

ALICE SENDS A MESSAGE TO BOB
SAYING TO MEET HER SOMEWHERE.

UH HUH.

BUT EVE SEES IT, TOO,
AND GOES TO THE PLACE.

WITH YOU SO FAR.

BOB IS DELAYED, AND
ALICE AND EVE MEET.

YEAH?



I'VE DISCOVERED A WAY TO GET COMPUTER
SCIENTISTS TO LISTEN TO ANY BORING STORY.

CS 492

Computer Security

Simple Security Protocols

Dr. Williams
Central Connecticut State University

(credit to xkcd.com)

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 even IPSec
- Implementation errors can occur
 - Such as IE implementation of SSL
- Not easy to get protocols right...

Ideal Security Protocol

- Satisfies security requirements
 - Requirements must be precise
- Efficient
 - Small computational requirement
 - Small bandwidth usage, network 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!

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

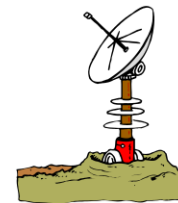
Angola



SAAF
Impala
K

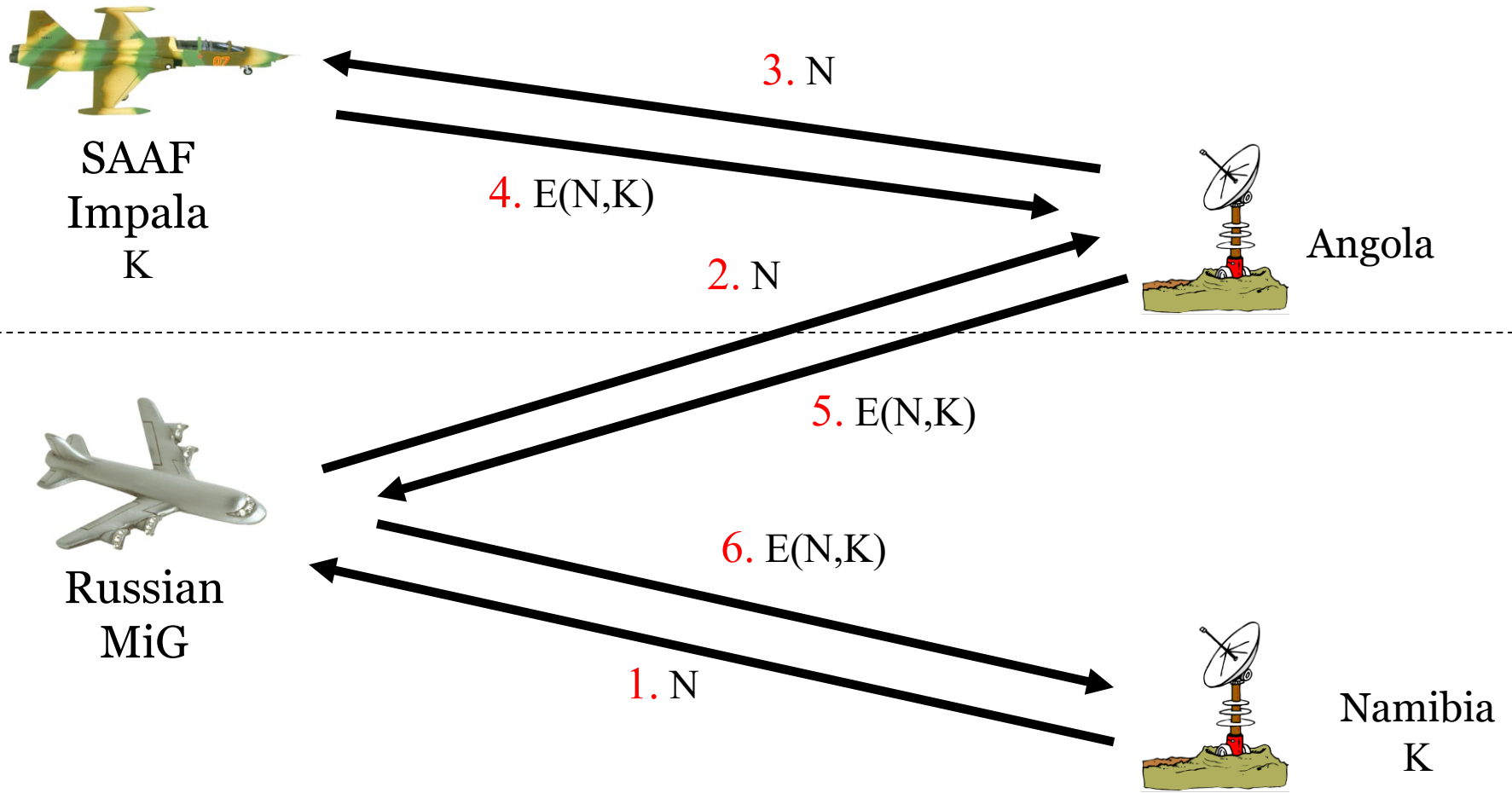
2. $E(N,K)$

1. N



Namibia
K

MIG in the Middle



Authentication Protocols

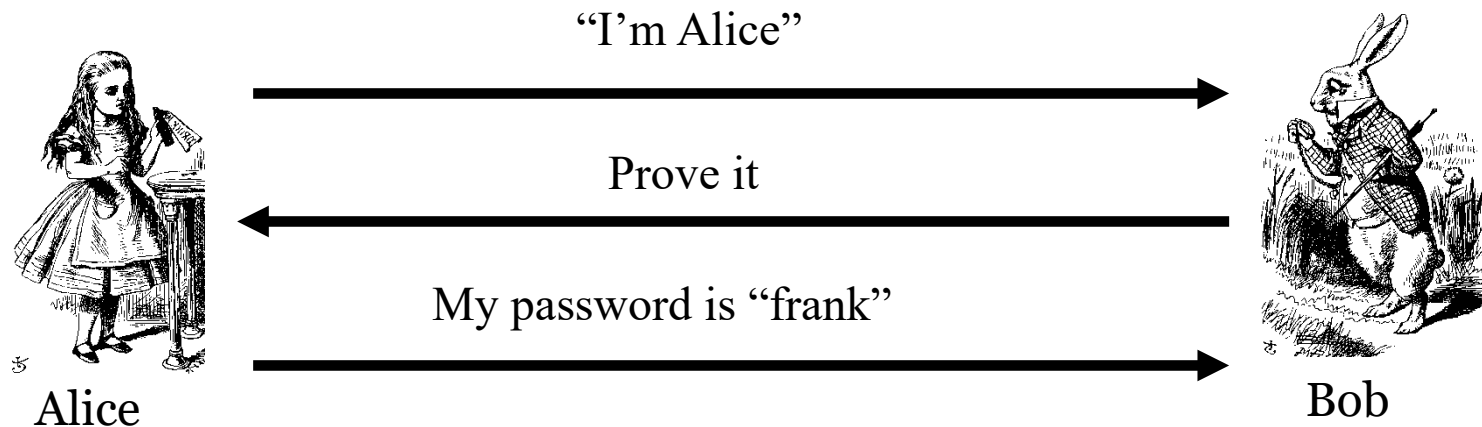
Authentication

- 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

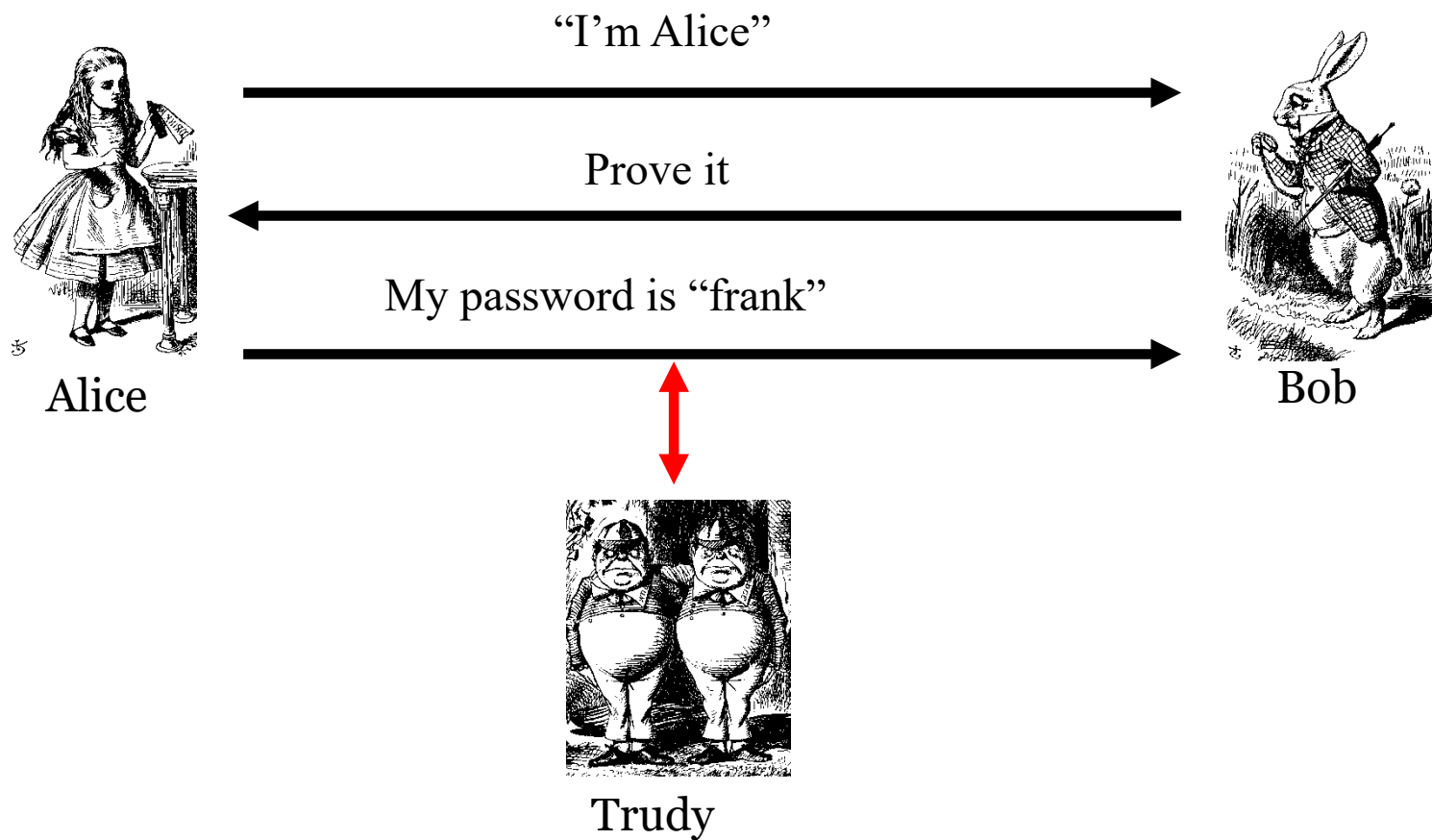
- Authentication on a stand-alone computer is relatively simple
 - Hash password with salt, etc.
 - “Secure path,” attacks on authentication software, keystroke logging, etc., are issues
- Authentication over a network is challenging
 - Attacker can passively observe messages
 - Attacker can replay messages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

