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

Protocol

The rules followed in some particular interaction.

Secure Entry into 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) Protocol

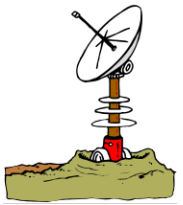
Soviet
MIG

Angola

South African Air Force
Impala
K

2. E(N,K)

1. N



Namibia
K

Fighting in Angola in the mid-1970s

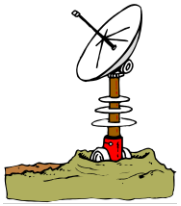
MIG-in-the-middle attack

South African Air Force
Impala
K

3. N

4. E(N,K)

2. N



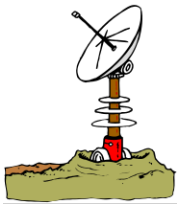
Angola
Cuban radar station

5. E(N,K)

Soviet
MiG aircraft

6. E(N,K)

1. N



Namibia
K
SAAF radar station

A security protocol failure

Authentication Protocols

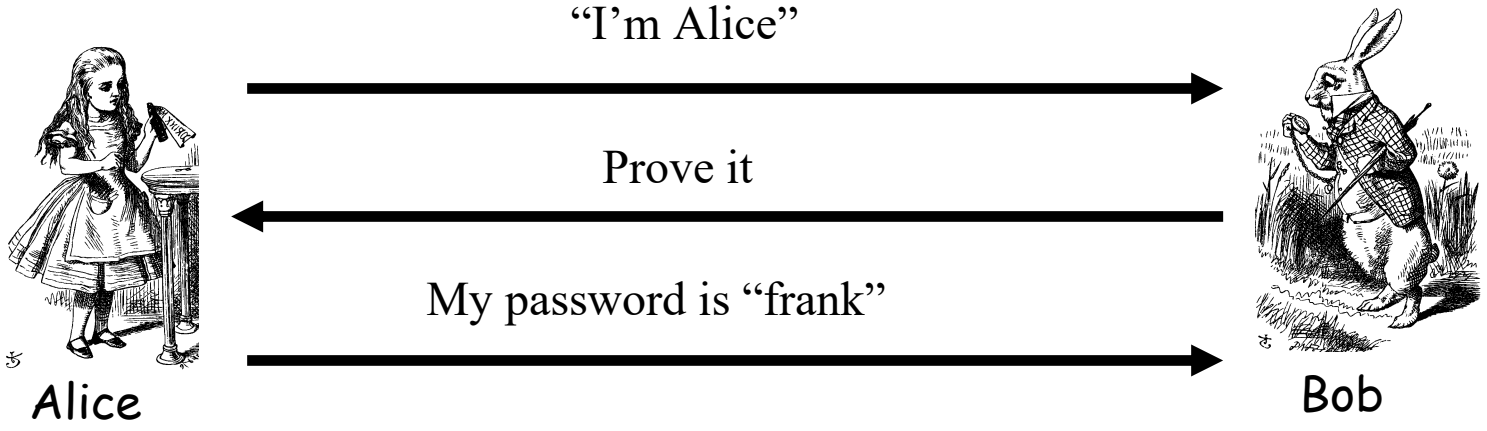
Authentication(认证)

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

Authentication

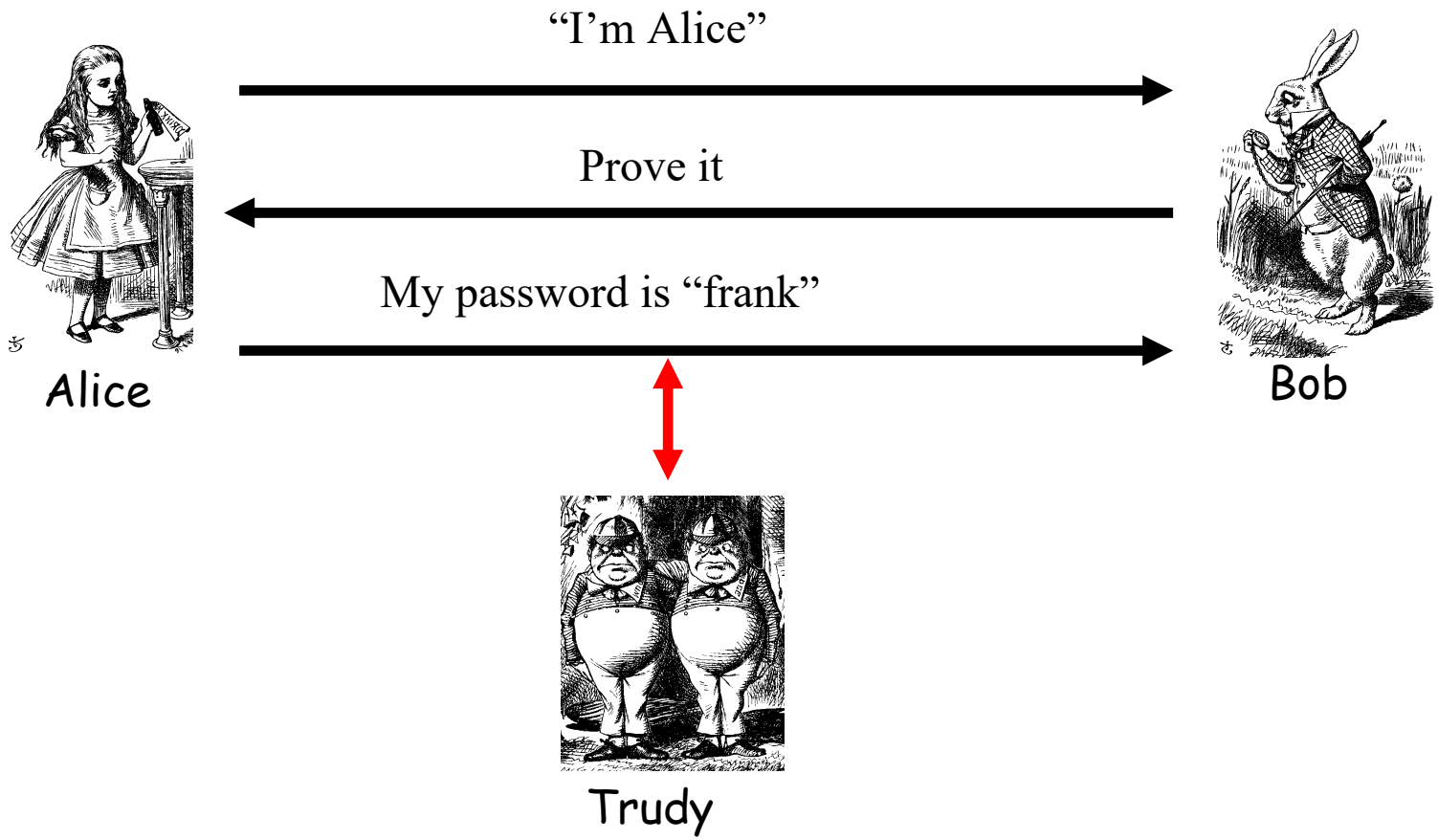
- Authentication on a **stand-alone computer** is relatively simple
  - For example, hash a password with a salt
  - "Secure path," attacks on authentication software, keystroke logging, etc., can be issues
- Authentication over a network is challenging
  - Attacker can passively observe messages
  -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

Simple Authentication

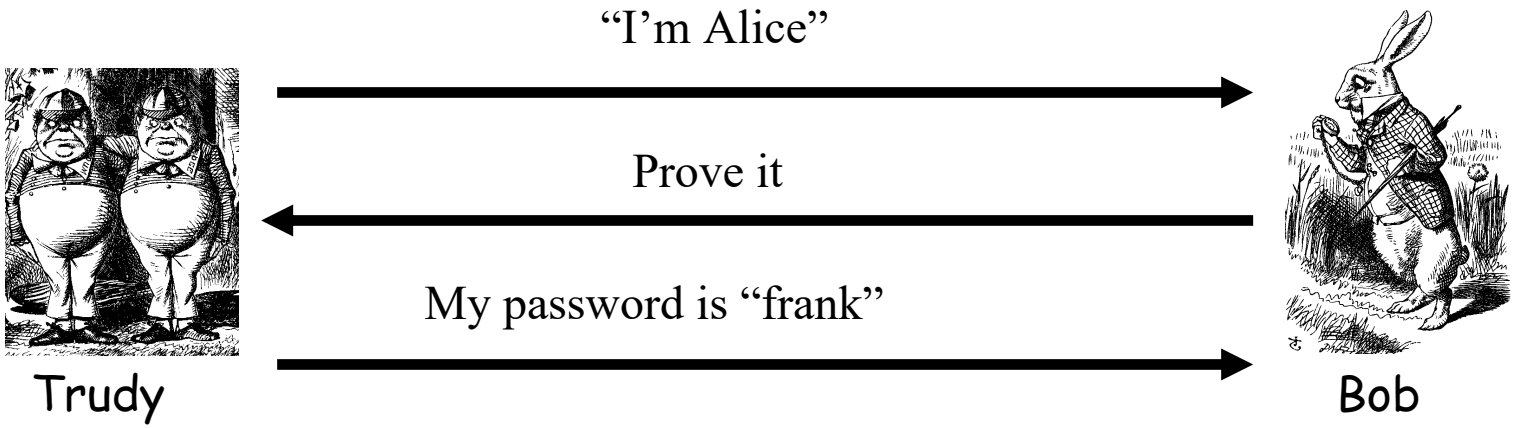


- Simple and may be OK for standalone system
- But highly insecure for networked system
  - Subject to a **replay attack**
  - Also, Bob must know Alice's password

