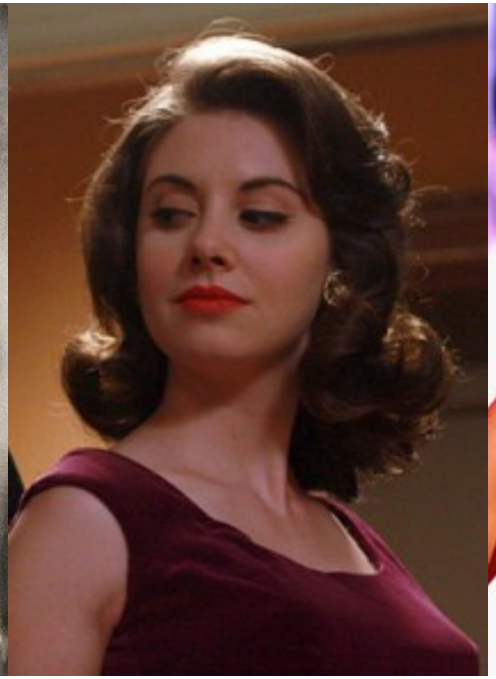
UNIVERSITY OF AMSTERDAM

Jaap van Ginkel

Security of Systems and Networks

September 30, 2024 Protocols

The Cast

Recap PKI

Alice opens Alice's Online Bank (AOB)

What are Alice's security concerns?

If Bob is a customer of AOB, what are his security concerns?

How are Alice's and Bob's concerns similar? How are they different?

How does Trudy view the situation?

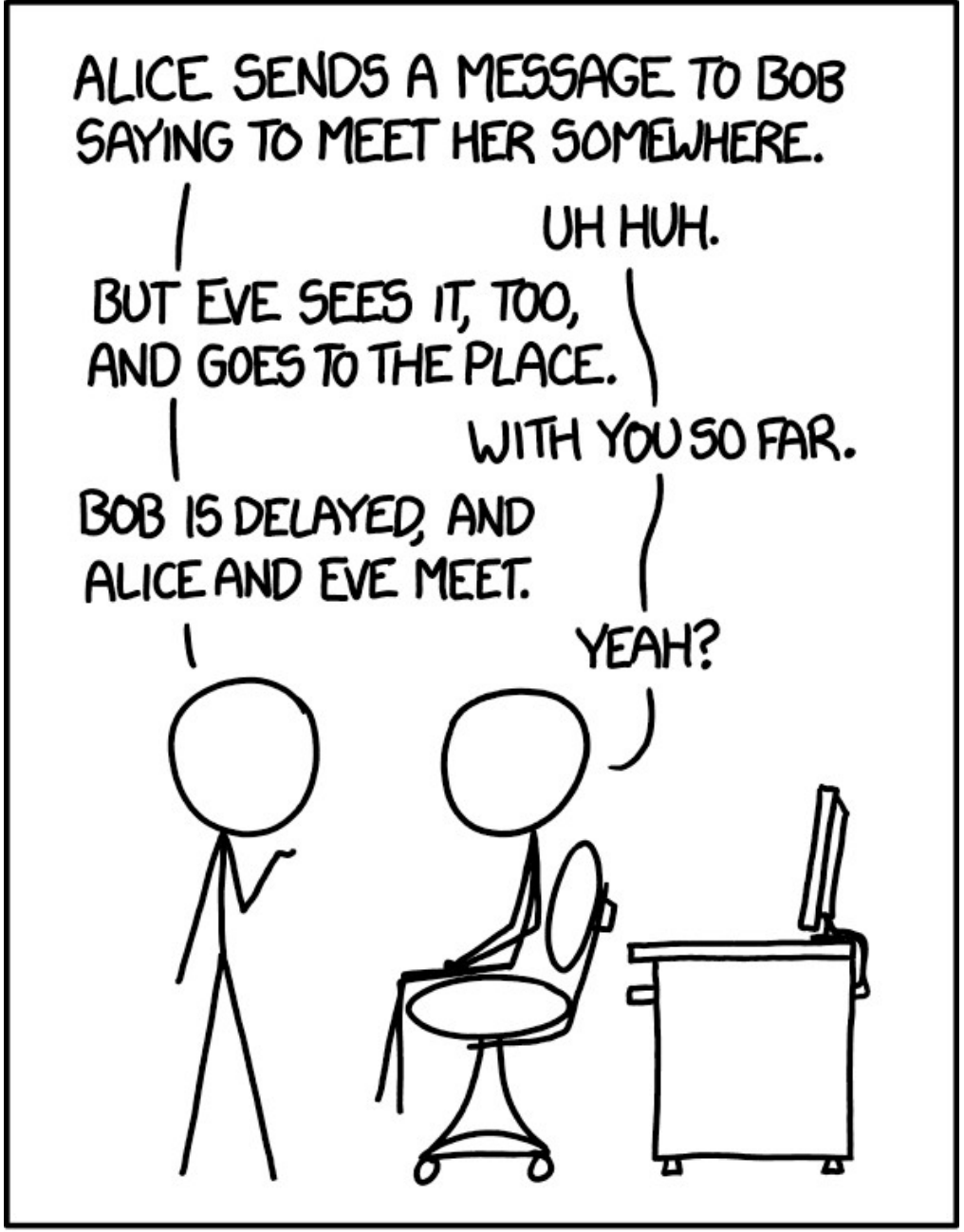
The Cast of Characters

Alice and Bob are the **good guys**

Trudy is the **bad "guy"** ⟶

Trudy is our generic "Intruder"

Sometimes Eve is used as "Eavesdropper"



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

Alice's Online Bank

Alice opens Alice's Online Bank (AOB)

What are Alice's security concerns?

If Bob is a customer of AOB, what are his security concerns?

How are Alice's and Bob's concerns similar? How are they different?

How does Trudy view the situation?

CIA

CIA == Confidentiality, Integrity, and Availability

AOB must prevent Trudy from learning Bob's account balance

Confidentiality: prevent unauthorized *reading* of information

Cryptography used for confidentiality

CIA

Trudy must not be able to change Bob's account balance

Bob must not be able to improperly change his own account balance

Integrity: detect unauthorized *writing* of information

Cryptography used for integrity

CIA

AOB's information must be available whenever it's needed

Alice must be able to make transaction

- If not, she'll take her business elsewhere

Availability: Data is available in a timely manner when needed

Availability is a "new" security concern

- Denial of service (DoS) attacks

Beyond CIA: Crypto

How does Bob's computer know that "Bob" is really Bob and not Trudy?

Bob's password must be verified

This requires some clever **cryptography**

What are security concerns of passwords?

Are there alternatives to passwords?

Beyond CIA: Protocols

When Bob logs into AOB, how does AOB know that “Bob” is really Bob?

As before, Bob’s password is verified

Unlike the previous case, **network** security issues arise

What are network security concerns?

- **Protocols** are critically important
- Crypto also important in protocols

Beyond CIA: Access Control

Once Bob is *authenticated* by AOB, then AOB must restrict actions of Bob

- Bob can't view Charlie's account info
- Bob can't install new software, etc., etc.

Enforcing these restrictions: *authorization*

Access control includes both authentication and authorization

Beyond CIA: Software

Cryptography, protocols, and access control are implemented in **software**

What are security issues of software?