Simple Authentication

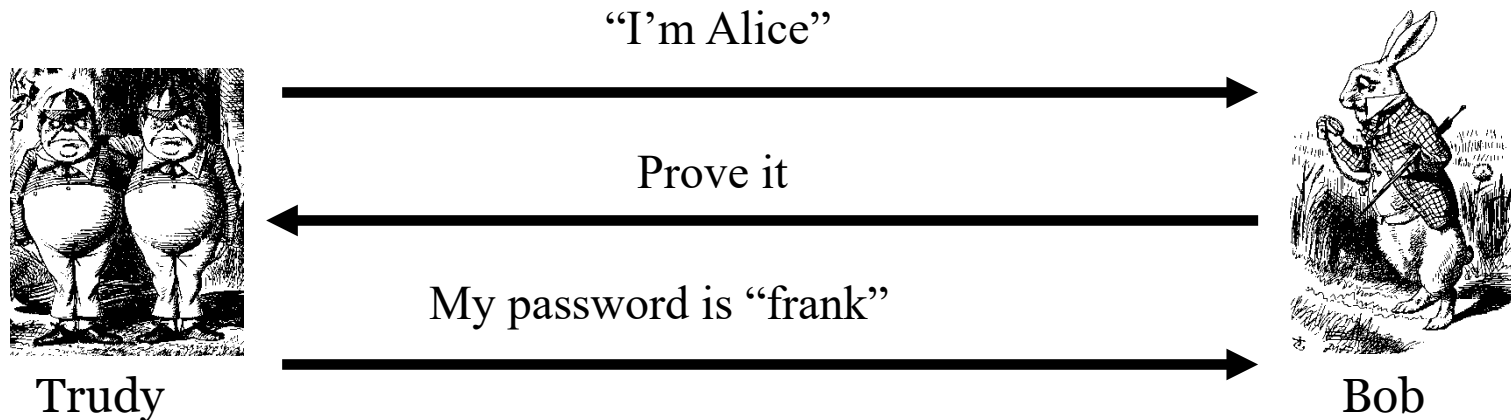


- Simple and may be OK for standalone system
- But insecure for networked system
 - Subject to a **replay** attack (next 2 slides)
 - Also, Bob must know Alice's password

Authentication Attack



Authentication Attack



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

Simple Authentication



Alice

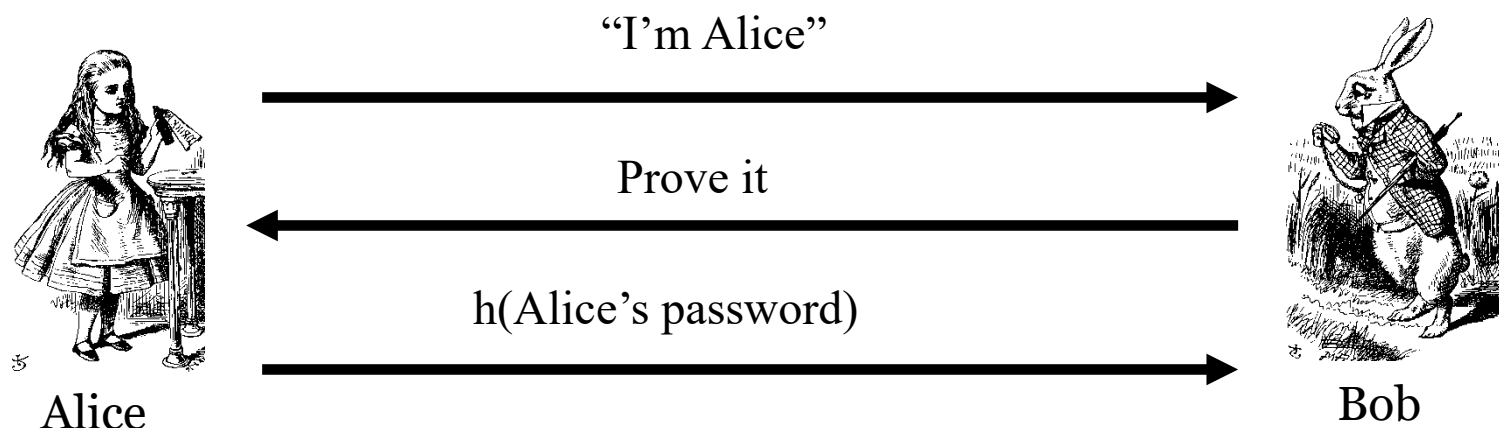
I'm Alice, my password is "frank"



Bob

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

Better Authentication



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

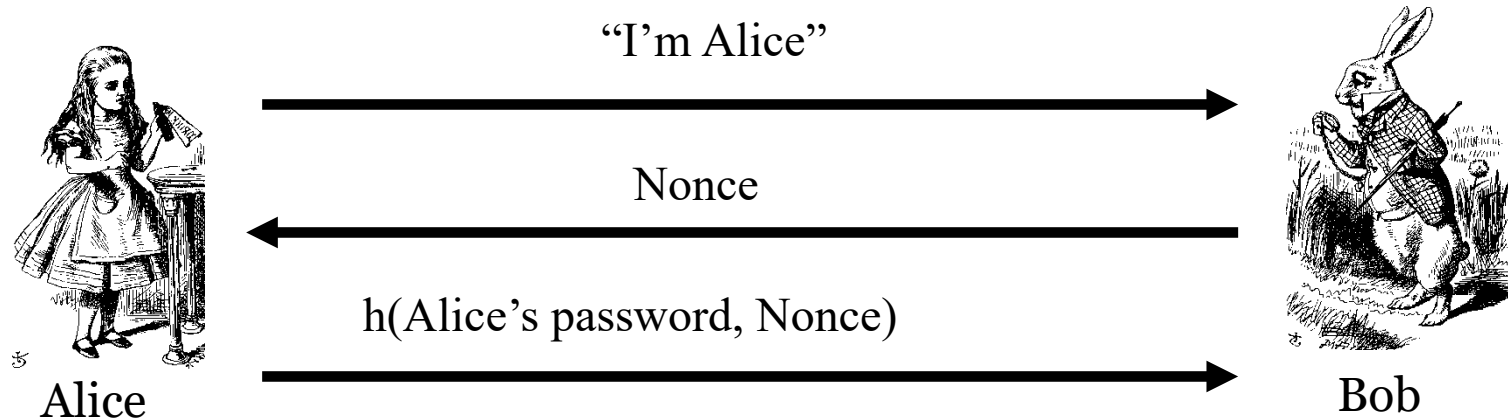
Challenge-Response

- 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

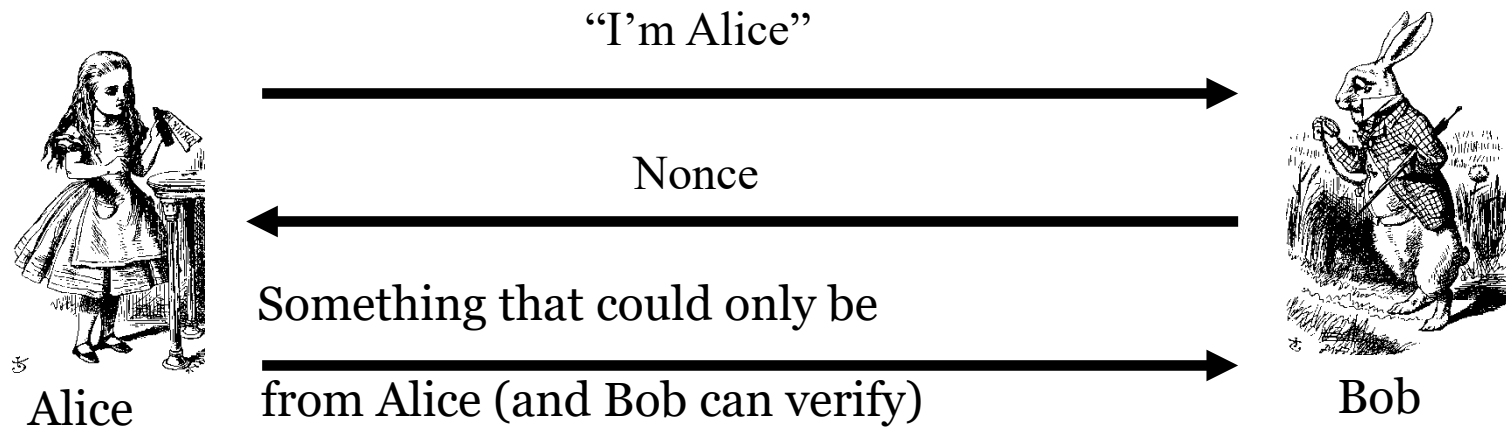
- To ensure freshness, can employ a **nonce**
 - Nonce == **number used once**
- What to use for nonces?
 - That is, what is the challenge?
- What should Alice do with the nonce?
 - That is, how to compute the response?
- How can Bob verify the response?
- Should we rely on passwords or keys?