Authentication Attack



Authentication Attack



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

Simple Authentication

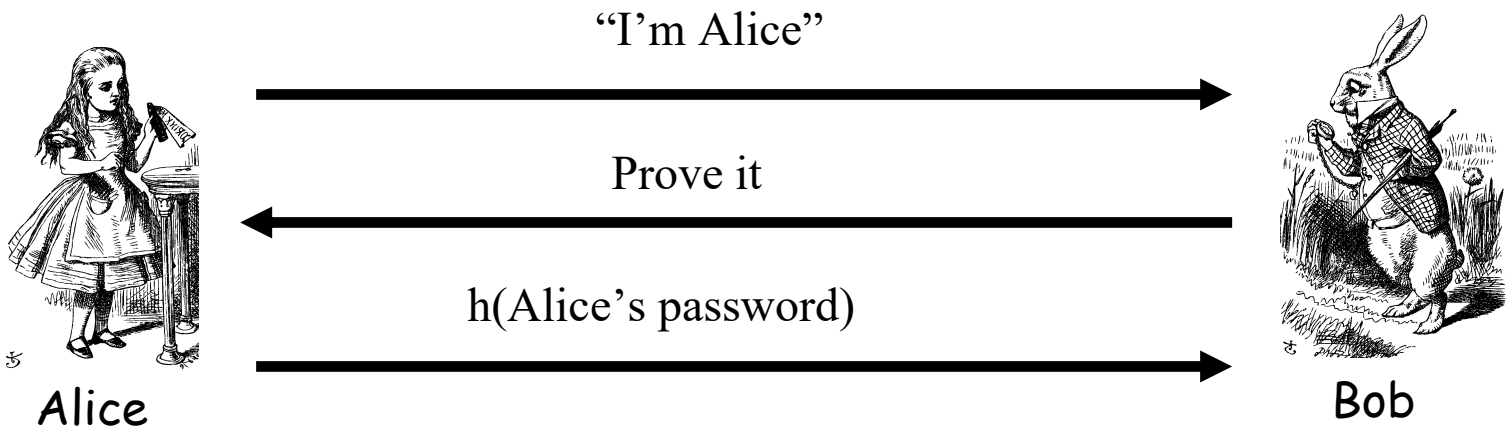
Alice

I'm Alice, my password is "frank"

Bob

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

Better Authentication



- This approach hides Alice's password
  - From both Bob and Trudy
- But still subject to **replay attack**

No longer needs to know Alice's password

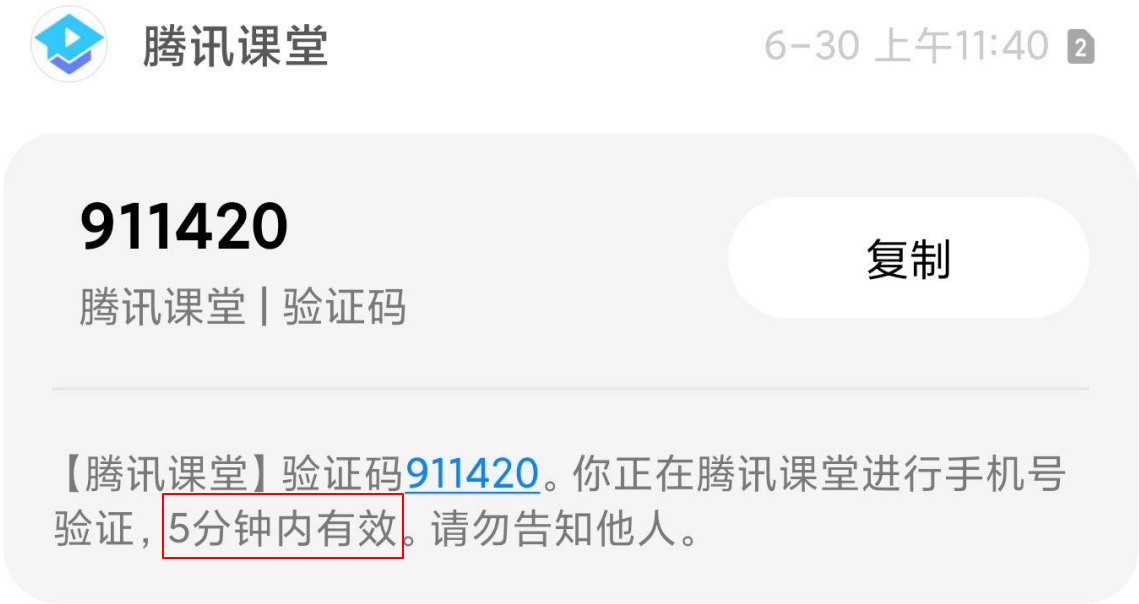
Challenge-Response Mechanism

- 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

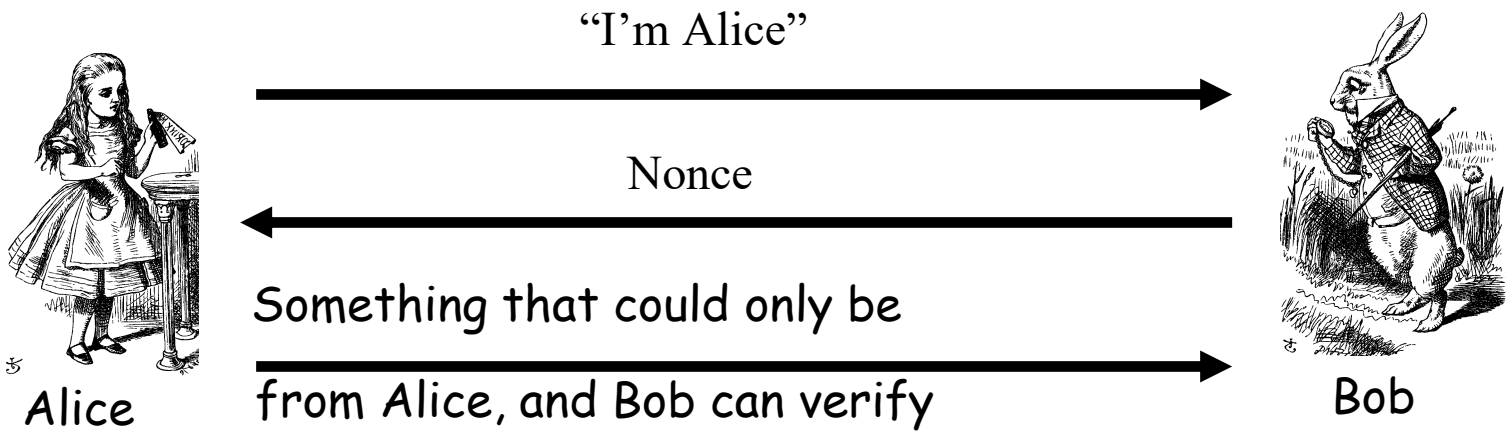
Nonce

- 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 use passwords or keys?

短信验证码

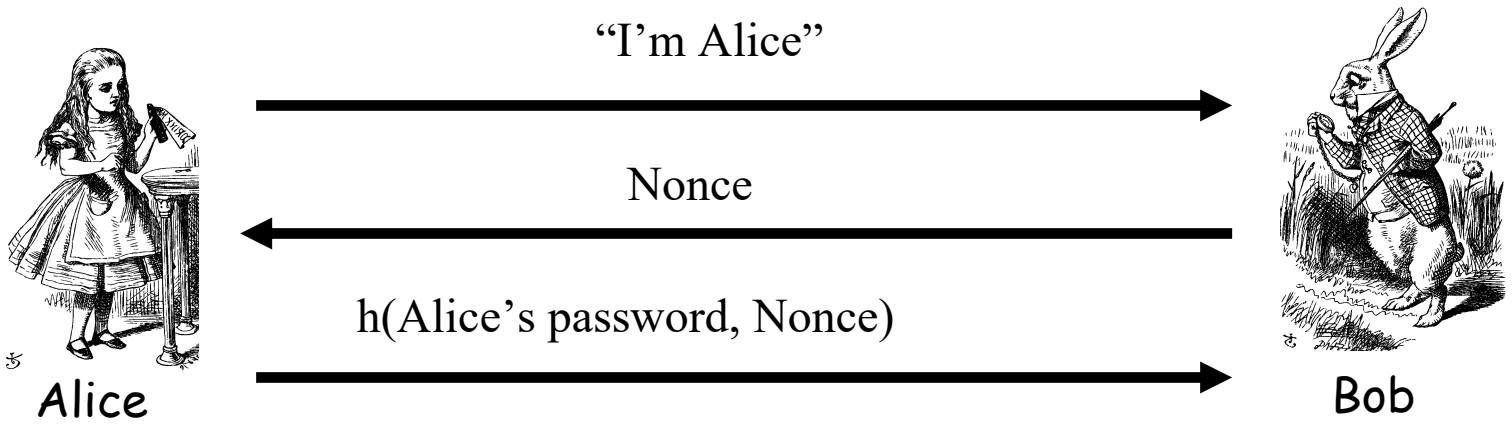

腾讯课堂
6-30 上午11:40
911420
腾讯课堂 | 验证码
复制
【腾讯课堂】验证码911420。你正在腾讯课堂进行手机号验证，5分钟内有效。请勿告知他人。

Generic Challenge-Response Mechanism



- In practice, how to achieve this?
- Hashed password works, but…
- …encryption is much better here (why?)

