COM3020J - Protocols

Dr. Anca Jurcut
E-mail: anca.jurcut@ucd.ie

School of Computer Science and Informatics
University College Dublin,
Ireland

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 also occur
  - Recently, IE implementation of SSL
- Not easy to get protocols right…

Ideal Security Protocol

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

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

ATM Machine Protocol

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

   Yes? Conduct your transaction(s)

   No? Machine (eventually) eats card

Authentication Protocols

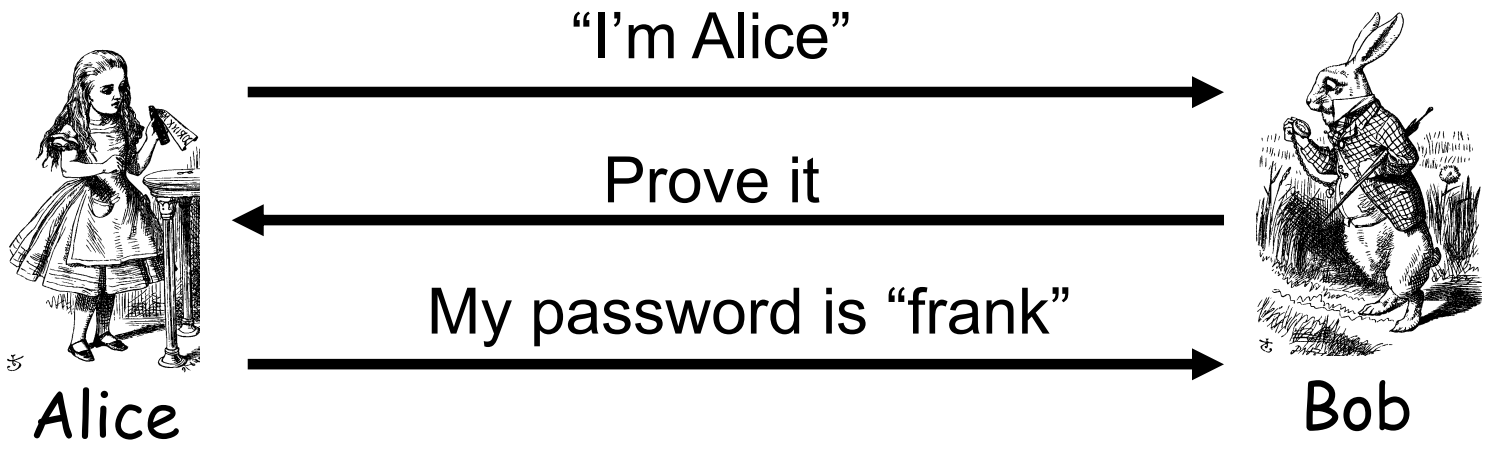
Authentication

- 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
  - Public keys, symmetric keys, hash functions, …
  - Anonymity, plausible deniability, perfect forward secrecy, etc.