Challenge-Response



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

Generic Challenge-Response



- In practice, how to achieve this?
- Hashed pwd works...
- Encryption is better here (Why?)

Symmetric Key Notation

- Encrypt plaintext P with key K

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

- Decrypt ciphertext C with key K

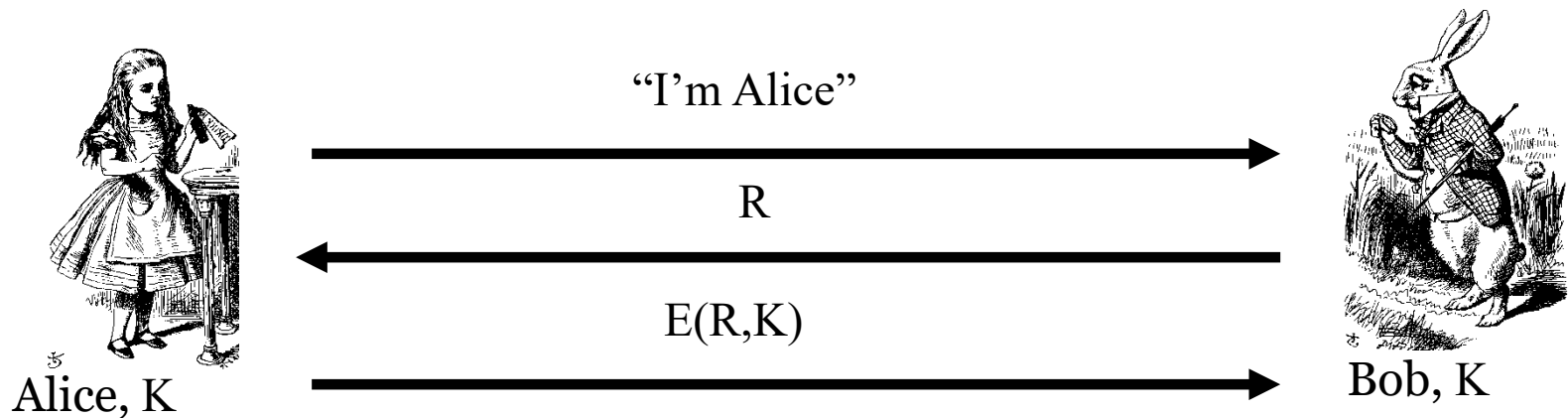
$$P = D(C, K)$$

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

Authentication: Symmetric Key

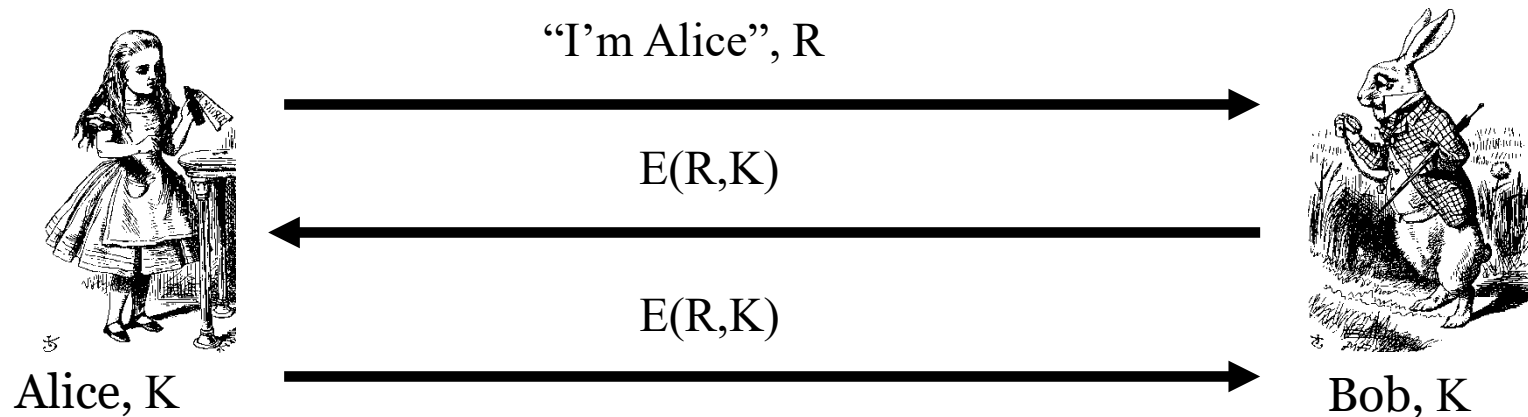
- Alice and Bob share symmetric key K
- Key K known only to Alice and Bob
- Authenticate by proving knowledge of shared symmetric key
- How to accomplish this?
 - Must not reveal key, must not allow replay (or other) attack, must be verifiable, ...

Authentication with Symmetric Key



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

Mutual Authentication?

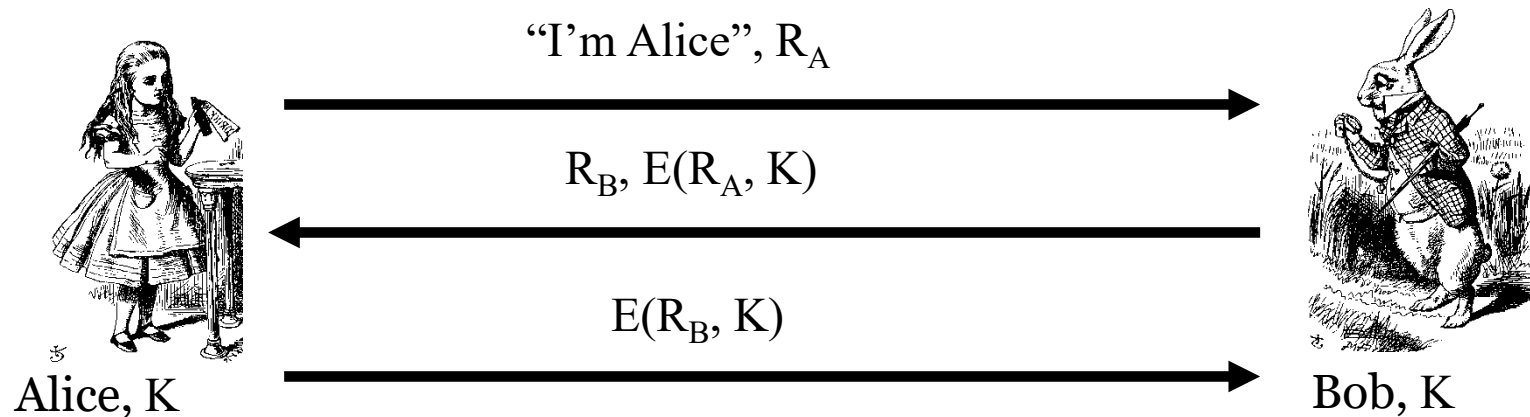


- What's wrong with this picture?
- "Alice" could be Trudy (or anybody else)!

Mutual Authentication

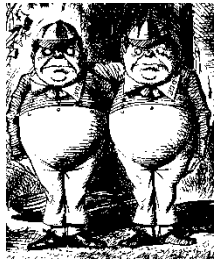
- Since we have a secure one-way authentication protocol...
- The obvious thing to do is to use the protocol twice
 - Once for Bob to authenticate Alice
 - Once for Alice to authenticate Bob
- This has got to work...

