COM3020J - Protocols

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

School of Computer Science and Informatics
University College Dublin,
Ireland

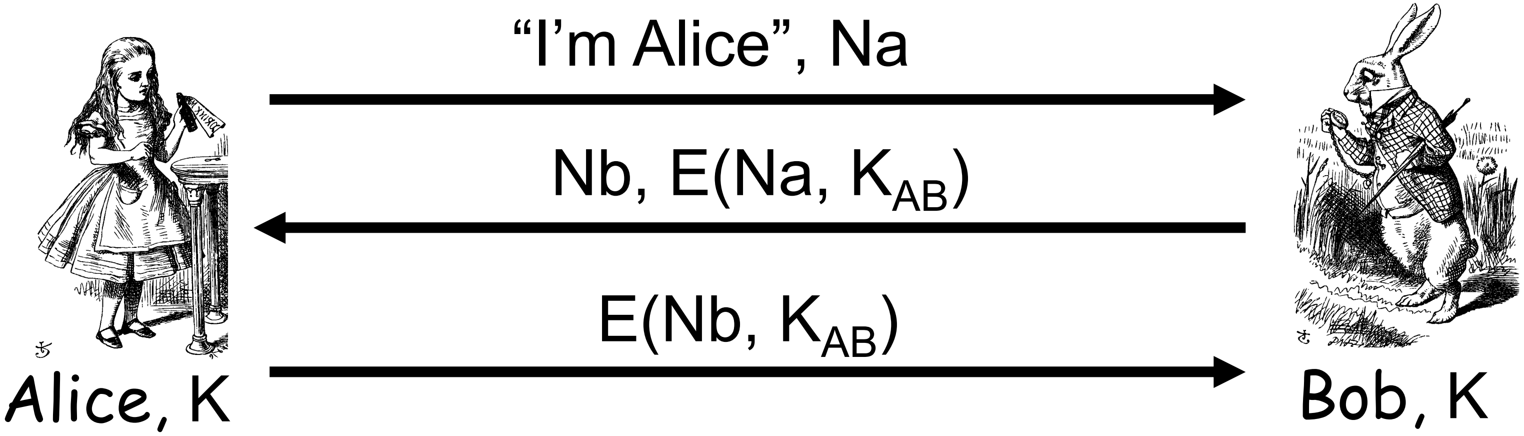
Authentication: Symmetric Key

- 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, …

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", Na

Nb, E(Na, K_{AB})

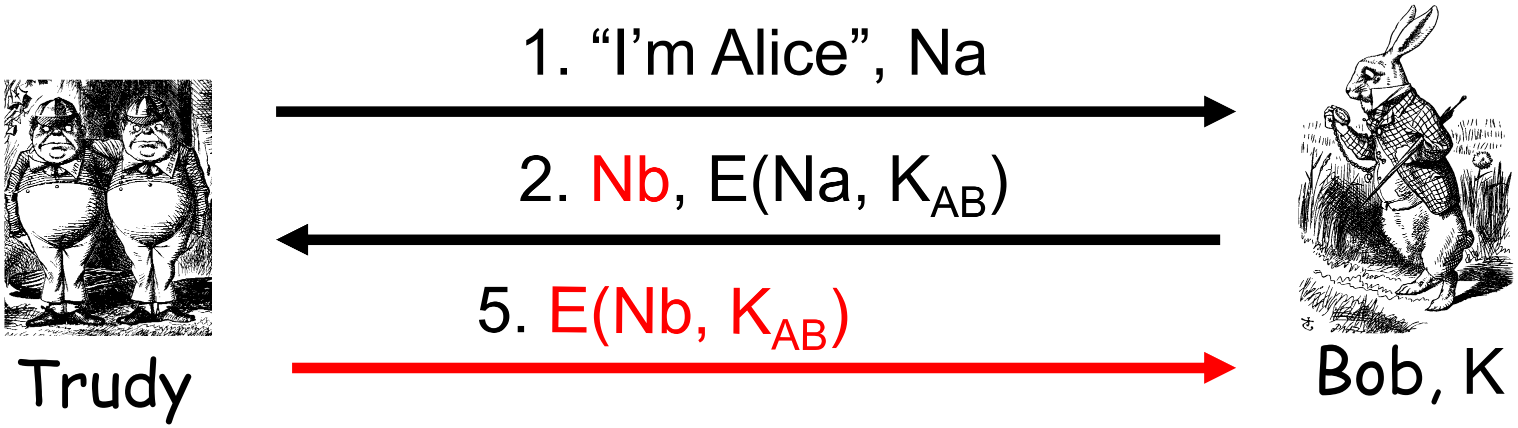
E(Nb, K_{AB})