Authentication

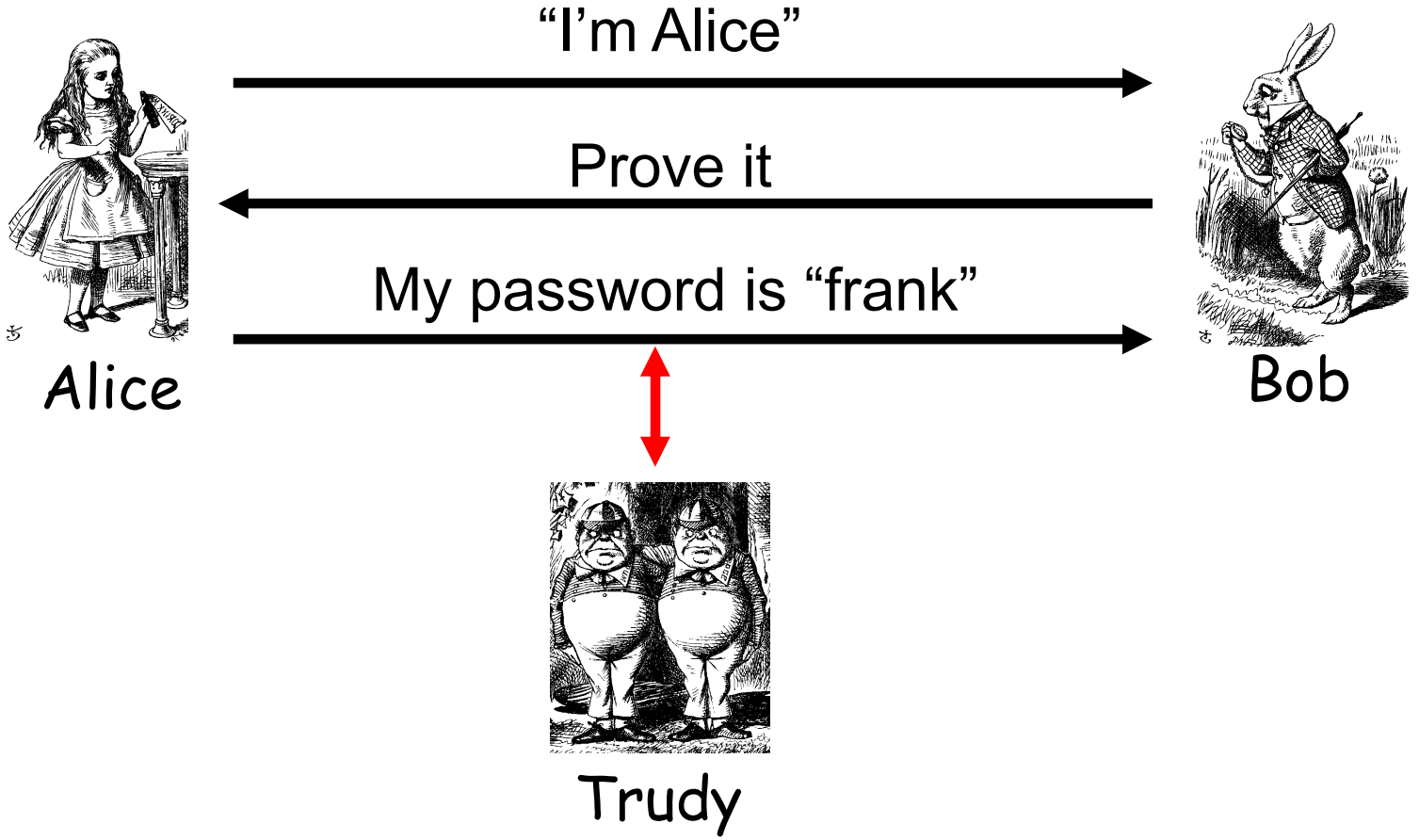
- Authentication on a stand-alone computer is relatively simple
  - Hash password with salt
  - “Secure path,” attacks on authentication software, keystroke logging, etc., can be issues
- Authentication over a network is challenging
  - Attacker can passively observe messages
  - Attacker can replay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

Simple Authentication



- Simple and may be OK for standalone system
- But highly 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

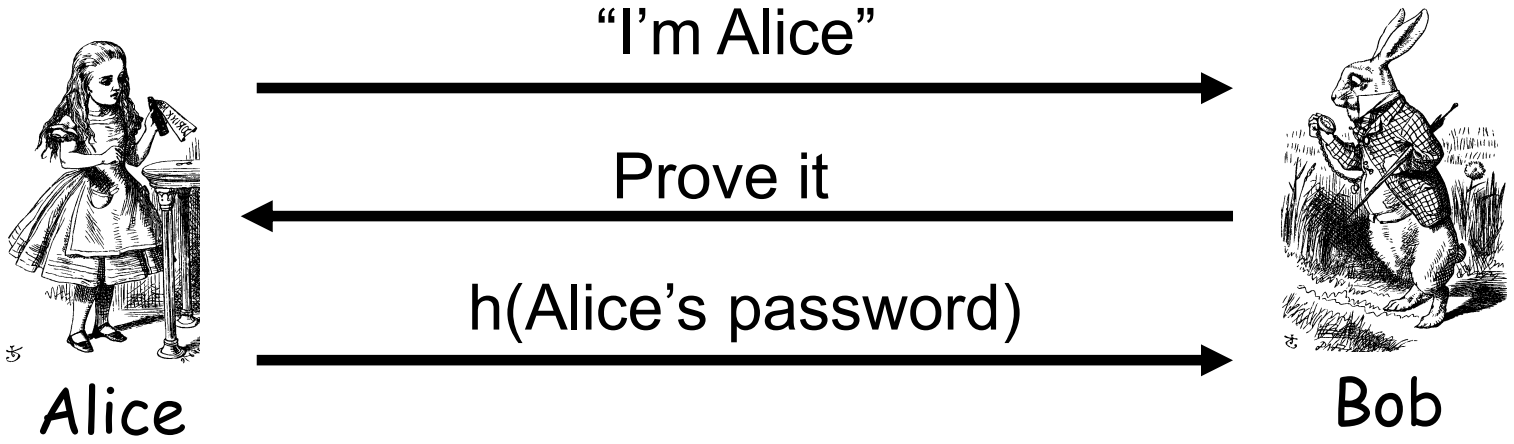
Alice

I'm Alice, my password is "frank"