Challenge-Response



- 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

Symmetric Key Notation

- Encrypt plaintext P with key K

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

- Decrypt ciphertext C with key K

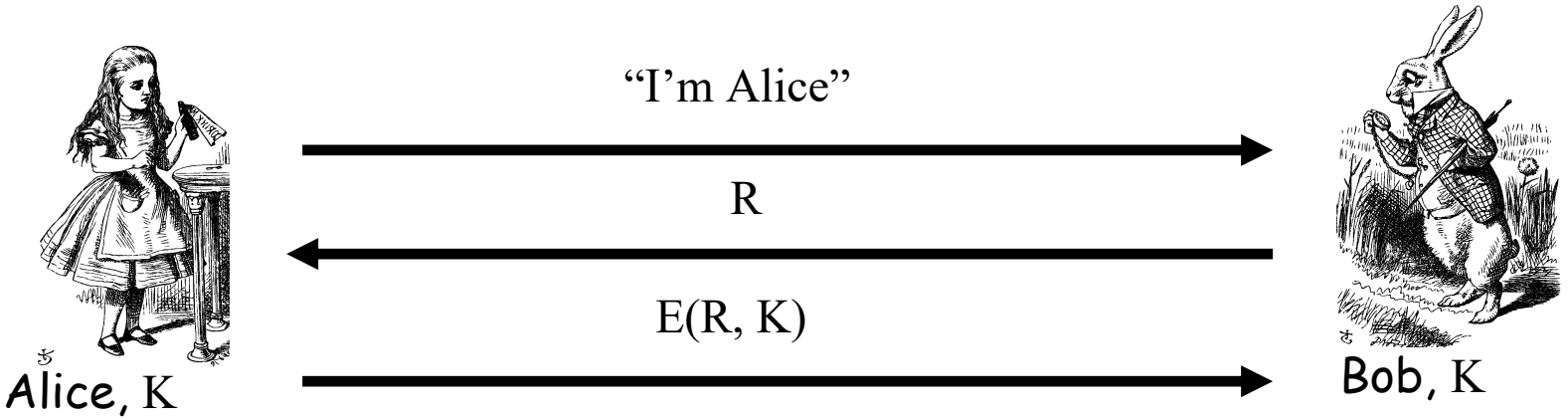
$$P = D(C, K)$$

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

Authentication: Symmetric Key

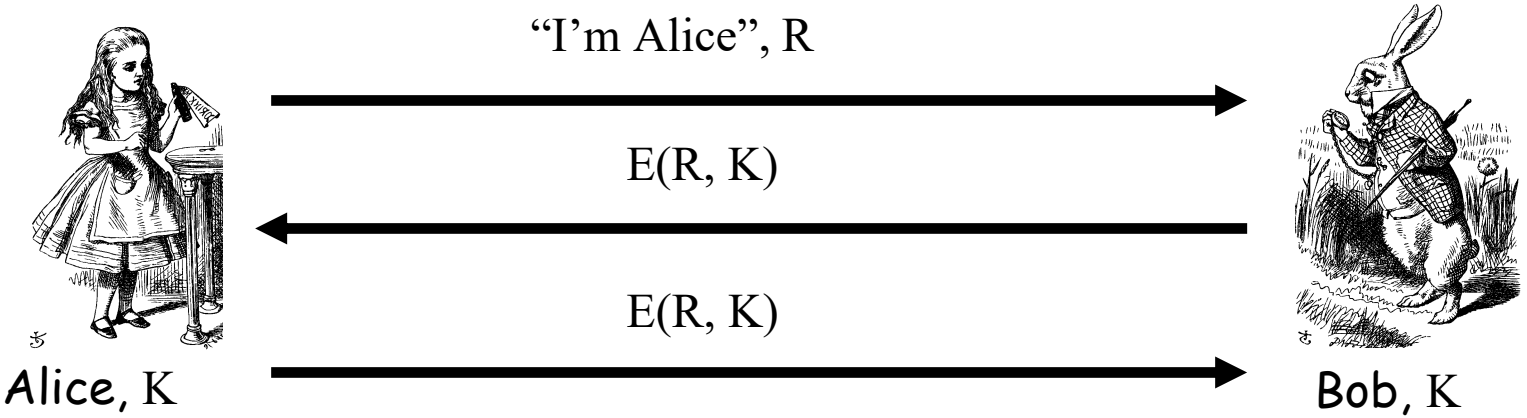
- Alice and Bob share symmetric key K
- Key K known only to Alice and Bob
- Authenticate by proving knowledge of shared symmetric key
- How to accomplish this?
  - Cannot reveal key, must not allow replay (or other) attack, must be verifiable, …

Authenticate Alice Using Symmetric Key



- Secure method for Bob to authenticate Alice
- But, Alice does not authenticate Bob
- So, can we achieve **mutual** authentication? No.

Mutual Authentication?

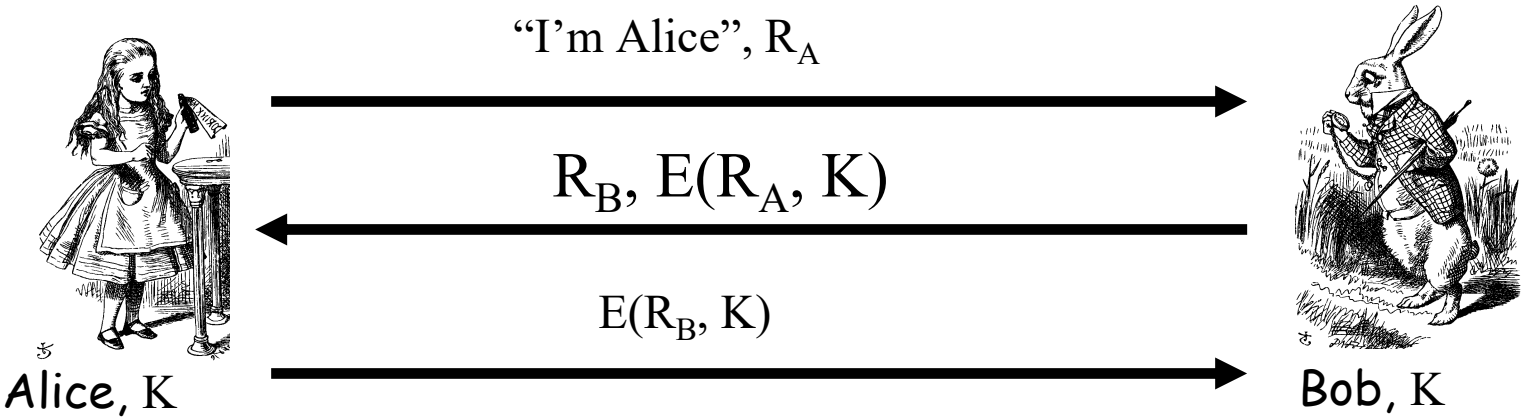


- What's wrong with this picture?
- "Alice" could be Trudy (or anybody else)!

Mutual Authentication

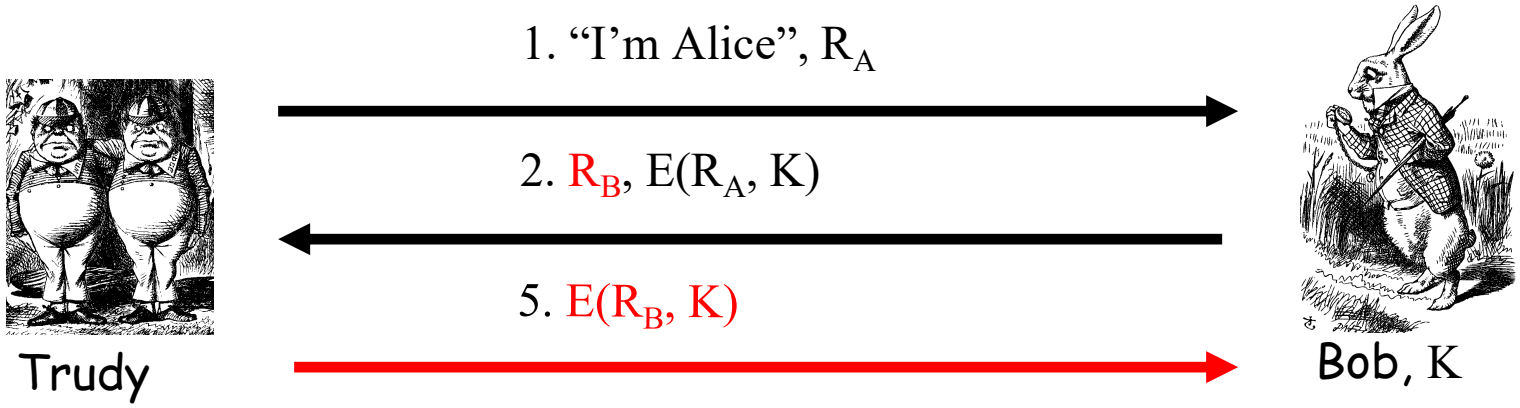
- Since we have a secure one-way authentication protocol…
- The obvious thing to do is to use the protocol twice
  - Once for Bob to authenticate Alice
  - Once for Alice to authenticate Bob
- This has got to work…

Secure Mutual Authentication?

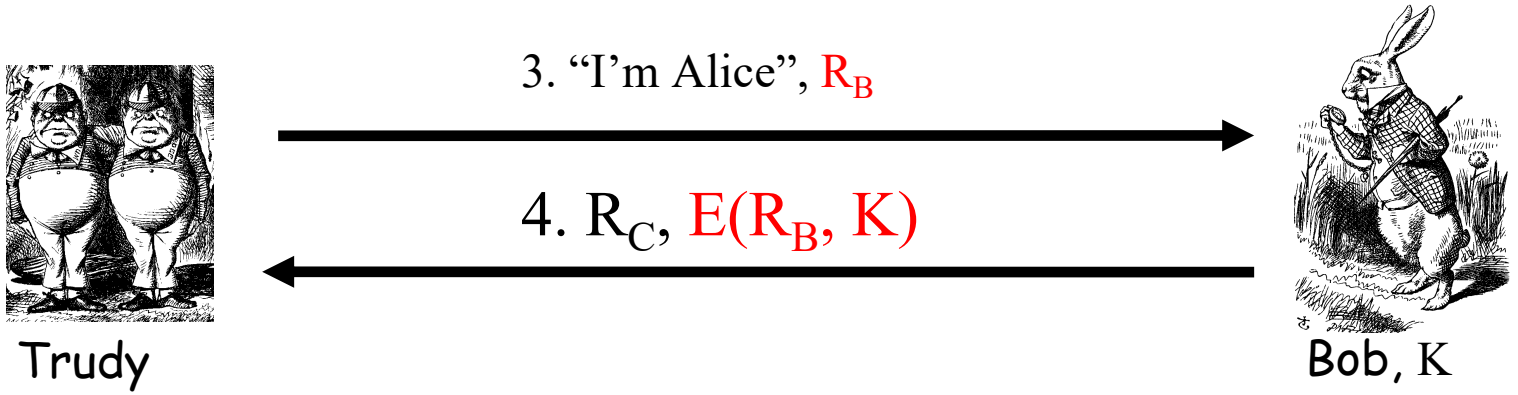


- This provides mutual authentication…
- …or does it? Subject to **reflection attack**

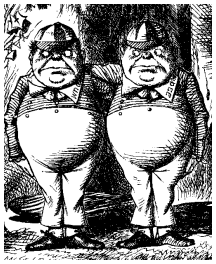
Analogue to MiG-in-the-middle ATTACK



3. Trudy cleverly opens a new connection to Bob where she again claims to be Alice and this time sends Bob his own "random" challenge R_B