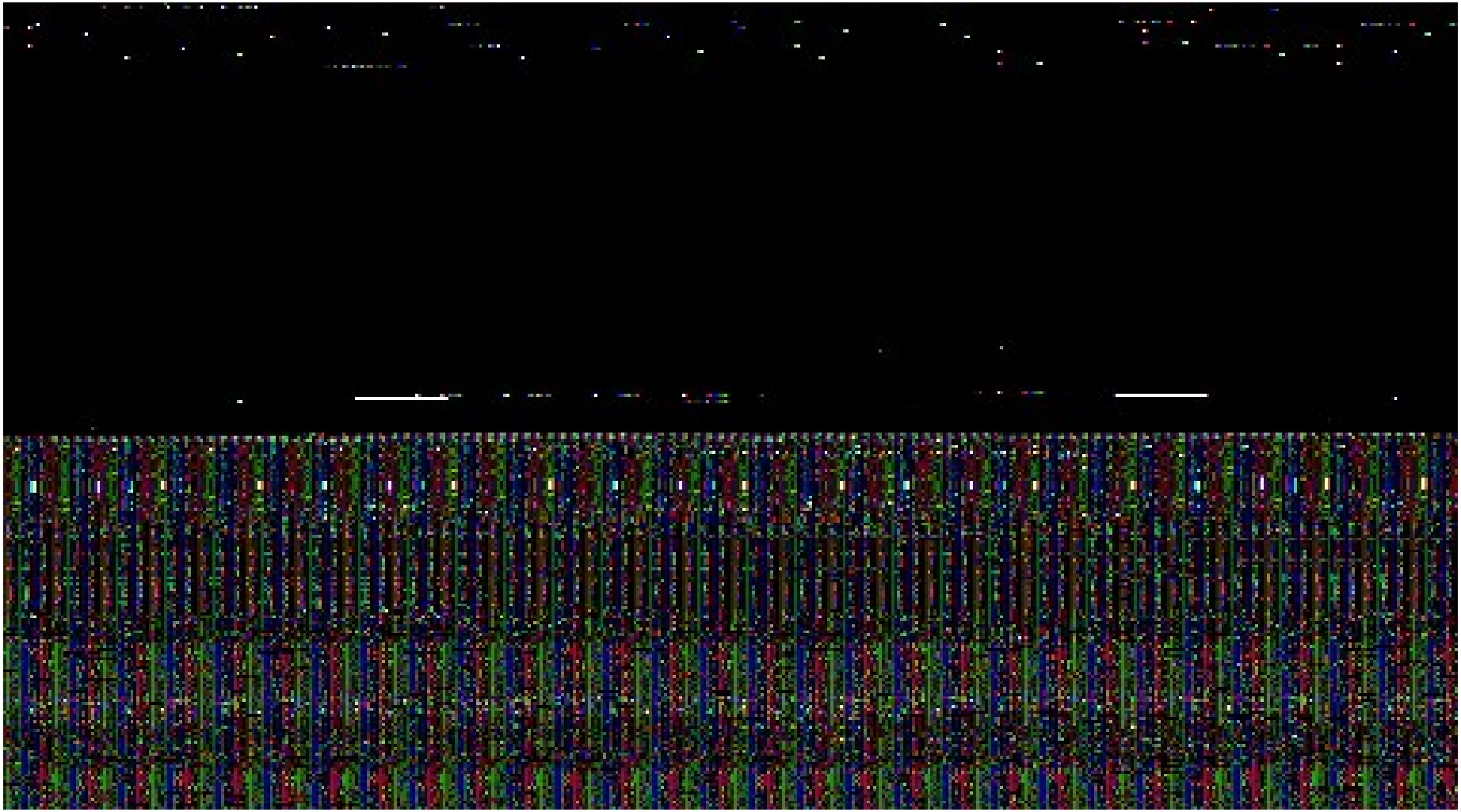
- Most software is complex and buggy
- Software flaws lead to security flaws
- How does Trudy attack software?
- How to reduce flaws in software development?

And what about malware?

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: be unambiguous



Ideal Security Protocol

- Satisfies security requirements
  - Requirements must be precise
- Efficient
  - Minimize computational requirement in particular, costly public key operations
  - Minimize delays/bandwidth
- Not fragile
  - Must work when attacker tries to break it
  - Works even if environment changes
- Easy to use and implement, flexible, etc.
- Very difficult to satisfy all of these!

Simple Security Protocols

Secure Entry to NSA

1. Insert badge into reader
2. Enter PIN
3. Correct PIN?

Yes?

Enter

No?

Get shot by security guard


NOBODY GETS IN,
NOBODY GETS OUT.
VIA THEMETAPICTURE.COM

Vulnerability type	Primitive	Protocol	Application
Plaintext disclosure			
Plaintext communication			38
Plaintext storage			32
Plaintext logging			9
Man-in-the-middle attacks			
Authentication logic error		10	22
Inadequate SSL cert. checks			42
Brute-force attacks			
Encryption logic error	1	4	
Weak encryption cipher		2	35
Weak keys	4	2	2
Hard-coded keys		1	25
Weak PRNG		2	2
Low PRNG seed entropy		1	16
PRNG seed reuse		4	
Side-channel attacks			
Timing	2	8	
Padding oracle		2	
Compression (CRIME)		2	
Memory disclosure		1	
Total	7	39	223

Figure 2: Categories of cryptographic vulnerabilities. Vulnerability types are grouped by their impacts (in bold text). Each number is a count of the CVEs in a category at the primitive, protocol, or application layer of cryptographic software.

ATM Machine Protocol

1. Insert ATM card
2. Enter PIN
3. Correct PIN?

   Yes?

   Conduct your transaction(s)

   No?

   Machine eats card

Identify Friend or Foe (IFF)

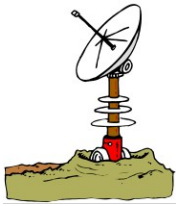
Russian
MIG

Angola

SAAF
Impala

2. E(N,K)

1. N



Namibia

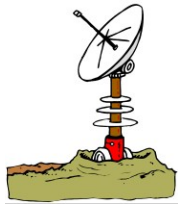
MIG in the Middle

SAAF
Impala

3. N

4. E(N,K)

2. N



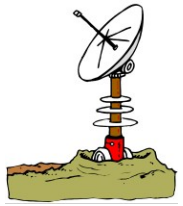
Angola

5. E(N,K)

Russian
MiG

6. E(N,K)

1. N



Namibia

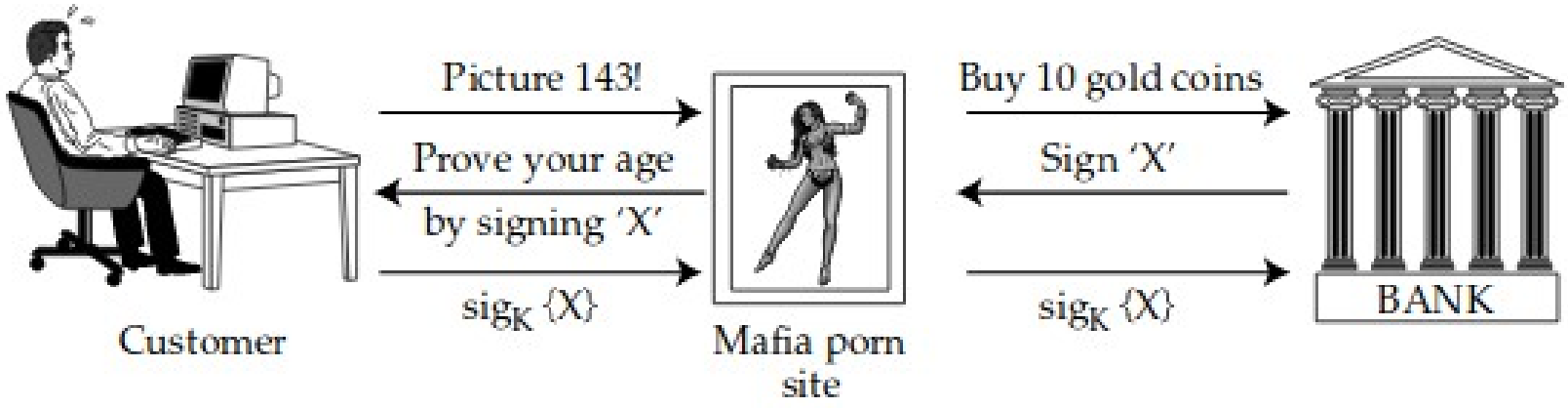


Figure 4.3: – the Mafia-in-the-middle attack

Authentication Protocols

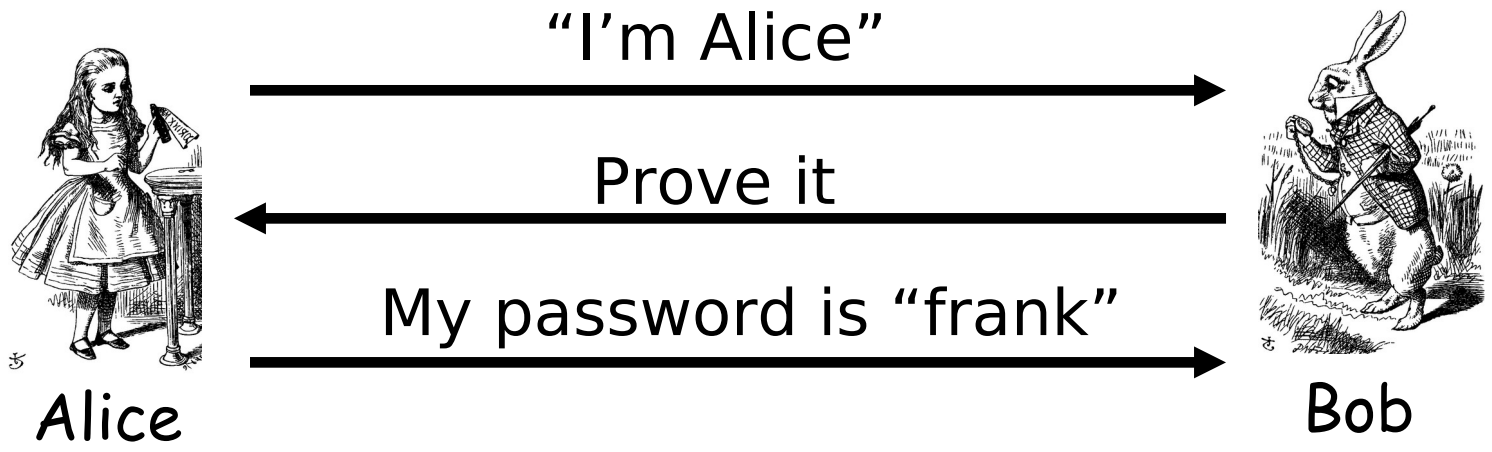
Authentication

- 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)
- May also need to establish a session key
- May have other requirements, such as
  - Use only public keys
  - Use only symmetric keys
  - Use only a hash function
  - Anonymity, plausible deniability, etc., etc.