Mutual Authentication



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

Mutual Authentication Attack



Trudy

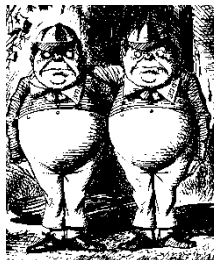
1. "I'm Alice", R_A

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

5. $E(R_B, K)$



Bob, K



Trudy

3. "I'm Alice", R_B

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

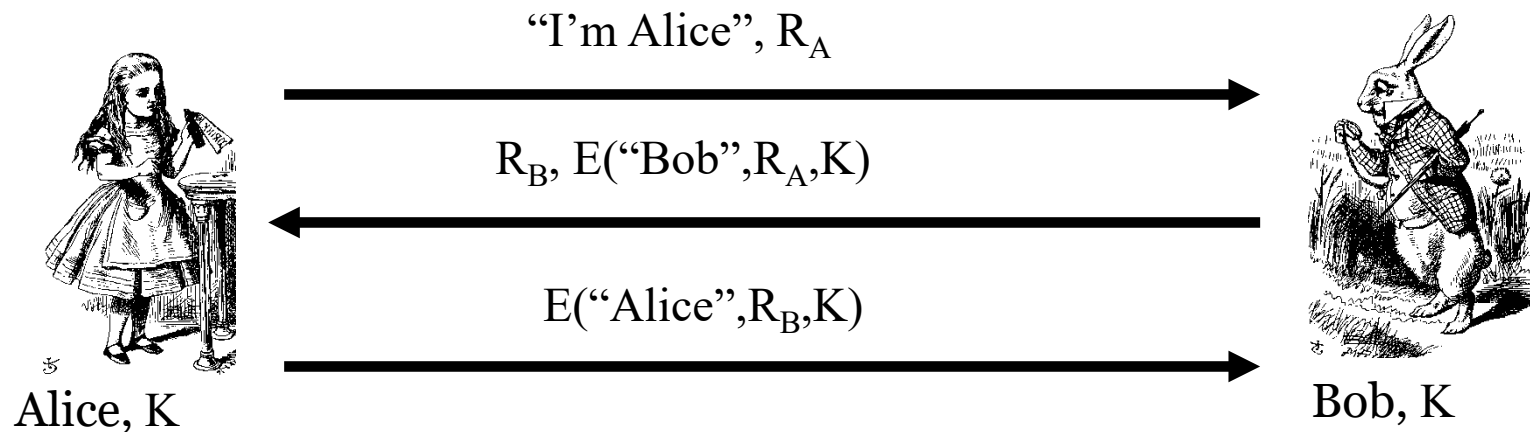


Bob, K

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

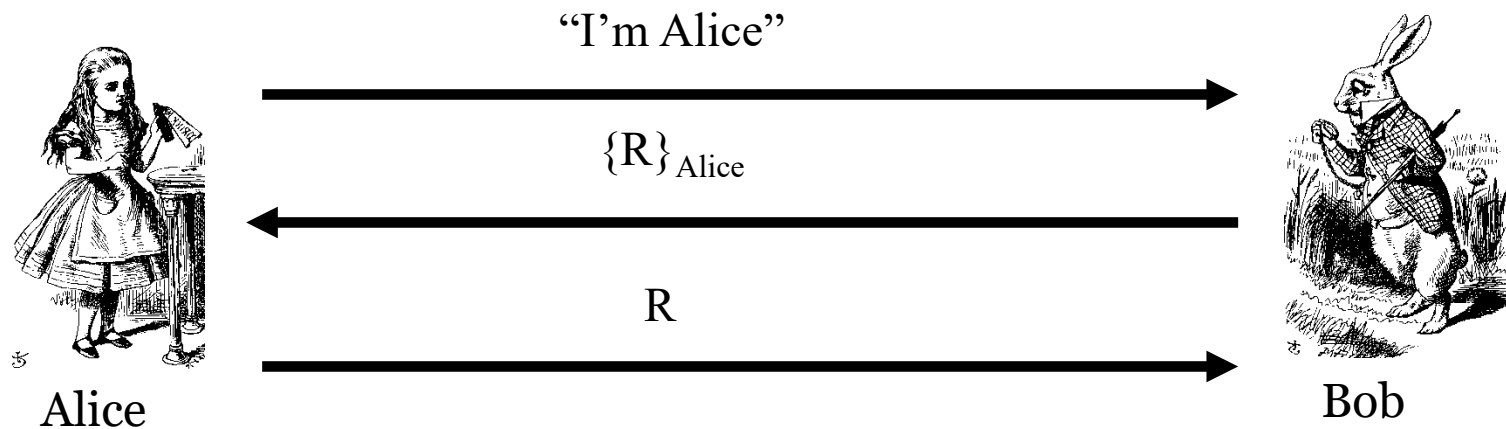


- Do these “insignificant” changes help?
- Yes!

Public Key Notation

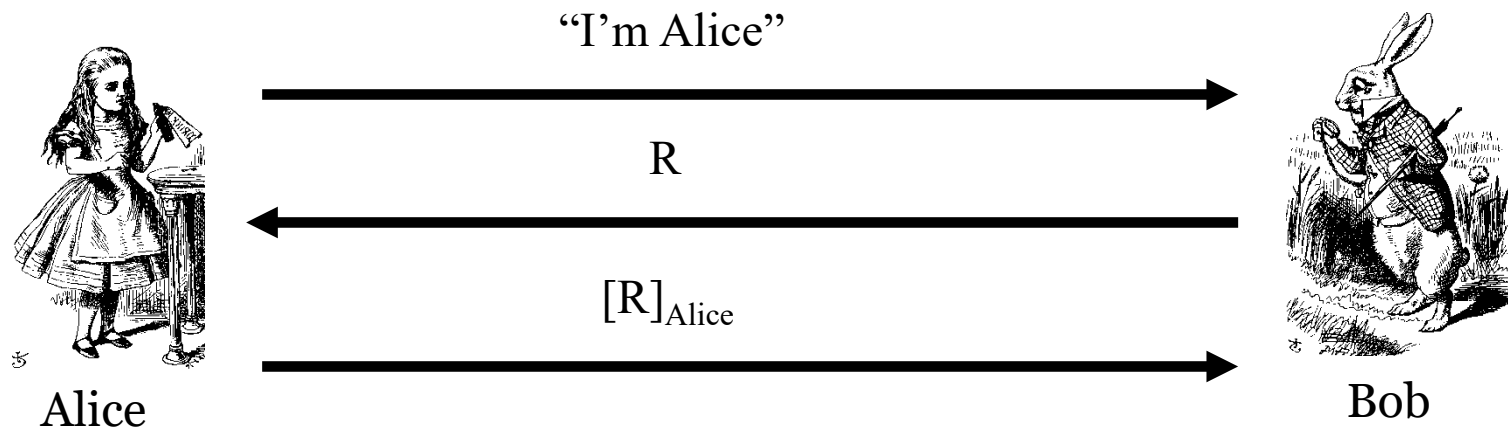
- Encrypt M with Alice's public key: $\{M\}_{\text{Alice}}$
- Sign M with Alice's private key: $[M]_{\text{Alice}}$
- Then
 - $[\{M\}_{\text{Alice}}]_{\text{Alice}} = M$
 - $\{[M]_{\text{Alice}}\}_{\text{Alice}} = M$
- **Anybody** can use Alice's **public key**
- Only **Alice** can use her **private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

