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, GSM, and IPSec
- Implementation errors can occur
  - Recent IE implementation of SSL
- Not easy to get protocols right…

Ideal Security Protocol

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

Chapter 9:
Simple Security Protocols

"I quite agree with you," said the Duchess; "and the moral of that is—
'Be what you would seem to be' —or
if you'd like it put more simply—'Never imagine yourself not to be
otherwise than what it might appear to others that what you were
or might have been was not otherwise than what you
had been would have appeared to them to be otherwise.' "
— Lewis Carroll, *Alice in Wonderland*

Seek simplicity, and distrust it.
— Alfred North Whitehead

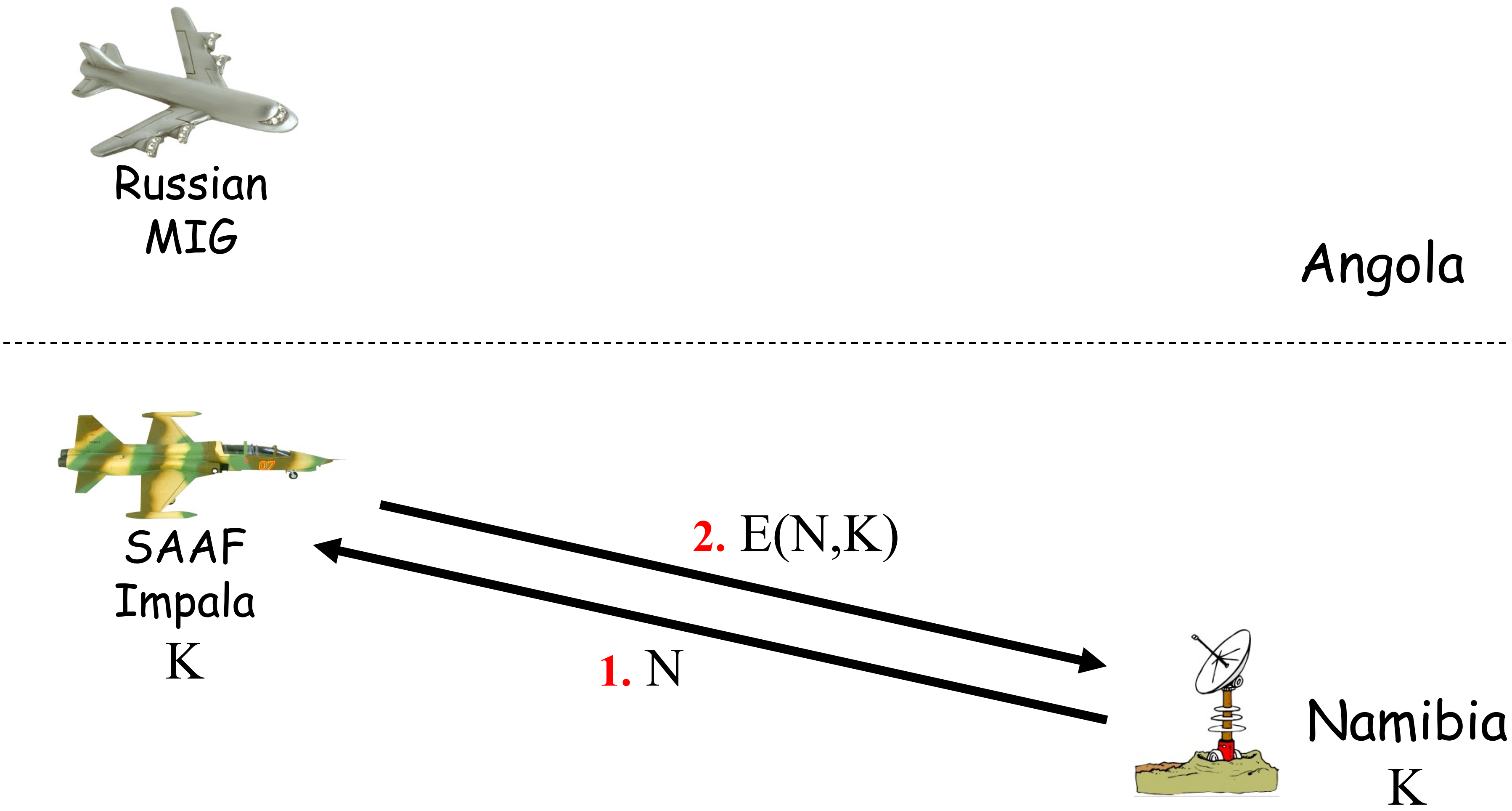
Secure Entry to NSA

1. Insert badge into reader
2. Enter PIN
3. Correct PIN?
    - Yes? Enter
    - No? Get shot by security guard

ATM Machine Protocol

1. Insert ATM card
2. Enter PIN
3. Correct PIN?
    - **Yes?** Conduct your transaction(s)
    - **No?** Machine (eventually) eats card

Identify Friend or Foe (IFF)



Russian
MIG

Angola

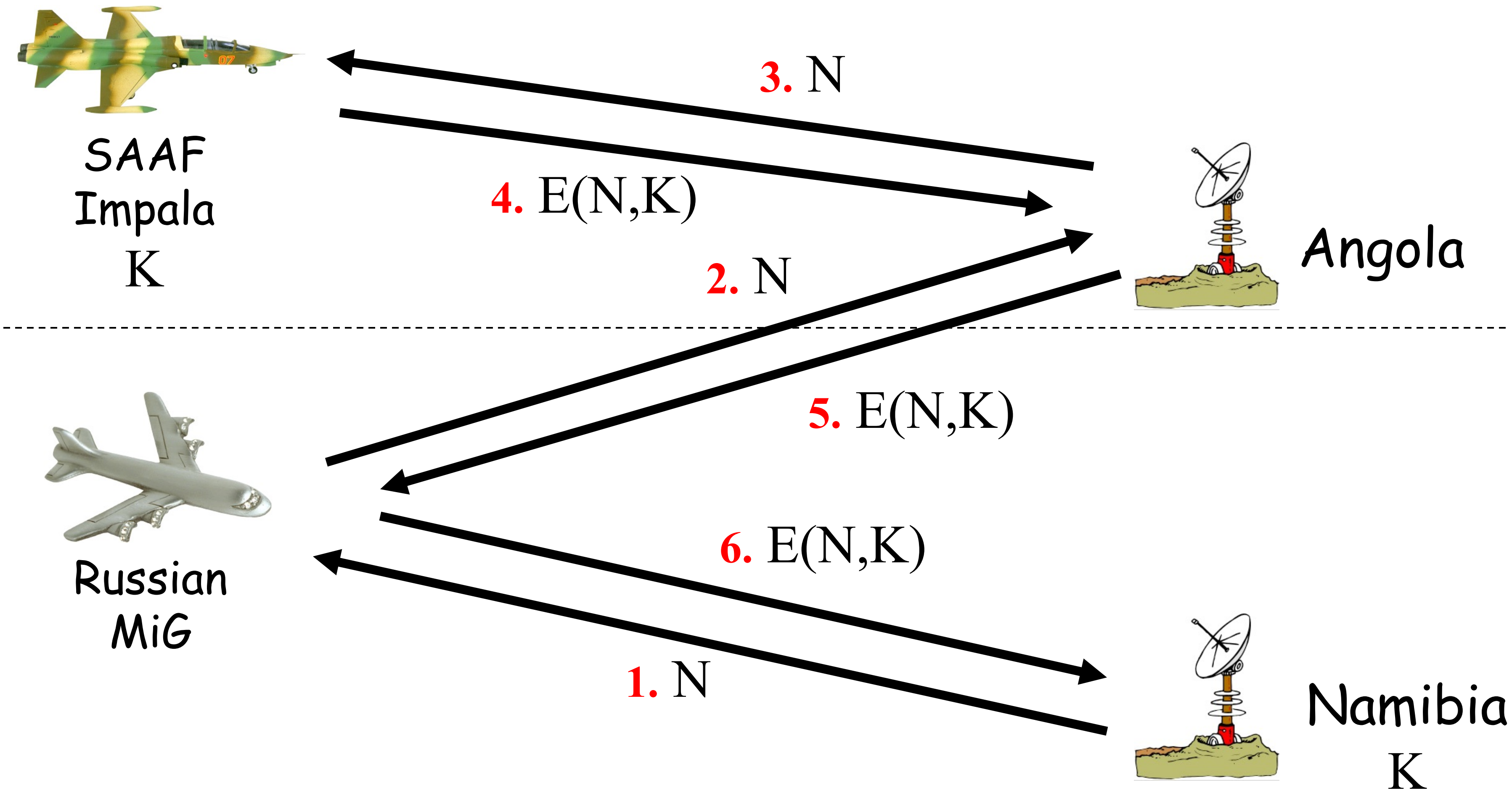
SAAF
Impala
K

2. E(N,K)

1. N

Namibia
K

MIG in the Middle



SAAF
Impala
K

3. N

4. E(N,K)

Angola

2. N

5. E(N,K)

Russian
MiG

6. E(N,K)

1. N

Namibia
K

Authentication Protocols

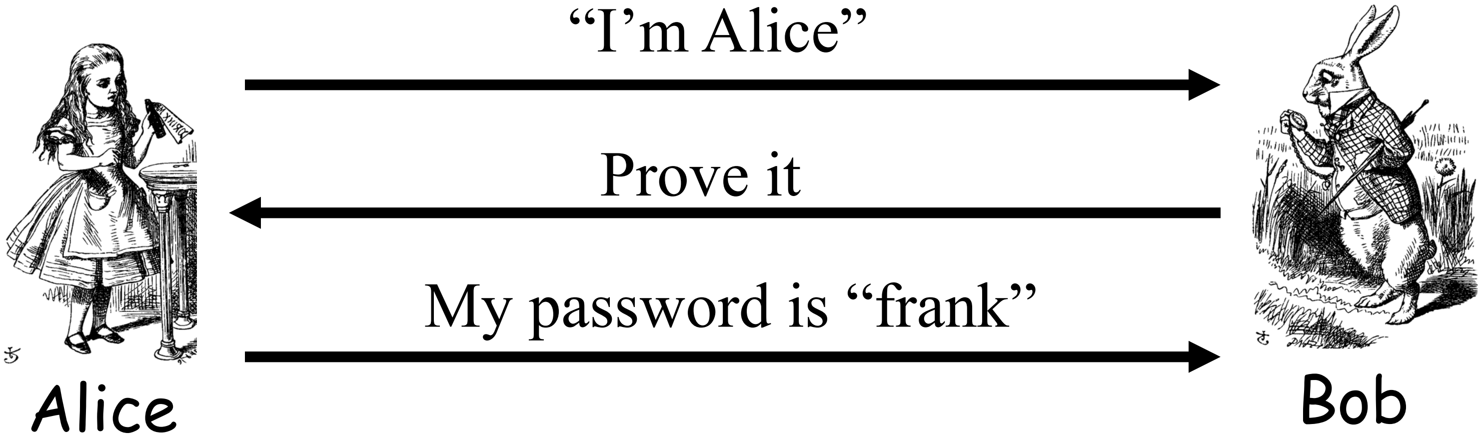
Authentication

- Alice must prove her identity to Bob
  - Alice and Bob can be humans or **computers**
- May also require Bob to prove he's Bob (mutual authentication)
- Probably need to establish a **session key**
- May have other requirements, such as
  - Use public keys
  - Use symmetric keys
  - Use hash functions
  - Anonymity, plausible deniability, etc., etc.

Authentication

- Authentication on a stand-alone computer is relatively simple
  - Hash password with salt
  - "Secure path," attacks on authentication software, keystroke logging, etc., can be issues
- Authentication over a network is challenging
  - Attacker can passively observe messages
  - Attacker can replay messages
  - Active attacks possible (insert, delete, change)

Simple Authentication



"I'm Alice"

Prove it

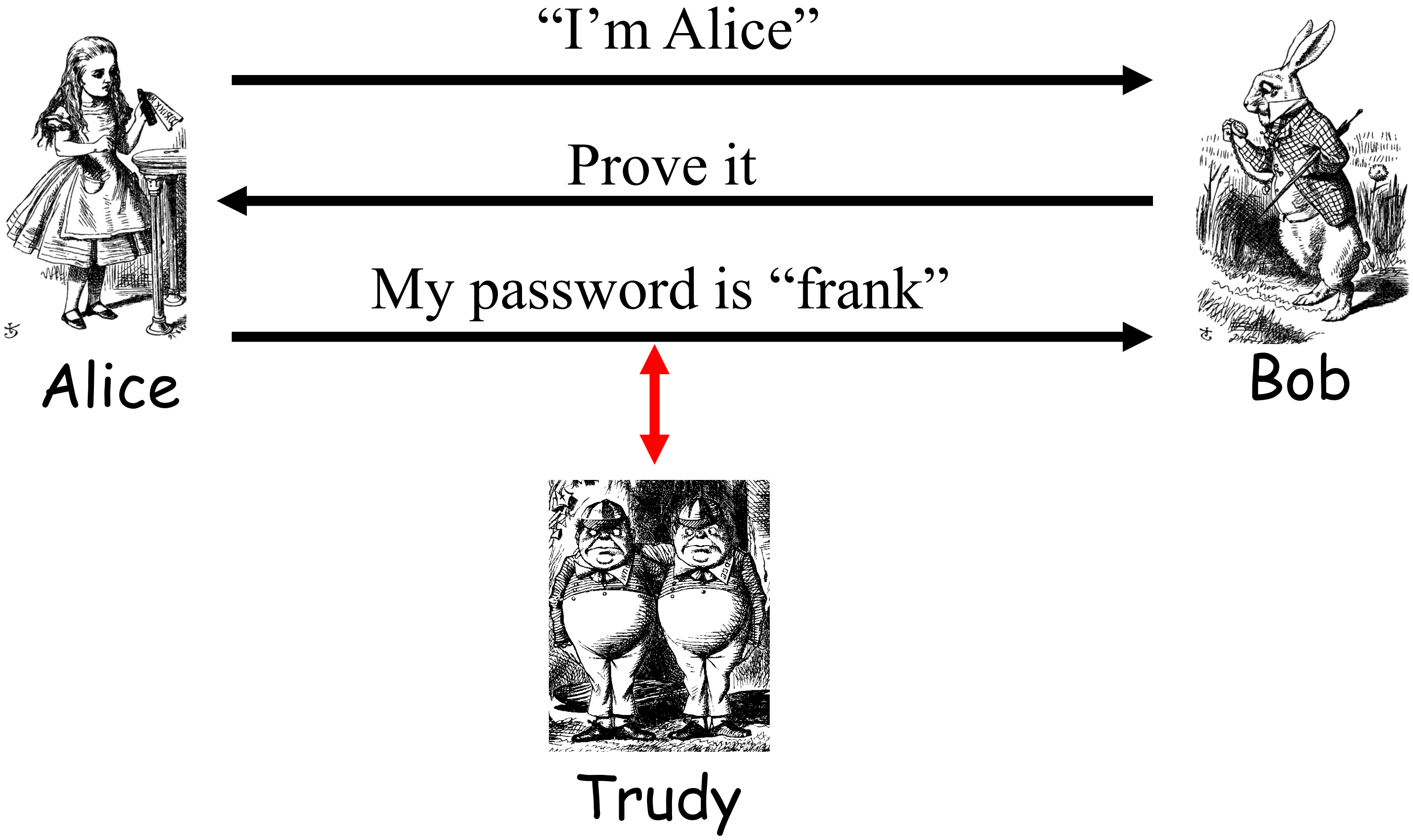
My password is "frank"

Alice

Bob

- Simple and may be OK for standalone system
- But insecure for networked system
    - Subject to a **replay** attack (next 2 slides)
    - Also, Bob must know Alice's password

Authentication Attack



“I’m Alice”

Prove it

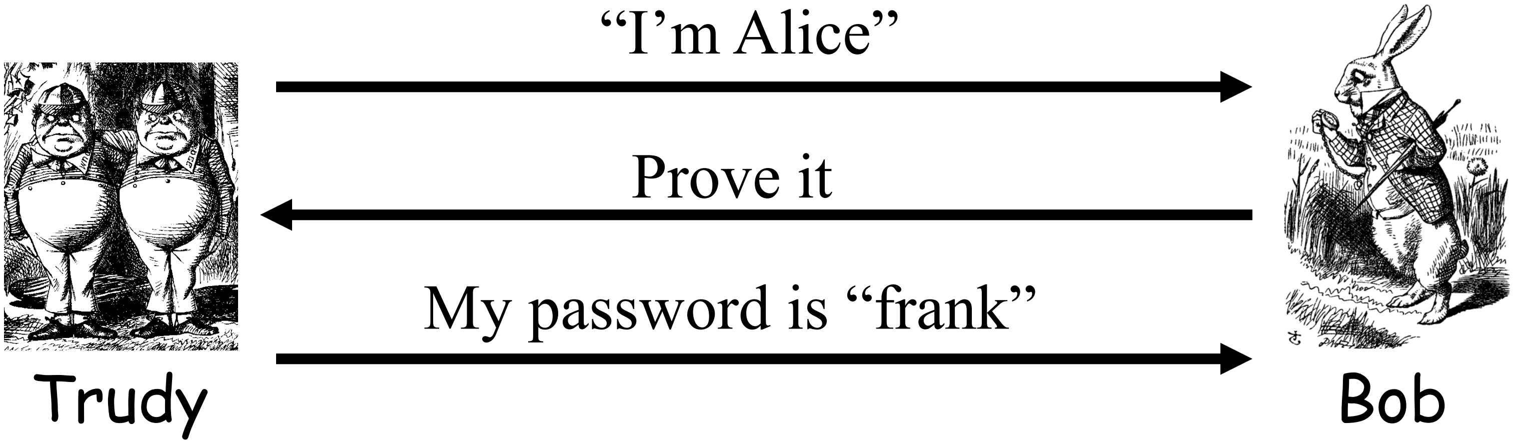
My password is “frank”

Alice

Bob

Trudy

Authentication Attack



"I'm Alice"

Prove it

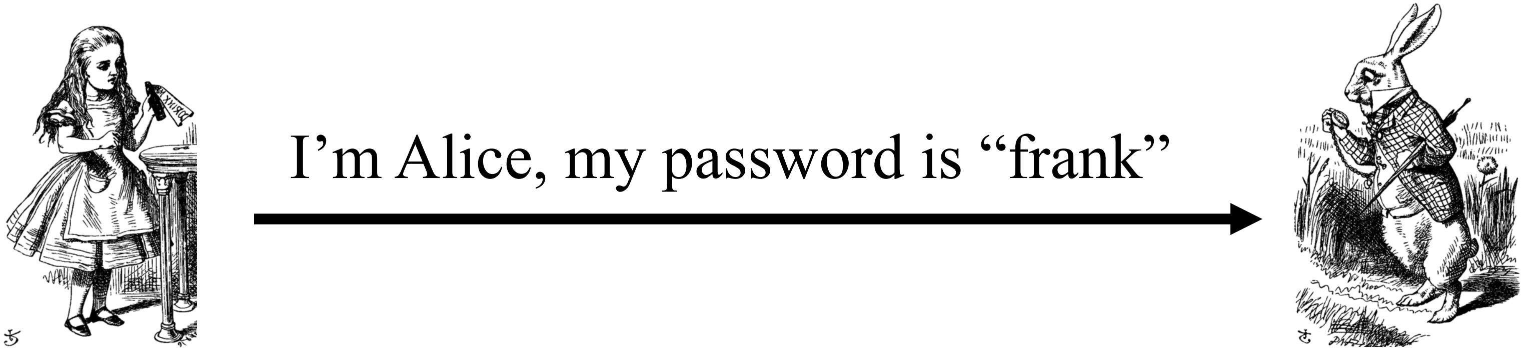
My password is "frank"

Trudy

Bob

- This is an example of a **replay** attack
- How can we prevent a replay?