Authentication

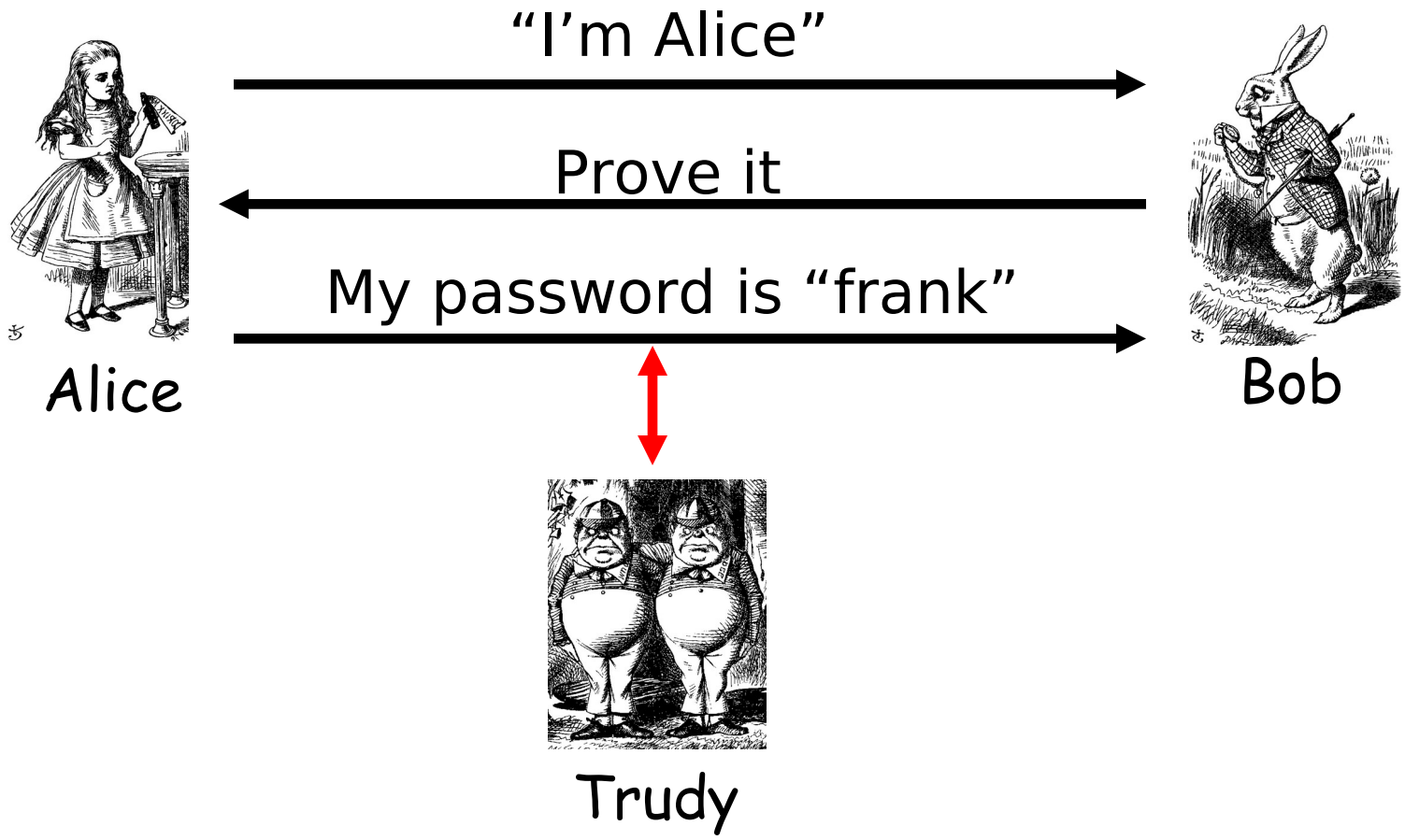
- Authentication on a stand-alone computer is relatively simple
  - “Secure path” is the primary issue
  - Main concern is an attack on authentication software
- Authentication over a network is much more complex
  - Attacker can passively observe messages
  - Attacker can replay messages
  - Active attacks may be possible (insert, delete, change messages)

Simple Authentication

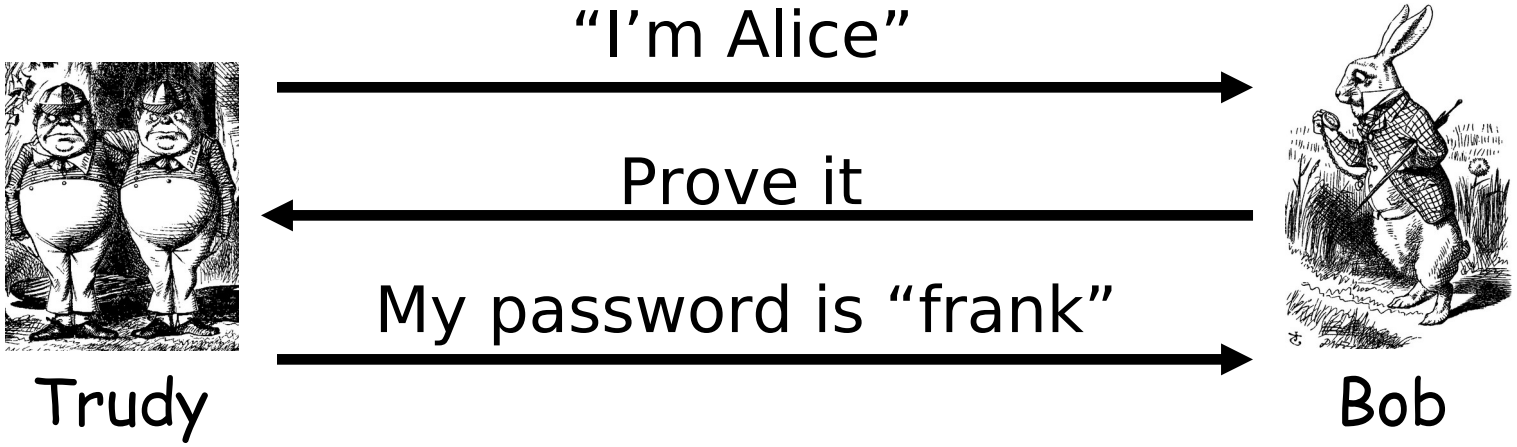


- Simple and may be OK for standalone system
- But insecure for networked system
  - Subject to a replay attack (next 2 slides)
  - Bob must know Alice's password

Authentication Attack



Authentication Attack



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

Simple Authentication

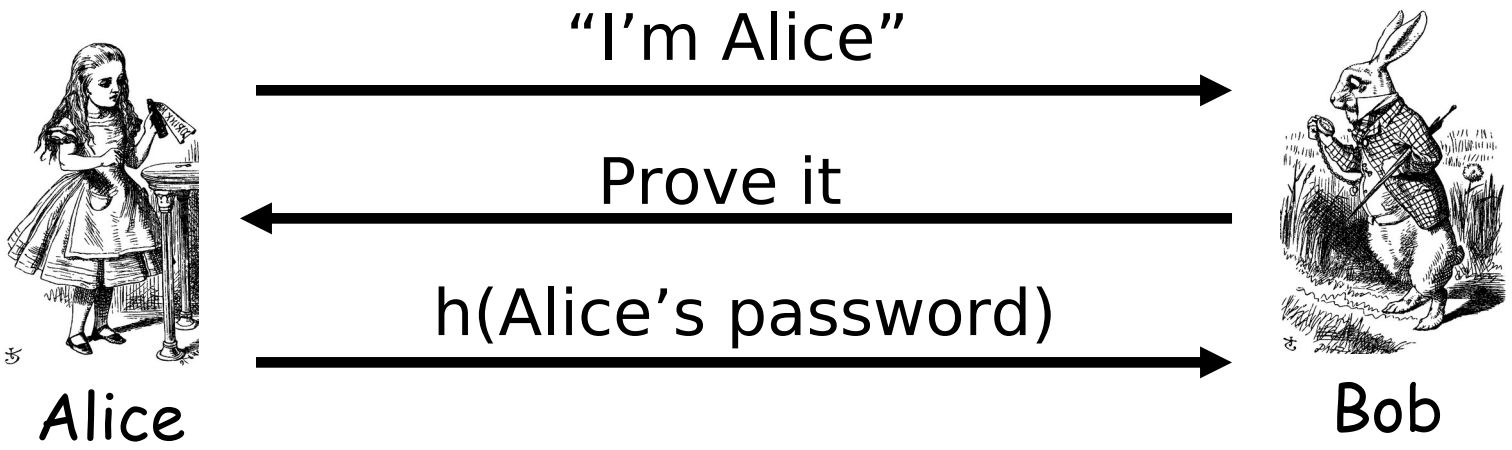
Alice

I'm Alice, My password is "frank"