Public Key Authentication



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

Public Key Authentication



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

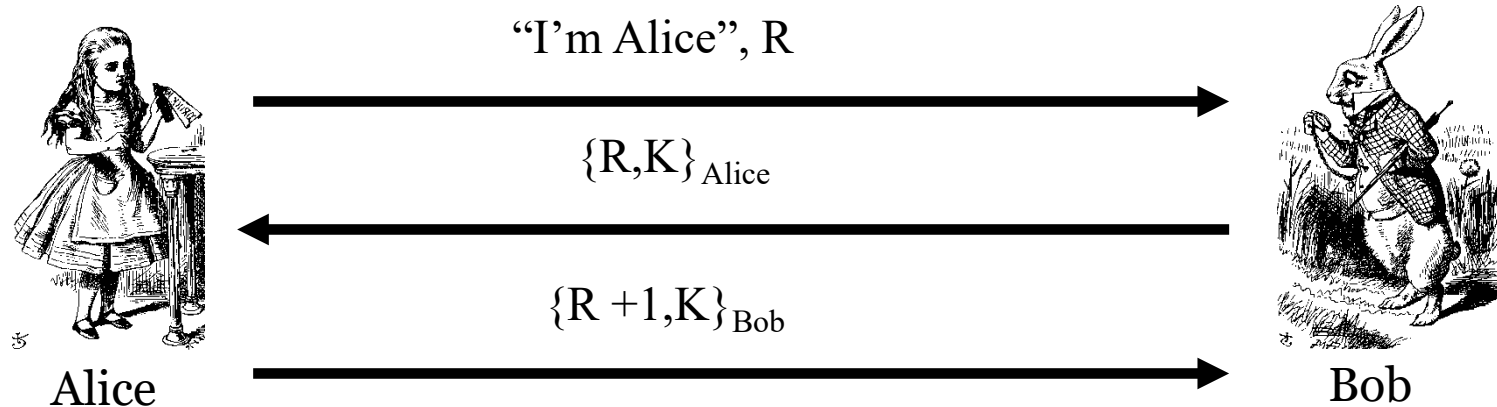
Public Keys

- 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

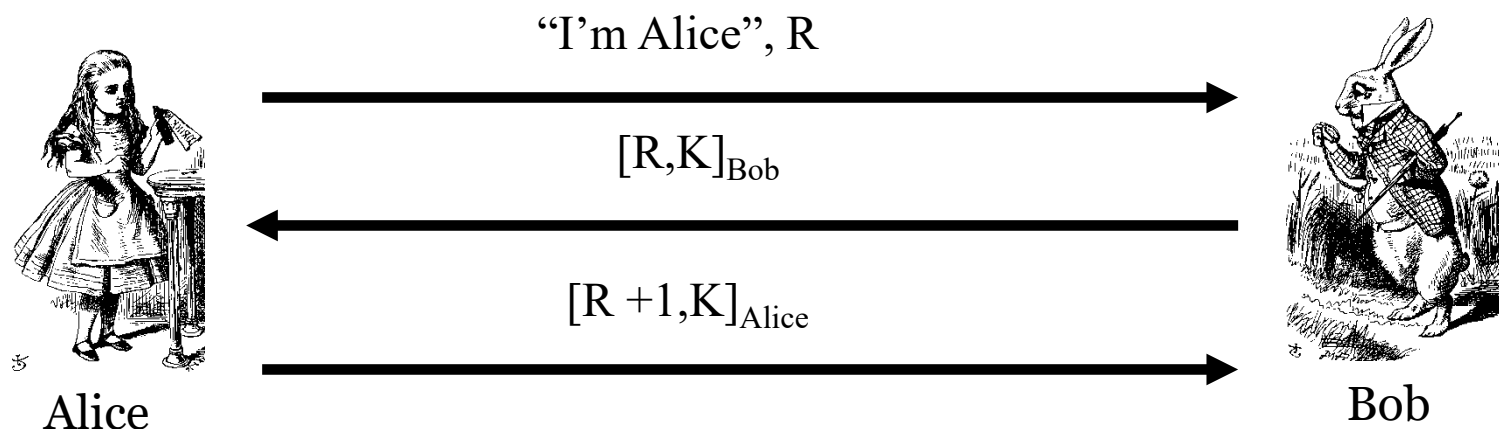
- Usually, a **session key** is required
 - I.e., a symmetric key for a particular session
 - Used for confidentiality and/or integrity
- How to authenticate and establish a session key (i.e., shared symmetric key)?
 - When authentication completed, want Alice and Bob to share a session key
 - Trudy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

Authentication & Session Key



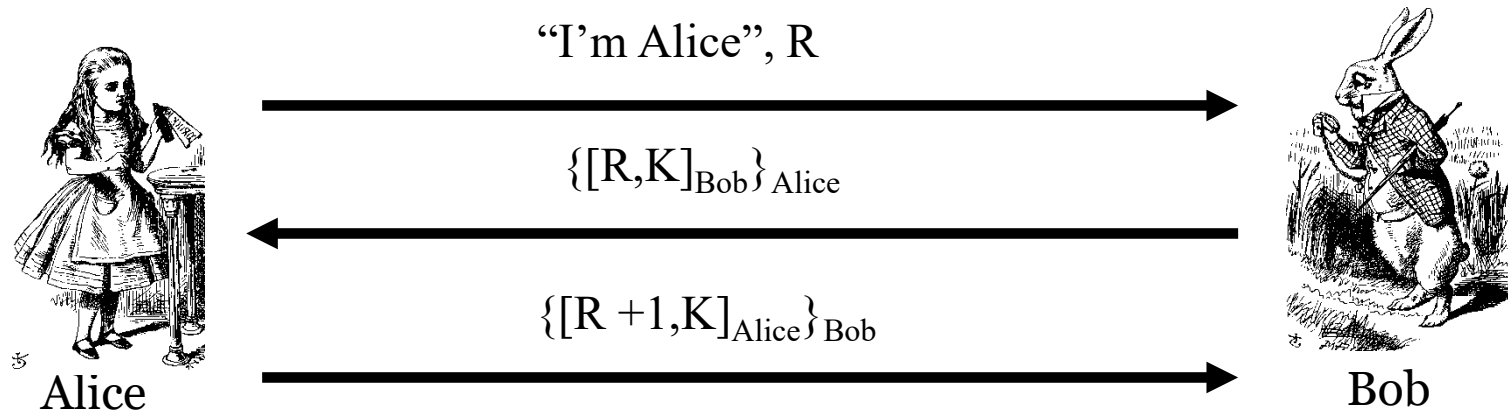
- Is this secure?
 - Alice is authenticated and session key is secure
 - Alice's "nonce", R , useless to authenticate Bob
 - The key K is acting as Bob's nonce to Alice
- No mutual authentication

Public Key Authentication and Session Key



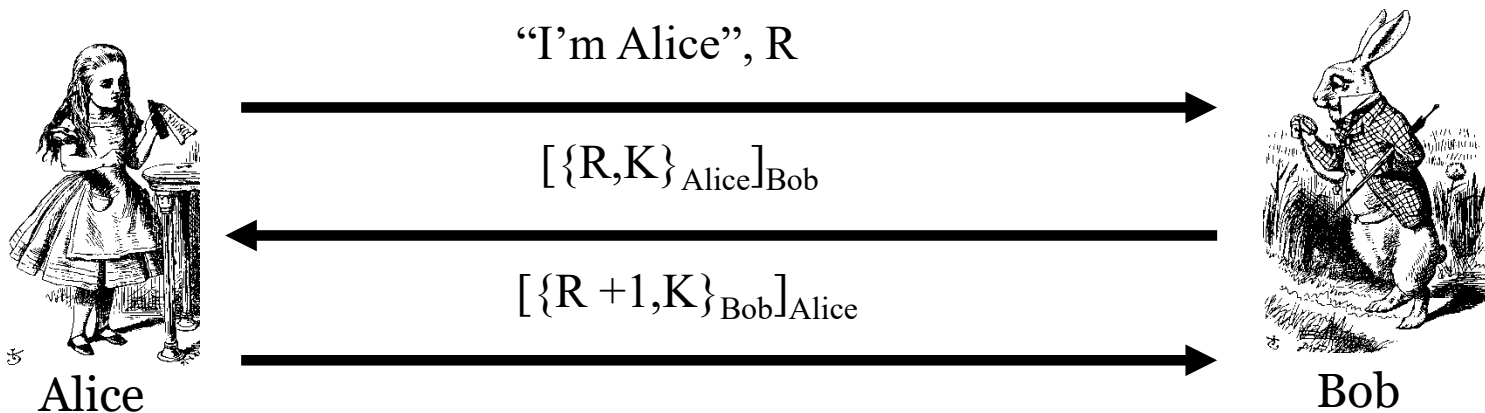
- Is this secure?
 - Mutual authentication (good), but...
 - ... session key is not secret (very bad)

Public Key Authentication and Session Key



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

Public Key Authentication and Session Key

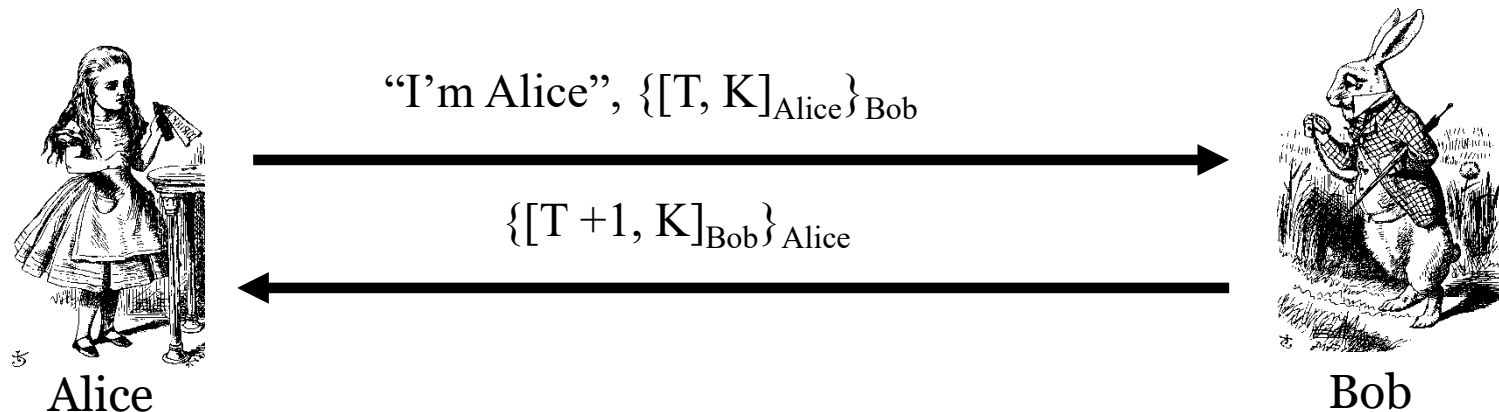


- Is this secure?
- Seems to be OK
 - Anyone can see $\{R, K\}_{\text{Alice}}$ and $\{R + 1, K\}_{\text{Bob}}$

Timestamps

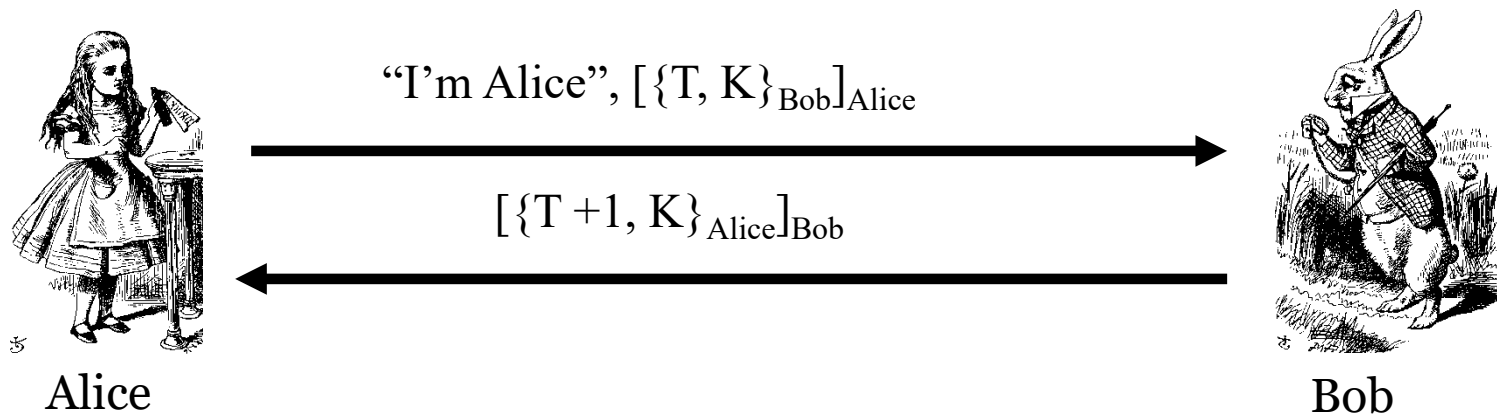
- A timestamp T is derived from current time
- Timestamps used in some security protocols
 - Kerberos, for example
- Timestamps reduce number of msgs (good)
 - Like a nonce that both sides know in advance
- “Time” is a security-critical parameter (bad)
- Clocks never exactly the same, so must allow for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

