocols Summary

- Generic authentication protocols
  - Protocols are subtle!
- SSH
- SSL
- IPSec
- Kerberos
- Wireless: next lecture…

Coming Attractions…

- Software and security
  - Software flaws   buffer overflow, etc.
  - Malware   viruses, worms, etc.
  - Software reverse engineering
  - Digital rights management
  - OS and security/NGSCB