Bob

- More efficient…
- But same problem as previous version

Better Authentication

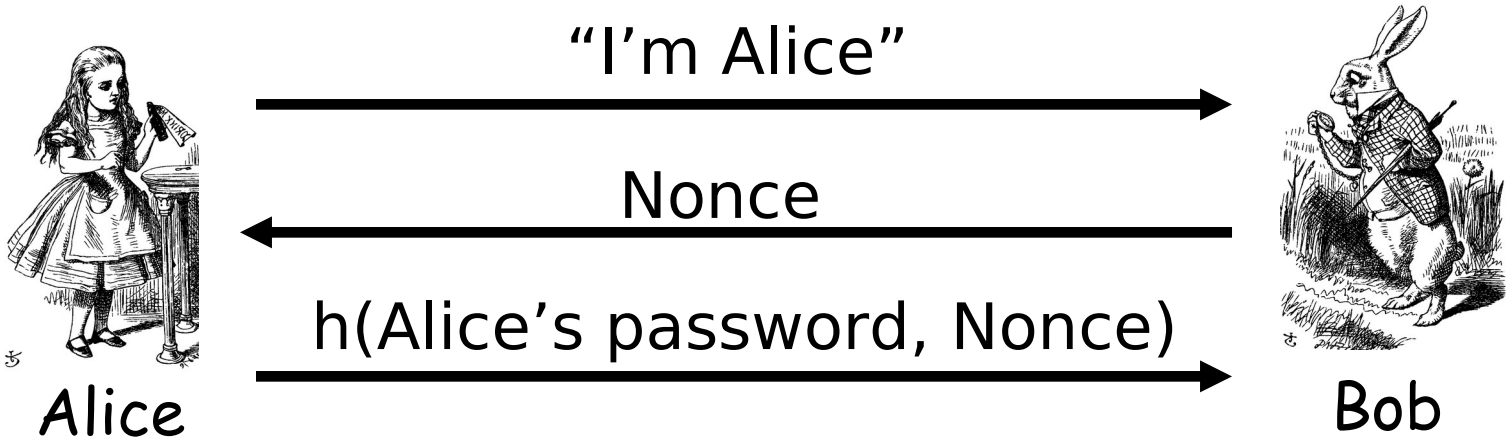


- Better since it hides Alice's password
  - From both Bob and attackers
- But still subject to replay

Challenge-Response

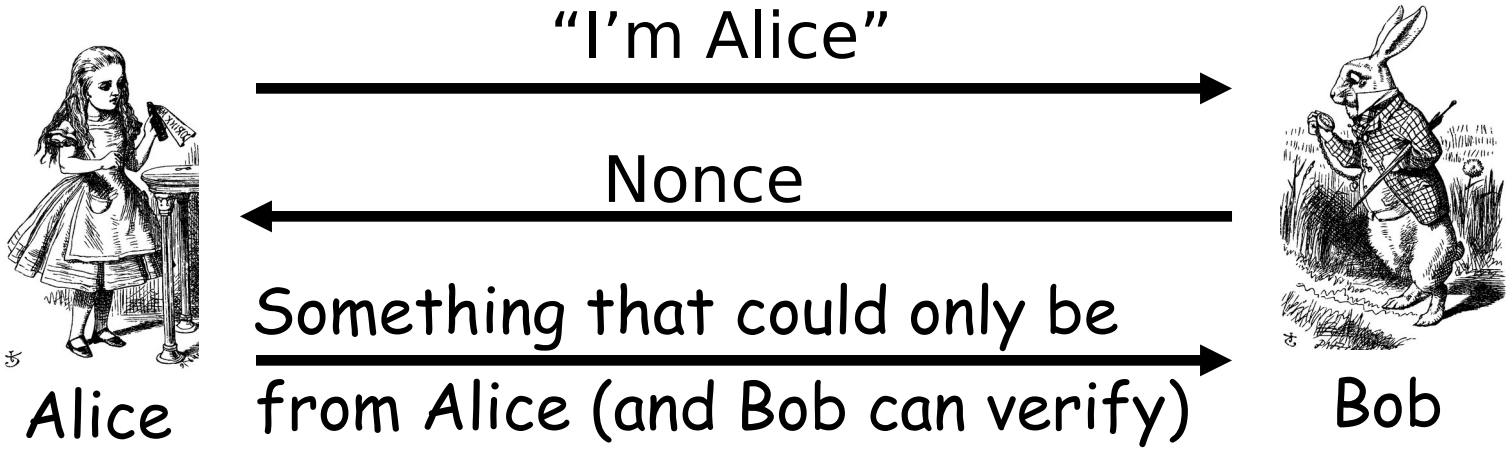
- To prevent replay, challenge-response used
- Suppose Bob wants to authenticate Alice
  - Challenge sent from Bob to Alice
  - Only Alice can provide the correct response
  - Challenge chosen so that replay is not possible
- How to accomplish this?
  - Password is something only Alice should know…
  - For freshness, a “number used once” or **nonce**

Challenge-Response



- Nonce is the **challenge**
- The hash is the **response**
- Nonce prevents replay, insures freshness
- Password is something Alice knows
- Note that Bob must know Alice's password

Challenge-Response



- What can we use to achieve this?
- Hashed password works, crypto might be better

- https://phished.be/product/

Hi Jaap,

You might want to add that screenshot to your slide deck to test how good students are in debunking bullshit advertising:

External validation and certifications

In addition to our extensive internal scanning and testing programs, we implement best practices to maintain full compliance and memberships.

SOC certification

Application servers and data are stored at google datacenters, which are among others SOC 1, SOC 2 and SOC 3 certified.

ISO 27001-compliant facilities

Application servers and data are stored at google datacenters, which are among others ISO 27001, ISO 27017.

SHA256 encryption

All personal data is stored in SHA256 encrypted databases with rotating keys and is inaccessible to non-authorized personel.

3rd party penetration test

Security testing is done on a continuous basis with the help of certified ethical hackers.

Symmetric Key Notation

- Encrypt plaintext P with key K

  C = E(P,K)

- Decrypt ciphertext C with key K

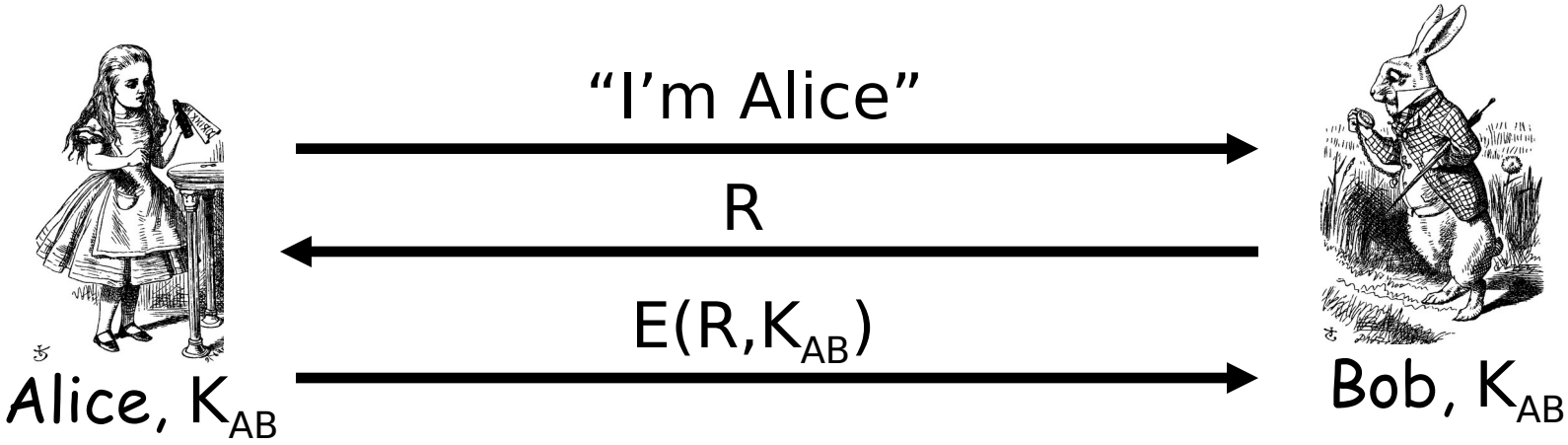
  P = D(C,K)

- Here, we are concerned with attacks on **protocols**, not directly on the crypto
- We assume that crypto algorithm is secure