Public Key Authentication with Timestamp T



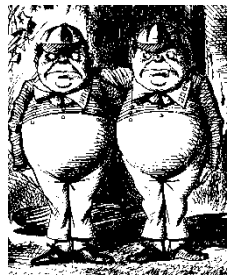
- ☐ Secure mutual authentication?
- ☐ Session key?
- ☐ Seems to be OK

Public Key Authentication with Timestamp T



- ❑ 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



Trudy

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



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



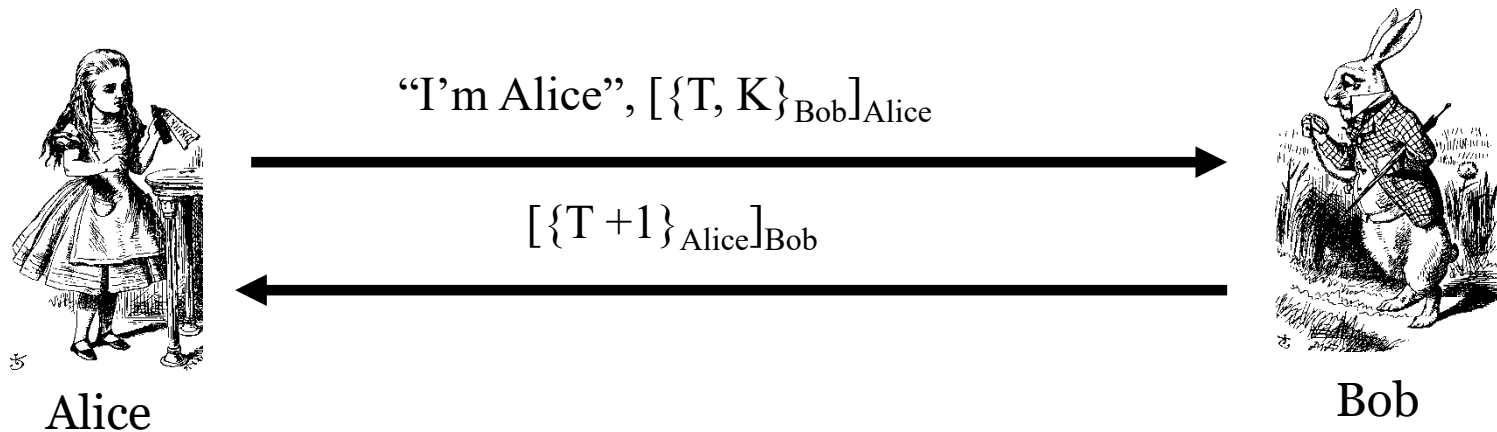
Bob

- ❑ 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



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

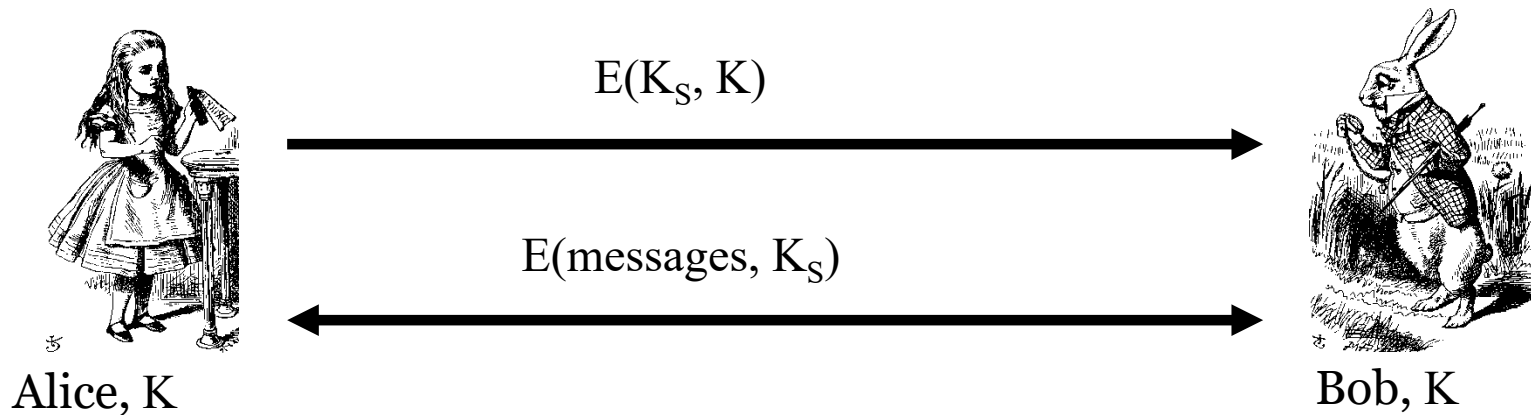
Perfect Forward Secrecy

- 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?