Simple Authentication



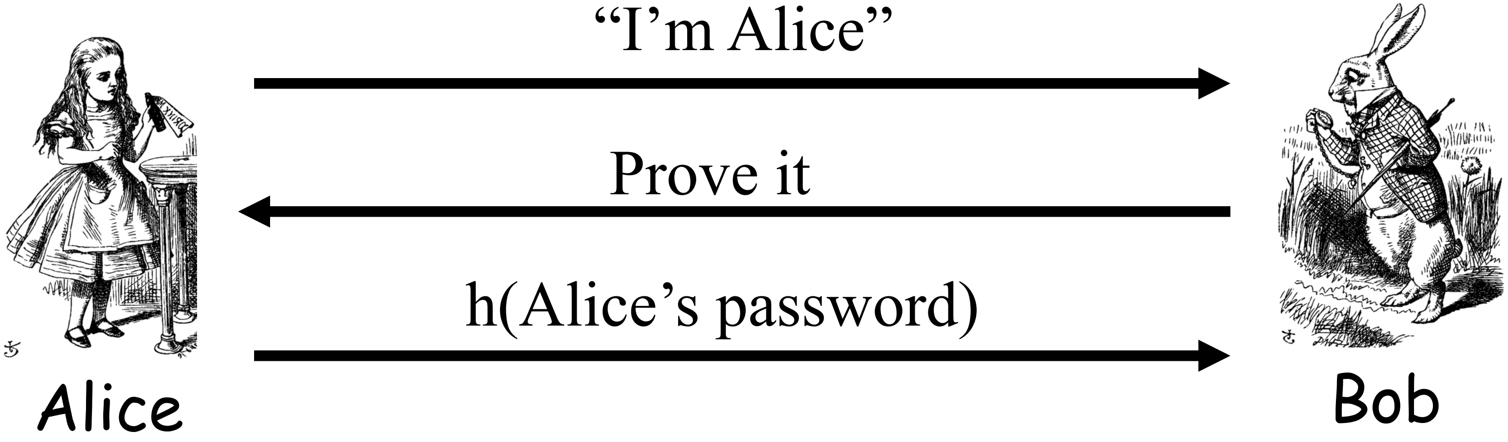
I’m Alice, my password is “frank”

Alice

Bob

- More efficient, but…
- … same problem as previous version

Better Authentication



Alice

"I'm Alice"

Prove it

h(Alice's password)

Bob

- Better since it hides Alice's password
  - From both Bob and Trudy
- But still subject to replay

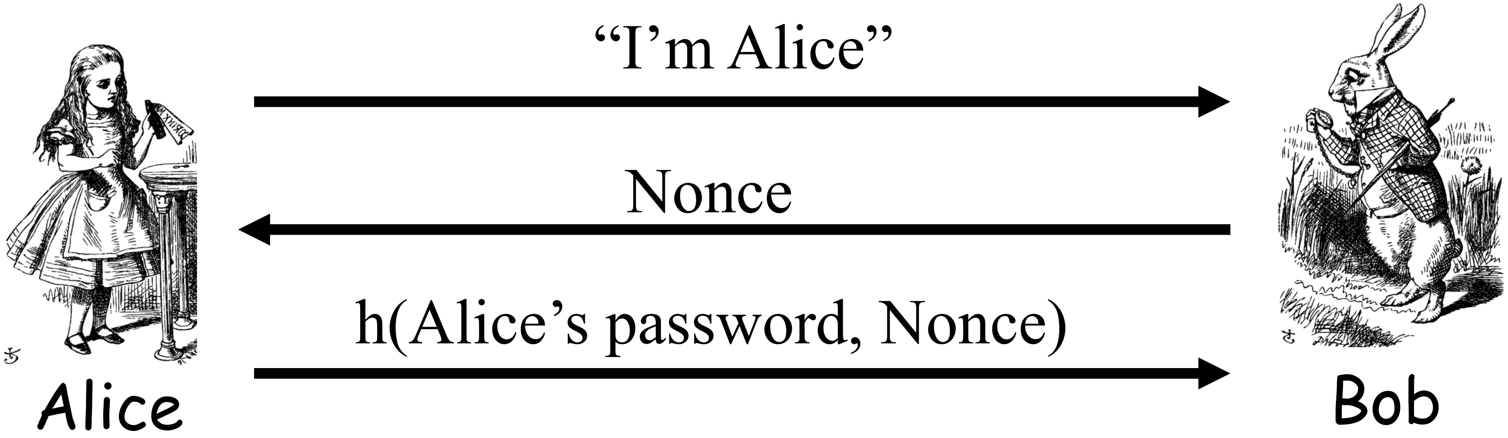
Challenge-Response

- To prevent replay, use ***challenge-response***
  - Goal is to ensure "freshness"
- Suppose Bob wants to authenticate Alice
  - ***Challenge*** sent from Bob to Alice
- Challenge is chosen so that…
  - Replay is not possible
  - Only Alice can provide the correct ***response***
  - Bob can verify the response

Nonce

- To ensure freshness, can employ a **nonce**
  - Nonce == **n**umber used **once**
- What to use for nonces?
  - That is, what is the challenge?
- What should Alice do with the nonce?
  - That is, how to compute the response?
- How can Bob verify the response?
- Should we rely on passwords or keys?

Challenge-Response



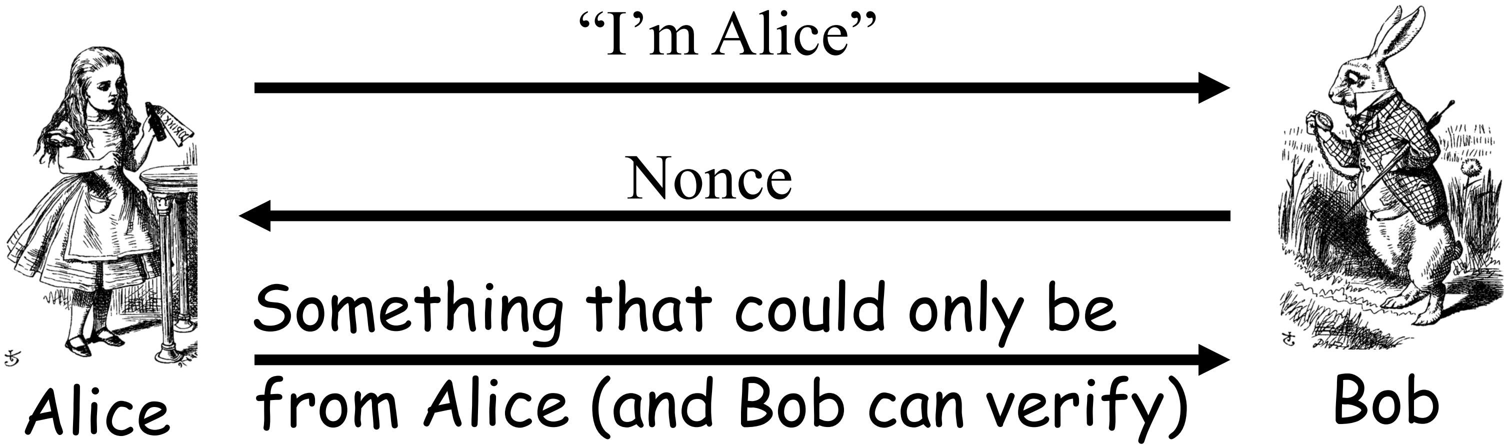
"I'm Alice"

Nonce

h(Alice's password, Nonce)

- Nonce is the **challenge**
- The hash is the **response**
- Nonce prevents replay, ensures freshness
- Password is something Alice knows
- Note: Bob must know Alice's pwd to verify

Generic Challenge-Response



- In practice, how to achieve this?
- Hashed password works, but…
- Encryption is better here (Why?)

Symmetric Key Notation

- Encrypt plaintext P with key K

  $$C = E(P,K)$$

- Decrypt ciphertext C with key K

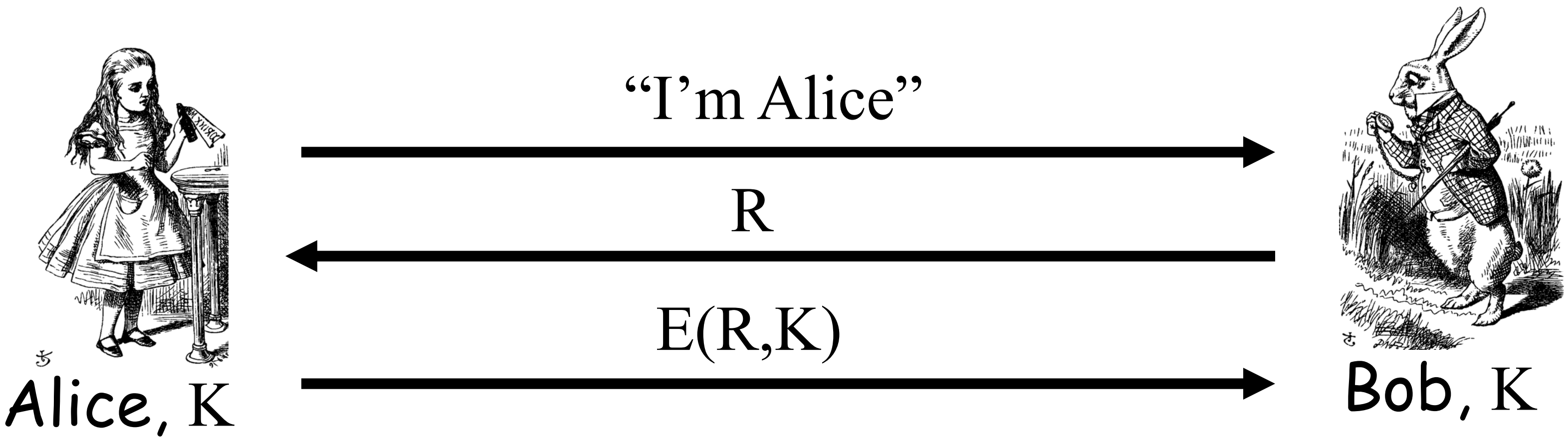
  $$P = D(C,K)$$

- Here, we are concerned with attacks on protocols, **not** attacks on crypto
  - So, we assume crypto algorithms are secure

Authentication: Symmetric Key

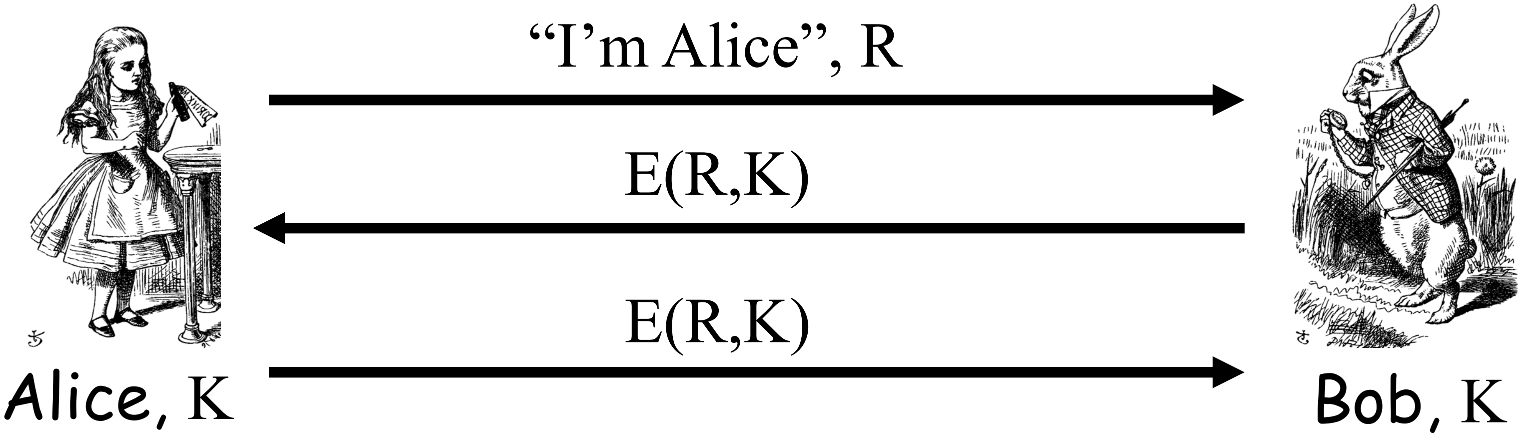
- Alice and Bob share symmetric key K
- Key K known only to Alice and Bob
- Authenticate by proving knowledge of shared symmetric key
- How to accomplish this?
  - Cannot reveal key, must not allow replay (or other) attack, must be verifiable, …

Authentication with Symmetric Key



- Secure method for Bob to authenticate Alice
- Alice does not authenticate Bob
- So, can we achieve mutual authentication?

Mutual Authentication?

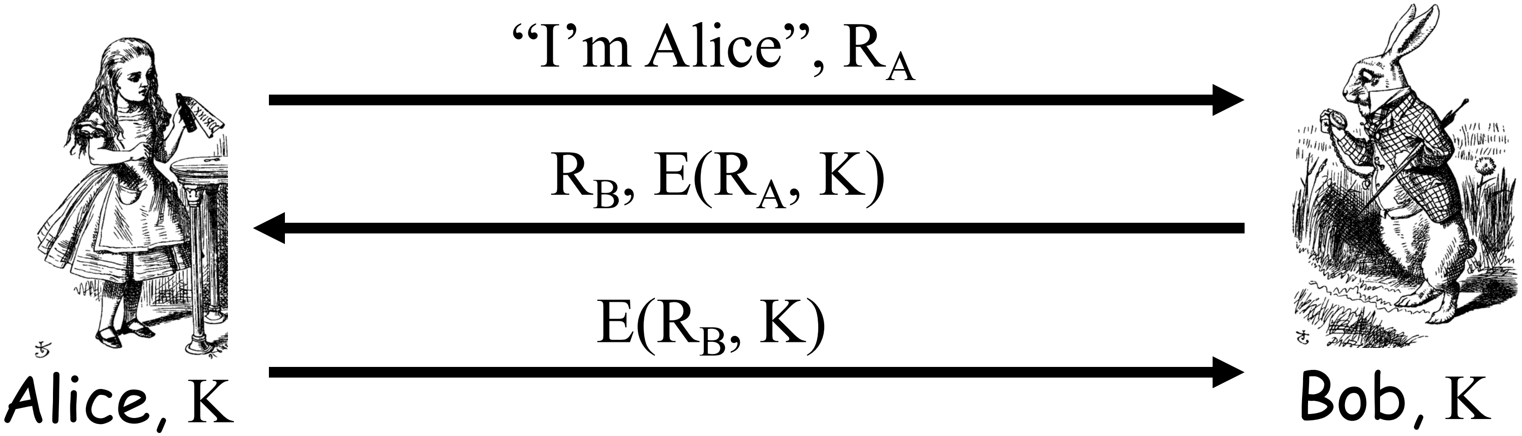


- What's wrong with this picture?
- "Alice" could be Trudy (or anybody else)!

Mutual Authentication

- Since we have a secure one-way authentication protocol…
- The obvious thing to do is to use the protocol twice
  - Once for Bob to authenticate Alice
  - Once for Alice to authenticate Bob
- This has got to work…

Mutual Authentication



“I’m Alice”, R_A

R_B, E(R_A, K)

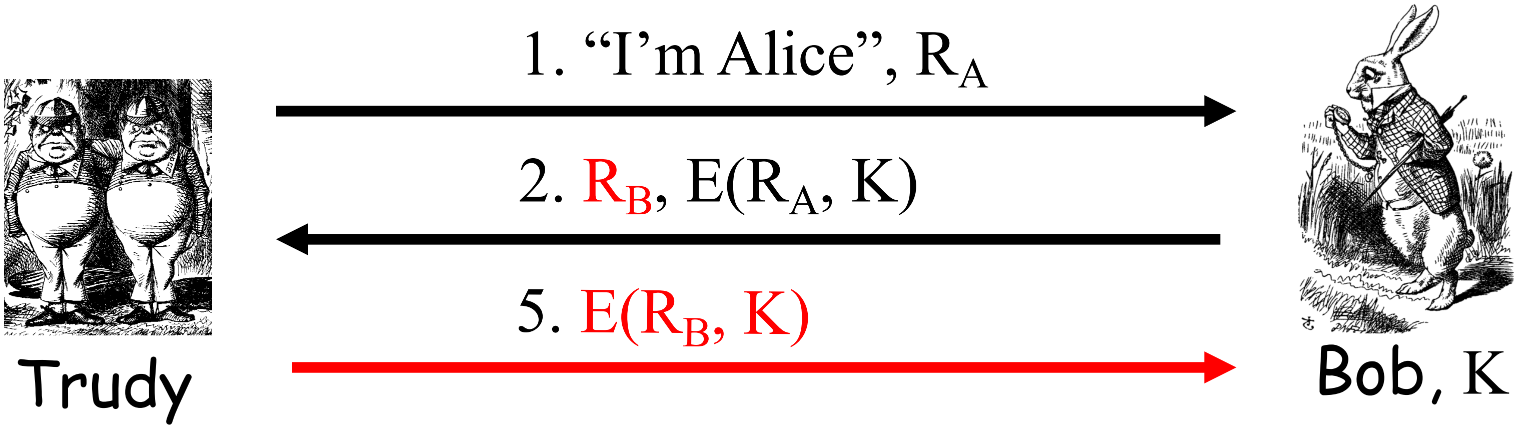
E(R_B, K)

Alice, K

Bob, K

- This provides mutual authentication…
- …or does it? See the next slide

Mutual Authentication Attack



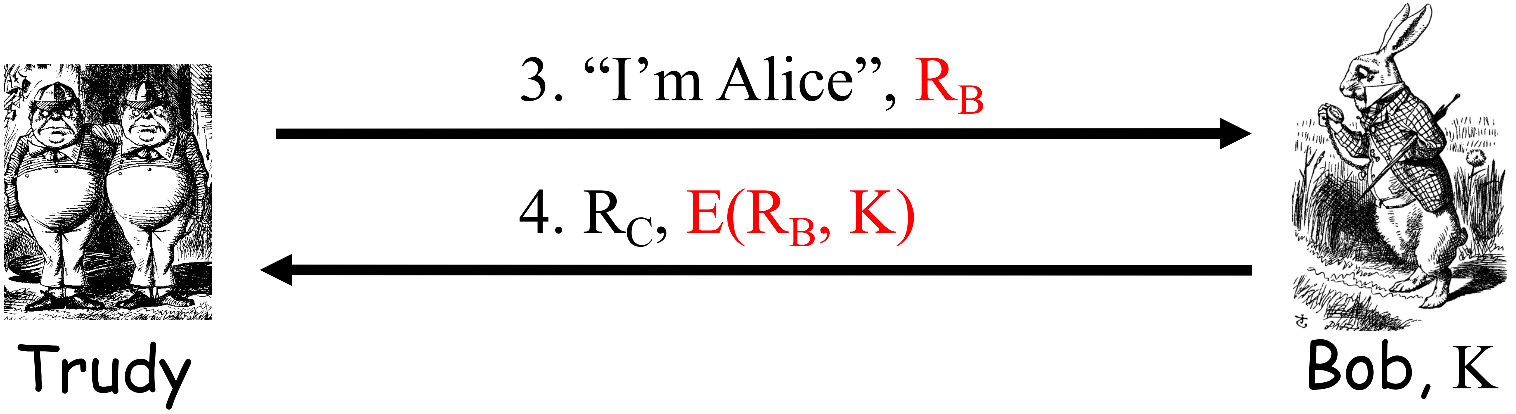
Trudy

1. “I’m Alice”, R_A

2. R_B, $E(R_A, K)$

5. $E(R_B, K)$

Bob, K



Trudy

3. “I’m Alice”, R_B

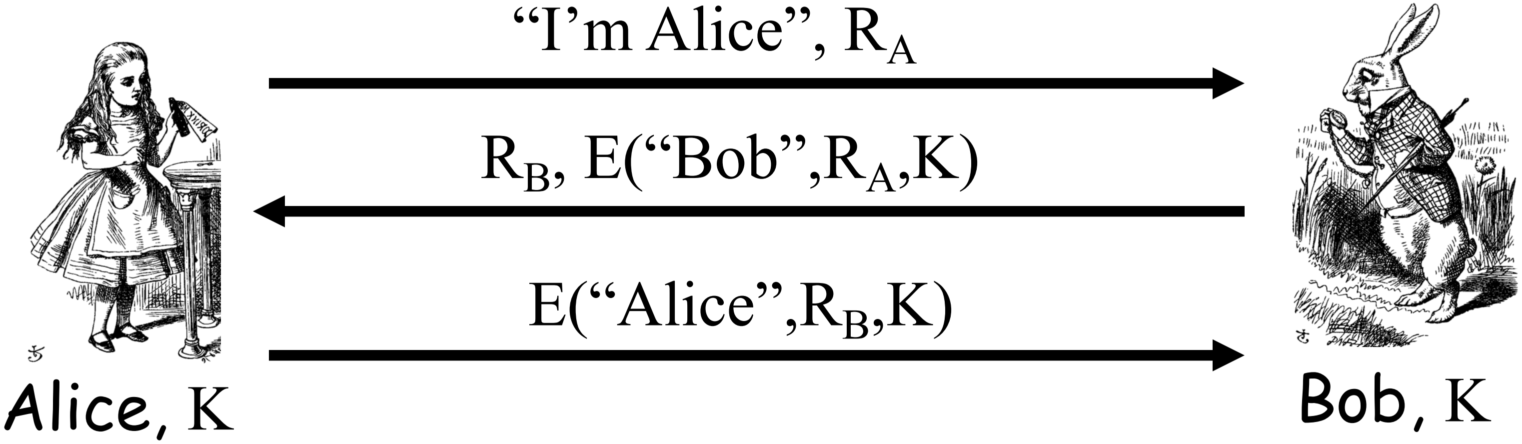
4. R_C, $E(R_B, K)$

Bob, K

Mutual Authentication

- Our one-way authentication protocol is **not** secure for mutual authentication
  - Protocols are subtle!
  - The "obvious" thing may not be secure
- Also, if assumptions or environment change, protocol may not be secure
  - This is a common source of security failure
  - For example, Internet protocols

Symmetric Key Mutual Authentication



“I’m Alice”, R_A

R_B, E(“Bob”,R_A,K)

E(“Alice”,R_B,K)

Alice, K

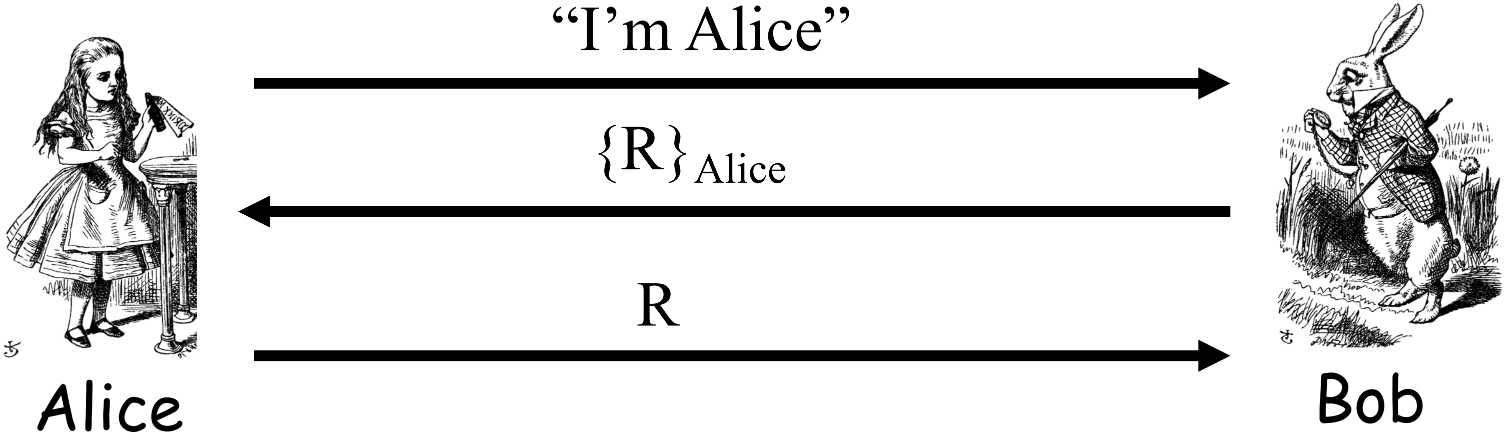
Bob, K

- Do these “insignificant” changes help?
- Yes!