Symmetric Key Authentication

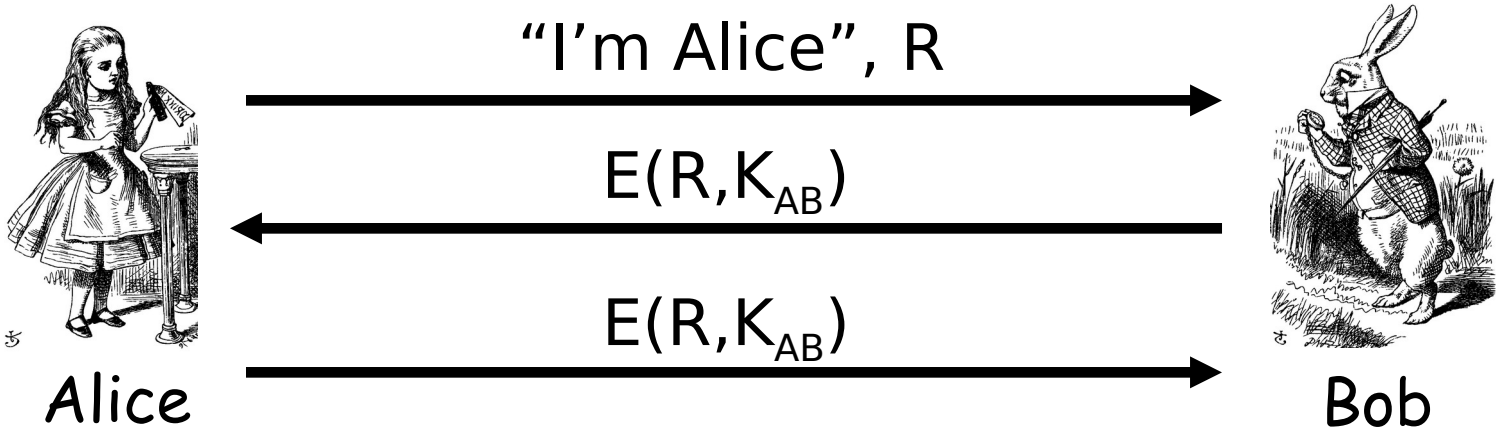
- Alice and Bob share symmetric key K_{AB}
- Key K_{AB} known only to Alice and Bob
- Authenticate by proving knowledge of shared symmetric key
- How to accomplish this?
  - Must not reveal key
  - Must not allow replay attack

Authentication with Symmetric Key



- Secure method for Bob to authenticate Alice
- Alice does not authenticate Bob
- Can we achieve mutual authentication?

Mutual Authentication?

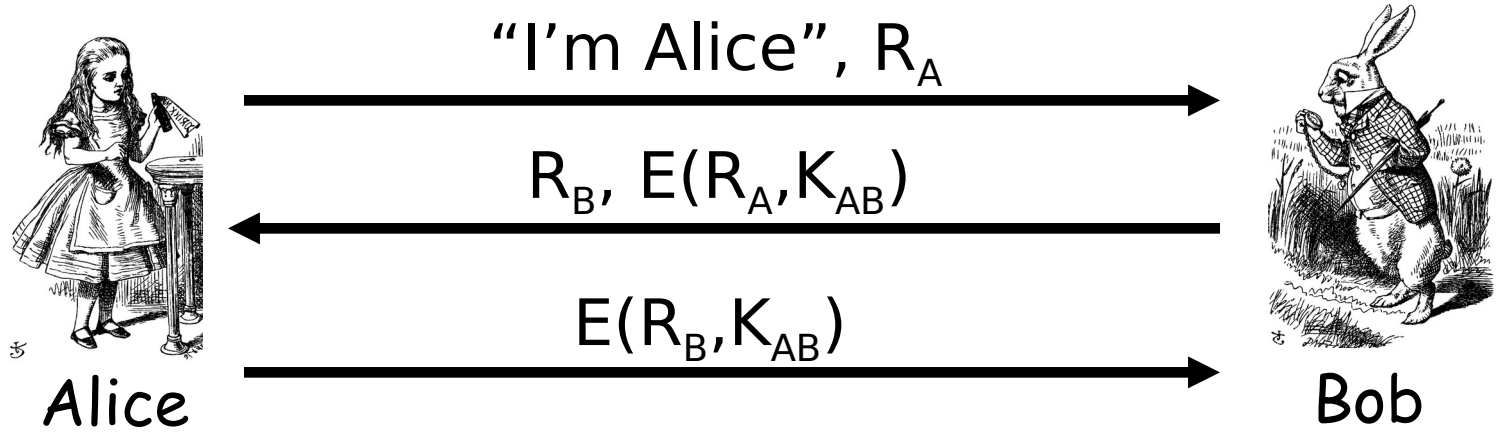


- What's wrong with this picture?
- "Alice" could be Trudy (or anybody else)!

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 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, $E(R_A,K_{AB})$

5. $E(R_B,K_{AB})$

Bob

Trudy

3. “I’m Alice”, R_B

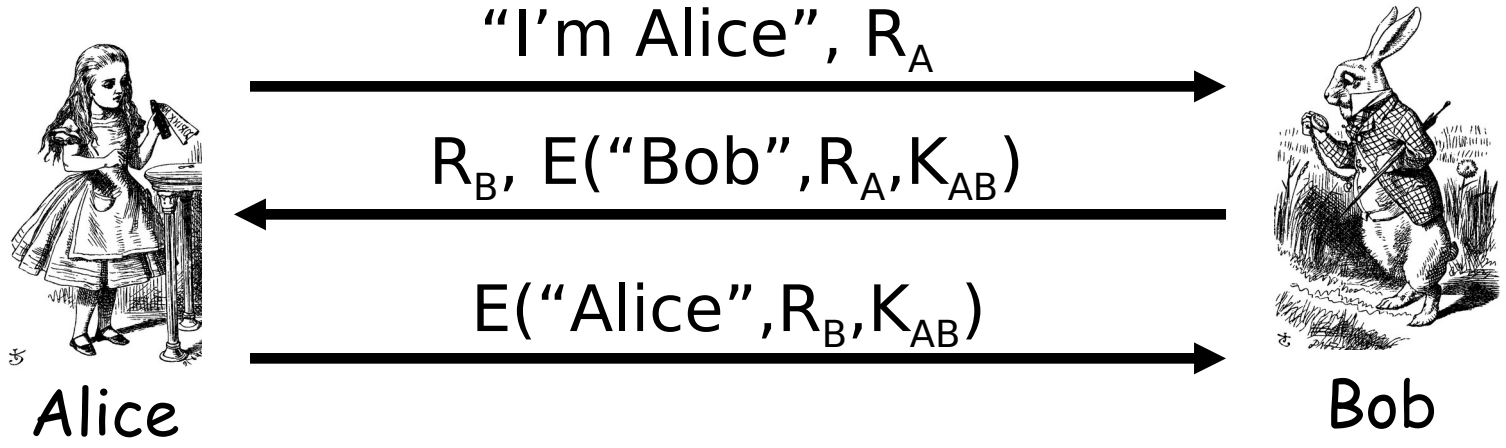
4. R_C, $E(R_B,K_{AB})$

Bob

Mutual Authentication

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

Symmetric Key Mutual Authentication



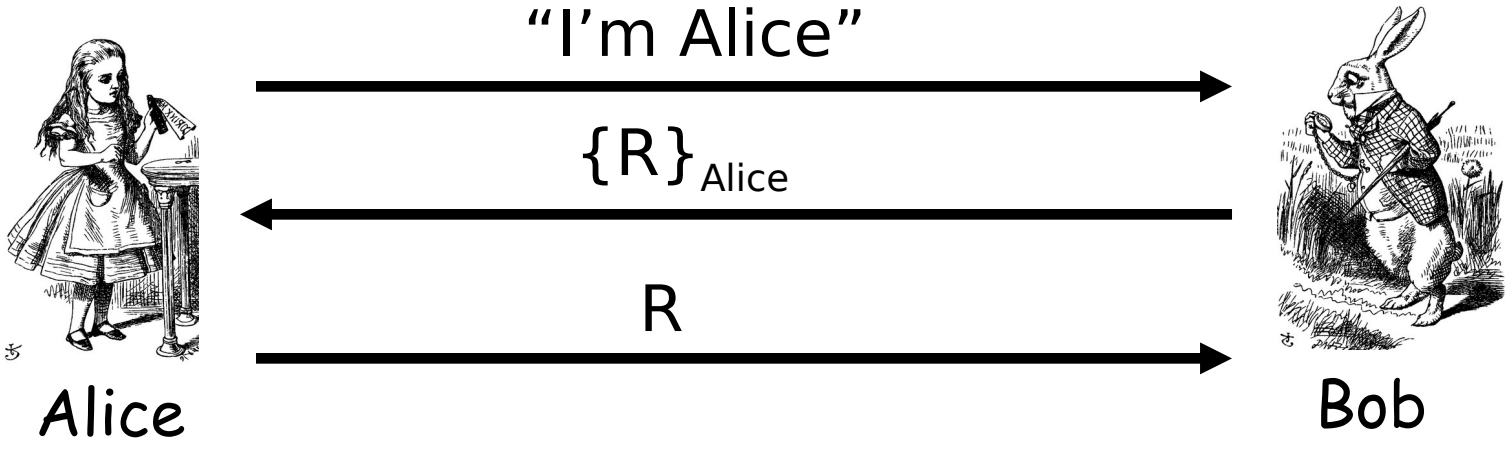
- Do these "insignificant" changes help?
- Yes!