Mutual Authentication Attack



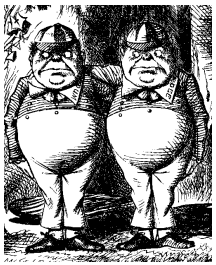
Trudy

1. "I'm Alice", R_A

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

5. $E(R_B, K)$

Bob, K



Trudy

3. "I'm Bob", R_B

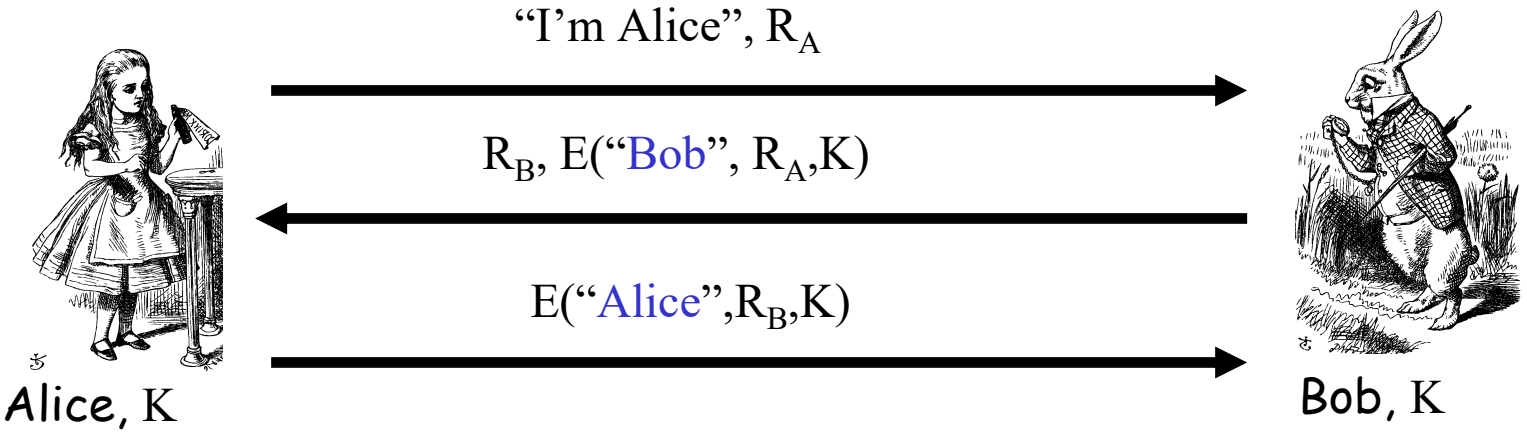
4. R_C, $E(R_B, K)$

Alice, K

Mutual Authentication Attack

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

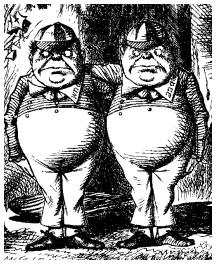
Strong Mutual Authentication Protocol



- Do these "insignificant" changes help?
- Yes! Trudy cannot use a response from Bob, E("Bob", R_A,K) , for message 5.
- Trudy may still can attack use more complicated way.

Encrypt users' identity. It's a bad idea to have the two sides in a protocol do the same thing.

Man-in-the-middle attacks (MIMA)



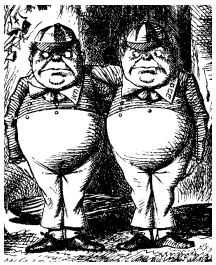
Trudy

1. "I'm Alice", R_A

2. R_B, E("Bob", R_A, K)

5. E("Alice", R_B, K)

Bob, K



Trudy

3. "I'm Bob", R_B

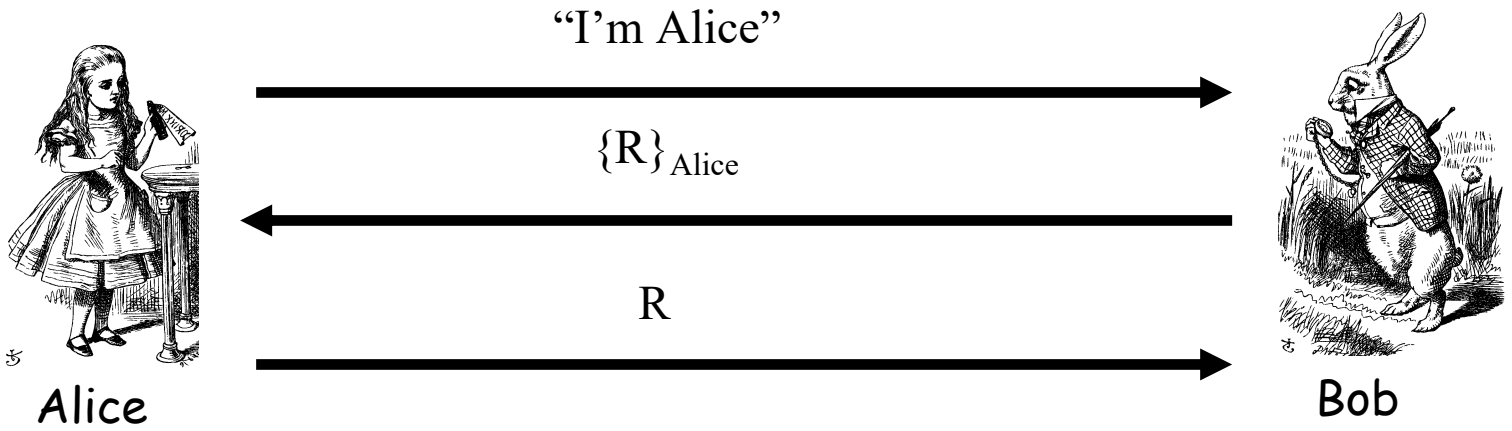
4. R_C, E("Alice", R_B, K)

Alice, K

Public Key Notation

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

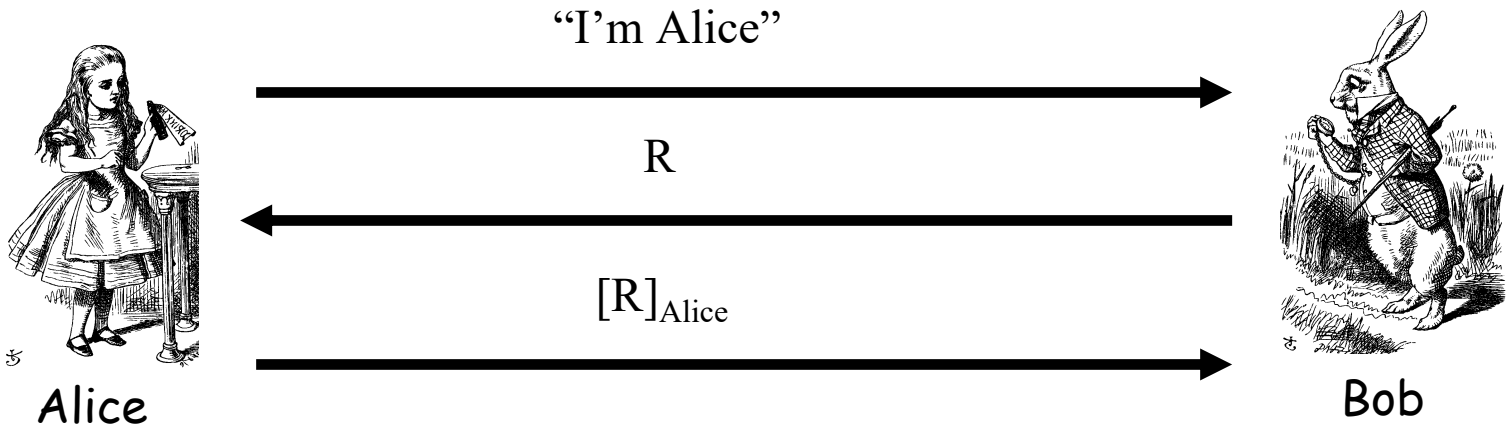
Public Key Authentication



- Is this secure?
- Trudy can get Alice to decrypt anything!

  Prevent this by having different key pairs for signing and encryption.

Public Key Authentication



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

Public Keys

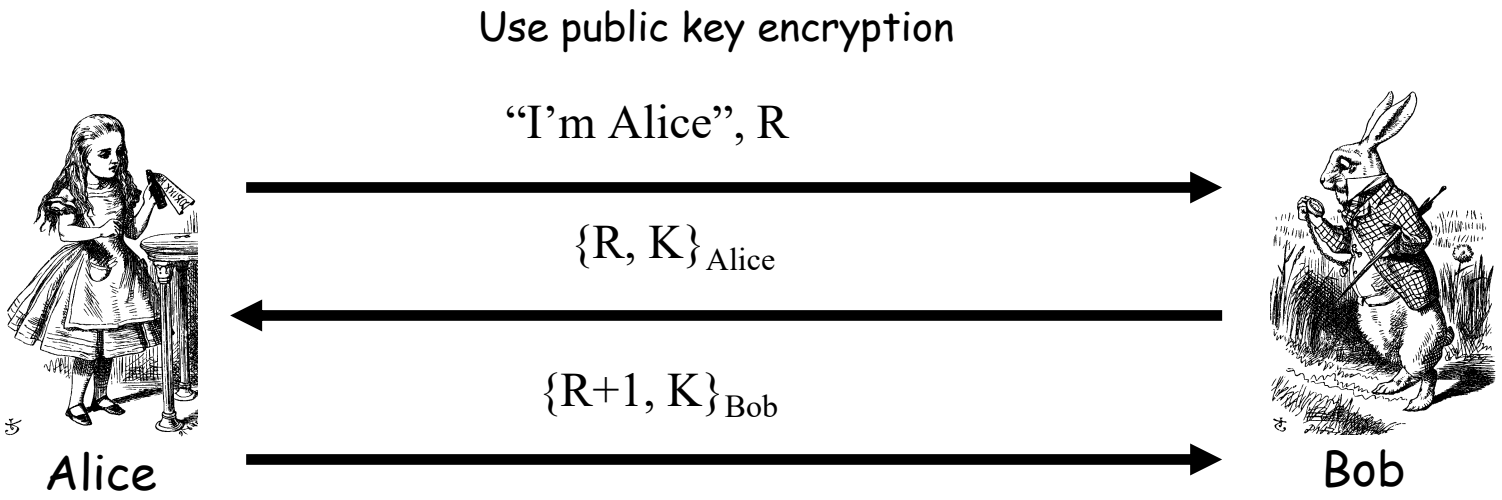
- Generally, a bad idea to use the same key pair for encryption and signing
- Instead, should have…
  - …one key pair for encryption/decryption and signing/verifying signatures…
  - …and a different key pair for authentication

Session Key

- Usually, a **session key** is required
  - A symmetric key for current session
  - Used for confidentiality and/or integrity
  - Limit the amount of data encryted with any one particular key
- How to authenticate ***and*** establish a session key (i.e., shared symmetric key)?
  - When authentication completed, Alice and Bob share a session key
  - Trudy cannot break the authentication…
  - …***and*** Trudy cannot determine the session key