Public Key Notation

- Encrypt M with Alice's public key: $\{M\}_{\text{Alice}}$
- Sign M with Alice's private key: $[M]_{\text{Alice}}$
- Then
  - $[\{M\}_{\text{Alice}}]_{\text{Alice}} = M$
  - $\{[M]_{\text{Alice}}\}_{\text{Alice}} = M$
- **Anybody** can use Alice's **public key**
- Only **Alice** can use her **private key**

Public Key Authentication



"I'm Alice"

$\{\text{R}\}_{\text{Alice}}$

R

Alice

Bob

- Is this secure?
- Trudy can get Alice to decrypt anything!
  - So, should have two key pairs

Public Key Authentication

"I'm Alice"

R

$[R]_{\text{Alice}}$

Alice

Bob

- Is this secure?
- Trudy can get Alice to sign anything!
  - Same a previous — should have two key pairs

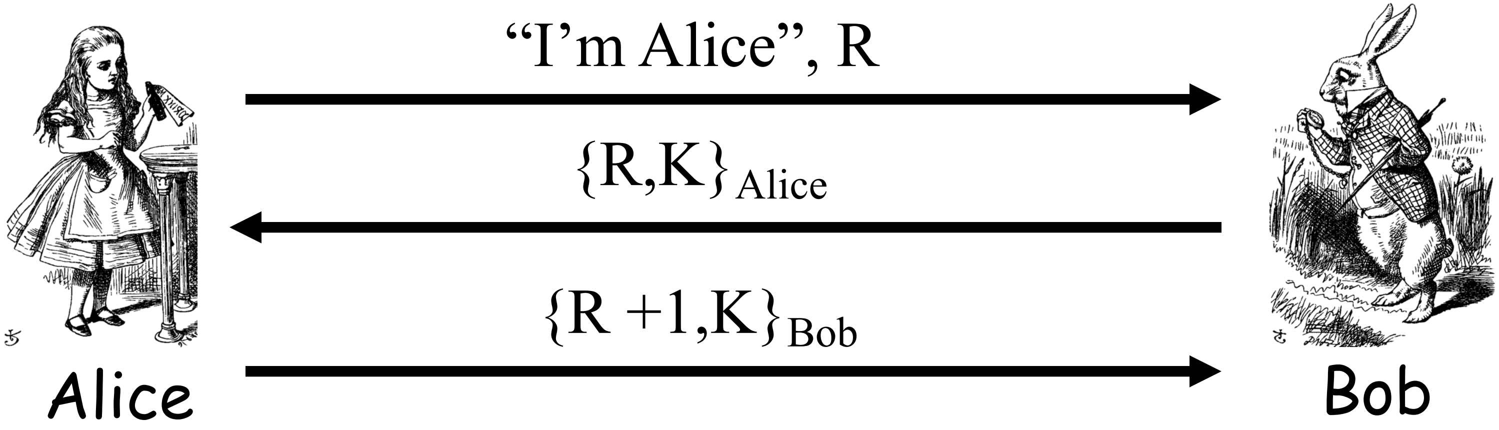
Public Keys

- Generally, a bad idea to use the same key pair for encryption and signing
- Instead, should have…
  - …one key pair for encryption/decryption…
  - …and a different key pair for signing/verifying signatures

Session Key

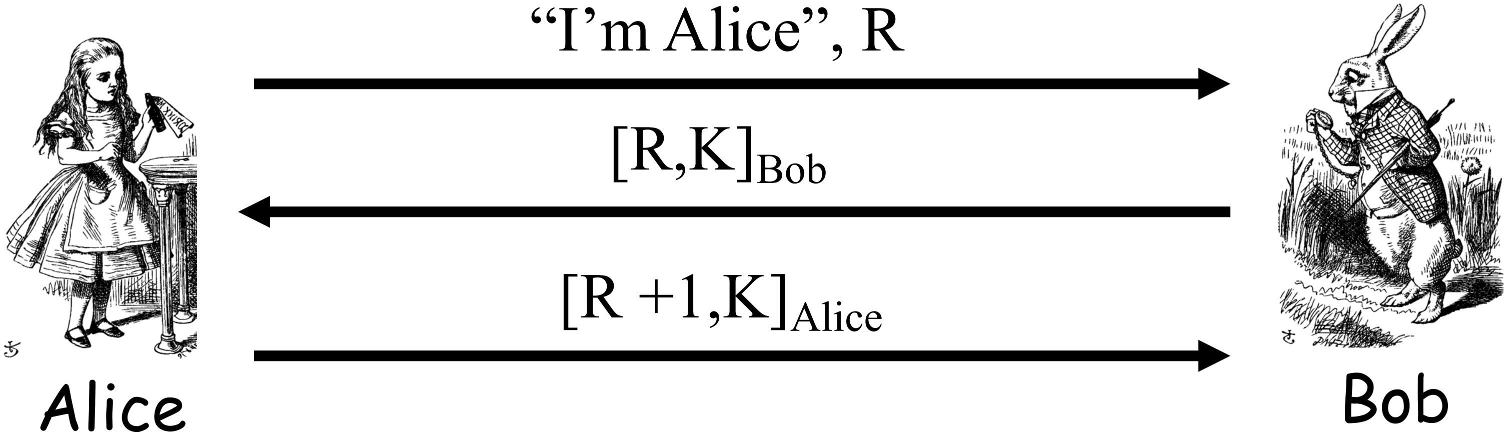
- Usually, a **session key** is required
  - I.e., a symmetric key for a particular session
  - Used for confidentiality and/or integrity
- How to authenticate and establish a session key (i.e., shared symmetric key)?
  - When authentication completed, want Alice and Bob to share a session key
  - Trudy cannot break the authentication…
  - …and Trudy cannot determine the session key

Authentication & Session Key



- Is this secure?
  - Alice is authenticated and session key is secure
  - Alice's "nonce", R, useless to authenticate Bob
  - The key K is acting as Bob's nonce to Alice
- No mutual authentication

Public Key Authentication and Session Key



Alice

"I'm Alice", R

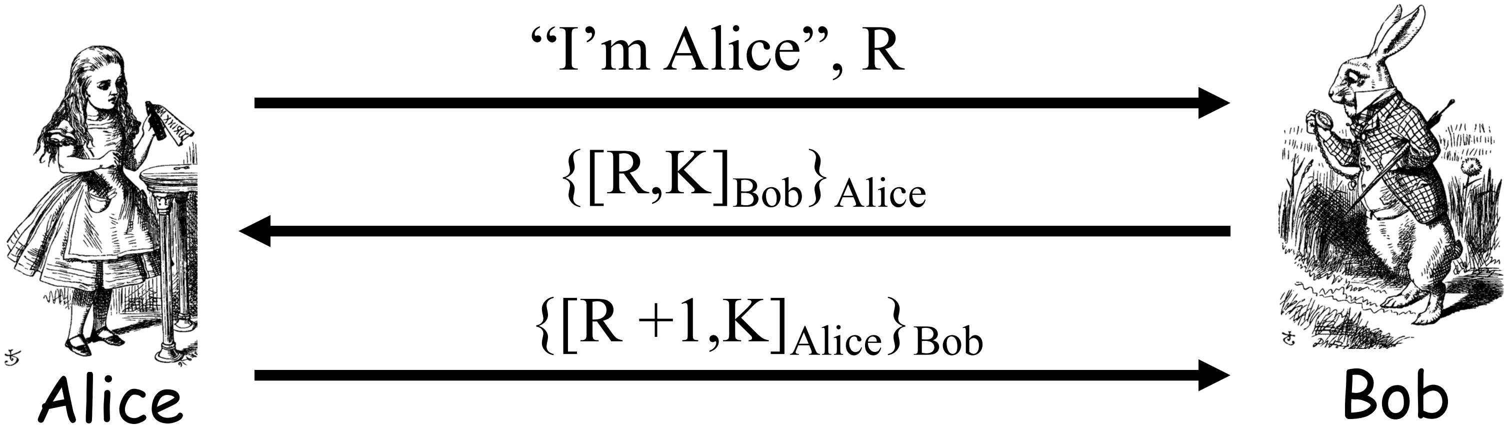
$[R,K]_{Bob}$

$[R+1,K]_{Alice}$

Bob

- Is this secure?
  - Mutual authentication (good), but…
  - … session key is not secret (very bad)

Public Key Authentication and Session Key



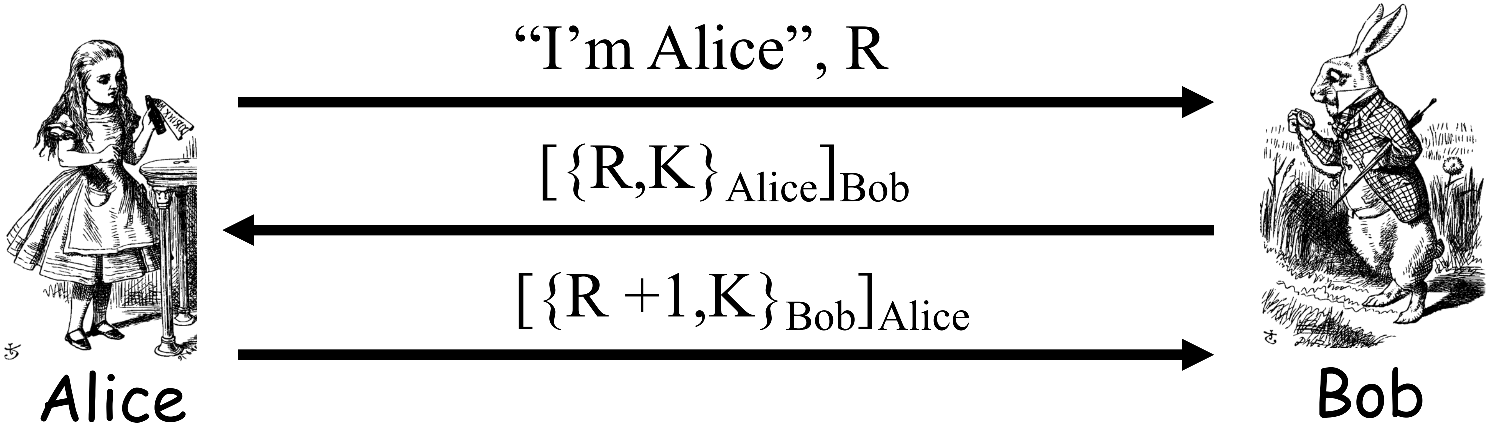
"I'm Alice", R

$\{[R,K]_{Bob}\}_{Alice}$

$\{[R+1,K]_{Alice}\}_{Bob}$

- Is this secure?
- Seems to be OK
- Mutual authentication and session key!

Public Key Authentication and Session Key



- Is this secure?
- Seems to be OK
  - Anyone can see $\{R,K\}_{Alice}$ and $\{R+1,K\}_{Bob}$

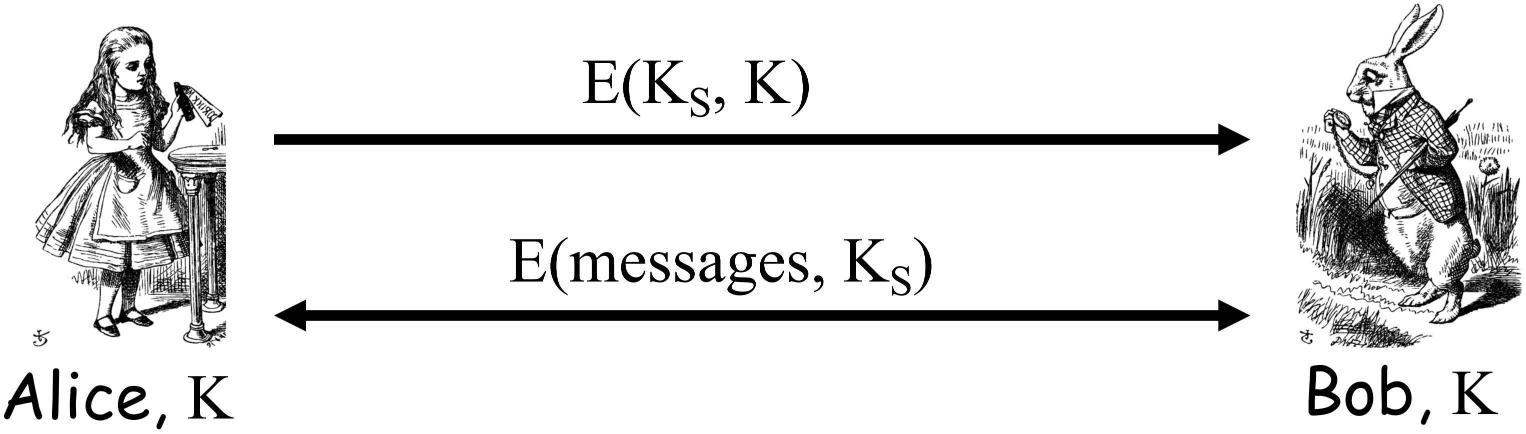
Perfect Forward Secrecy

- Consider this "issue"…
  - Alice encrypts message with shared key K and sends ciphertext to Bob
  - Trudy records ciphertext and later attacks Alice's (or Bob's) computer to recover K
  - Then Trudy decrypts recorded messages
- **Perfect forward secrecy (PFS):** Trudy cannot later decrypt recorded ciphertext
  - Even if Trudy gets key K or other secret(s)
- Is PFS possible?

Perfect Forward Secrecy

- Suppose Alice and Bob share key K
- For perfect forward secrecy, Alice and Bob cannot use K to encrypt
- Instead they must use a session key K_S and forget it after it's used
- Can Alice and Bob agree on session key K_S in a way that ensures PFS?

Naïve Session Key Protocol



E(K_S, K)

E(messages, K_S)

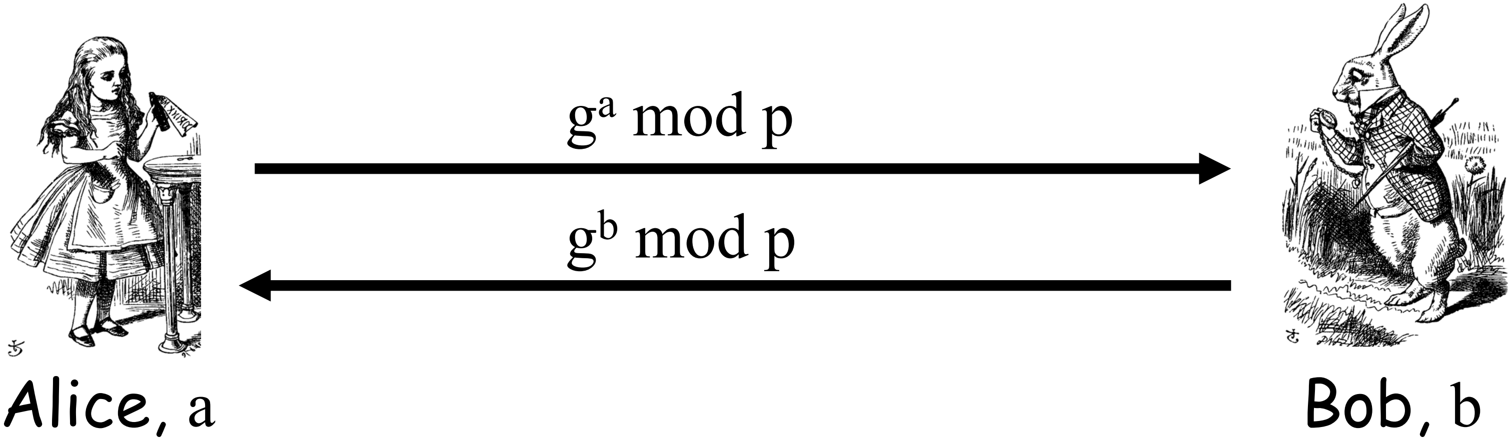
Alice, K

Bob, K

- Trudy could record E(K_S, K)
- If Trudy later gets K then she can get K_S
  - Then Trudy can decrypt recorded messages

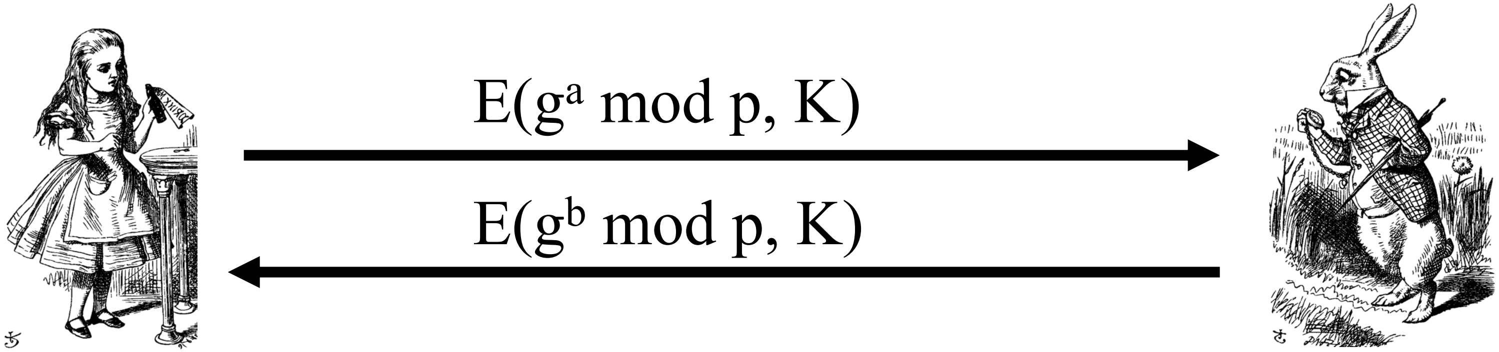
Perfect Forward Secrecy

- We use **Diffie-Hellman** for PFS
- Recall: public g and p



- But Diffie-Hellman is subject to MiM
- How to get PFS and prevent MiM?

Perfect Forward Secrecy



$E(g^a \bmod p, K)$

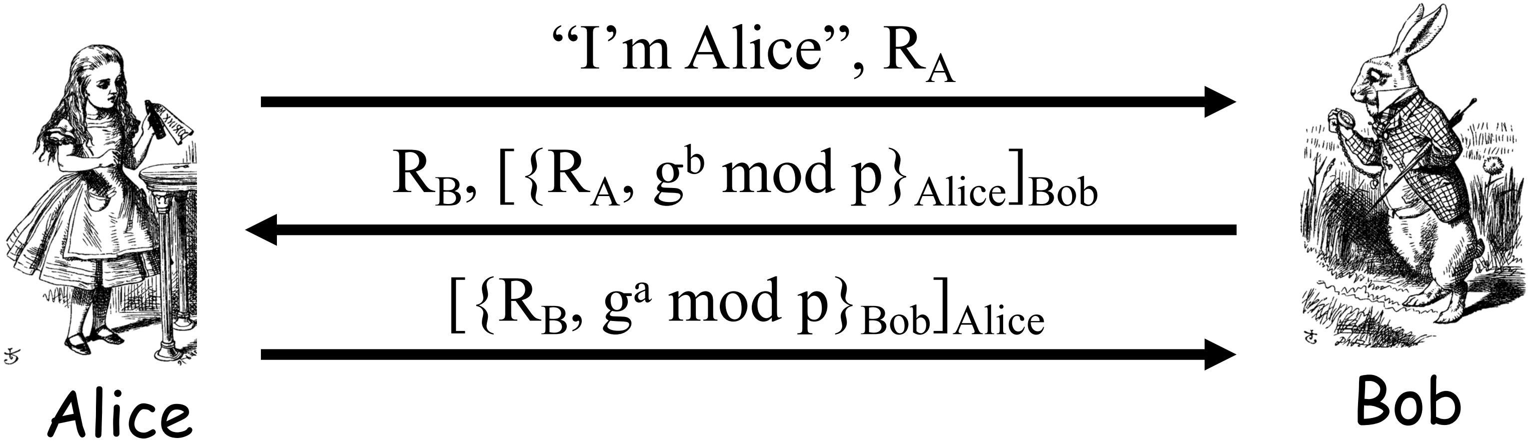
$E(g^b \bmod p, K)$

Alice: K, a

Bob: K, b

- Session key $K_S = g^{ab} \bmod p$
- Alice **forgets** a, Bob **forgets** b
- So-called **Ephemeral Diffie-Hellman**
- Neither Alice nor Bob can later recover K_S
- Are there other ways to achieve PFS?