Public Key Notation

- Encrypt M with Alice's public key: $\{M\}_{Alice}$
- Sign M with Alice's private key: $[M]_{Alice}$
- **Anybody** can do **public key** operations
- Only **Alice** can use her **private key** (sign)

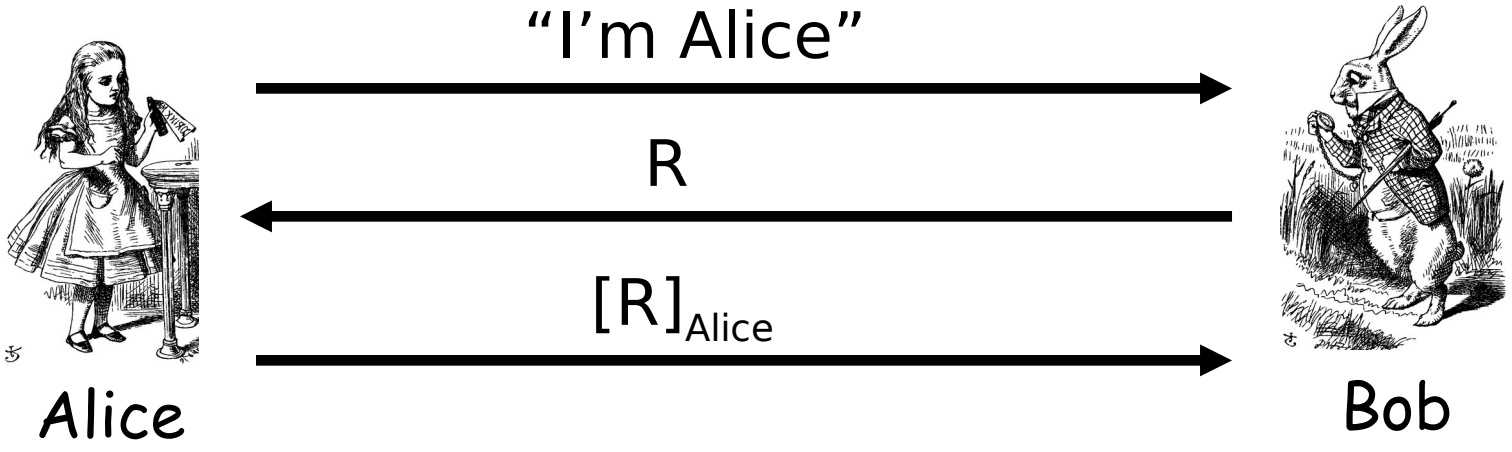
- Encryption and Decryption are inverse
- $[\{M\}_{Alice}]_{Alice} = \{[M]_{Alice}\}_{Alice}$

Public Key Authentication



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

Public Key Authentication



- Is this secure?
- Trudy can get Alice to sign anything!
  - Must have two key pairs

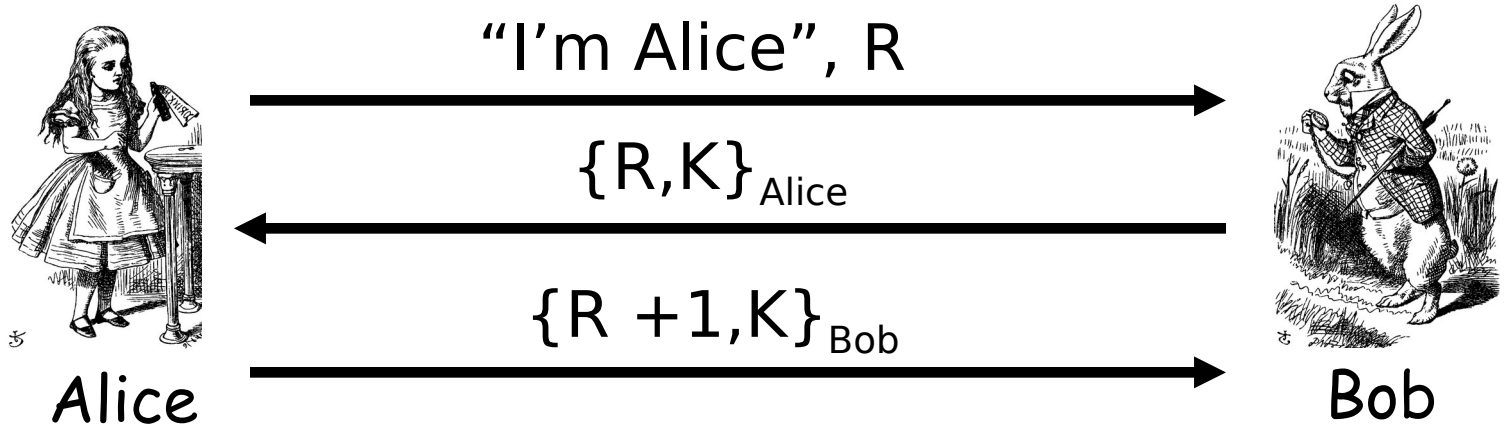
Public Keys

- Never use the same key pair for encryption and signing
- One key pair for encryption/decryption
- A different key pair for signing/verifying signatures

Session Key

- Usually, a session key is required
  - Symmetric key for a particular session
- Can we authenticate and establish a shared symmetric key?
  - Key can be used for confidentiality
  - Key can be used for integrity
- In some cases, we may also require perfect forward secrecy (PFS)
  - Discussed later…

Authentication & Session Key

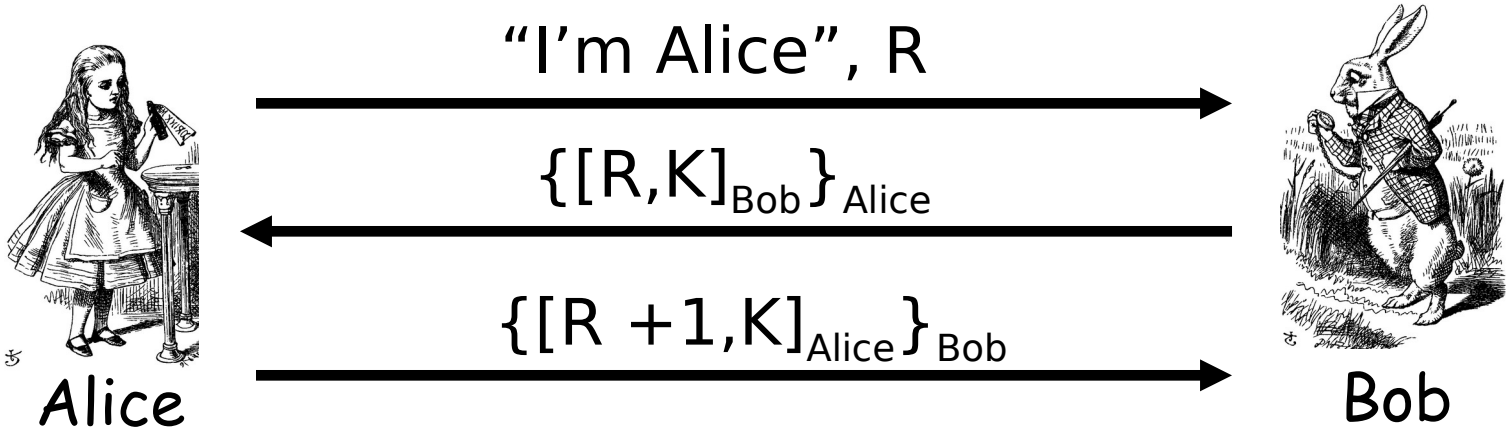


- Is this secure?
- OK for key, but no mutual authentication
- **Note** that K is acting as Bob's nonce

Public Key Authentication and Session Key

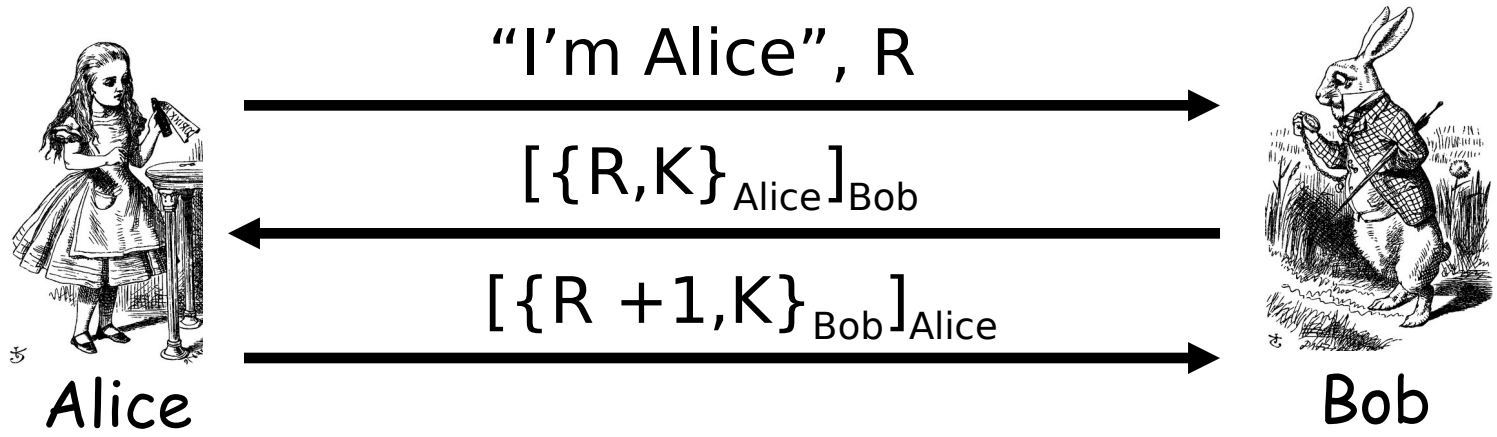
- Is this secure?
- Mutual authentication but key is not secret!