Bob

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

Better Authentication



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

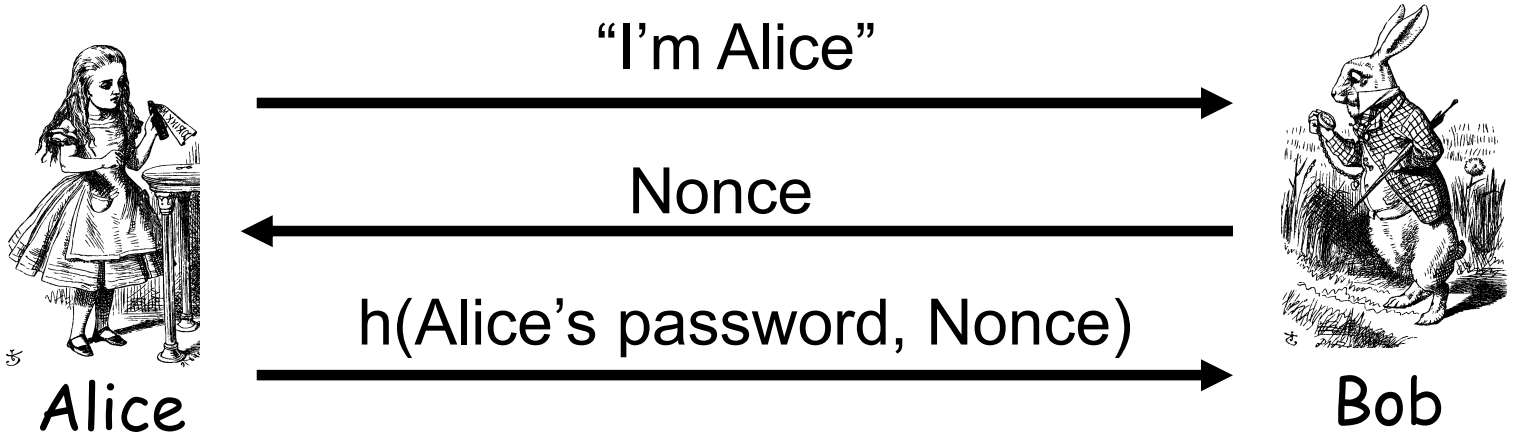
Challenge-Response

- 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

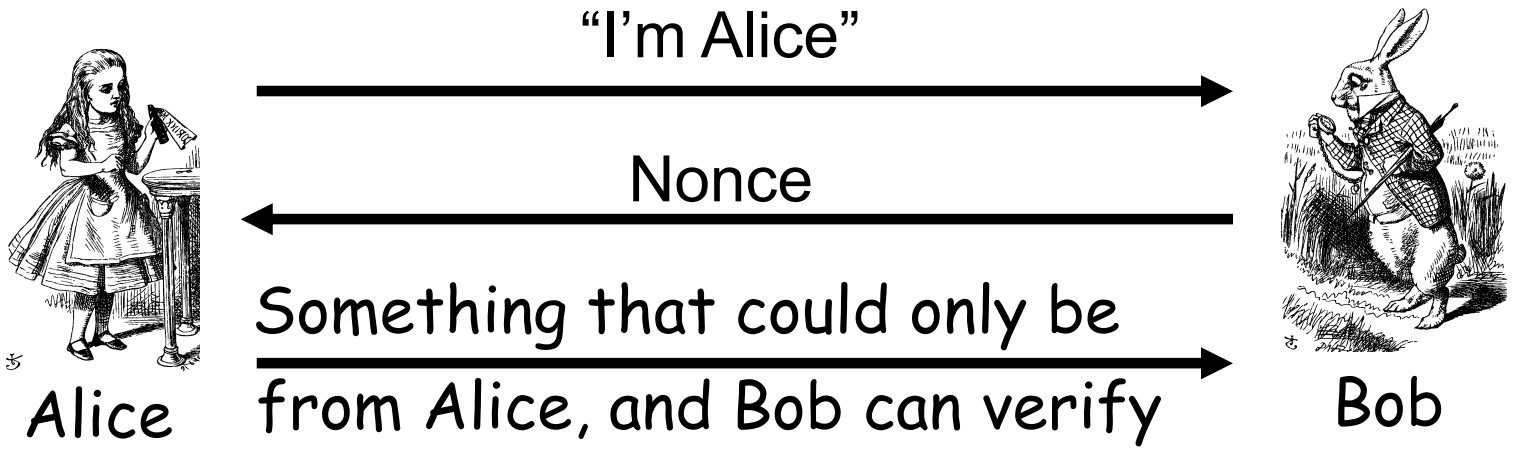
- To ensure freshness, can employ a **nonce**
  - Nonce == **n**umber used **once**
- What to use for nonces?
  - That is, what is the challenge?
- What should Alice do with the nonce?
  - That is, how to compute the response?
- How can Bob verify the response?
- Should we use passwords or keys?