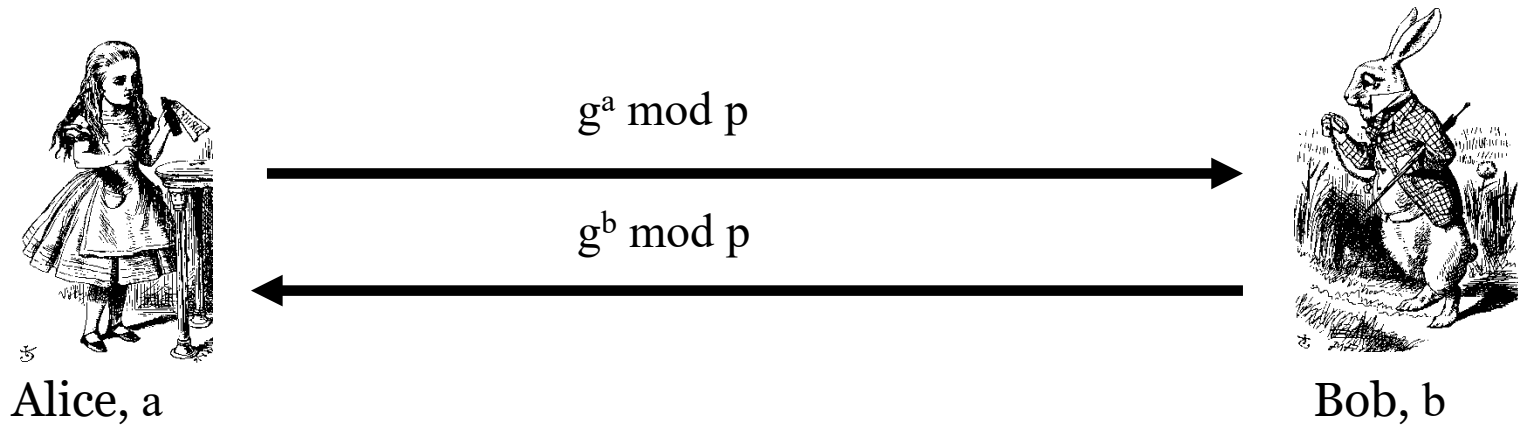
Naïve Session Key Protocol



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

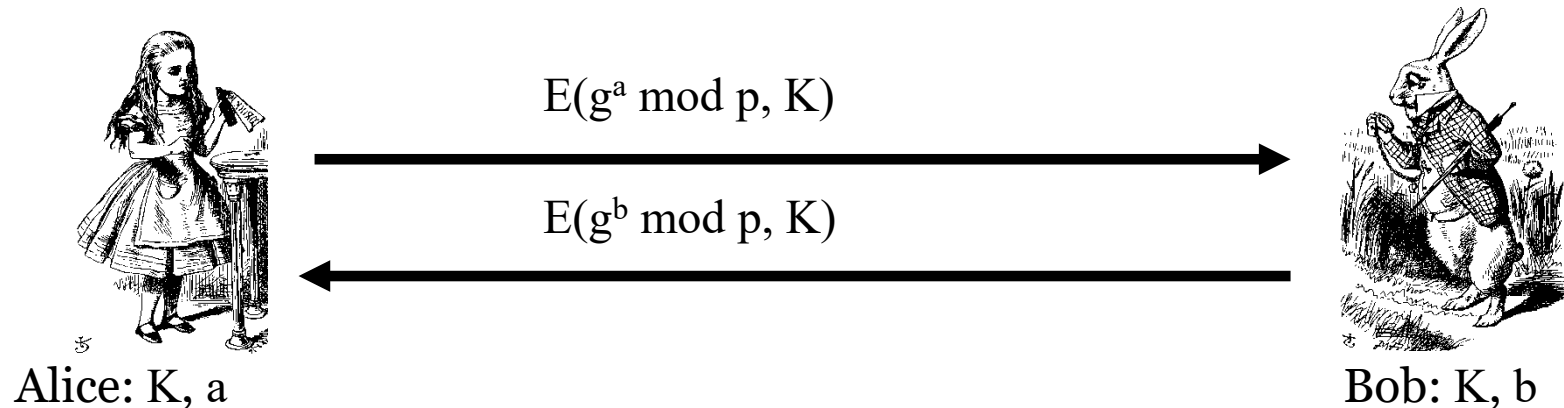
Perfect Forward Secrecy

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



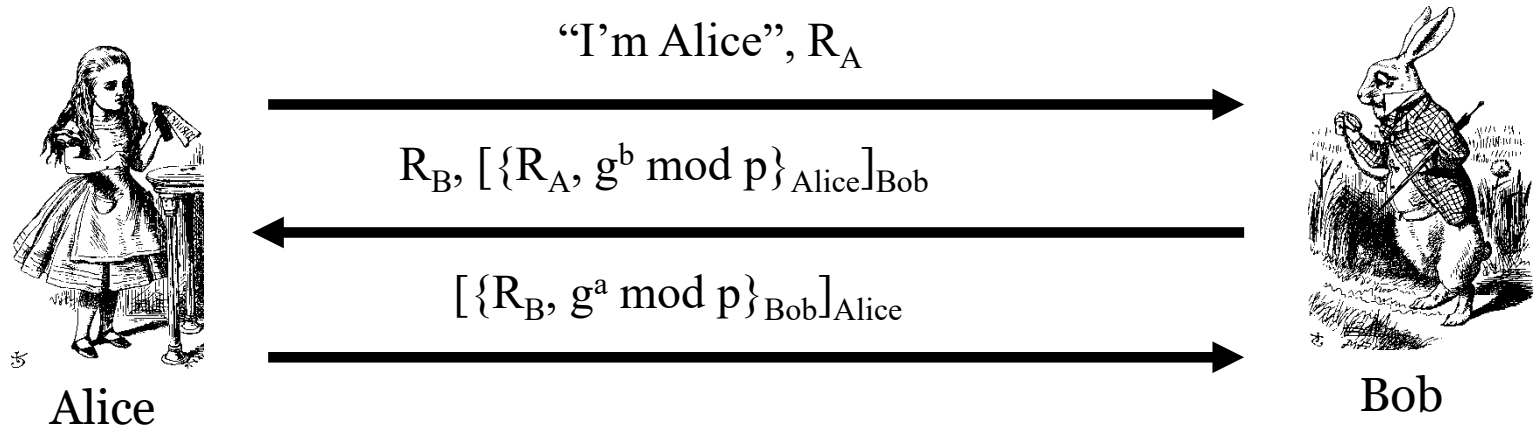
- ❑ But Diffie-Hellman is subject to MiM
- ❑ How to get PFS and prevent MiM?

Perfect Forward Secrecy



- 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



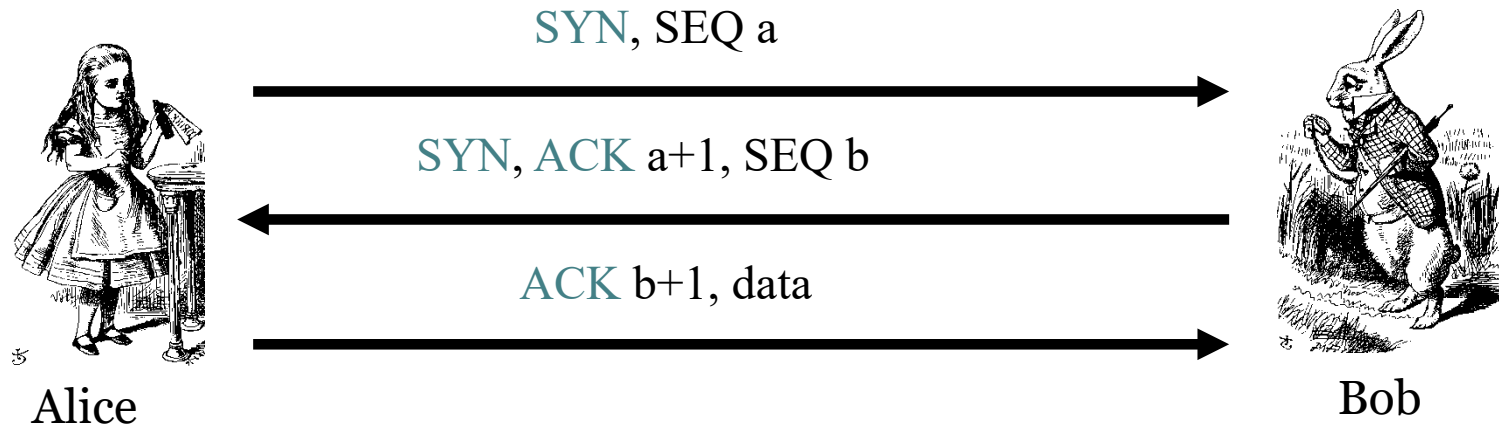
- ❑ Session key is $K = g^{ab} \bmod p$
- ❑ Alice forgets a and Bob forgets b
- ❑ If Trudy later gets Bob's and Alice's secrets, she cannot recover session key K

Authentication and TCP

TCP-based Authentication

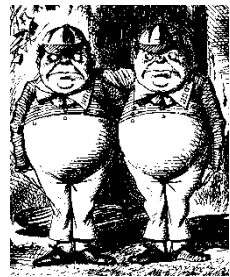
- TCP not intended for use as an authentication protocol
- But IP address in TCP connection often used for authentication
- One mode of IPSec uses IP address for authentication
- This can cause problems

TCP 3-way Handshake



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

TCP Authentication Attack



Trudy

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



Bob

5. 5. 5. 5.

Alice

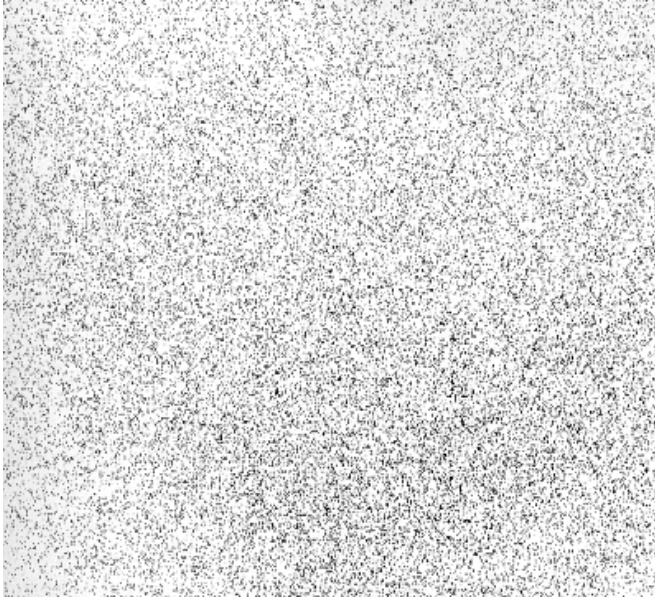


4. SYN, ACK = $t+1$, SEQ = b_2

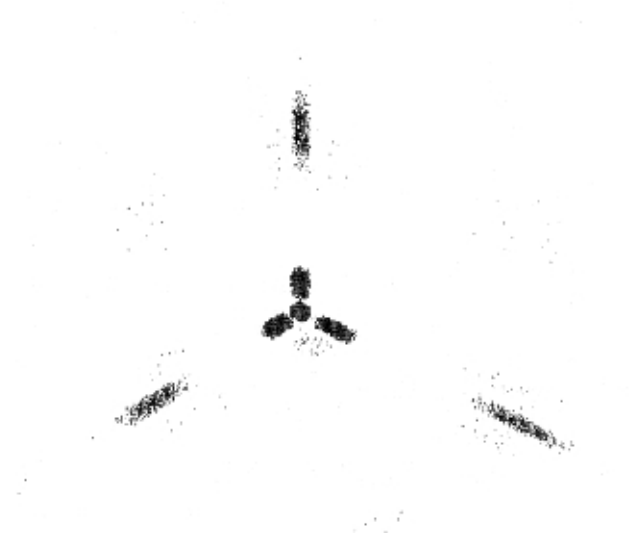
TCP Authentication Attack

- 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

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

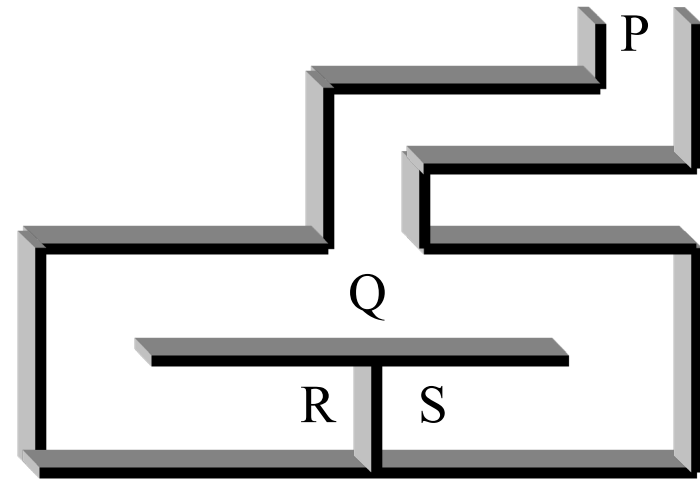
Zero Knowledge Proofs

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”