Establish the session key as part of the authentication protocol

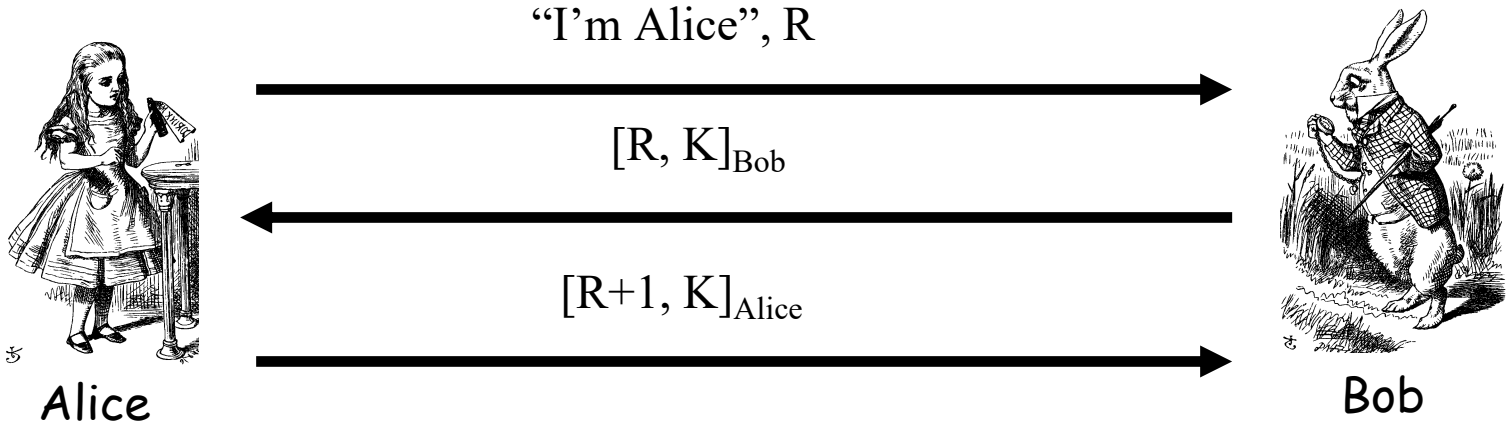
Authentication & Session Key



- 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 (only Alice is authenticated)

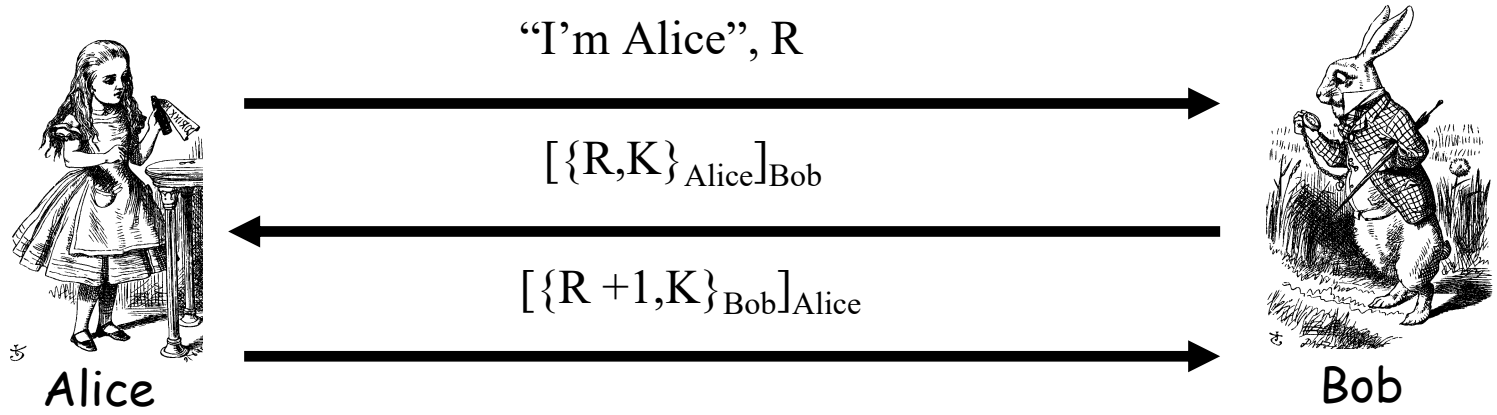
Public Key Authentication and Session Key

Use digital signatures



- Is this secure?
  - Does provide mutual authentication (good), but…
  - … session key is not protected (very bad)

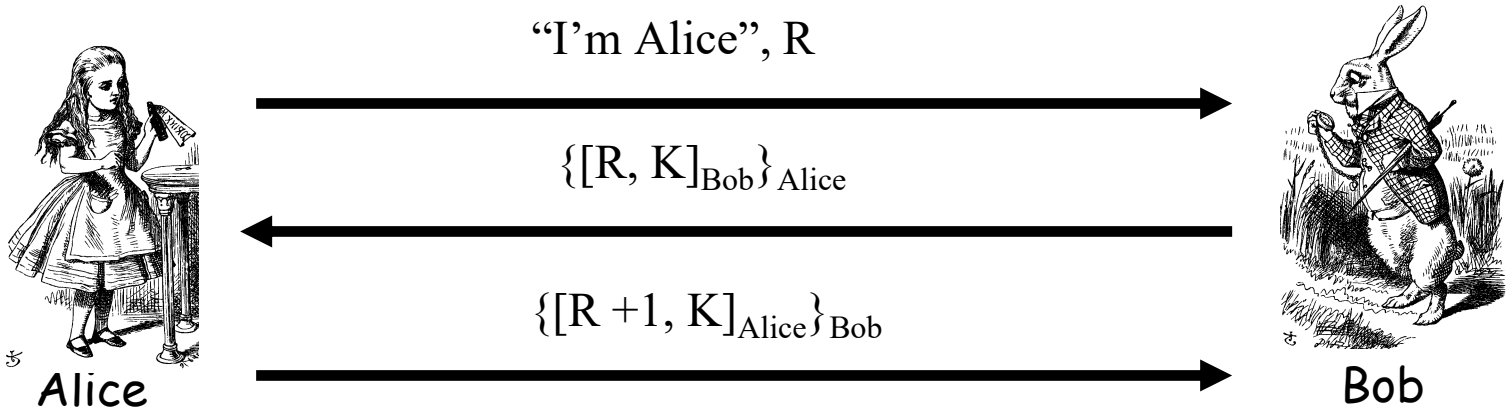
Encrypt and Sign Mutual Authentication



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

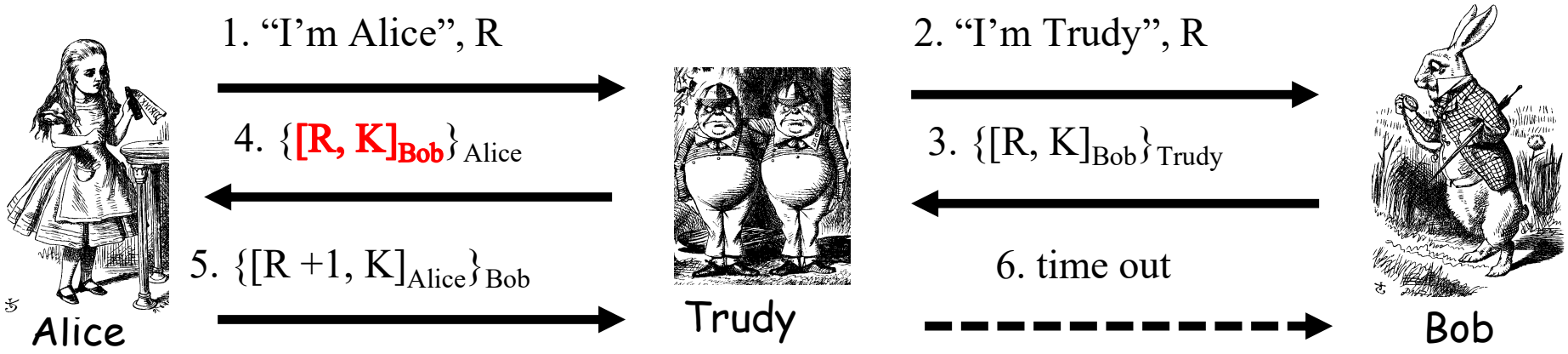
Public Key Authentication and Session Key

Sign and then encrypt



- Is this secure?
- No! It's subject to subtle MiM attack
  - See the next slide…

Public Key Authentication and Session Key



- Trudy can get $[R, K]_{Bob}$ and then K from 3.
- Alice uses this same key K
- Alice thinks she's talking to Bob

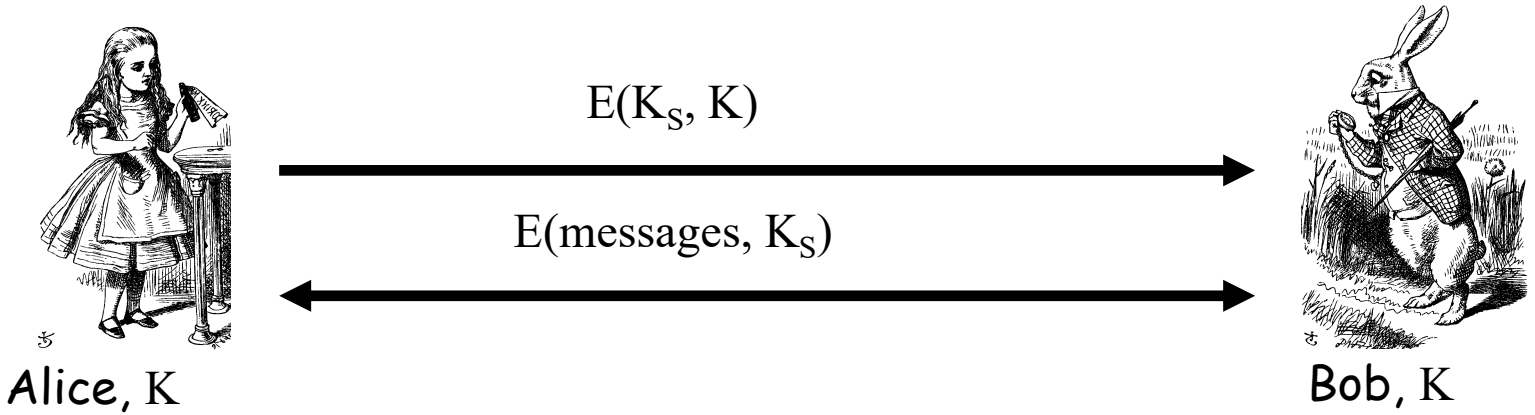
Perfect Forward Secrecy

- 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 provides PFS?