Alice, K

Bob, K

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

Mutual Authentication Attack



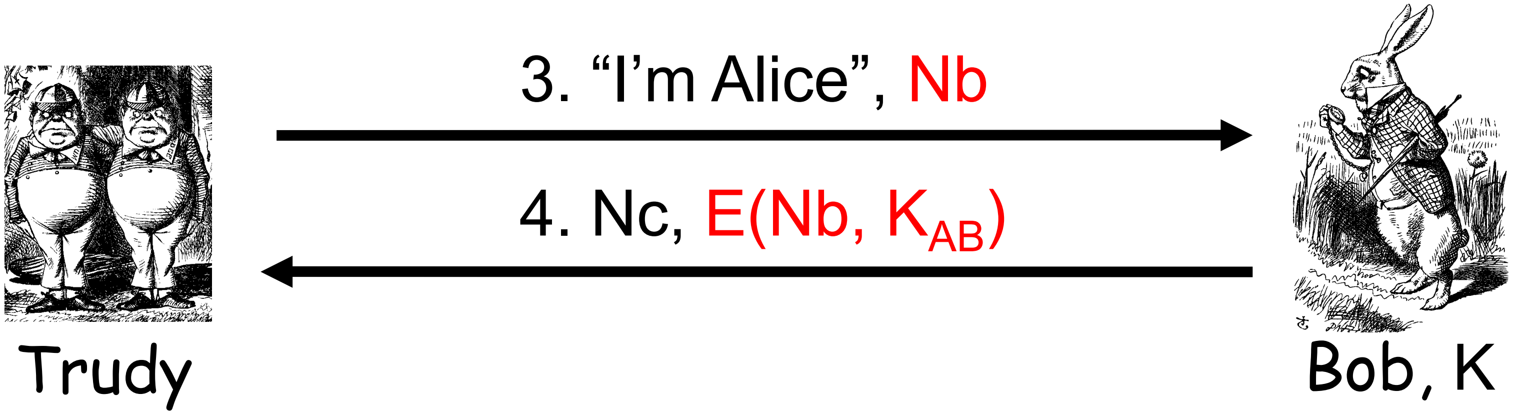
1. “I’m Alice”, Na

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

5. E(Nb, K_{AB})

Trudy

Bob, K



3. “I’m Alice”, Nb

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

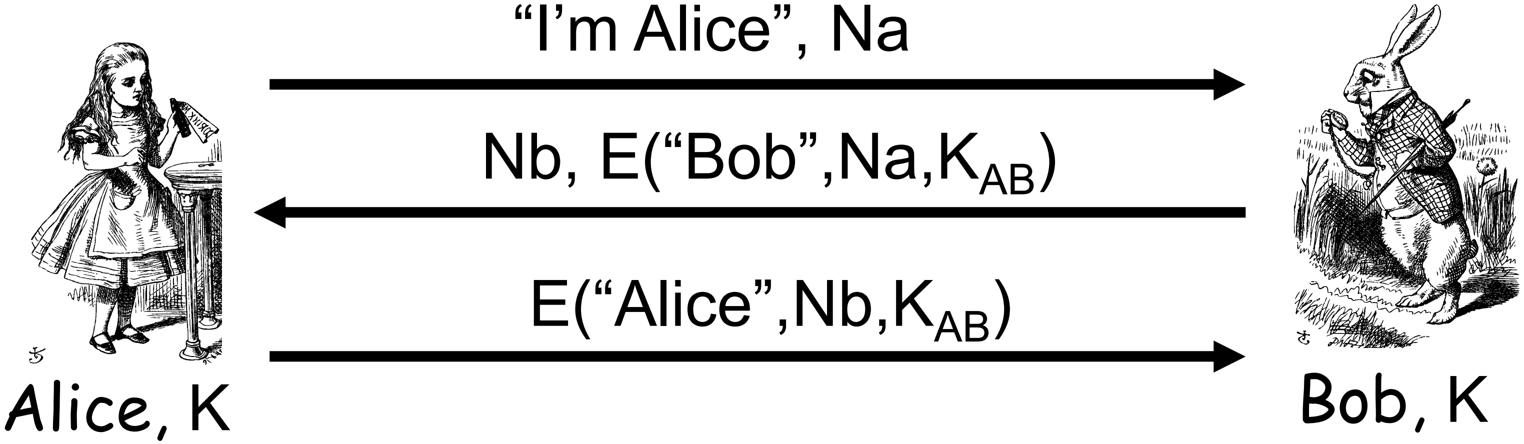
Trudy

Bob, K

Mutual Authentication

- 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

Symmetric Key Mutual Authentication