Mutual Authentication, Session Key and PFS



“I’m Alice”, R_A

R_B, $[\{R_A, g^b \bmod p\}_{\text{Alice}}]_{\text{Bob}}$

$[\{R_B, g^a \bmod p\}_{\text{Bob}}]_{\text{Alice}}$

Alice

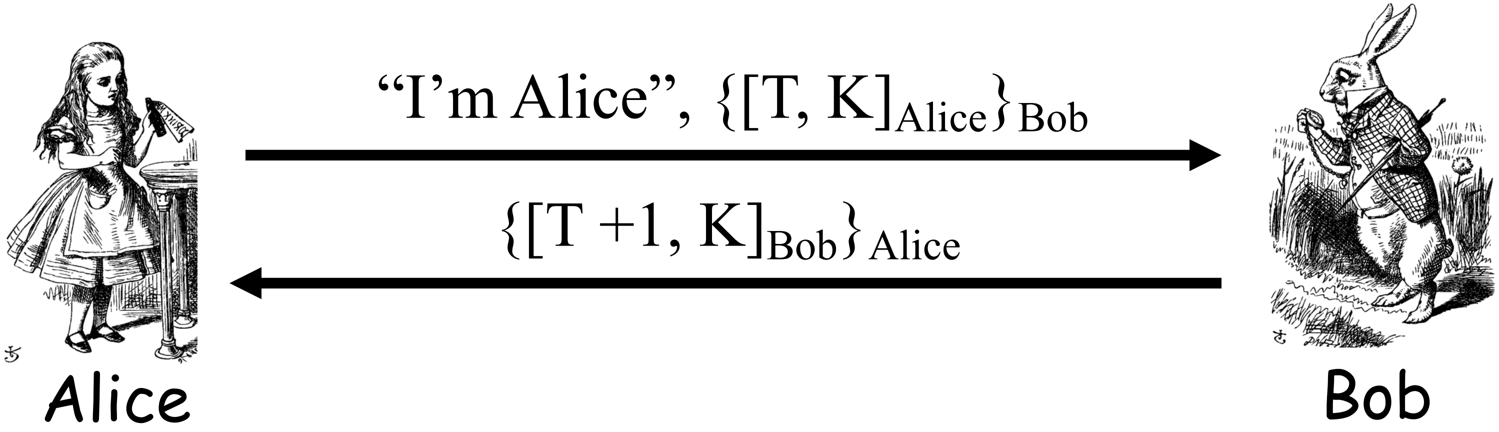
Bob

- Session key is $K = g^{ab} \bmod p$
- Alice forgets a and Bob forgets b
- If Trudy later gets Bob’s and Alice’s secrets, she cannot recover session key K

Timestamps

- A timestamp T is derived from current time
- Timestamps used in some security protocols
  - Kerberos, for example
- Timestamps reduce number of msgs (good)
  - Like a nonce that both sides know in advance
- "Time" is a security-critical parameter (bad)
- Clocks never exactly the same, so must allow for **clock skew** — creates risk of replay
  - How much clock skew is enough?

Public Key Authentication with Timestamp T

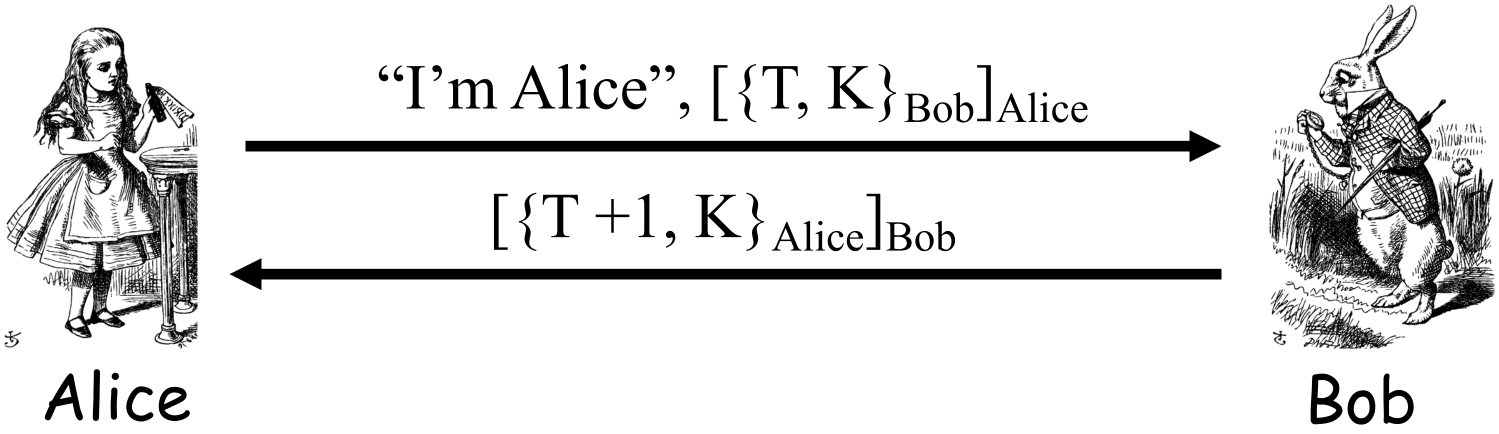


"I'm Alice", $\{[T, K]_{\text{Alice}}\}_{\text{Bob}}$

$\{[T+1, K]_{\text{Bob}}\}_{\text{Alice}}$

- Secure mutual authentication?
- Session key?
- Seems to be OK

Public Key Authentication with Timestamp T

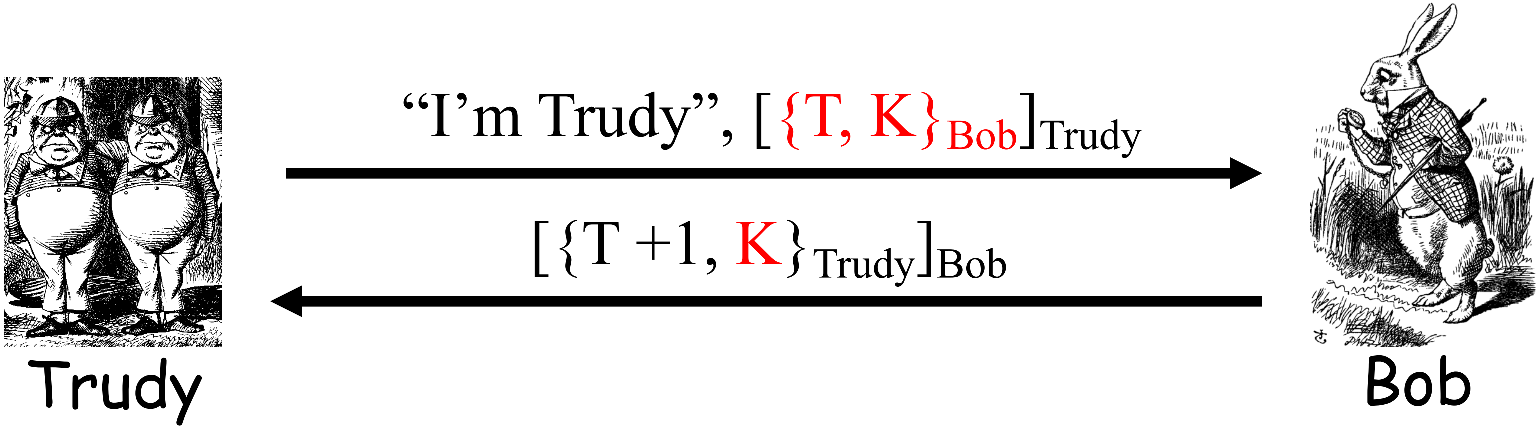


“I’m Alice”, $[\{T, K\}_{Bob}]_{Alice}$

$[\{T+1, K\}_{Alice}]_{Bob}$

- Secure authentication and session key?
- Trudy can use Alice’s public key to find $\{T, K\}_{Bob}$ and then…

Public Key Authentication with Timestamp T



“I’m Trudy”, $[\{T, K\}_{\text{Bob}}]_{\text{Trudy}}$

$[\{T+1, K\}_{\text{Trudy}}]_{\text{Bob}}$

Trudy

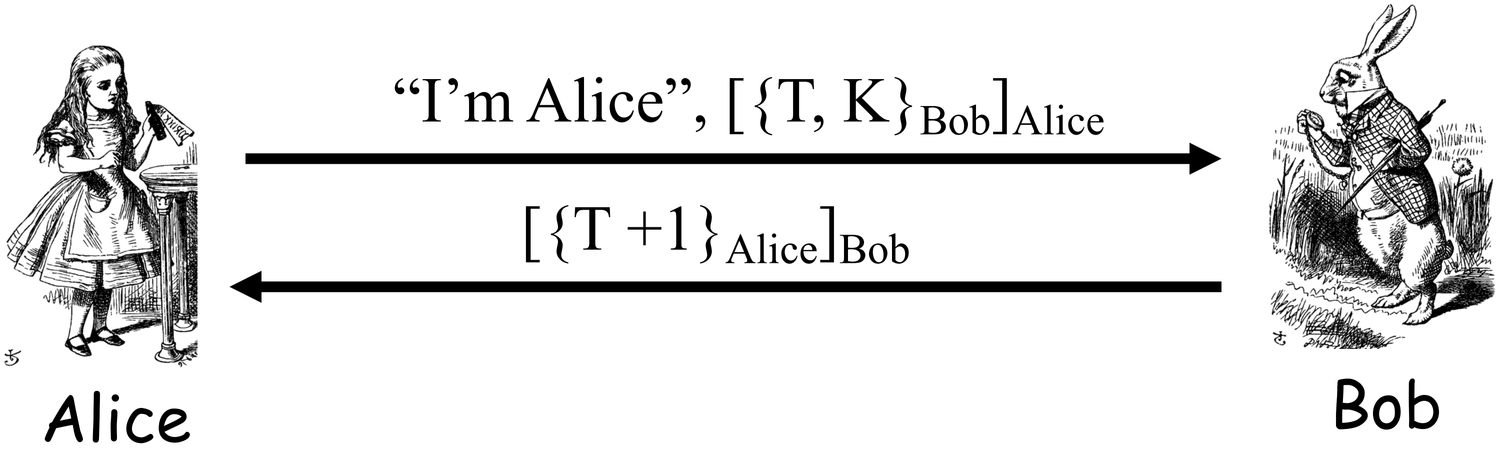
Bob

- Trudy obtains Alice-Bob session key K
- **Note:** Trudy must act within clock skew

Public Key Authentication

- Sign and encrypt with nonce…
  - Secure
- Encrypt and sign with nonce…
  - Secure
- Sign and encrypt with timestamp…
  - Secure
- Encrypt and sign with timestamp…
  - Insecure
- Protocols can be subtle!

Public Key Authentication with Timestamp T



“I’m Alice”, $[\{T, K\}_{Bob}]_{Alice}$

$[\{T+1\}_{Alice}]_{Bob}$

Alice

Bob

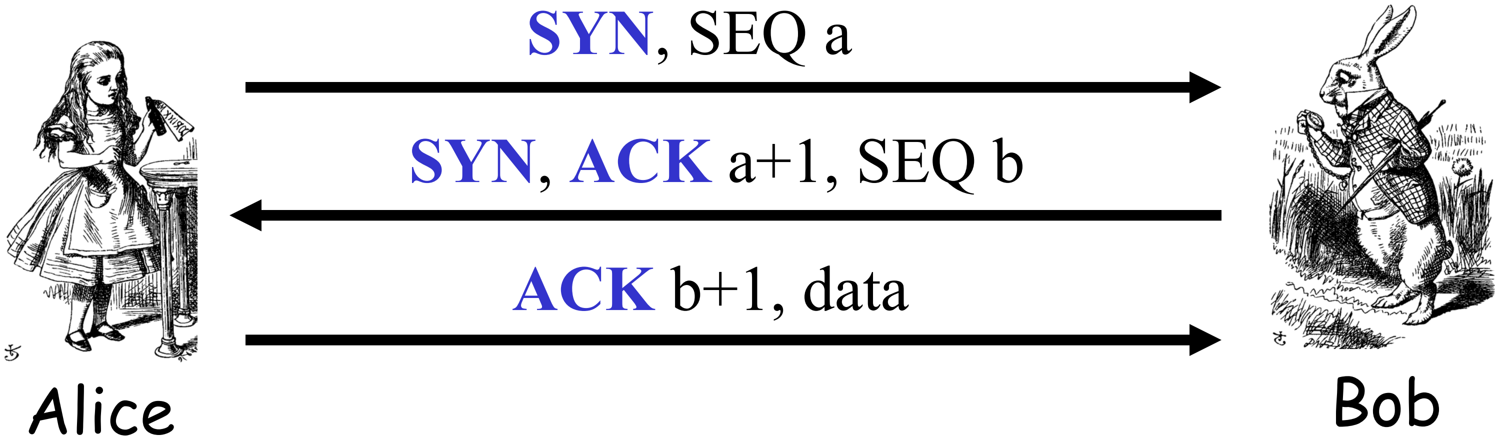
- Is this “encrypt and sign” secure?
  - Yes, seems to be OK
- Does “sign and encrypt” also work here?

Authentication and TCP

TCP-based Authentication

- TCP not intended for use as an authentication protocol
- But IP address in TCP connection often used for authentication
- One mode of IPSec relies on IP address for authentication

TCP 3-way Handshake



SYN, SEQ a

SYN, **ACK** a+1, SEQ b

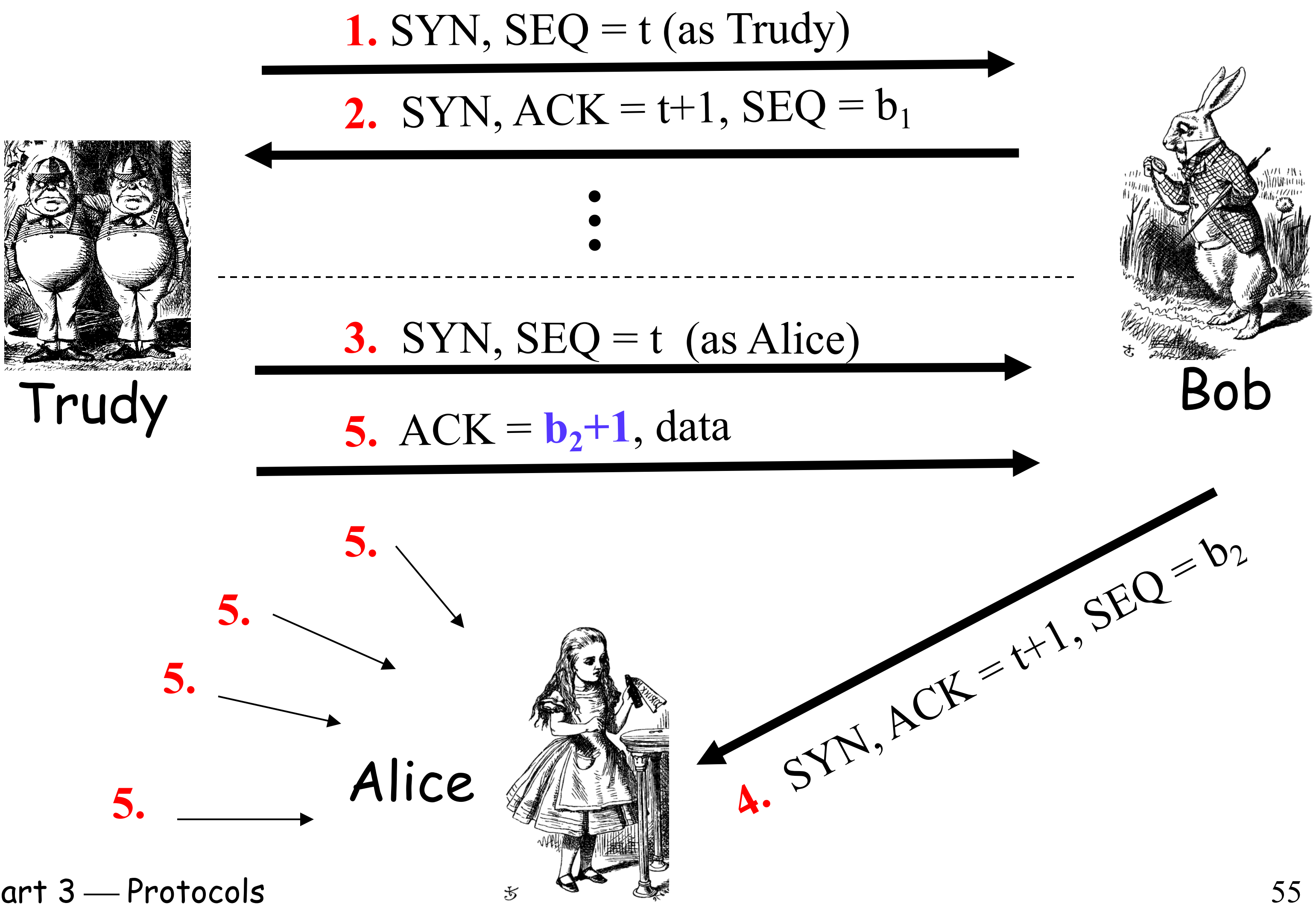
ACK b+1, data

Alice

Bob

- Recall the TCP three way handshake
- Initial sequence numbers: SEQ a and SEQ b
    - Supposed to be selected at random
- If not…

TCP Authentication Attack



1. SYN, SEQ = t (as Trudy)

2. SYN, ACK = t+1, SEQ = b_1

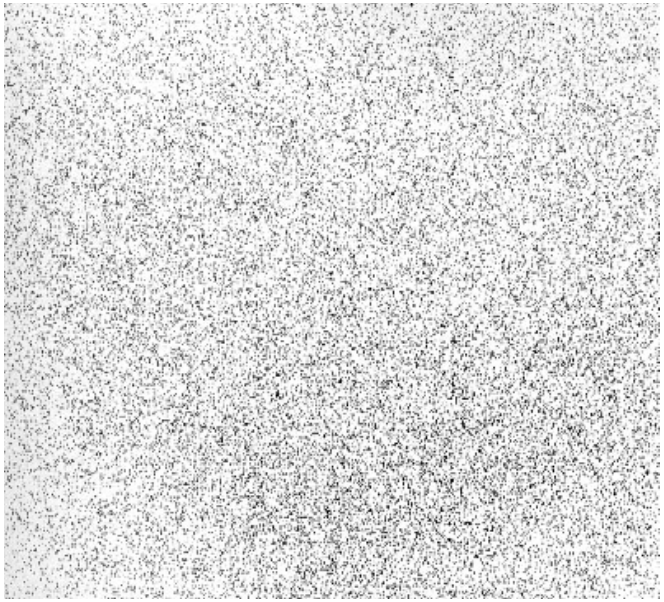
⋮

3. SYN, SEQ = t (as Alice)

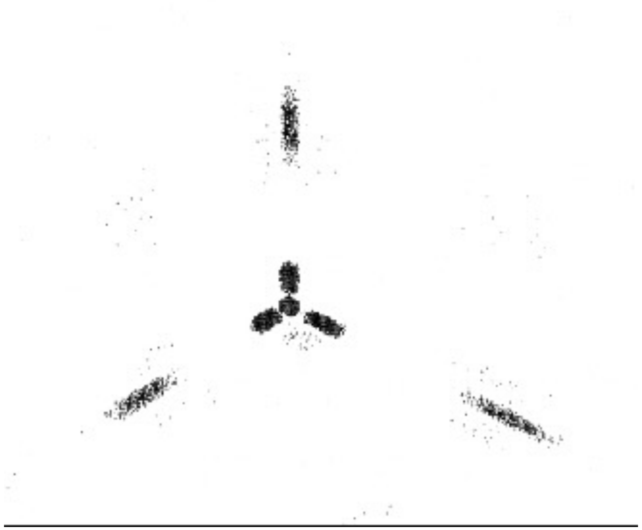
5. ACK = b_2+1, data

4. SYN, ACK = t+1, SEQ = b_2

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 packets (or else connection will terminate)
- If **password** (or other authentication) required, this attack fails
- If TCP connection is relied on for authentication, then attack can succeed
- **Bad idea** to rely on TCP for authentication

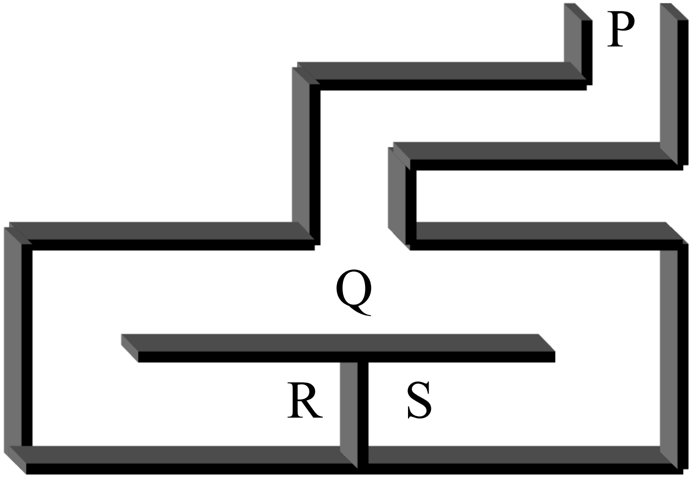
Zero Knowledge Proofs

Zero Knowledge Proof (ZKP)

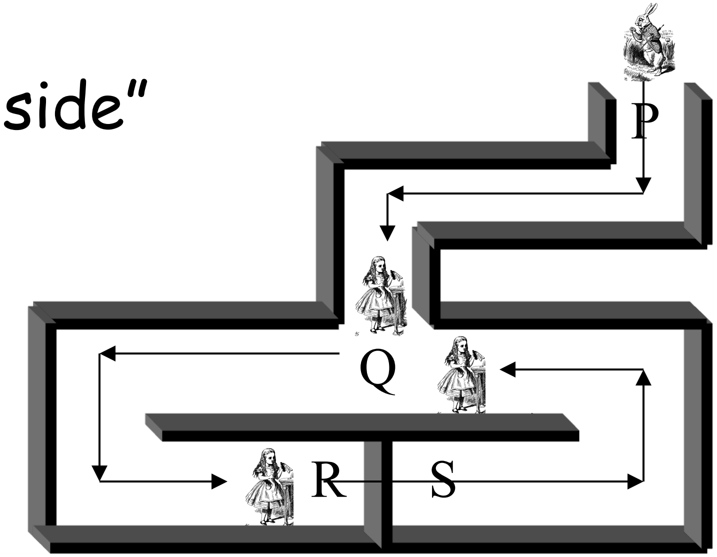
- Alice wants to prove that she knows a secret without revealing **any** info about it
- Bob must verify that Alice knows secret
  - But, Bob gains no info about the secret
- Process is probabilistic
  - Bob can verify that Alice knows the secret to an arbitrarily high probability
- An "interactive proof system"

Bob's Cave

- Alice knows secret phrase to open path between R and S ("open sarsaparilla")
- Can she convince Bob that she knows the secret without revealing phrase?