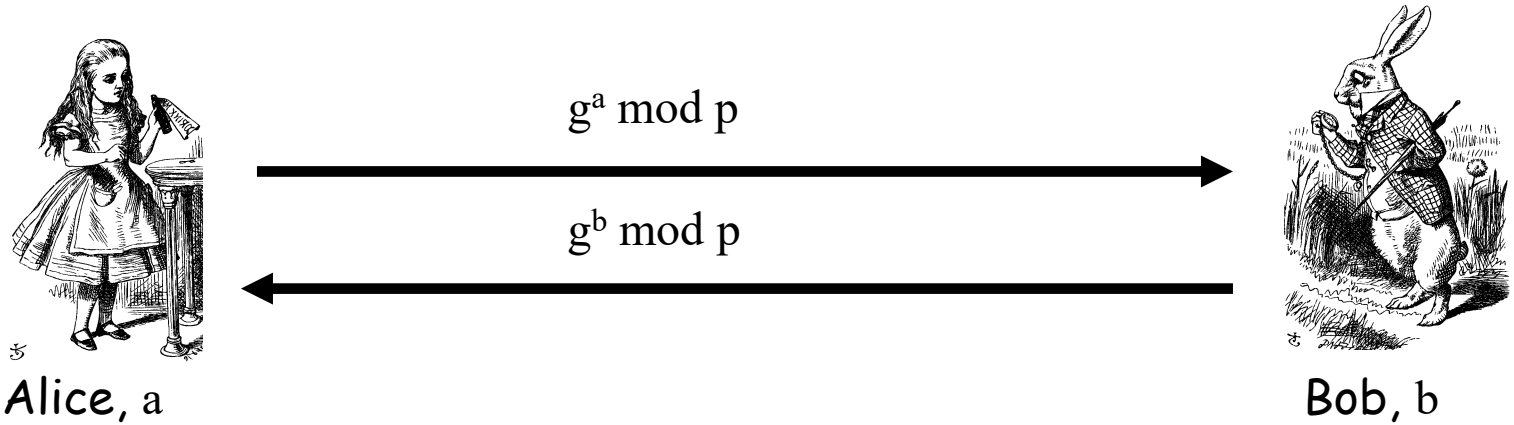
Naïve Session Key Protocol



- Trudy could record $E(K_S, K)$
- If Trudy later gets K then she can get K_S
  - Then Trudy can decrypt recorded messages
- ***No*** perfect forward secrecy in this case

Perfect Forward Secrecy

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

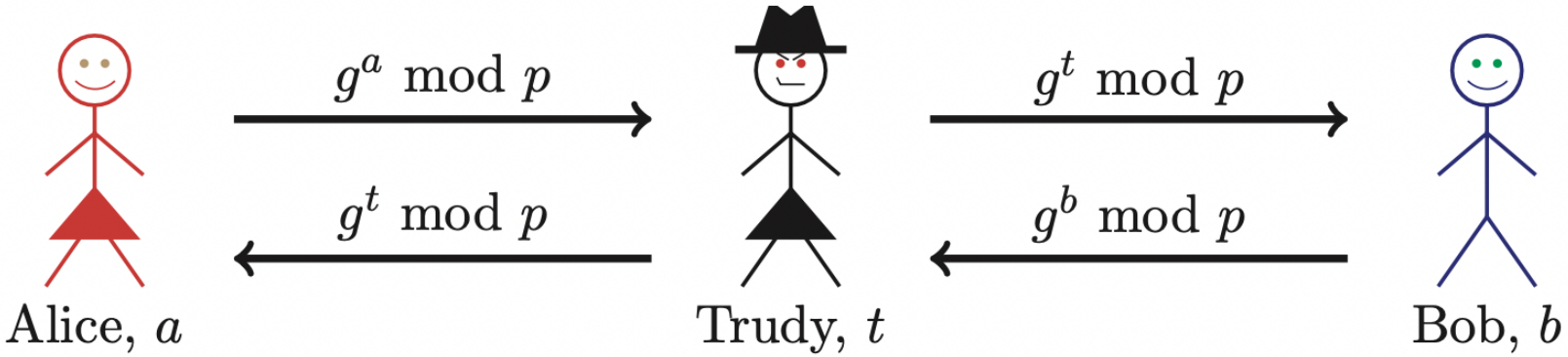


- But Diffie-Hellman is subject to MiM
- How to get PFS and prevent MiM?

For inventing and promulgating both asymmetric public-key cryptography, including its application to digital signatures, and a practical cryptographic key-exchange method.
https://amturing.acm.org/award_winners/diffie_8371646.cfm

MiM attack on Diffie-Hellman

- Subject to man-in-the-middle (MiM) attack



- Trudy shares secret $g^{at} \bmod p$ with Alice
- Trudy shares secret $g^{bt} \bmod p$ with Bob
- Alice and Bob don't know Trudy is MiM

Perfect Forward Secrecy

Alice: K, a

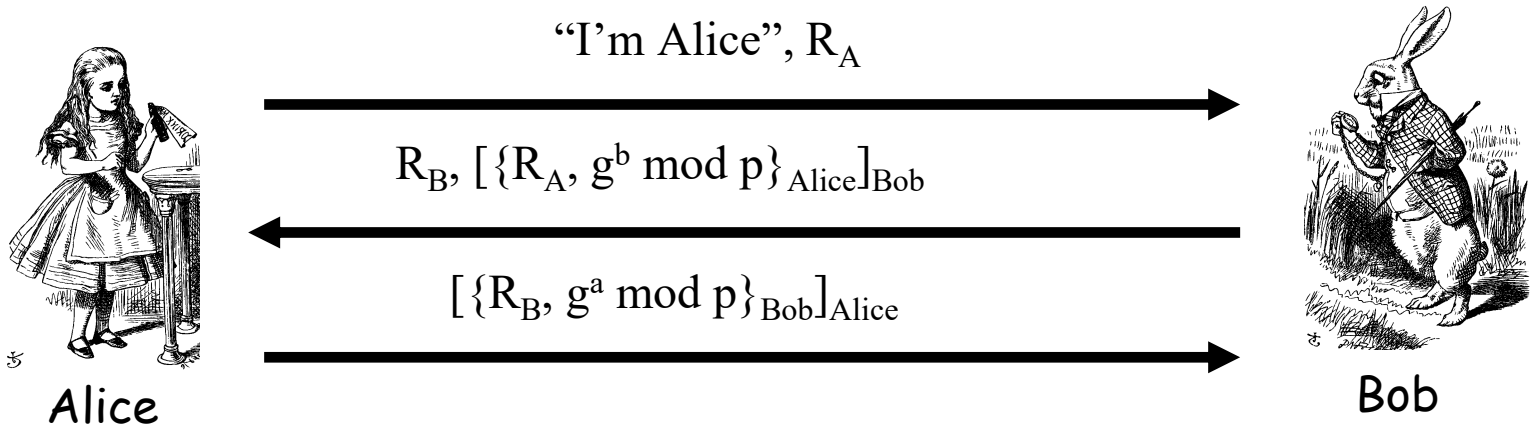
$E(g^a \bmod p, K)$

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

Bob: K, b

- Session key $K_S = g^{ab} \bmod p$
- Alice **forgets** a, Bob **forgets** b
- This is known as **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

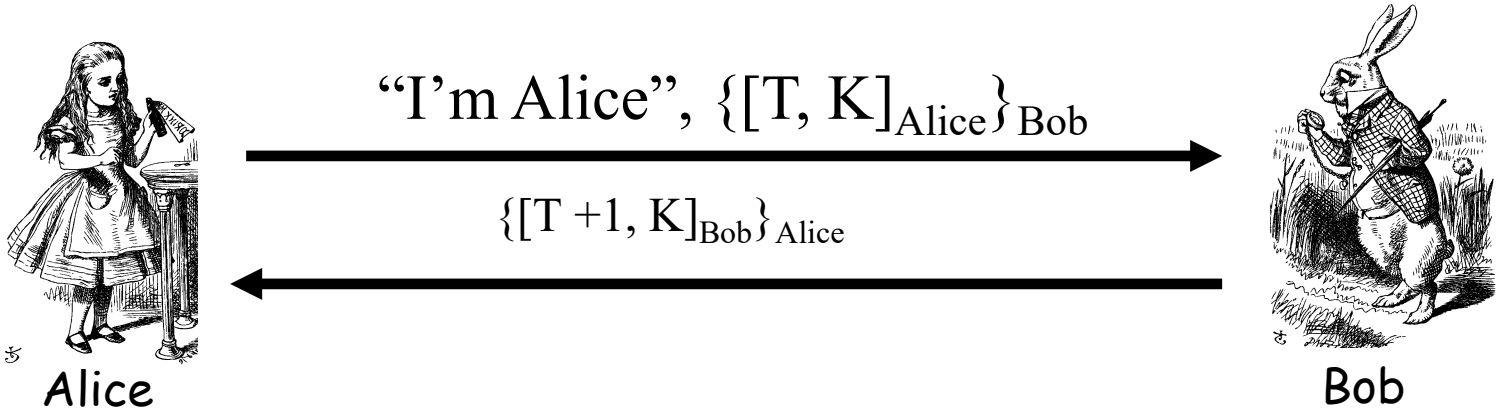


- Session key is $K = g^{ab} \bmod p$
- Alice forgets a and Bob forgets b
- If Trudy later gets Bob's and Alice's secrets, she cannot recover session key K