“I’m Alice”, Na

Nb, E(“Bob”,Na,K_{AB})

E(“Alice”,Nb,K_{AB})

Alice, K

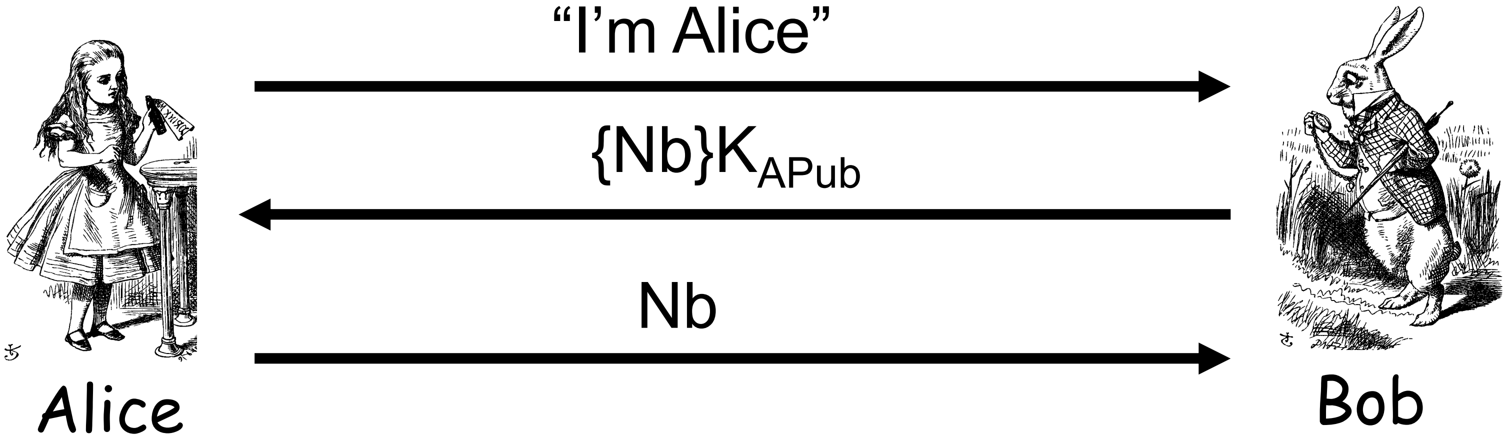
Bob, K

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

Public Key Notation

- Encrypt M with Alice’s public key: $\{M\}K_{APub}$
- Sign M with Alice’s private key: $\{M\}K_{APriv}$
- Then
  - $\{\{M\}K_{APub}\}K_{APriv} = M$
  - $\{\{M\}K_{APriv}\}K_{APub} = M$
- **Anybody** can use Alice’s **public key**
- Only **Alice** can use her **private key**

Public Key Authentication



"I'm Alice"