Bob's Cave

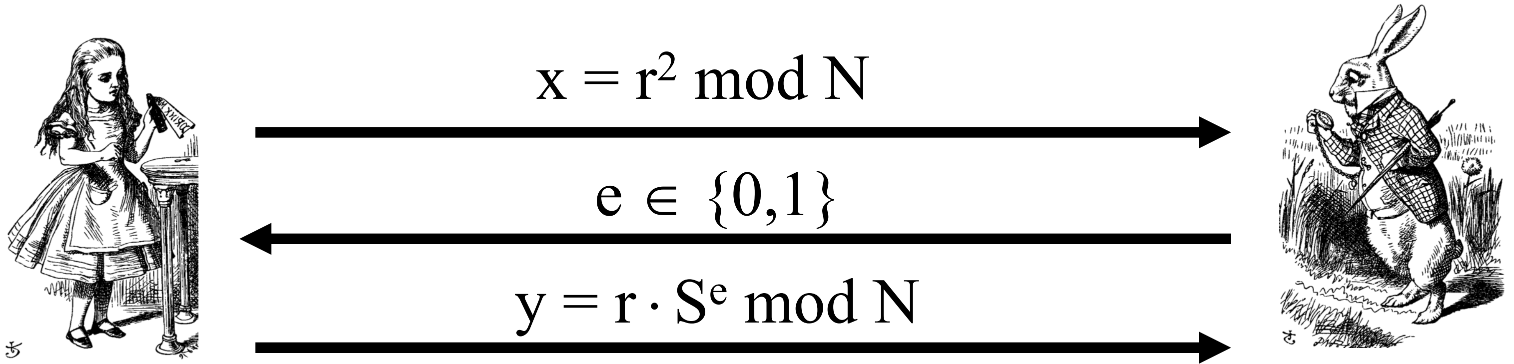


- Bob: "Alice come out on S side"
- Alice (quietly): "Open sarsaparilla"
- If Alice does not know the secret…
- …then Alice could come out from the correct side with probability 1/2
- If Bob repeats this n times, then Alice (who does not know secret) can only fool Bob with probability $1/2^n$

Fiat-Shamir Protocol

- Cave-based protocols are inconvenient
  - Can we achieve same effect without the cave?
- Finding square roots modulo N is difficult
  - Equivalent to factoring
- Suppose $N = pq$, where p and q prime
- Alice has a secret S
- N and $v = S^2 \bmod N$ are **public**, S is **secret**
- Alice must convince Bob that she knows S without revealing any information about S

Fiat-Shamir



$x = r^2 \bmod N$

$e \in \{0,1\}$

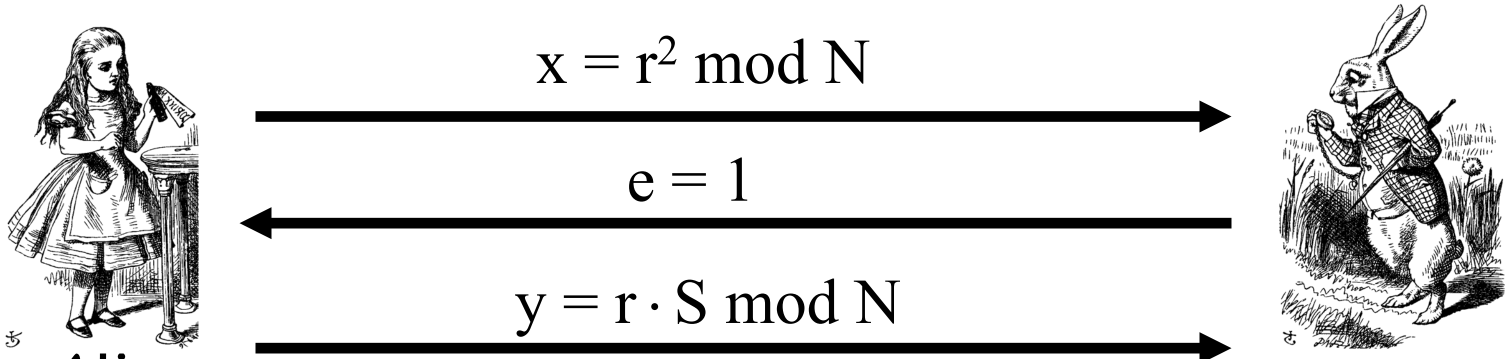
$y = r \cdot S^e \bmod N$

Alice
secret S
random r

Bob
random e

- **Public:** Modulus N and $v = S^2 \bmod N$
- Alice selects random r, Bob chooses $e \in \{0,1\}$
- Bob verifies: $\mathbf{y^2 = x \cdot v^e \bmod N}$
  - Why? Because… $y^2 = r^2 \cdot S^{2e} = r^2 \cdot (S^2)^e = x \cdot v^e \bmod N$

Fiat-Shamir: $e = 1$



$x = r^2 \bmod N$

$e = 1$

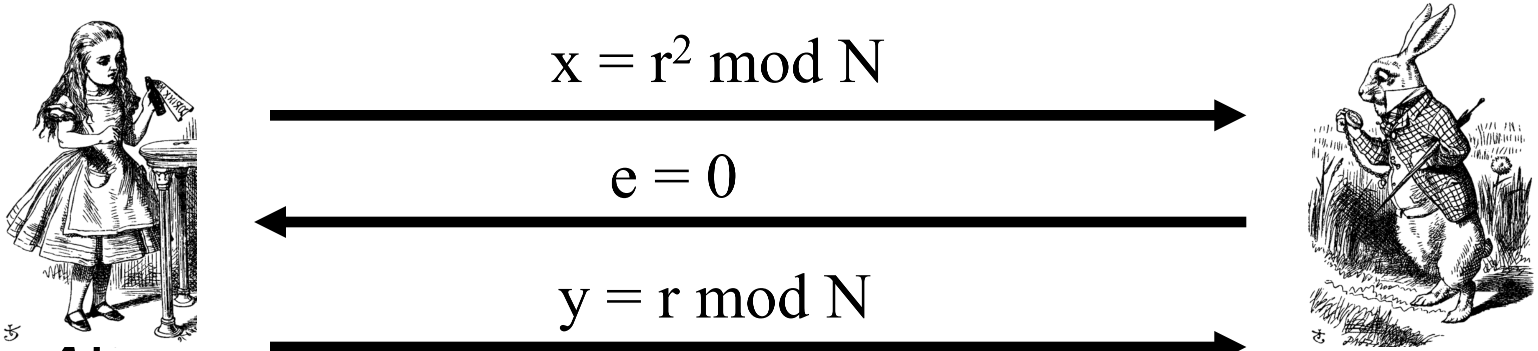
$y = r \cdot S \bmod N$

Alice
secret S
random r

Bob
random e

- **Public:** Modulus N and $v = S^2 \bmod N$
- Alice selects random r, Bob chooses $e = 1$
- If $y^2 = x \cdot v \bmod N$ then Bob accepts it
  - I.e., "Alice" passes this iteration of the protocol
- Note that Alice must know S in this case

Fiat-Shamir: e = 0



$x = r^2 \bmod N$

$e = 0$

$y = r \bmod N$

Alice
secret S
random r

Bob
random e

- **Public:** Modulus N and $v = S^2 \bmod N$
- Alice selects random r, Bob chooses $e = 0$
- Bob must checks whether $y^2 = x \bmod N$
- Alice does **not** need to know S in this case!

Fiat-Shamir

- **Public:** modulus N and $v = S^2 \bmod N$
- **Secret:** Alice knows S
- Alice selects random r and **commits** to r by sending $x = r^2 \bmod N$ to Bob
- Bob sends **challenge** $e \in \{0,1\}$ to Alice
- Alice **responds** with $y = r \cdot S^e \bmod N$
- Bob checks whether $y^2 = x \cdot v^e \bmod N$
  - Does this prove response is from Alice?

Does Fiat-Shamir Work?

- If everyone follows protocol, math works:
  - Public: $v = S^2 \bmod N$
  - Alice to Bob: $x = r^2 \bmod N$ and $y = r \cdot S^e \bmod N$
  - Bob verifies: $y^2 = x \cdot v^e \bmod N$
- Can Trudy convince Bob she is Alice?
  - If Trudy expects $e = 0$, she sends $x = r^2$ in msg 1 and $y = r$ in msg 3 (i.e., follow the protocol)
  - If Trudy expects $e = 1$, sends $x = r^2 \cdot v^{-1}$ in msg 1 and $y = r$ in msg 3
- If Bob chooses $e \in \{0,1\}$ at random, Trudy can only trick Bob with probability $1/2$

Fiat-Shamir Facts

- Trudy can trick Bob with probability 1/2, but…
  - …after n iterations, the probability that Trudy can convince Bob that she is Alice is only $1/2^n$
  - Just like Bob's cave!
- Bob's $e \in \{0,1\}$ must be unpredictable
- Alice must use new r each iteration, or else…
  - If $e = 0$, Alice sends $r \bmod N$ in message 3
  - If $e = 1$, Alice sends $r \cdot S \bmod N$ in message 3
  - Anyone can find S given $r \bmod N$ and $r \cdot S \bmod N$

Fiat-Shamir Zero Knowledge?

- Zero knowledge means that nobody learns *anything* about the secret S
    - **Public:** $v = S^2 \bmod N$
    - Trudy sees $r^2 \bmod N$ in message 1
    - Trudy sees $r \cdot S \bmod N$ in message 3 (if $e = 1$)
- If Trudy can find r from $r^2 \bmod N$, gets S
    - But that requires modular square root
    - If Trudy could find modular square roots, she could get S from **public** v
- Protocol does not seem to "help" to find S

ZKP in the Real World

- Public key certificates identify users
    - No anonymity if certificates sent in plaintext
- ZKP offers a way to authenticate without revealing identities
- ZKP supported in MS's Next Generation Secure Computing Base (NGSCB), where…
    - …ZKP used to authenticate software "without revealing machine identifying data"
- ZKP is **not** just pointless mathematics!

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

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 — 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)

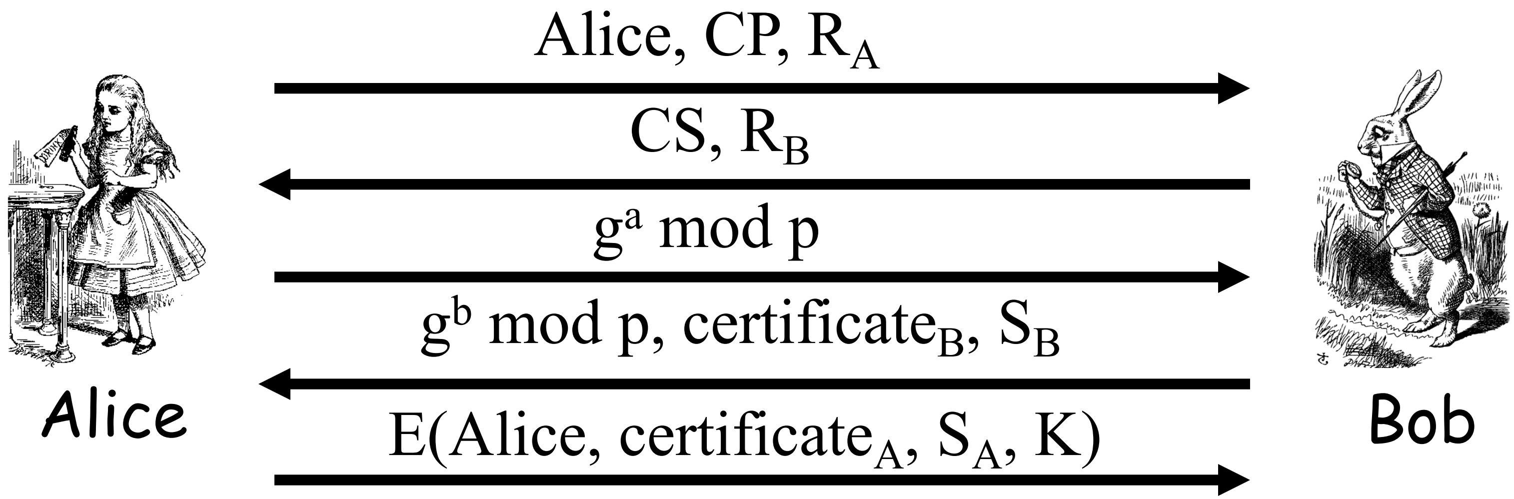
SSH

- 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



Alice, CP, R_A

CS, R_B

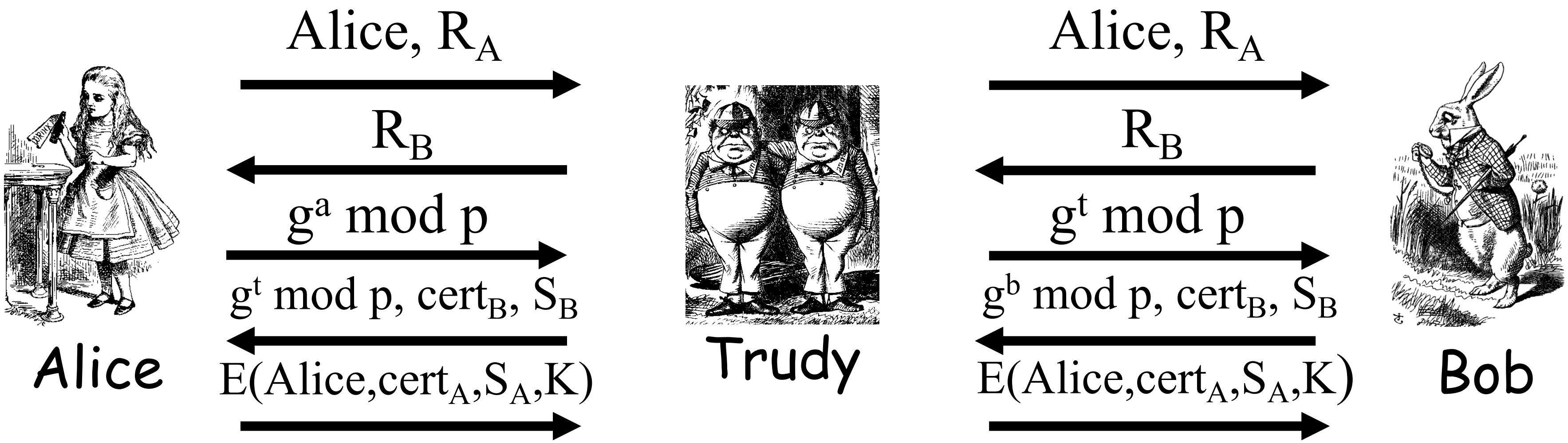
$g^a \bmod p$

$g^b \bmod p$, certificate$_B$, S_B

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

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

MiM Attack on SSH?


Alice, R_A

R_B

g^a 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)

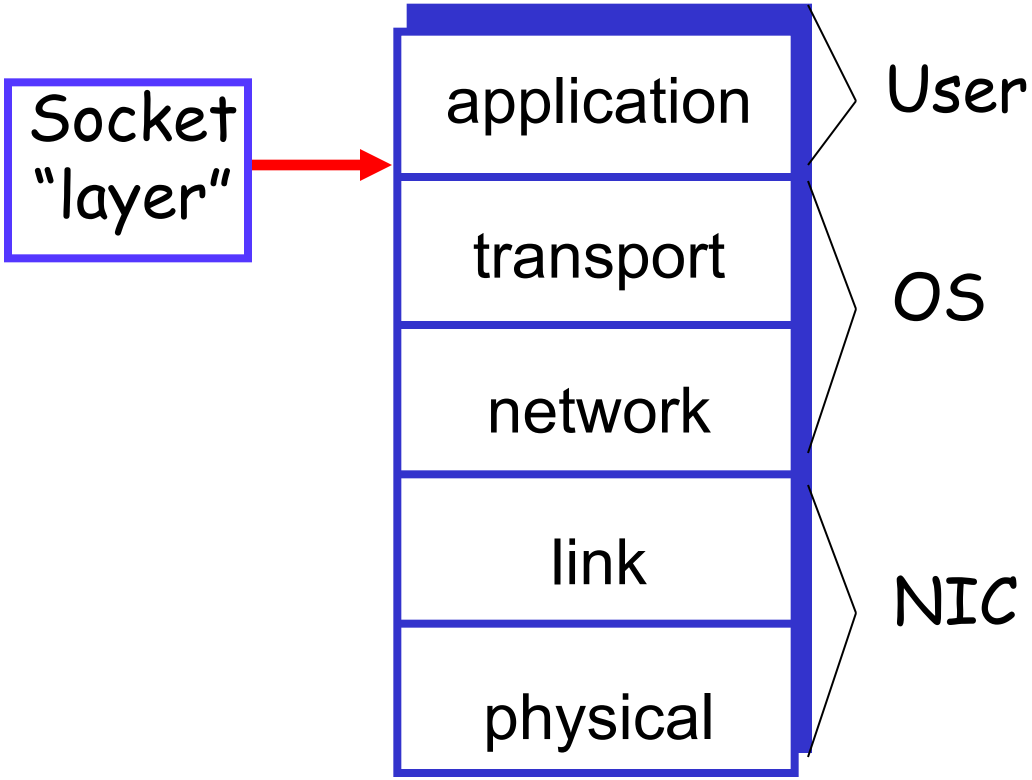
Bob

- 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

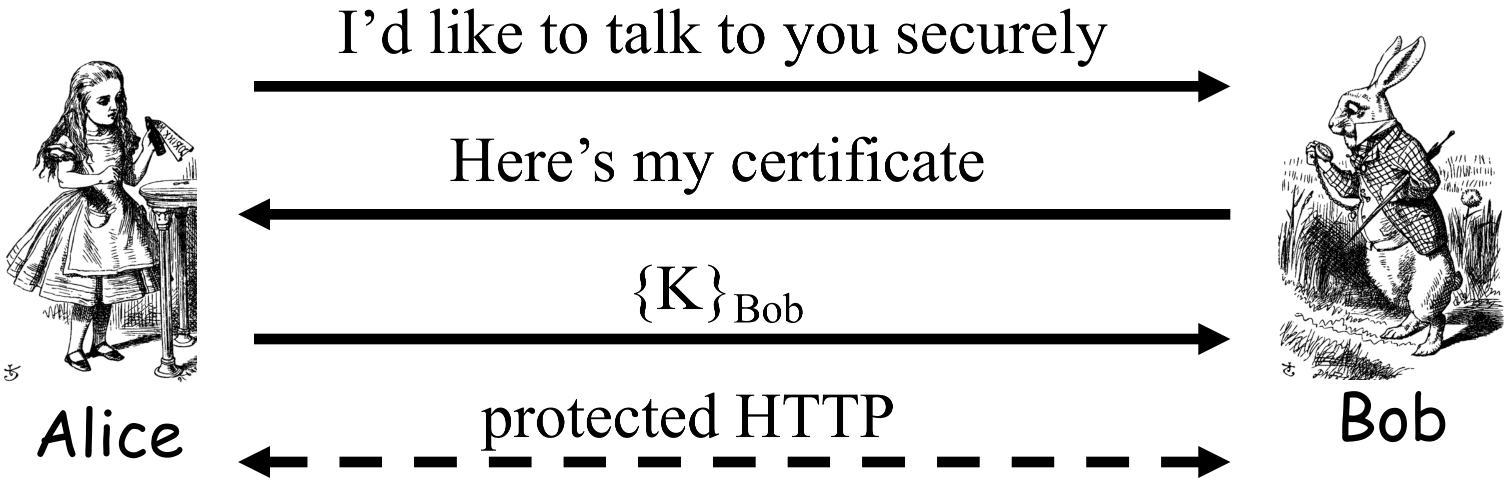
- "Socket layer" lives between application and transport layers
- SSL usually between HTTP and TCP



What is SSL?