Timestamps

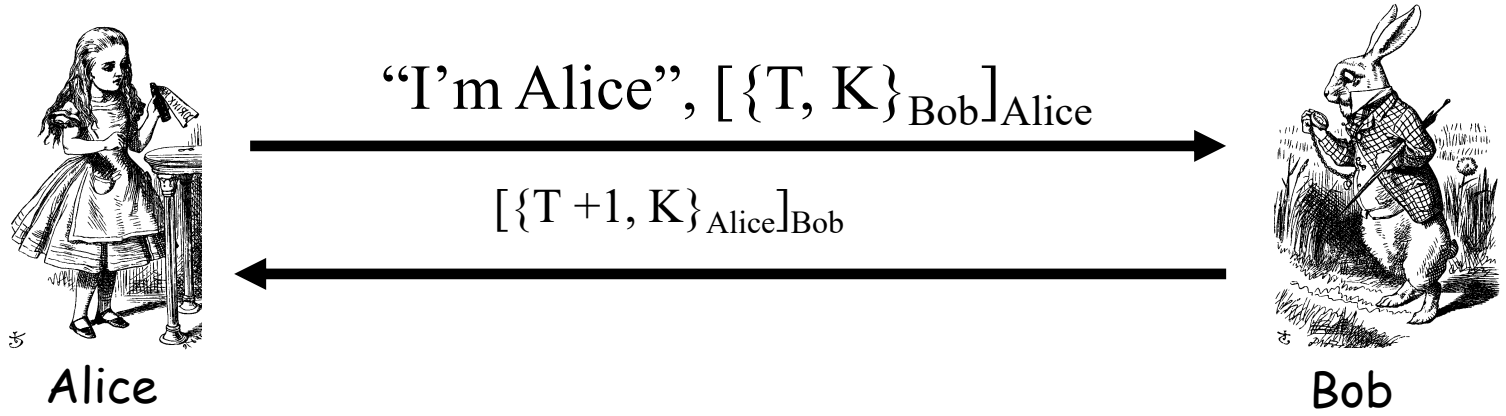
- A timestamp T is derived from current time
- Timestamps can be used to prevent replay
  - Used in Kerberos, for example
- Timestamps reduce number of msgs (good) exchanging nonces
  - A challenge that both sides know in advance
- "Time" is a security-critical parameter (bad)
  - Clocks not same and/or network delays, so must allow for **clock skew** — creates risk of replay
  - How much clock skew is enough?

Public Key Authentication with Timestamp T



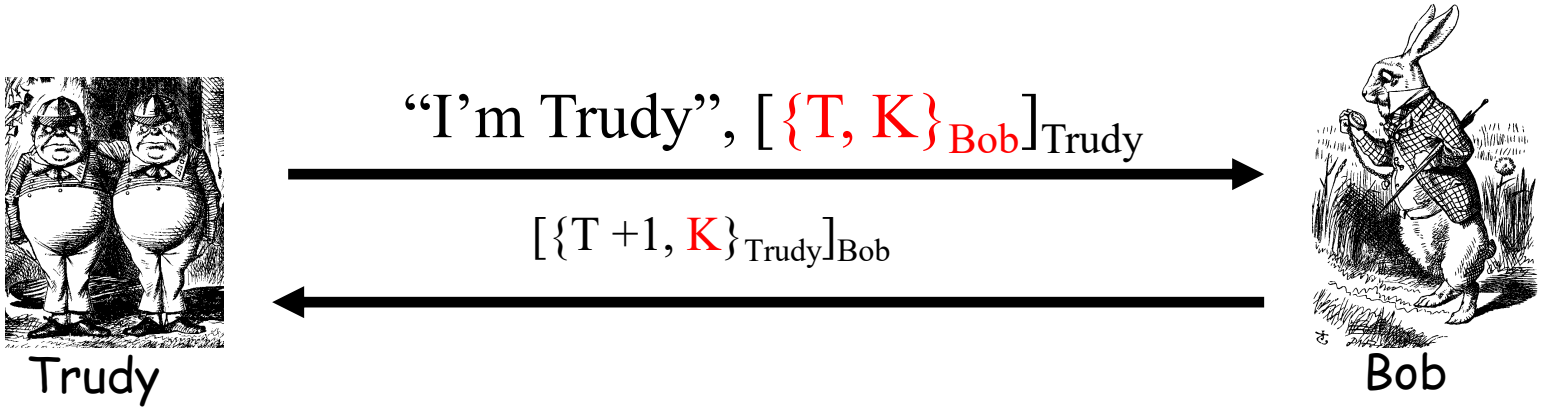
- Secure mutual authentication?
- Session key secure?
- Seems to be OK
- Reduce the number of messages by a third

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

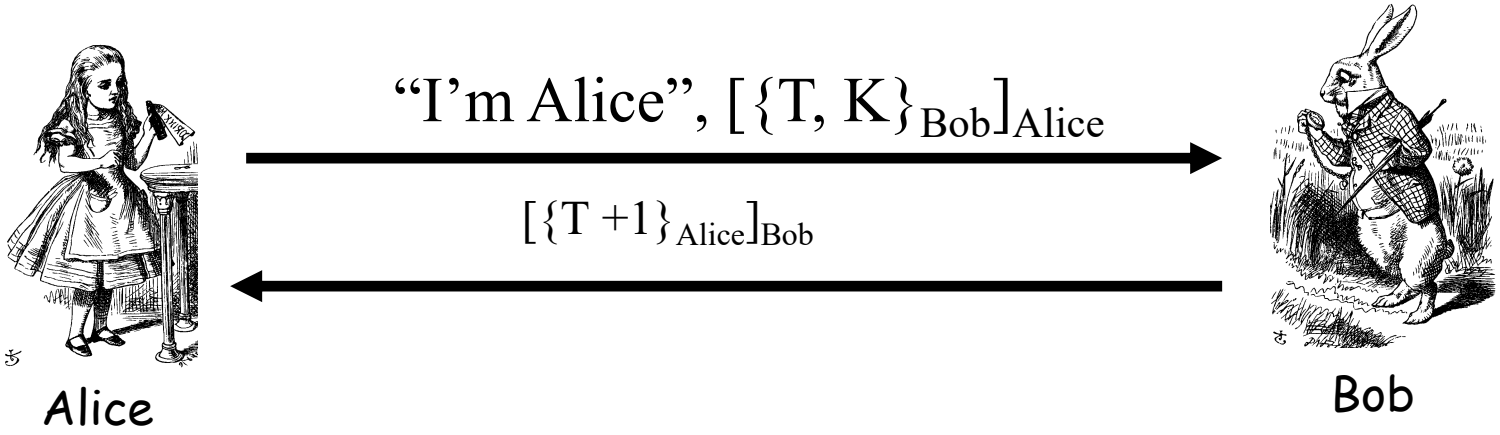


- Trudy obtains Alice-Bob session key K
- **Note:** Trudy must act within clock skew

Public Key Authentication

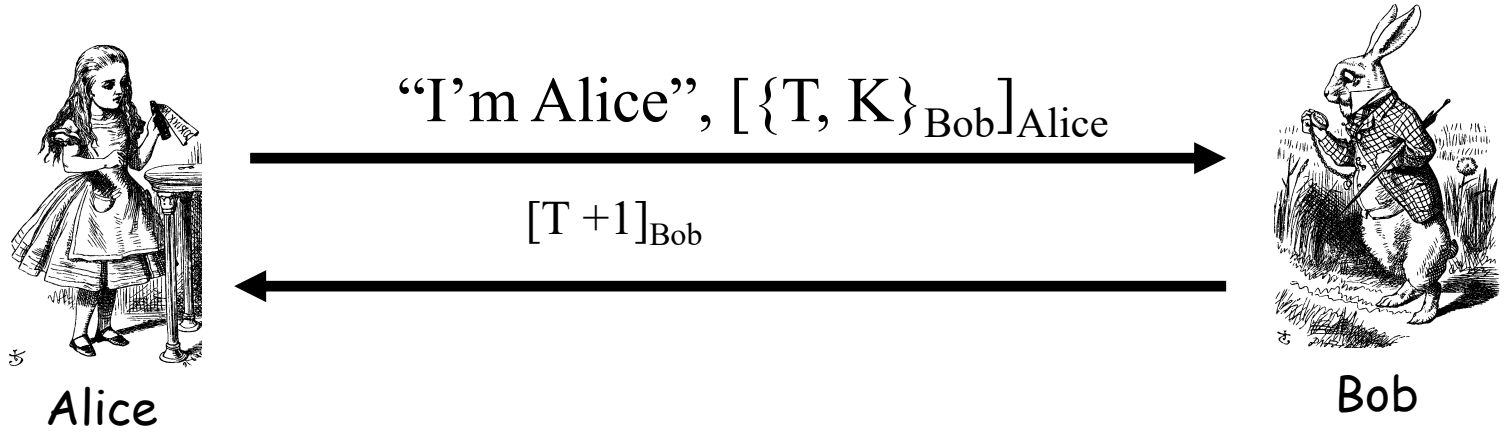
- Sign and encrypt with nonce…
  - **Insecure**
- Encrypt and sign with nonce…
  - **Secure**
- Sign and encrypt with timestamp…
  - **Secure**
- Encrypt and sign with timestamp…
  - **Insecure**
- Protocols can be subtle!

Encrypt and sign using a timestamp



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

Secure encrypt and sign with a Timestamp

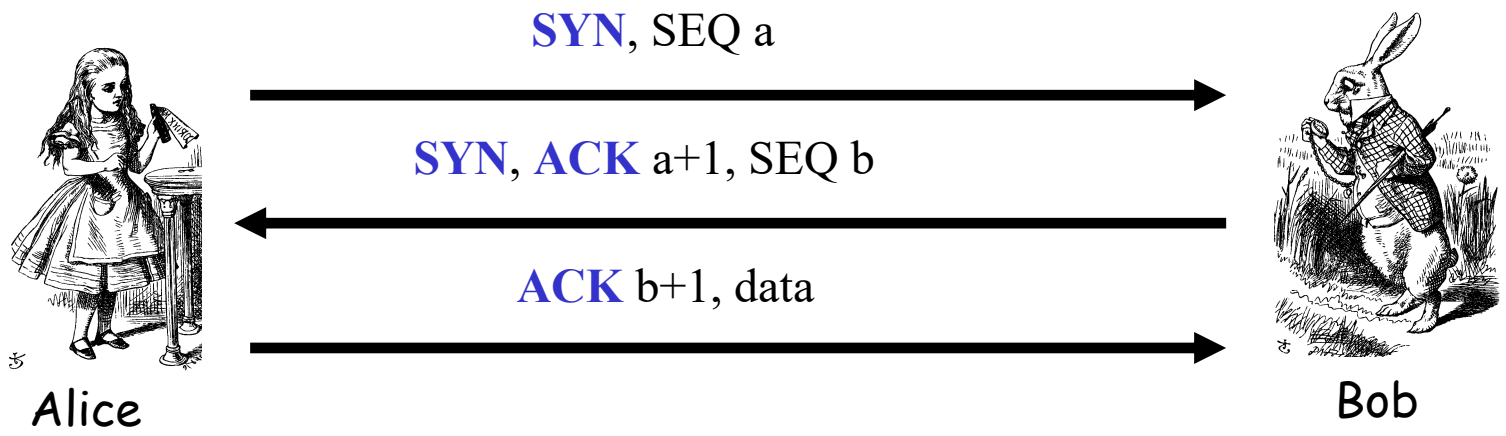


The timestamp in the message two is sufficient to authenticate Bob.