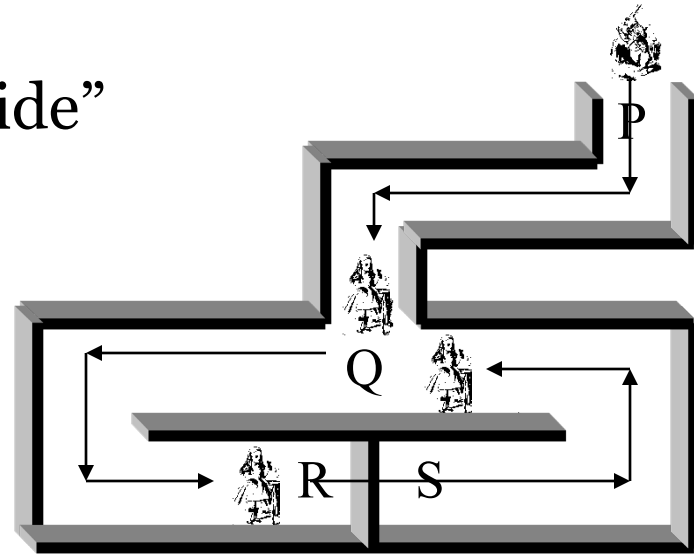
Bob's Cave

- 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 $1/2^n$



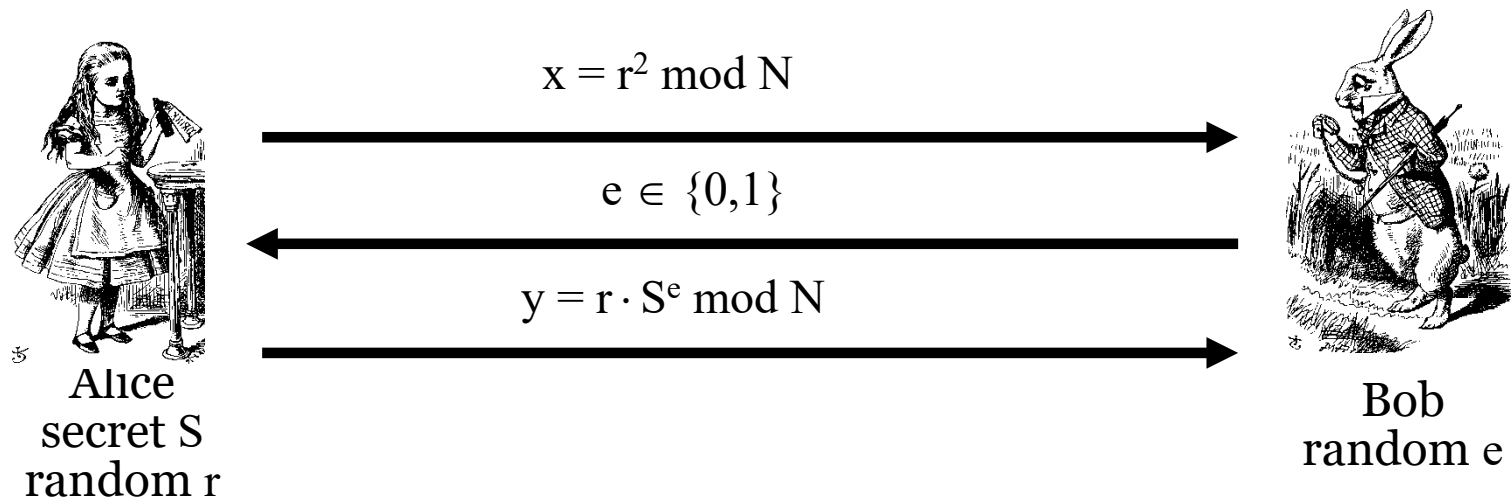
Fiat-Shamir Protocol

- 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

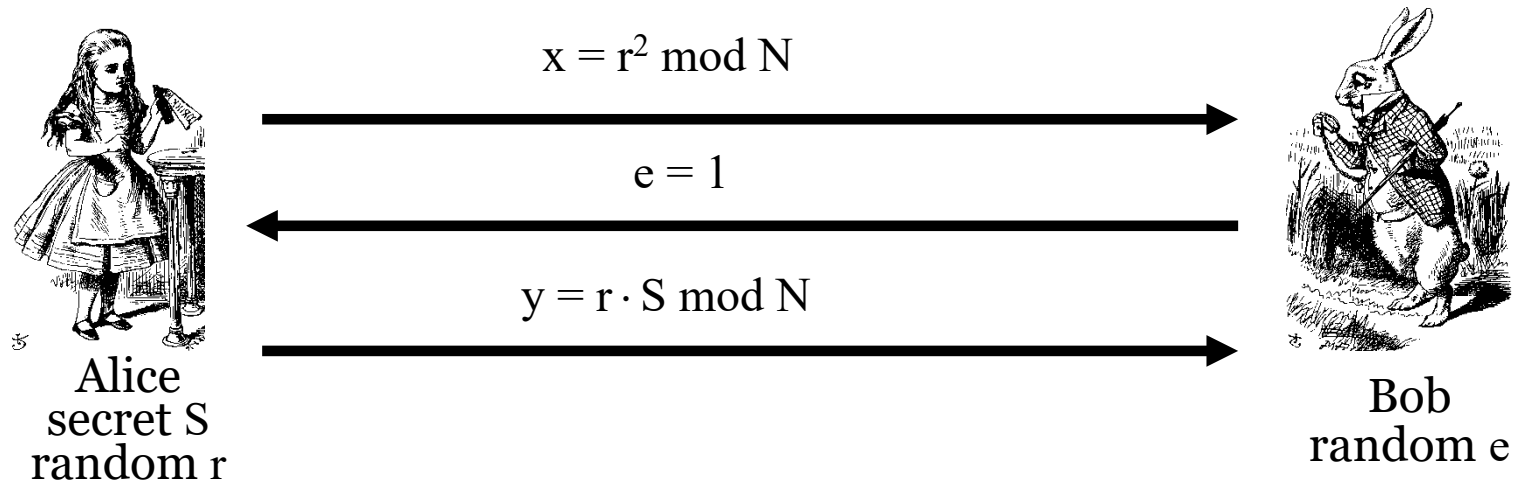
- **Public:** modulus N and $v = S^2 \bmod N$
- **Secret:** Alice knows S
- Alice selects random r and **commits** to r by sending $x = r^2 \bmod N$ to Bob
- Bob sends **challenge** $e \in \{0,1\}$ to Alice
- Alice **responds** with $y = r \cdot S^e \bmod N$
- Bob checks whether $y^2 = x \cdot v^e \bmod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

Fiat-Shamir



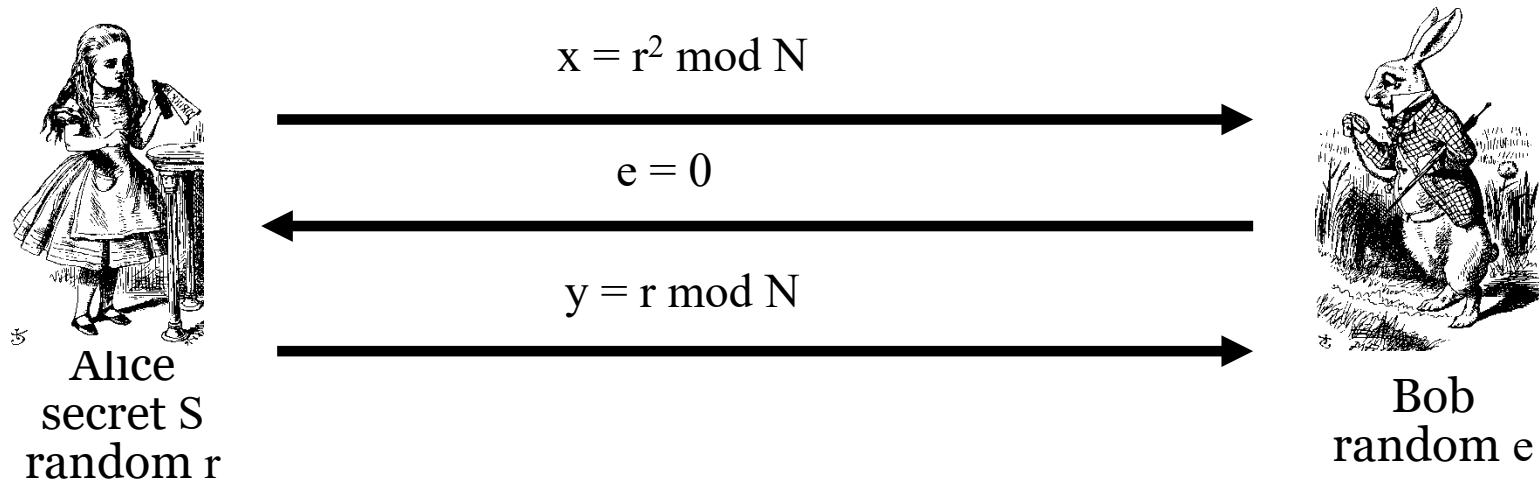
- **Public:** Modulus N and $v = S^2 \bmod N$
- Alice selects random r , Bob chooses $e \in \{0,1\}$
- Bob must verify: $y^2 = x \cdot v^e \bmod N$
 - Why? Because... $y^2 = r^2 \cdot S^{2e} = r^2 \cdot (S^2)^e = x \cdot v^e \bmod N$

Fiat-Shamir: $e = 1$



- **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: $e = 0$



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

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