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

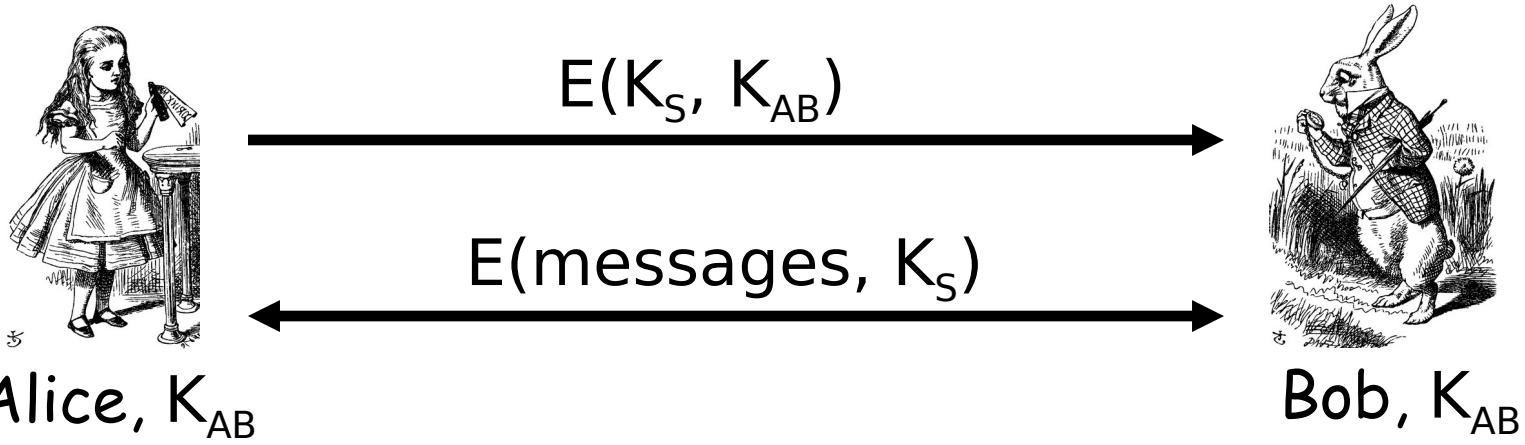
Perfect Forward Secrecy

- The concern…
  - Alice encrypts message with shared key K_{AB} and sends ciphertext to Bob
  - Trudy records ciphertext and later attacks Alice's (or Bob's) computer to find K_{AB}
  - Then Trudy decrypts recorded messages
- **Perfect forward secrecy (PFS):** Trudy cannot later decrypt recorded ciphertext
  - Even if Trudy gets key K_{AB} or other secret(s)
- Is PFS possible?

Perfect Forward Secrecy

- Suppose Alice and Bob share key K_{AB}
- For perfect forward secrecy, Alice and Bob cannot use K_{AB} to encrypt
- Instead they must use a **session key** K_S and forget it after it's used
- Problem: How can Alice and Bob agree on session key K_S and ensure PFS?

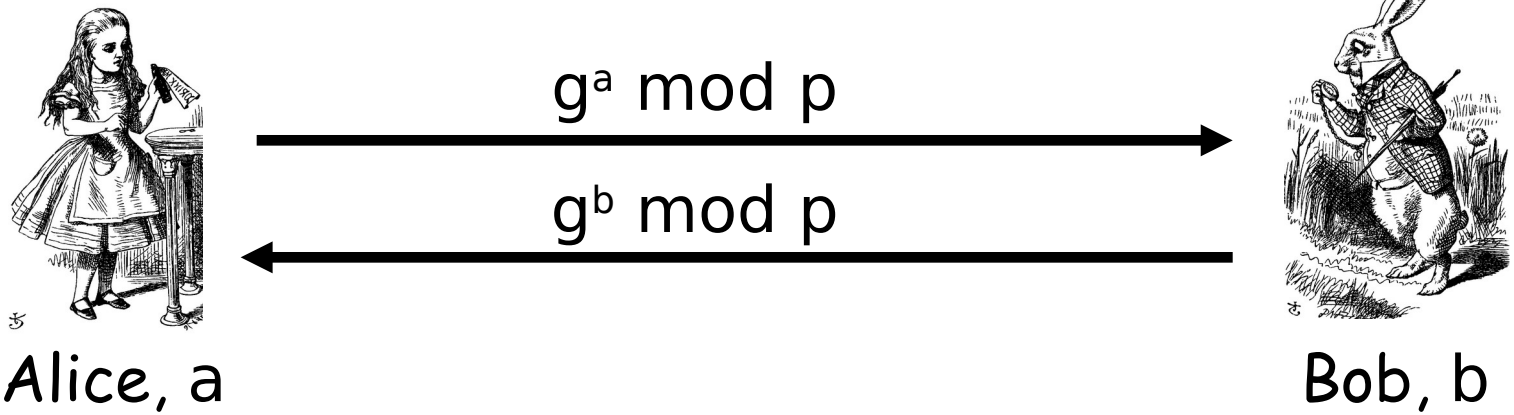
Naïve Session Key Protocol



- Trudy could also record $E(K_S, K_{AB})$
- If Trudy gets K_{AB}, she gets K_S

Perfect Forward Secrecy

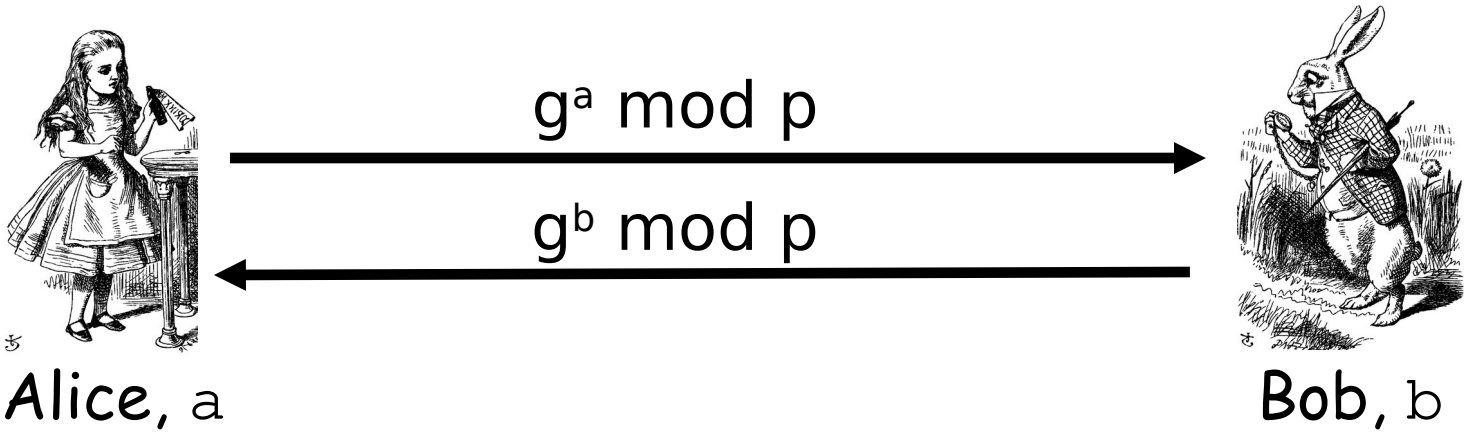
- Can use **Diffie-Hellman** for PFS
- Recall Diffie-Hellman: public g and p



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

Diffie-Hellman

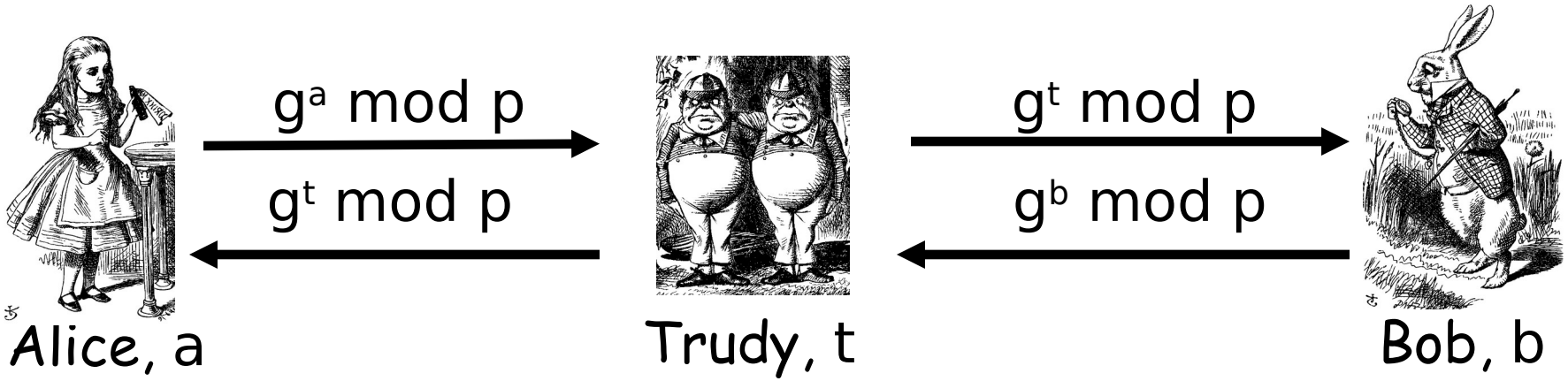
- **Public:** g and p
- **Secret:** Alice's exponent a, Bob's exponent b