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

Generic Challenge-Response



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

Symmetric Key Notation

- Encrypt plaintext P with key K

  C = E(P,K)

- Decrypt ciphertext C with key K

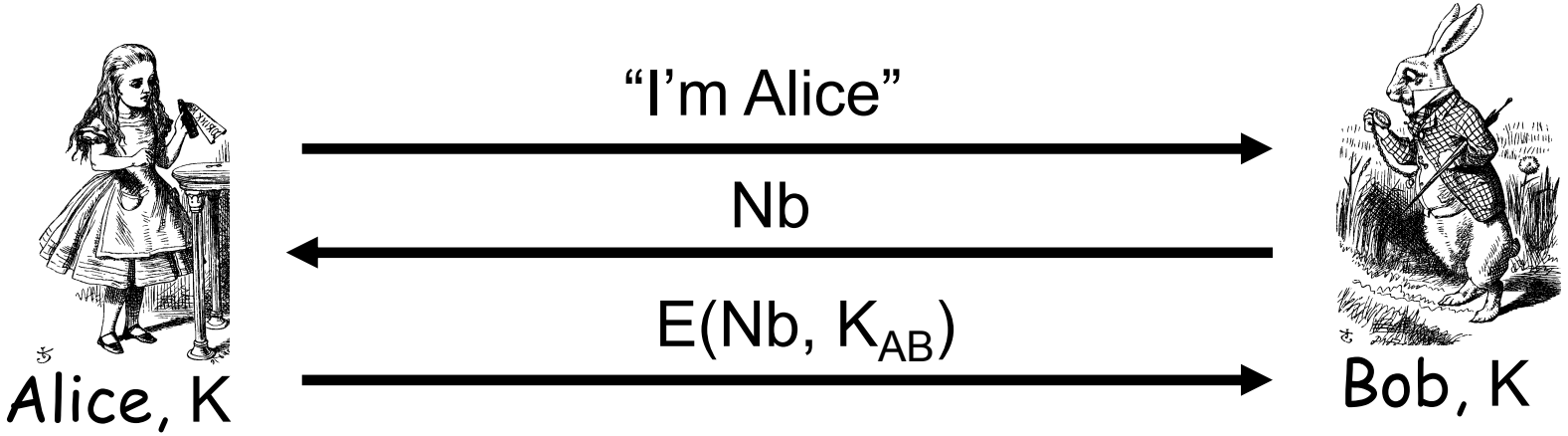
  P = D(C,K)

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

Authentication: Symmetric Key

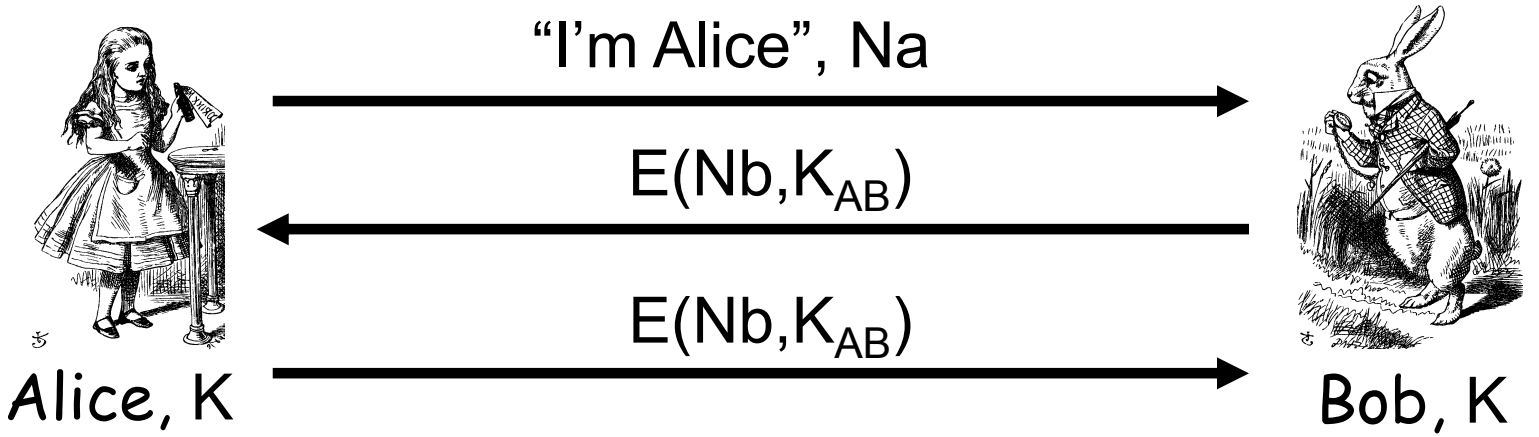
- Alice and Bob share symmetric key K_{AB}
- Key K_{AB} known only to Alice and Bob
- Authenticate by proving knowledge of shared symmetric key
- How to accomplish this?
  - Cannot reveal key, must not allow replay (or other) attack, must be verifiable, …