- SSL is the protocol used for majority of secure 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



I'd like to talk to you securely

Here's my certificate

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

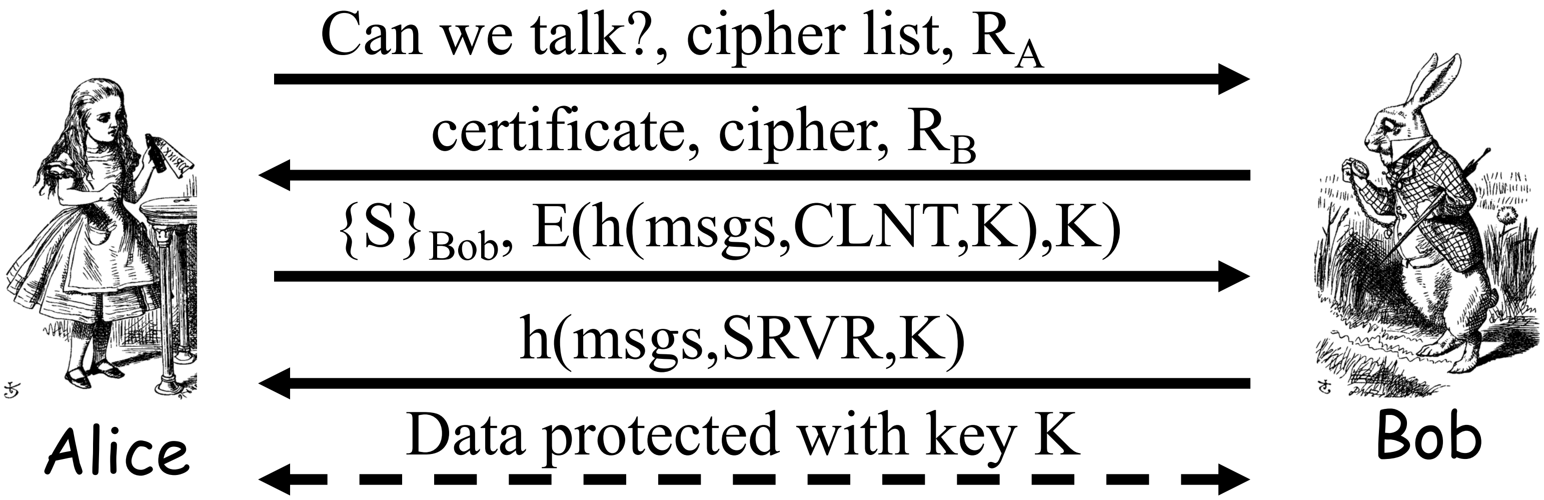
protected HTTP

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

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

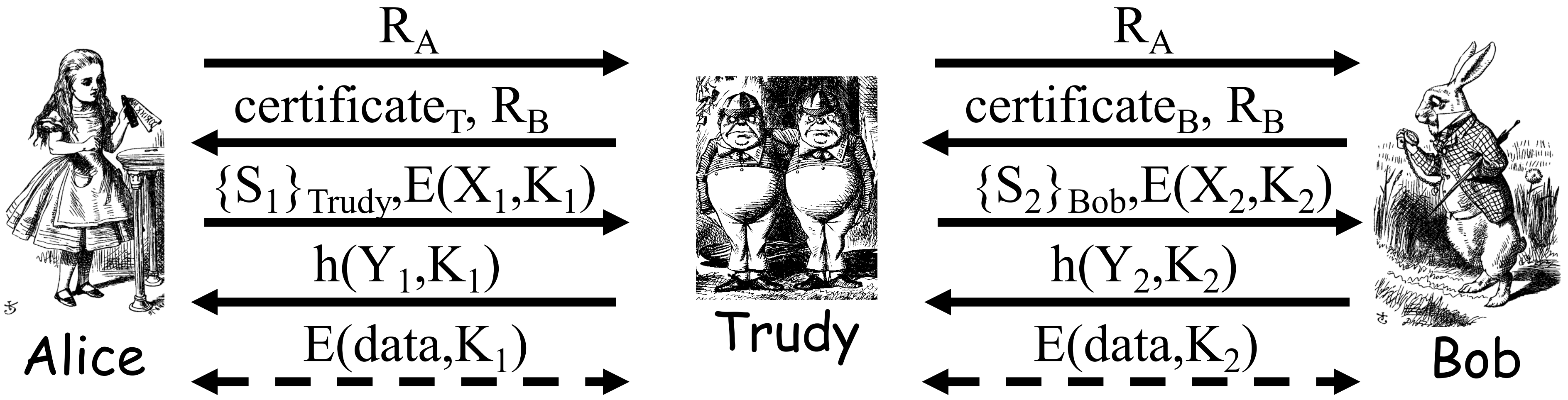
SSL Keys

- 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 $h(\text{msgs},\text{CLNT},K)$ 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?



Alice

R_A

certificate$_T$, R_B

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

$h(Y_1,K_1)$

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

Trudy

R_A

certificate$_B$, R_B

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

$h(Y_2,K_2)$

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

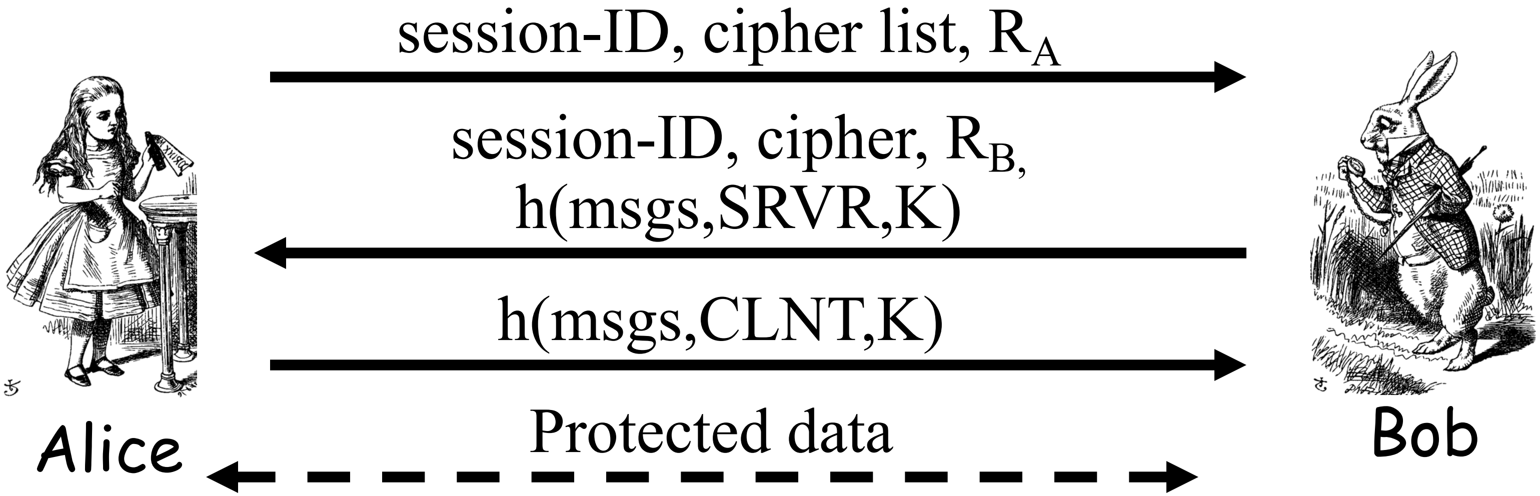
Bob

- **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)

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

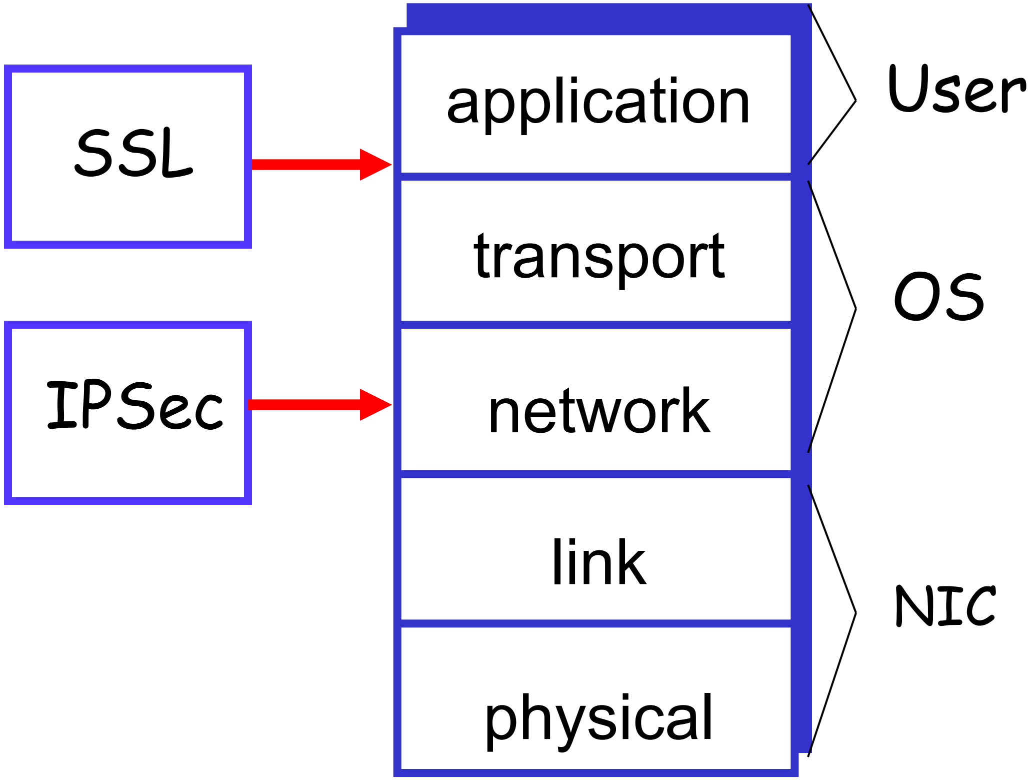
SSL vs IPSec

- 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 should 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 encryption and/or integrity of IP packets
  - 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
- 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!
- Need more evidence it's over-engineered?

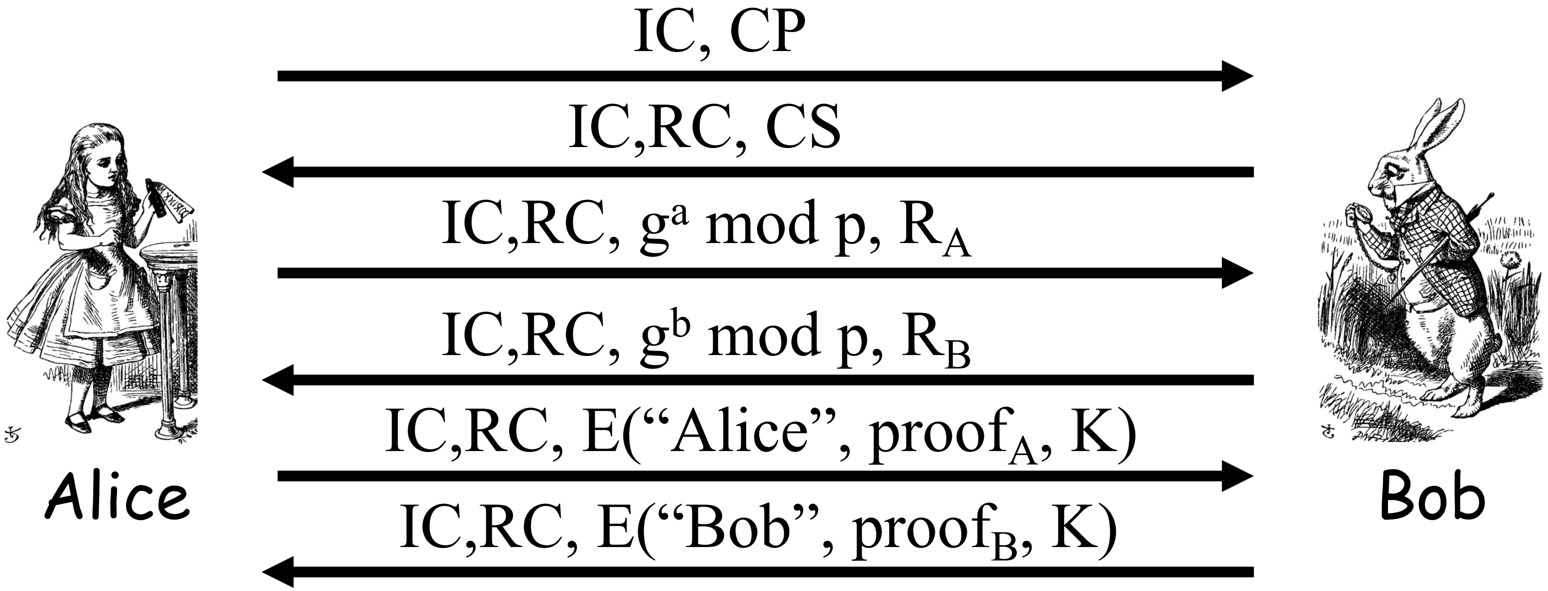
IKE Phase 1

- 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 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

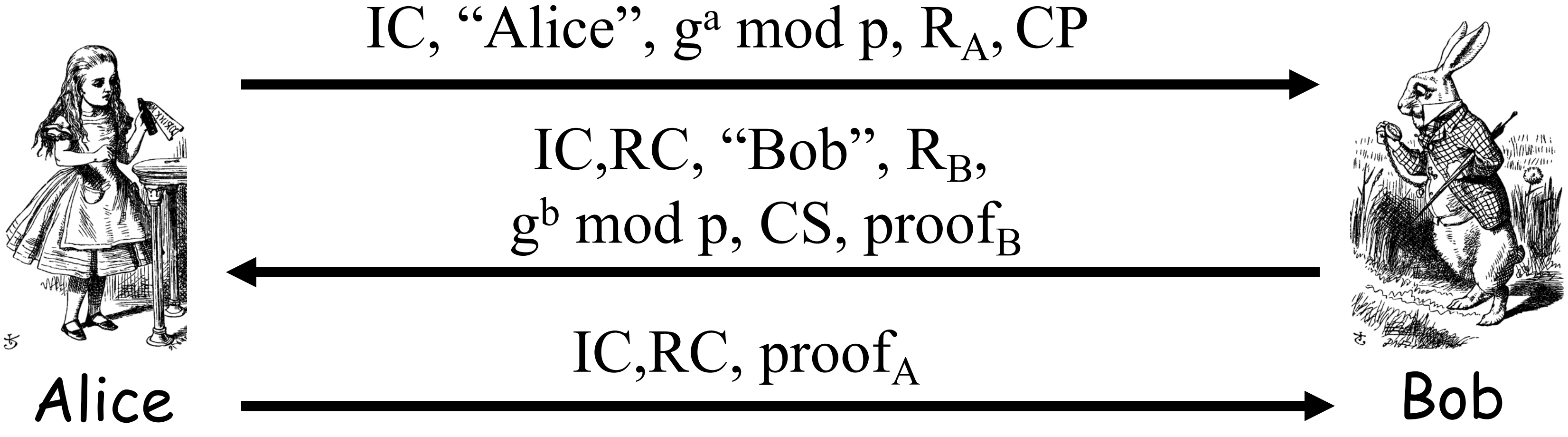
IC,RC, g^b mod p, R_B

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 mod p, R_A, CP

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

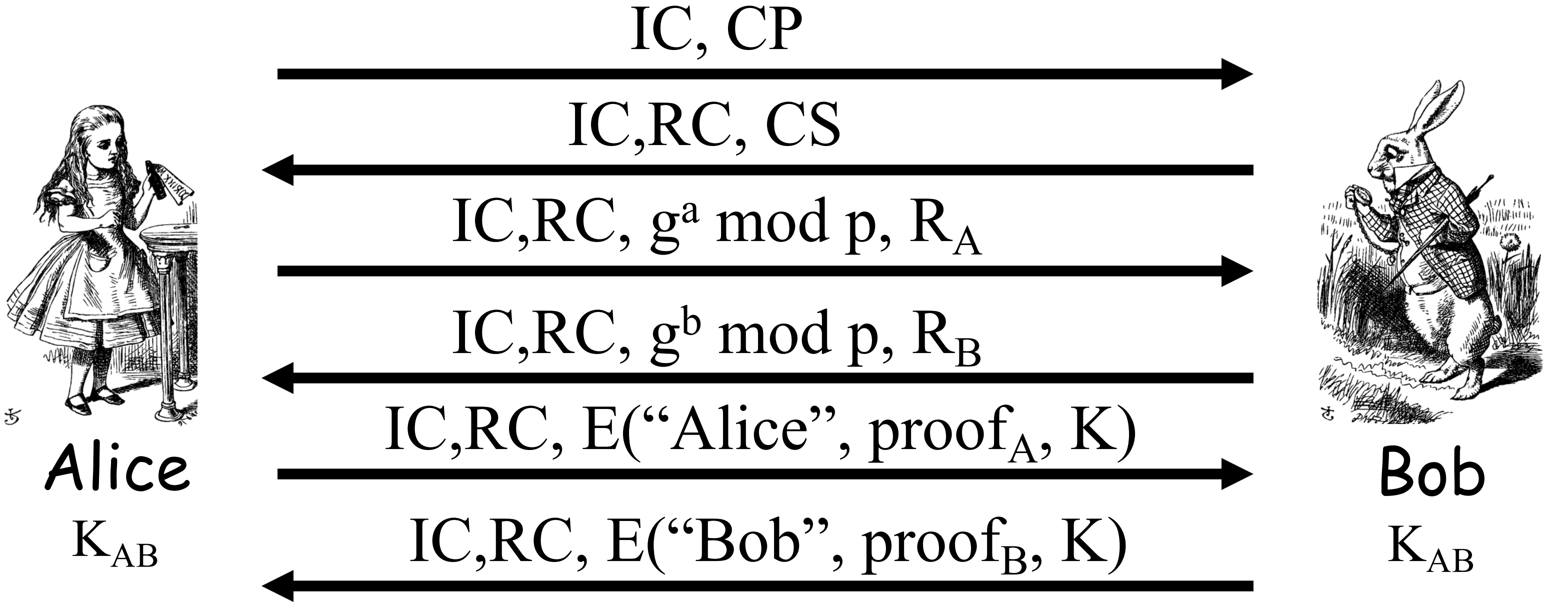
IC,RC, proof_A

- 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
- 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)

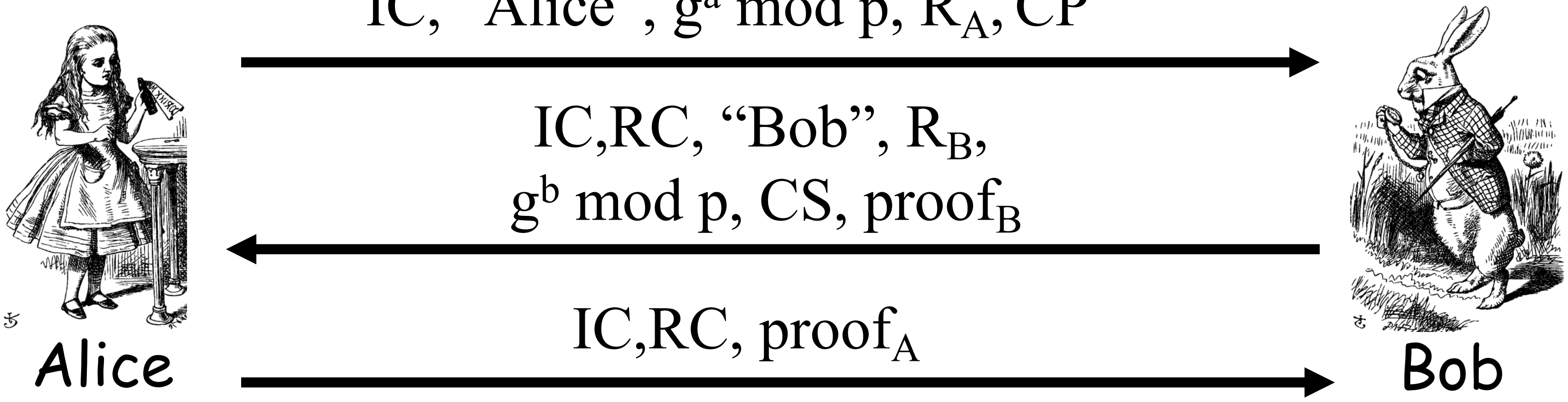
IC,RC, E(“Bob”, proof_B, K)

- 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 ID must be IP address!**
- 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)



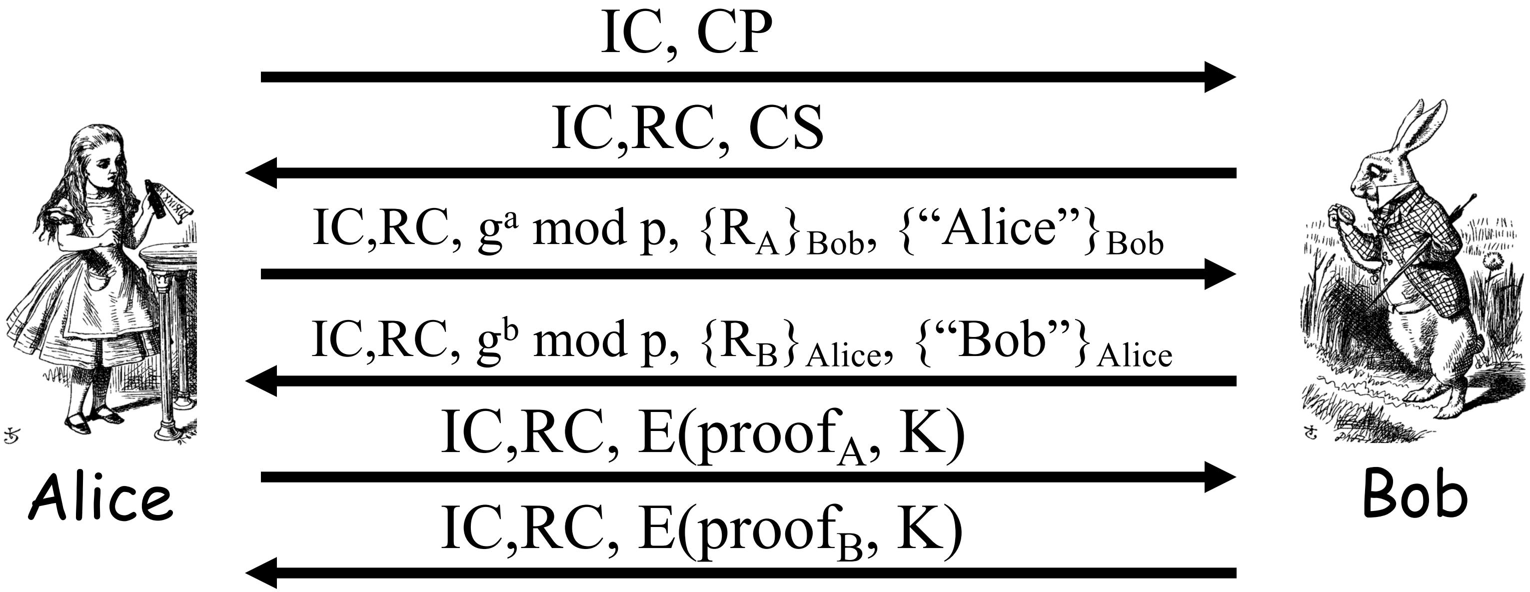
IC, “Alice”, g^a mod p, R_A, CP

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

IC,RC, proof_A

- 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)