TCP-based Authentication

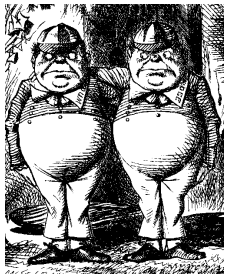
- TCP not intended for use as an authentication protocol
- But IP address in TCP connection may be (mis)used for authentication
- Also, one mode of IPSec relies on IP address for authentication

TCP 3-way Handshake



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

TCP Authentication Attack



Trudy

1. SYN, SEQ = t (as Trudy)

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

Bob

3. SYN, SEQ = t (as Alice)

5. ACK = b_2+1, data

5.

5.

5.

5.

Alice

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

发送给Alice的IP地址的响应不能被收到，否则会被终止，因为Alice并未完成三次握手过程，所以需发起DOS攻击。

TCP Authentication Attack

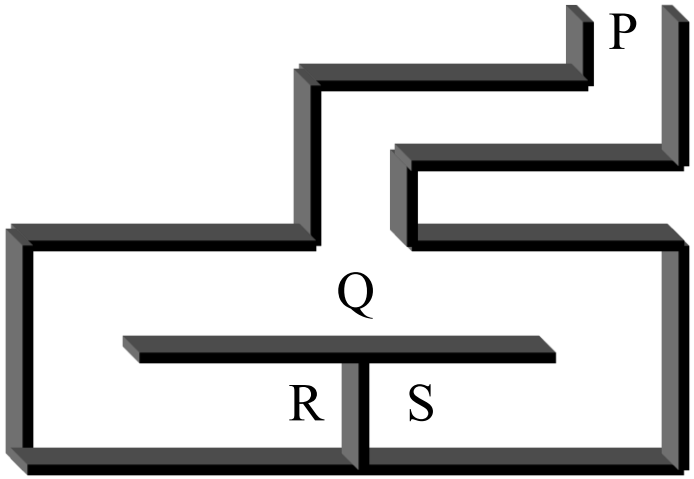
- Trudy cannot see what Bob sends, but she can send packets to Bob, while posing as **Alice**
- Trudy must prevent Alice from receiving Bob's response (or else connection will terminate)
- If **password** (or other authentication) required, this attack fails
- If TCP connection is relied on for authentication,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

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 information about the secret
- Process is probabilistic
  - Bob can verify that Alice knows the secret to an arbitrarily high probability
- An "interactive proof system"

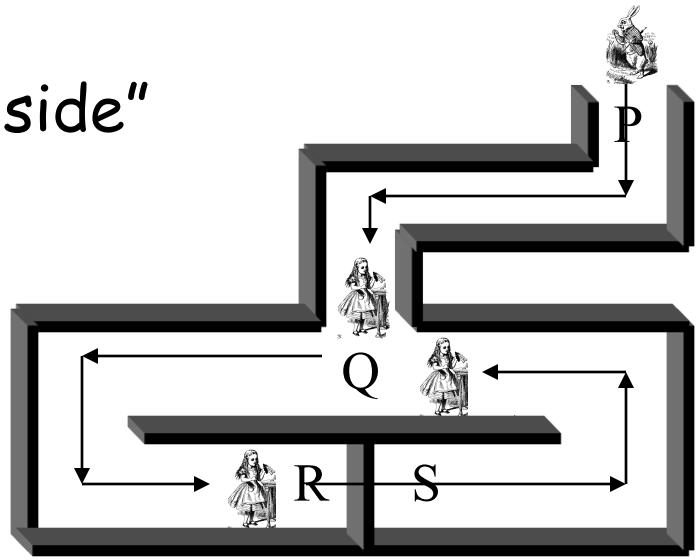
Bob's Cave

- 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…

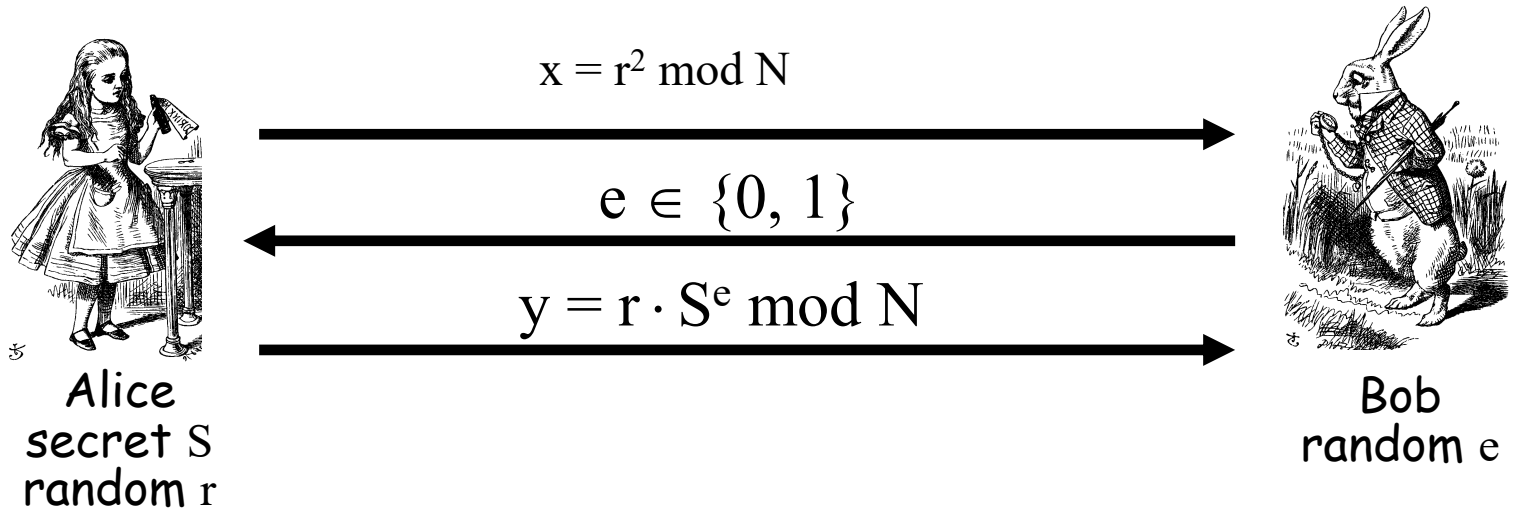


- …then Alice could come out from the correct side with probability $1/2$
- If Bob repeats this n times and Alice does not know secret, she can only fool Bob with probability $1/2^n$

Fiat-Shamir Protocol

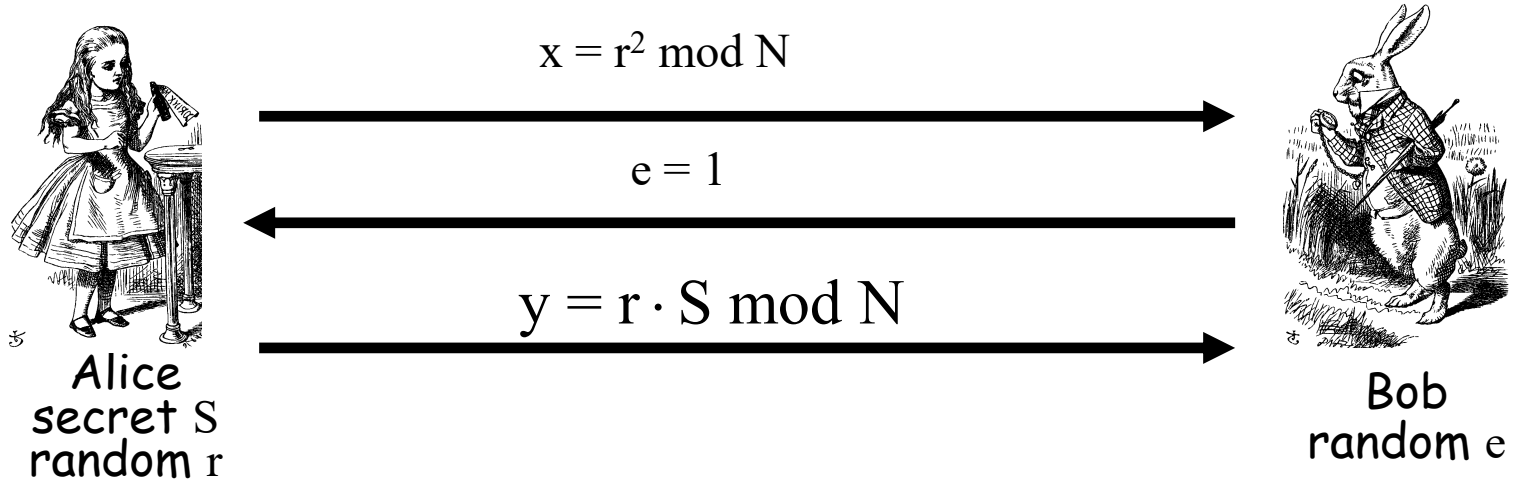
- Cave-based protocols are inconvenient
  - Can we achieve same effect without the cave?
- Finding square roots modulo N is difficult
  - Equivalent to factoring
- Suppose $N = pq$, where p and q are prime
- Alice has a secret S
- N and $v = S^2 \bmod N$ are **public**, S is **secret**
- Alice must convince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

Fiat-Shamir



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

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
  - And Alice passes this iteration of the protocol
- Note that Alice must know S in this case

Fiat-Shamir: e = 0

Alice
secret S
random r

$x = r^2 \bmod N$

$e = 0$

$y = r \bmod N$

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 follows the protocol: send $x = r^2$ in msg 1 and $y = r$ in msg 3
  - If Trudy expects $e = 1$, she 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 $r^2 \bmod N$, she gets S
  - But that requires modular square root calculation
  - 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

Homework

- 每组从第9章题目中随机选3题作答，并给出选择理由。
- 使用Kali虚拟机实现WEP口令攻击，完成实验报告 https://zhuanlan.zhihu.com/p/57677761