- Alice computes $(g^b)^a = g^{ba} = g^{ab}$ mod p
- Bob computes $(g^a)^b = g^{ab}$ mod p
- Could use $K = g^{ab}$ mod p as symmetric key

Diffie-Hellman

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



- Trudy shares secret g^{at} mod p with Alice
- Trudy shares secret g^{bt} mod p with Bob
- Alice and Bob don't know Trudy exists!

Diffie-Hellman

- How to prevent MiM attack?
  - Encrypt DH exchange with symmetric key
  - Encrypt DH exchange with public key
  - Sign DH values with private key
  - Other?
- You **MUST** be aware of MiM attack on Diffie-Hellman

Perfect Forward Secrecy

Alice, a

$E(g^a \bmod p, K_{AB})$

$E(g^b \bmod p, K_{AB})$

Bob, b

- Session key $K_S = g^{ab} \bmod p$
- Alice forgets a, Bob forgets b
- **Ephemeral Diffie-Hellman**
- Not even Alice and Bob can later recover K_S
- Other ways to do PFS?

Mutual Authentication, Session Key and PFS

Alice

"I'm Alice", R_A

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

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

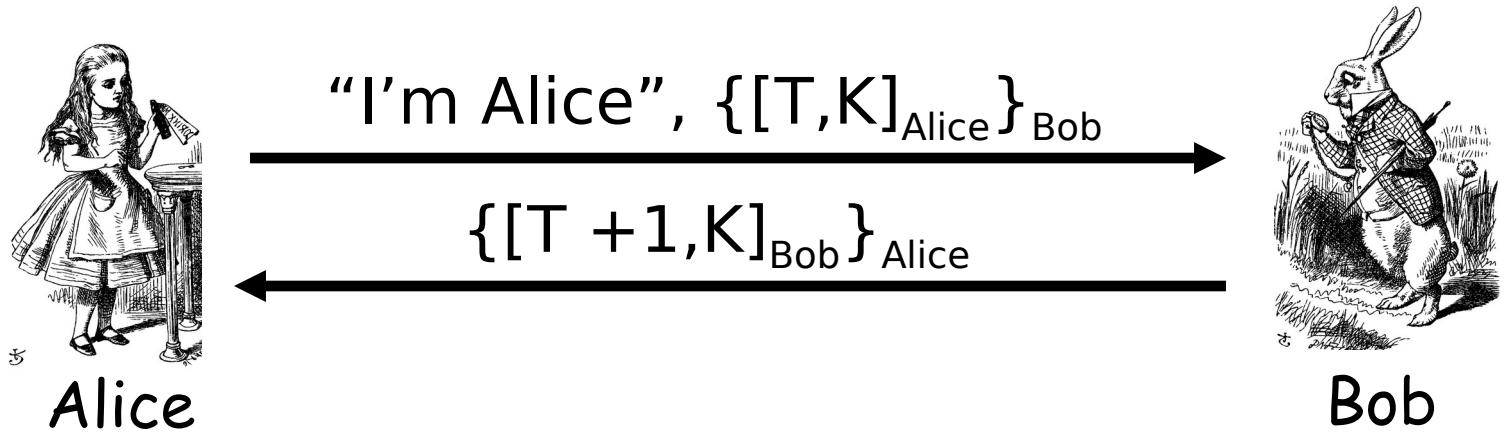
Bob

- 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 the current time
- Timestamps used in many security protocols (Kerberos, for example)
- Timestamps reduce number of messages
  - Like a nonce that both sides know in advance
- But, use of timestamps implies that time is a security-critical parameter
- Clocks never exactly the same, so must allow for **clock skew** risk of replay
- How much clock skew is enough?

Public Key Authentication with Timestamp T



- Is this secure?
- Seems to be OK

Public Key Authentication with Timestamp T

Alice

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

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

Bob

- Is this secure?
- Trudy can use Alice's public key to find $\{T,K\}_{Bob}$ and then…

Public Key Authentication with Timestamp T

Trudy

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

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

Bob

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

Public Key Authentication

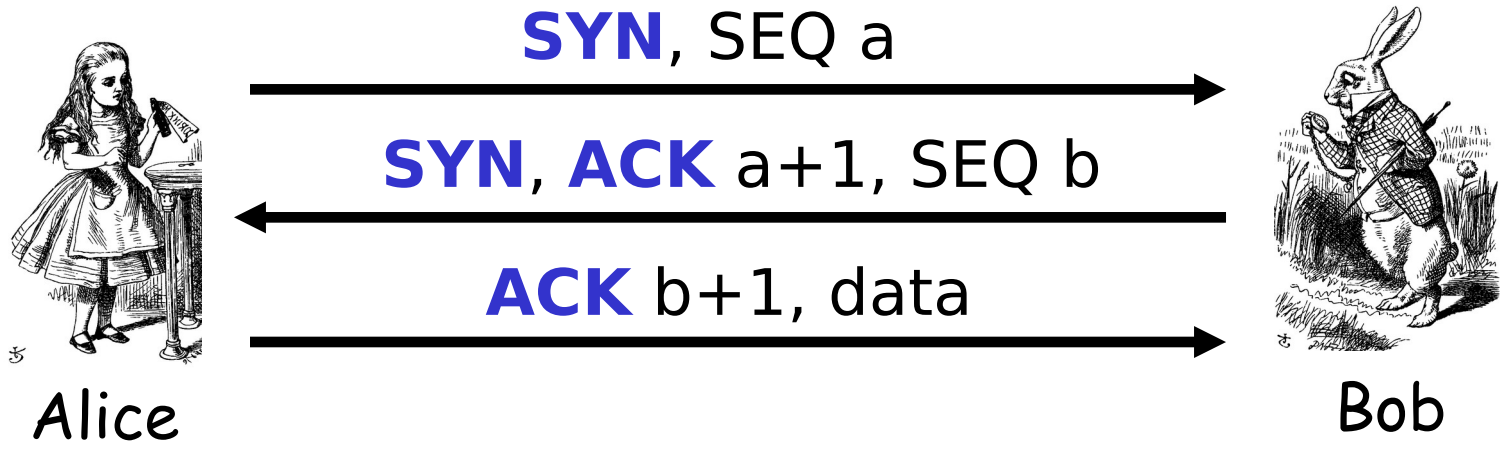
- Sign and encrypt with nonce…
  - **Secure**
- Encrypt and sign with nonce…
  - **Secure**
- Sign and encrypt with timestamp…
  - **Secure**
- Encrypt and sign with timestamp…
  - **Insecure**
- Protocols can be subtle!

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 SEQ number must be random
- Why? See the next slide…

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 = $\mathbf{b_2+1}$, data

Bob

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

5.

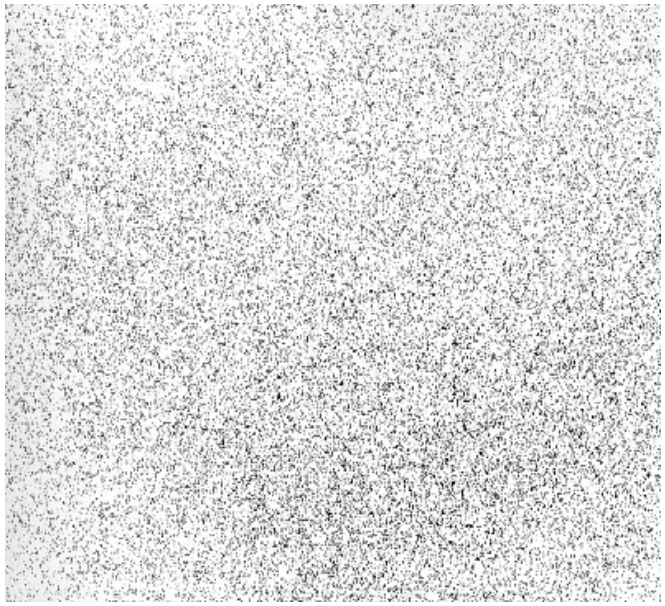
5.

5.

5.

Alice

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 server 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 succeeds
- **Bad idea** to rely on TCP for authentication