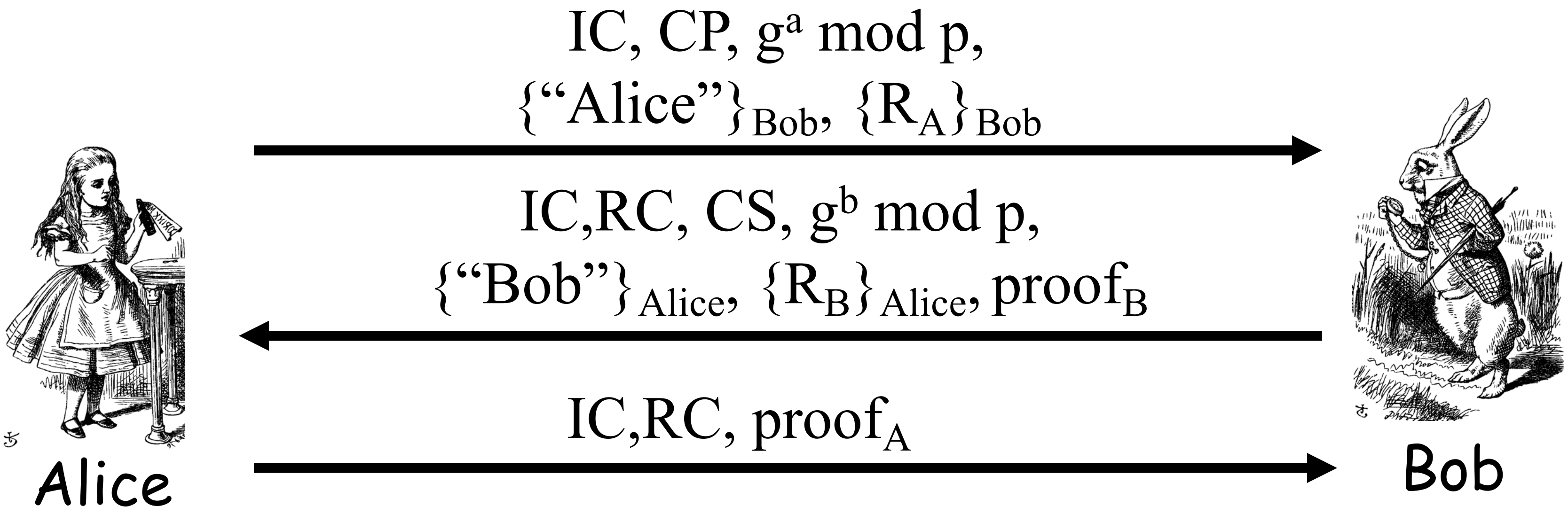
IC,RC, E(proof$_B$, K)

Alice

Bob

- 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

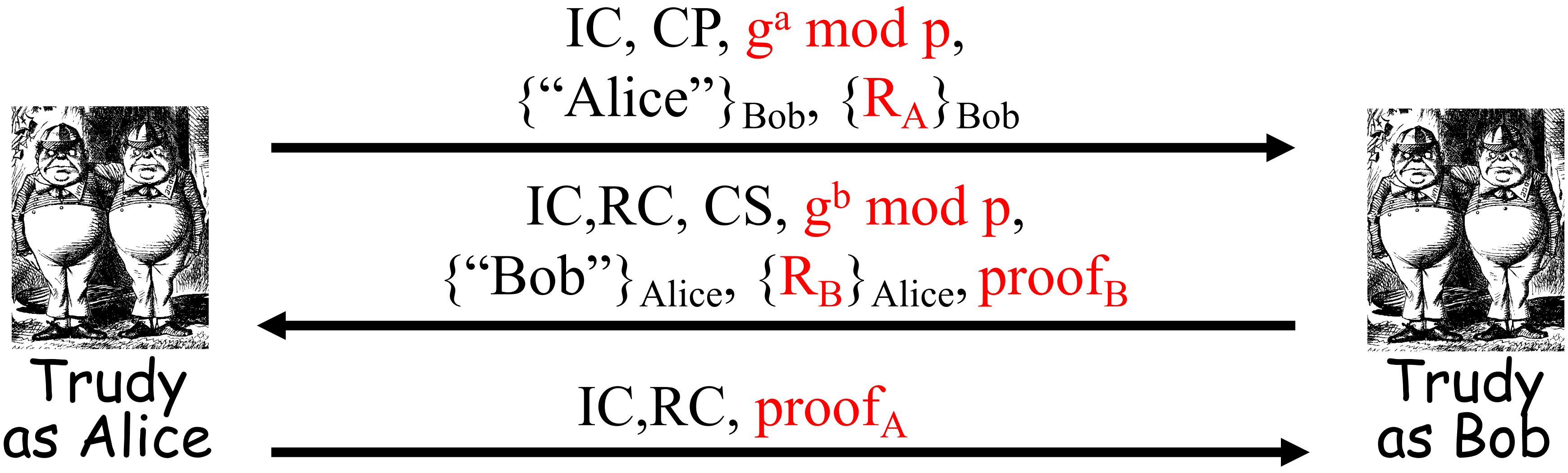
IC,RC, proof_A

- 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 R_A and R_B
- Trudy can compute "valid" keys and proofs: $g^{ab} \bmod p$, K, **SKEYID**, proof_A and proof_B
- 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 exchange that appears to be between Alice and Bob
- Appears valid to any observer, **including Alice and Bob!**

Plausible Deniability

- Trudy can create "conversation" that appears to be between Alice and Bob
- Appears valid, even to Alice and Bob!
- A security *failure*?
- In this IPSec 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

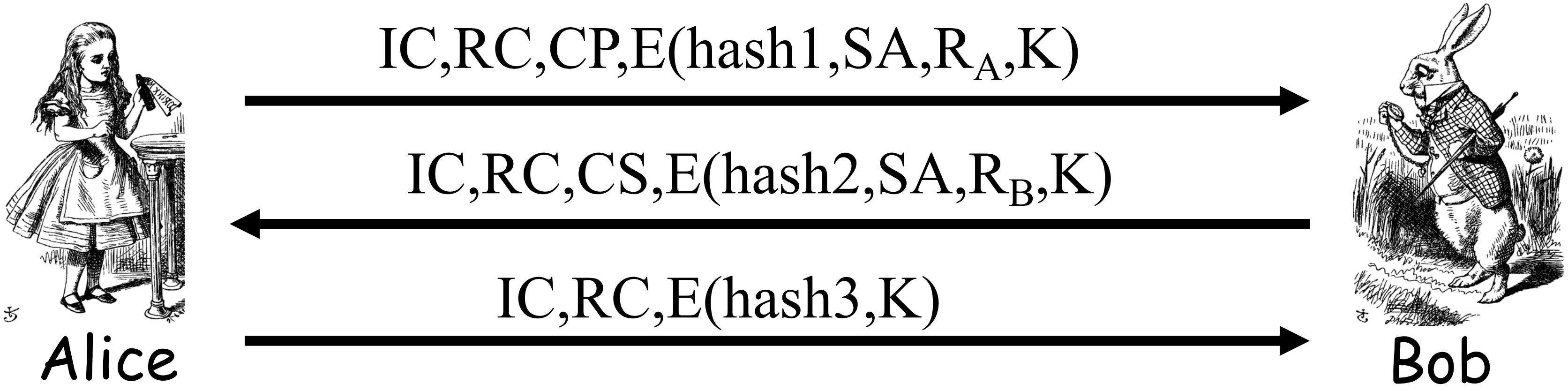
IKE Phase 1 Summary

- 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)

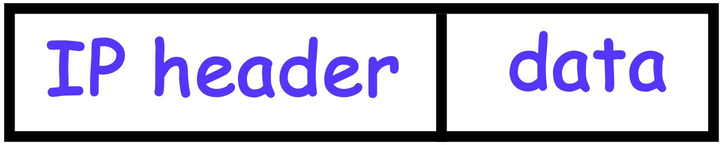
- Key K, IC, RC and SA known from Phase 1
- Proposal CP includes ESP and/or AH
- Hashes 1,2,3 depend on SKEYID, SA, R_A and R_B
- Keys derived from KEYMAT = h(SKEYID,R_A,R_B,junk)
- Recall SKEYID depends on phase 1 key method
- Optional PFS (ephemeral Diffie-Hellman exchange)

IPSec

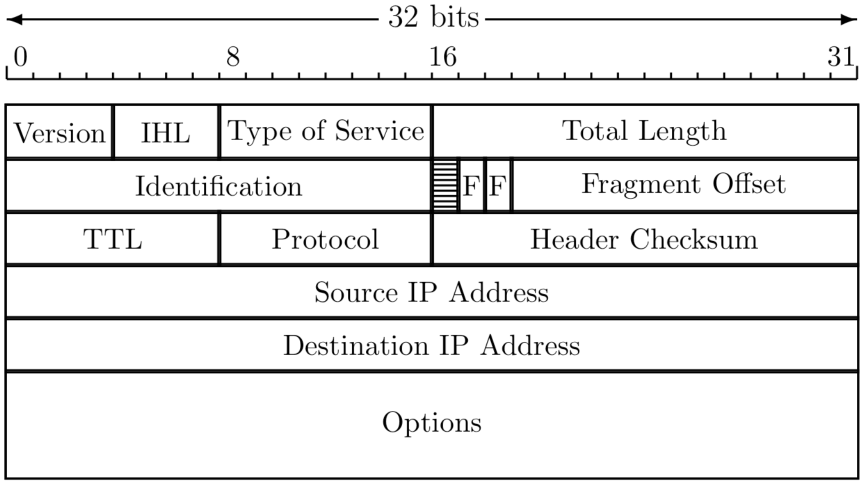
- After IKE Phase 1, we have an IKE SA
- After IKE Phase 2, we have an IPSec SA
- Both sides have a shared symmetric key
- Now what?
  - We want to protect **IP datagrams**
- But what is an IP datagram?
  - Considered from the perspective of IPSec…

IP Review

- IP datagram is of the form

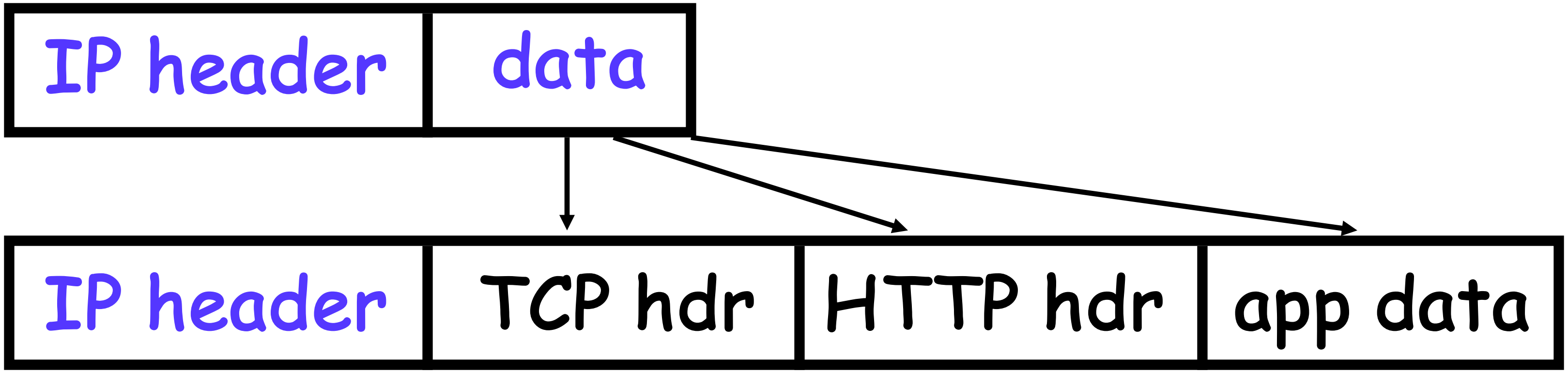


- Where IP header is



IP and TCP

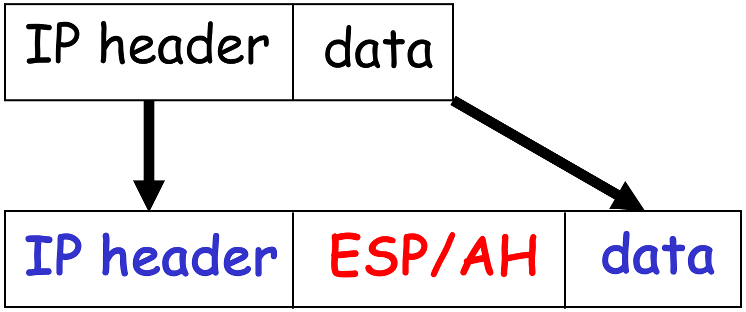
- Consider Web traffic
  - IP encapsulates TCP and…
  - …TCP encapsulates HTTP



- IP data includes TCP header, etc.

IPSec Transport Mode

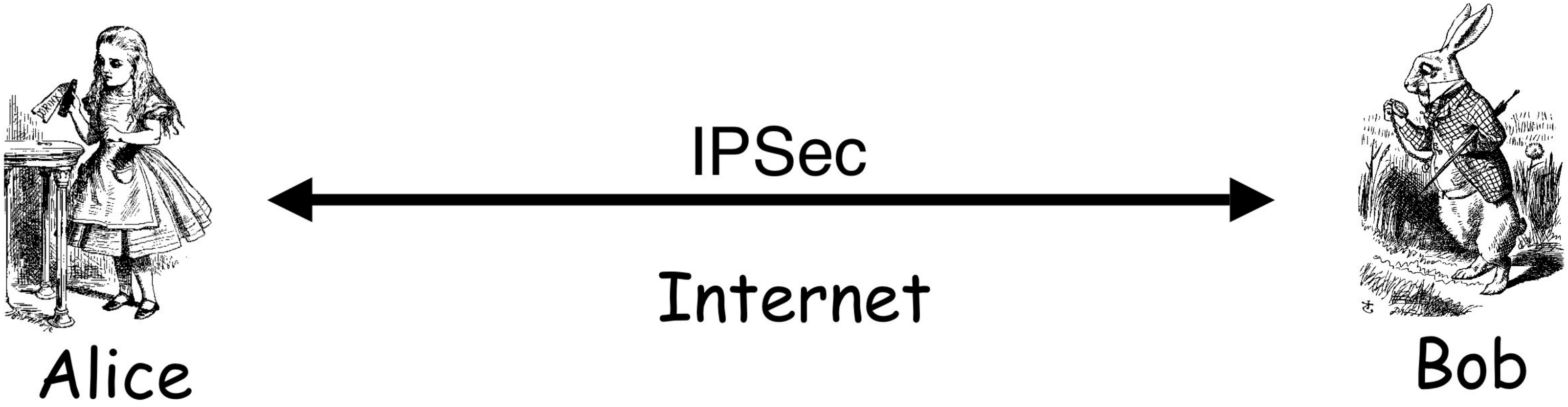
- IPSec Transport Mode



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

IPSec: Host-to-Host

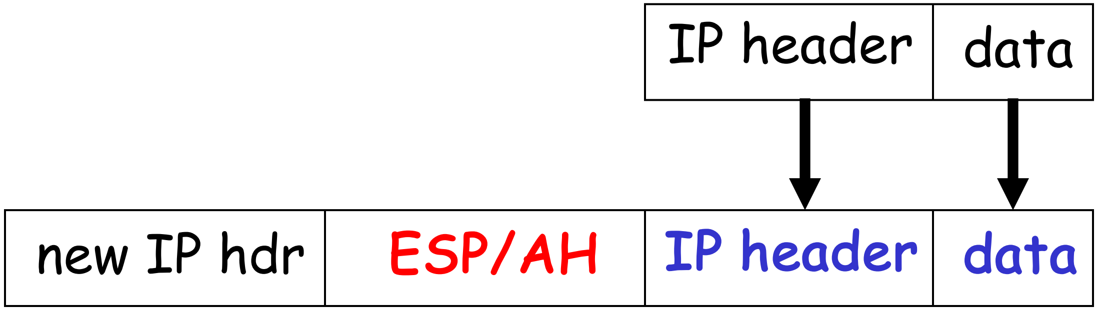
- IPSec transport mode



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

IPSec Tunnel Mode

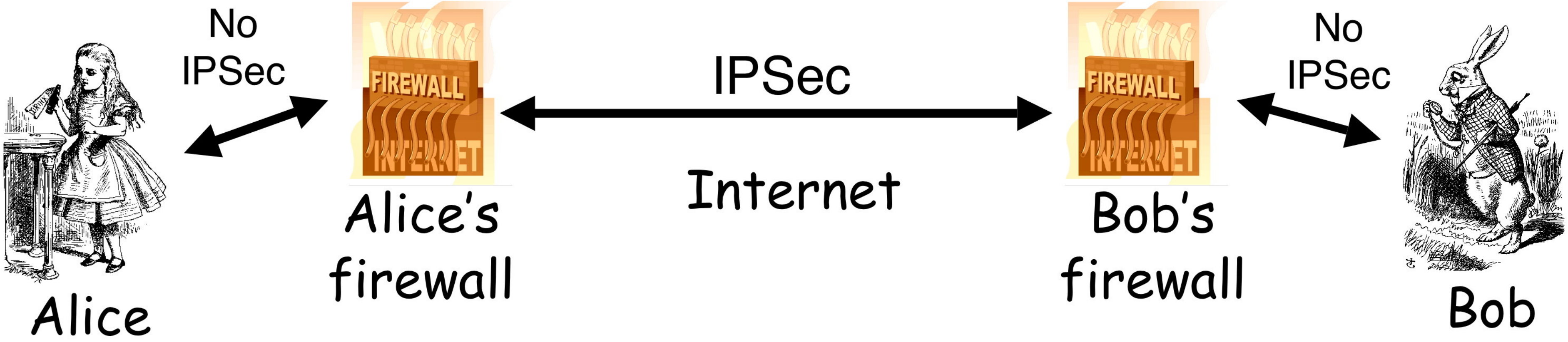
- IPSec **Tunnel Mode**



- 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

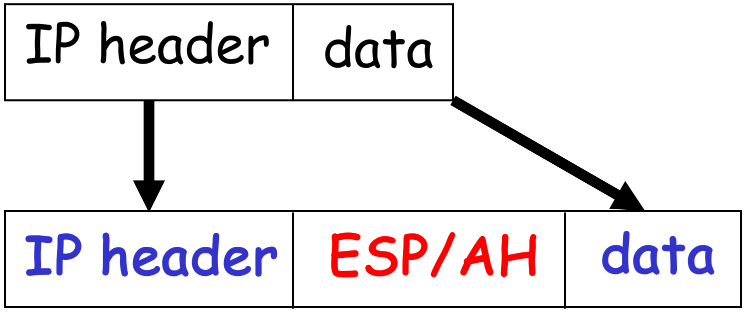
- IPSec tunnel mode



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

Comparison of IPSec Modes

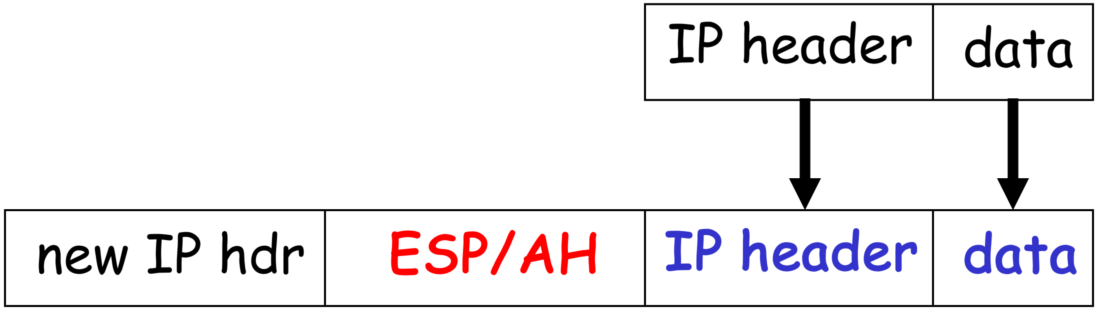
- Transport Mode



IP header | data

IP header | ESP/AH | data

- Tunnel Mode



IP header | data

new IP hdr | ESP/AH | IP header | data

- 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)
  - 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'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? (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 (e.g., port numbers)
- 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 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 (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