Authenticate Alice Using Symmetric Key



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

Mutual Authentication?

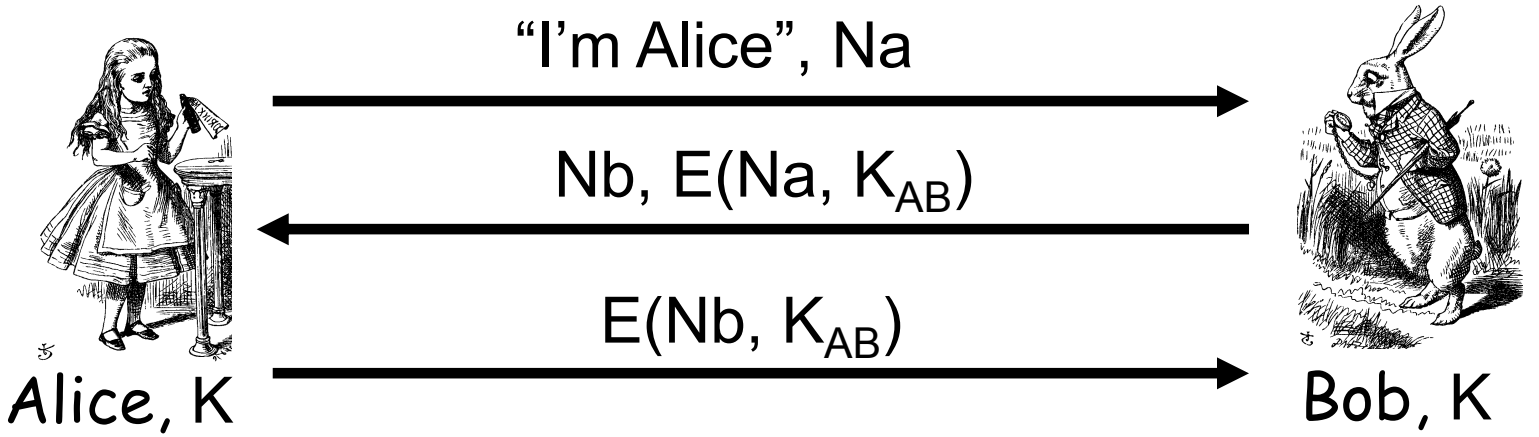


- What's wrong with this picture?
- "Alice" could be Trudy (or anybody else)!

Mutual Authentication

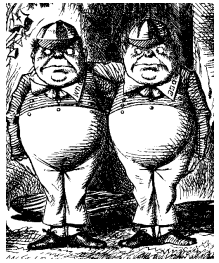
- Since we have a secure one-way authentication protocol…
- The obvious thing to do is to use the protocol twice
  - Once for Bob to authenticate Alice
  - Once for Alice to authenticate Bob
- This has got to work…

Mutual Authentication



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

Mutual Authentication Attack



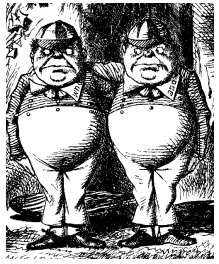
Trudy

1. “I’m Alice”, Na

2. Nb, E(Na, K_{AB})

5. E(Nb, K_{AB})

Bob, K



Trudy

3. “I’m Alice”, Nb

4. Nc, E(Nb, K_{AB})

Bob, K