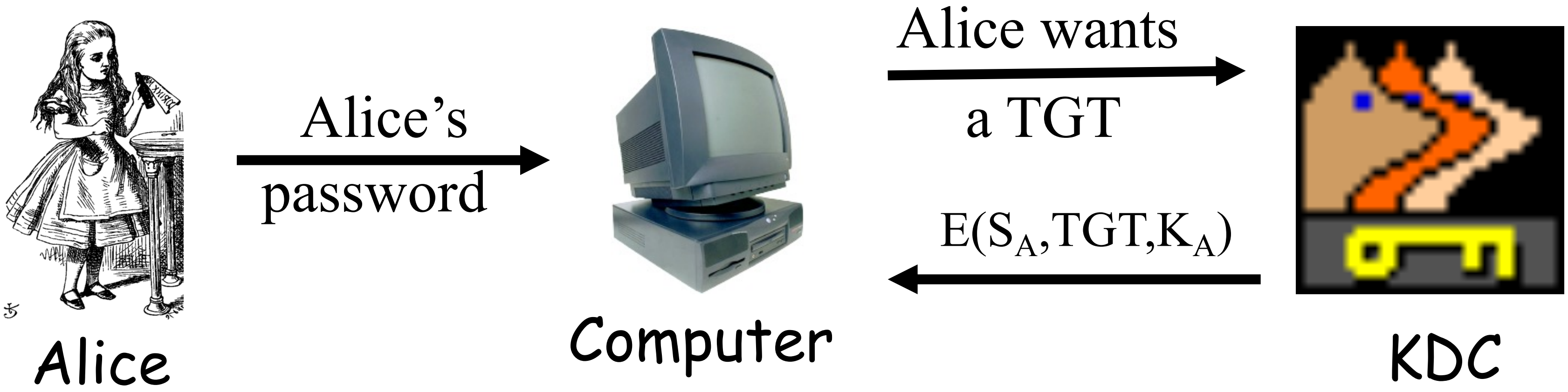
Kerberos Tickets

- 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

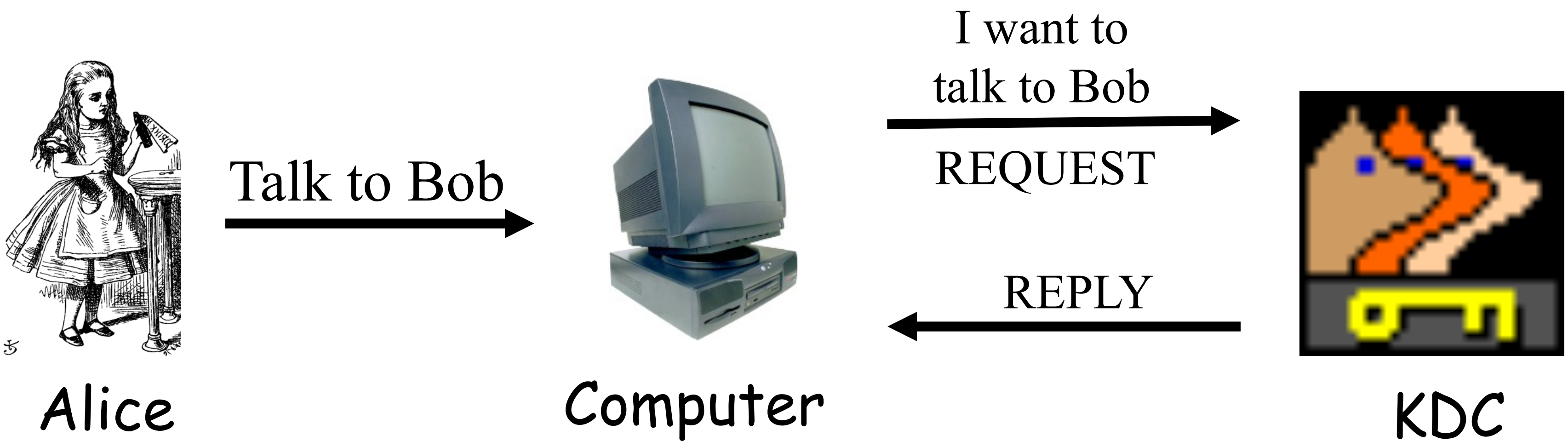
- Alice enters her password
- Then Alice's computer does following:
    - Derives K_A from Alice's password
    - Uses K_A to get TGT for Alice from KDC
- Alice then uses her TGT (credentials) to securely access network resources
- **Plus:** Security is transparent to Alice
- **Minus:** KDC *must* be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



Talk to Bob

I want to
talk to Bob

REQUEST

REPLY

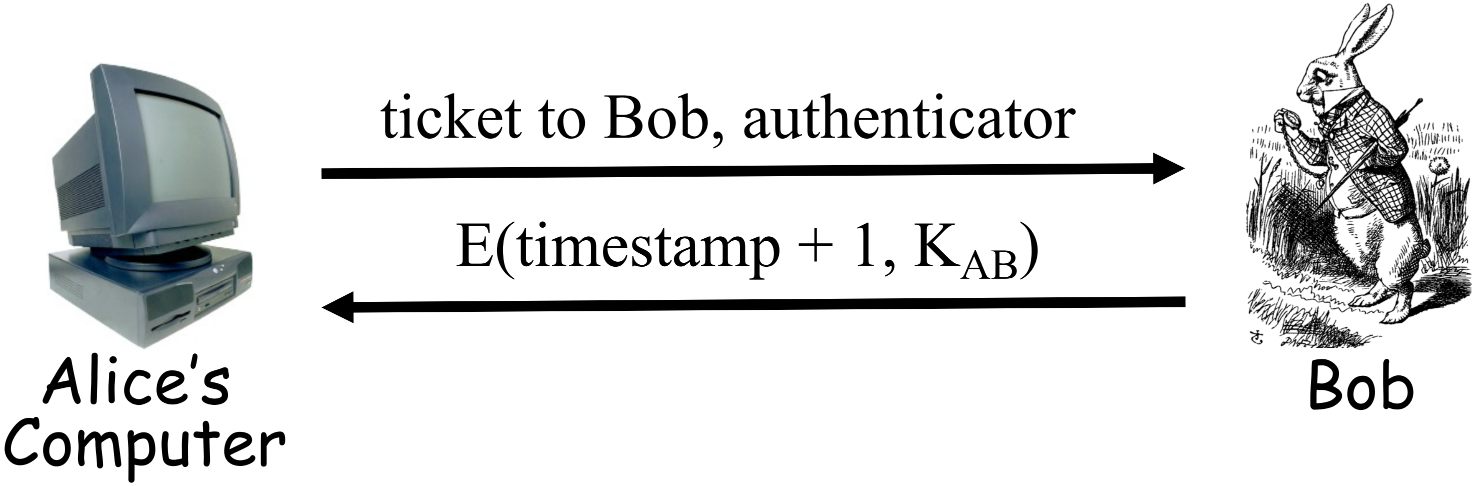
Alice

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

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

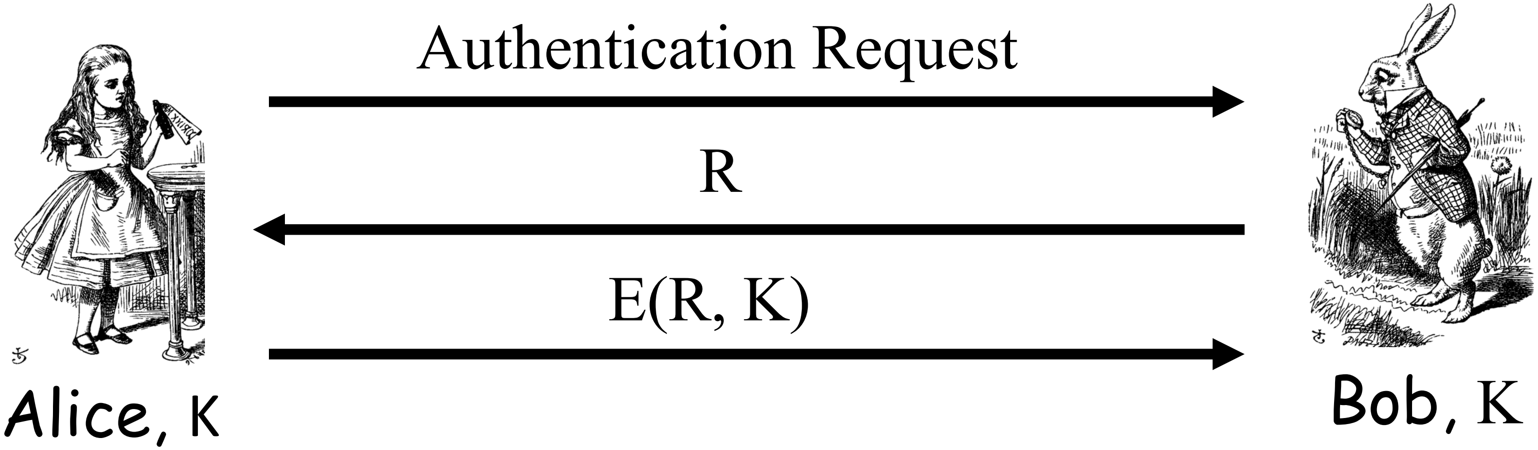
- In Kerberos, K_A = h(Alice's password)
- Could instead generate random K_A
  - Compute K_h = h(Alice's password)
  - And Alice's computer stores $E(K_A, K_h)$
- Then K_A need not change when Alice changes her password
  - But $E(K_A, K_h)$ must be stored on computer
- This alternative approach is often 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 is considered a strong cipher
  - But WEP introduces a subtle flaw…
  - …making cryptanalytic attacks feasible
- WEP uses CRC for "integrity"
  - Should have used a MAC or HMAC instead
  - CRC is for error detection, not crypto integrity
  - Everyone knows NOT to use CRC for this…

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 remains correct
  - Then Trudy’s introduced errors go undetected
  - Requires no knowledge of the plaintext!
- CRC does ***not*** provide a cryptographic integrity check
  - CRC designed to detect random errors
  - Not able to detect intelligent changes

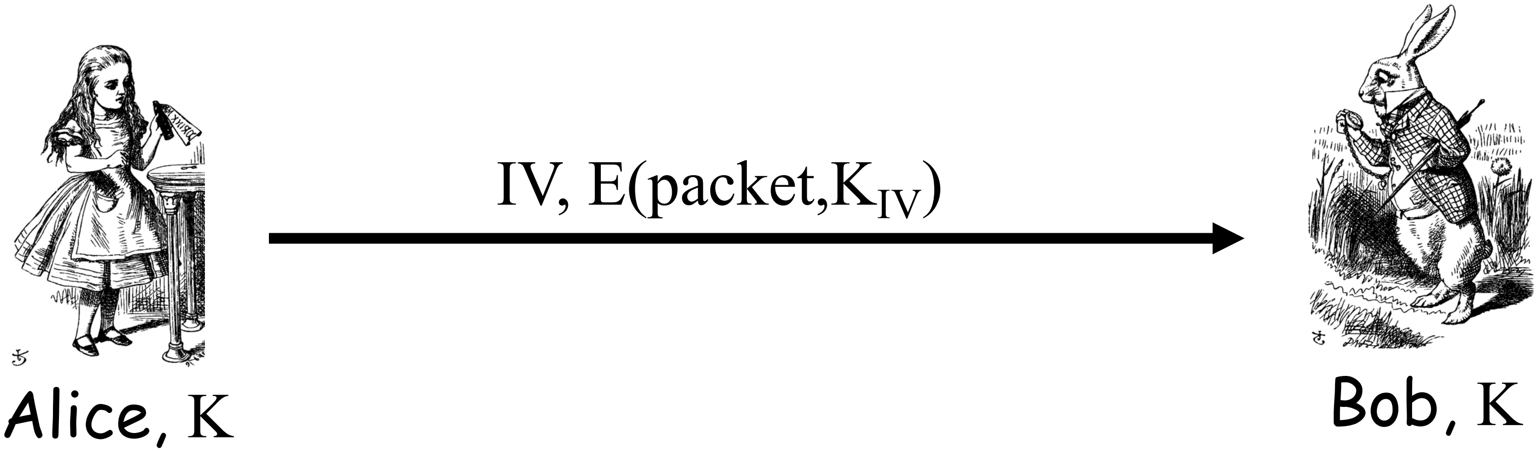
More WEP Integrity Issues

- 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 problem when integrity fails

WEP Key

- Recall WEP uses a long-term secret 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
- Note that the IV is known to Trudy

WEP IV Issues

- WEP uses 24-bit (3 byte) IV
  - Each packet gets a new 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 $1500 \cdot 8/(11 \cdot 10^6) \cdot 2^{24} = 18{,}000$ seconds…
  - …an IV must repeat in about 5 hours
- 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(s) 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 and 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 IVs and ciphertext
  - Trudy wants to find the key K

Cryptanalytic Attack

- 3-byte IV pre-pended to key
- Denote the RC4 key **bytes**…
  - …as $K_0,K_1,K_2,K_3,K_4,K_5, \ldots$
  - 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 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

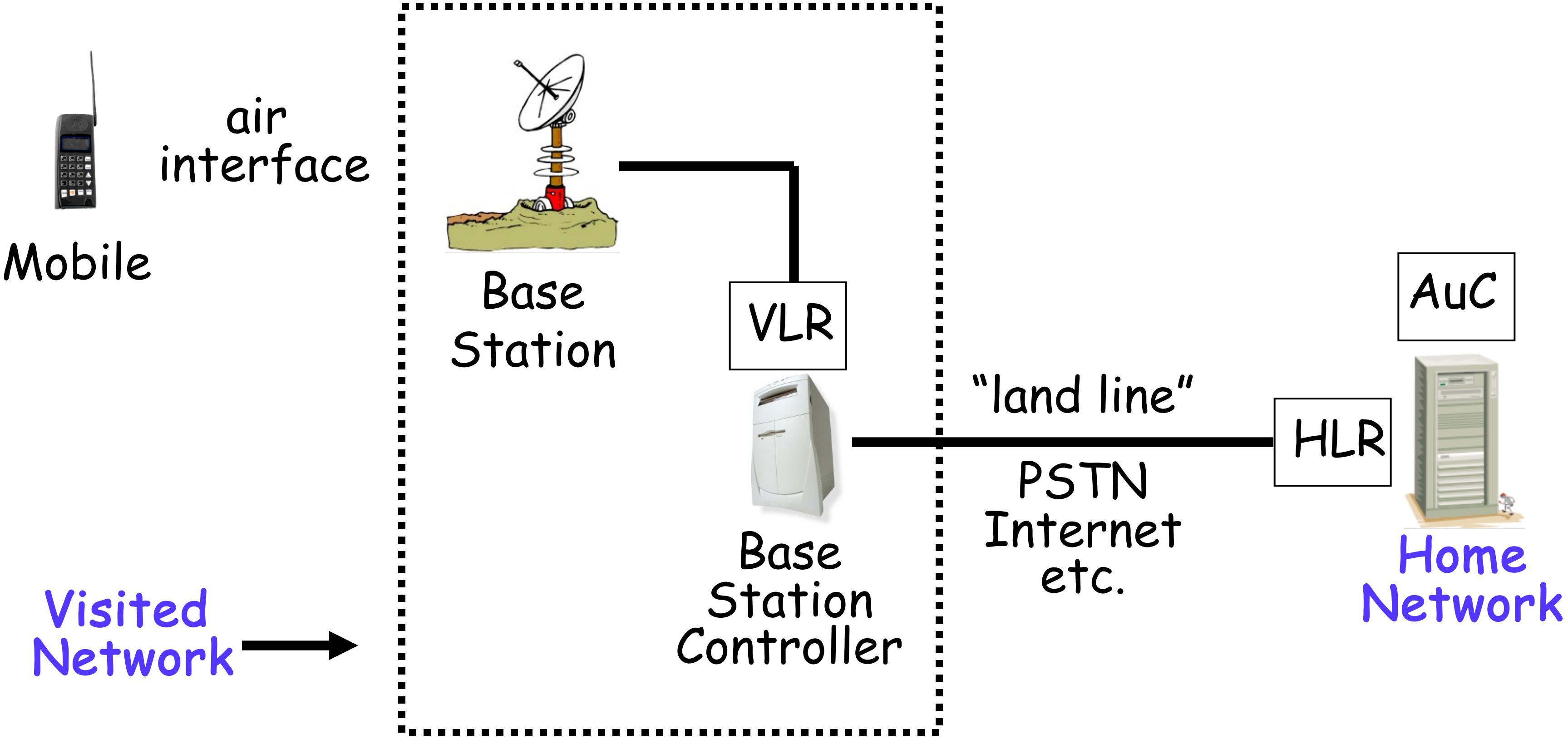
- Many attacks are practical
- Attacks have been used to recover keys and break real WEP traffic
- How to prevent WEP attacks?
  - Don't use WEP
  - Good alternatives: WPA, WPA2, etc.
- How to make WEP a little better?
  - Restrict MAC addresses, don't broadcast ID, …

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



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 (usually 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 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
  - May be important 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 sufficient for most uses

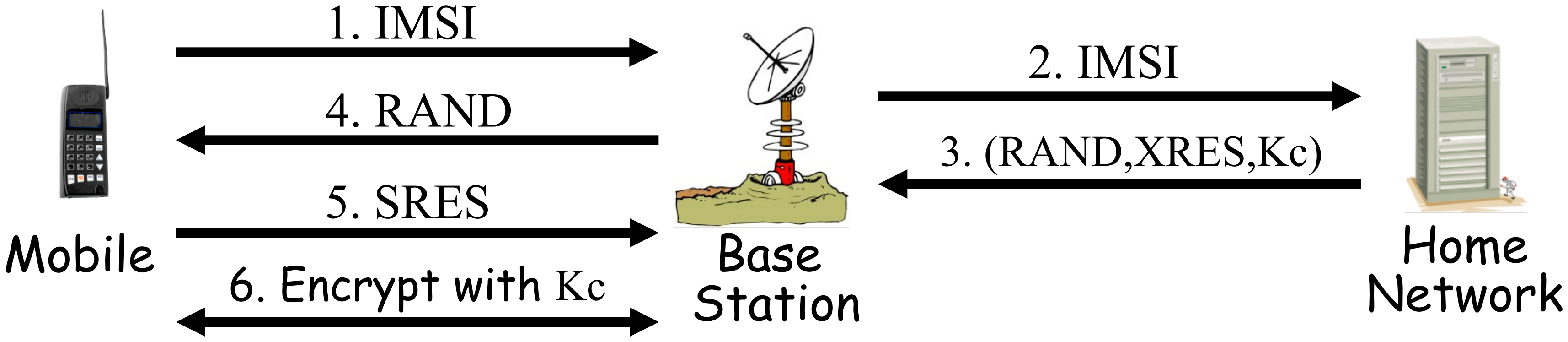
GSM: Authentication

- 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!

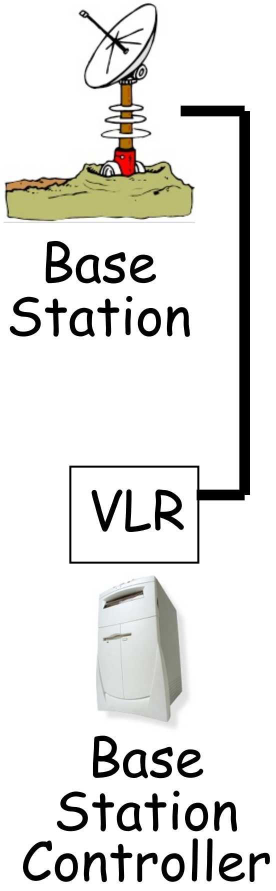
GSM Security



- 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

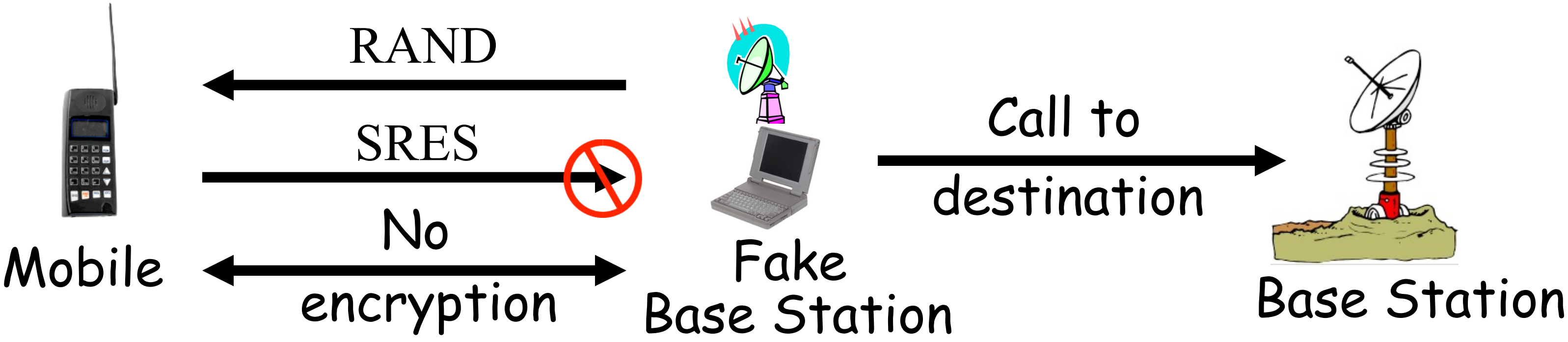


GSM Insecurity (2)

- 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
  - Encryption not automatic
  - 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?

3GPP: 3rd Generation Partnership Project

- 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: GSM and WEP

Coming Attractions…

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