$\{Nb\}K_{APub}$

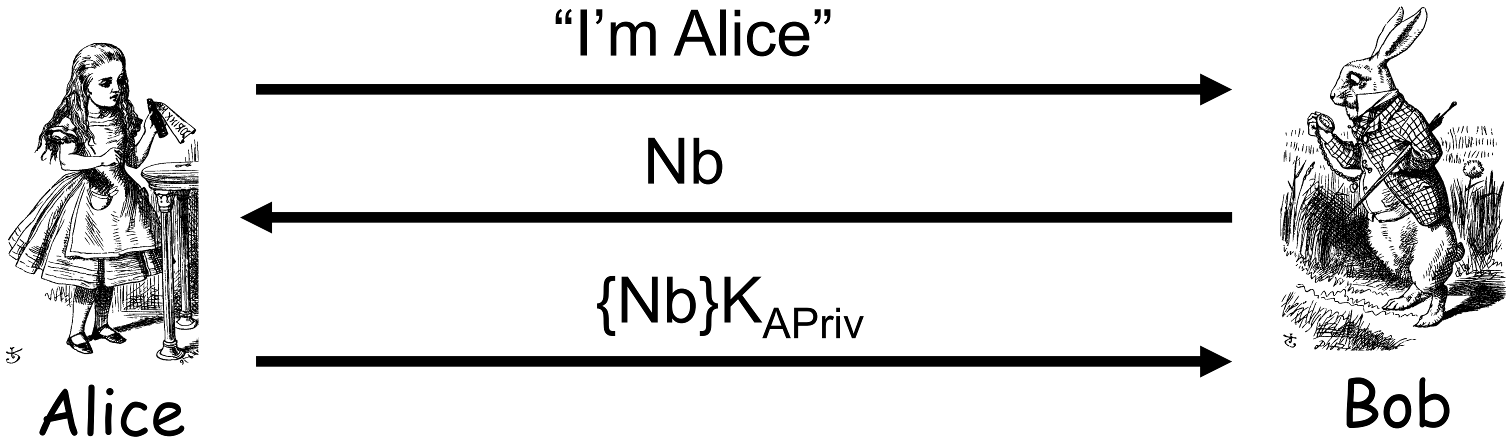
Nb

Alice

Bob

- Is this secure?
- Trudy can get Alice to decrypt anything!
  - Prevent this by having two key pairs

Public Key Authentication



"I'm Alice"

Nb

$\{Nb\}K_{APriv}$

Alice

Bob

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

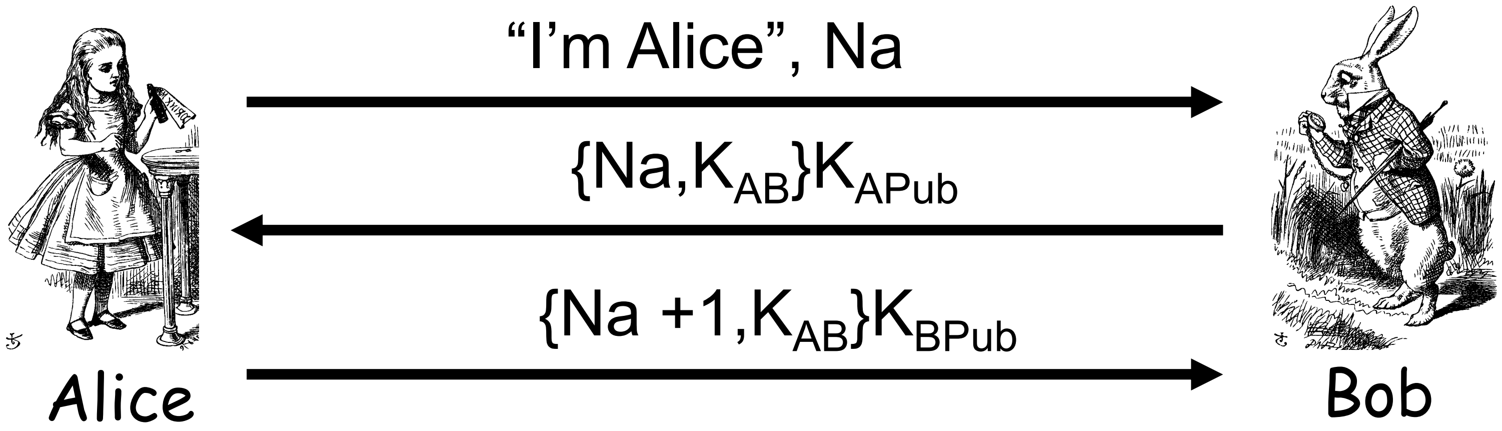
Public Keys

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

Session Key

- Usually, a **session key** is required
  - i.e., a symmetric key for current session
  - Used for confidentiality and/or integrity
- 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

Authentication & Session Key



"I'm Alice", Na

$\{Na, K_{AB}\}K_{APub}$

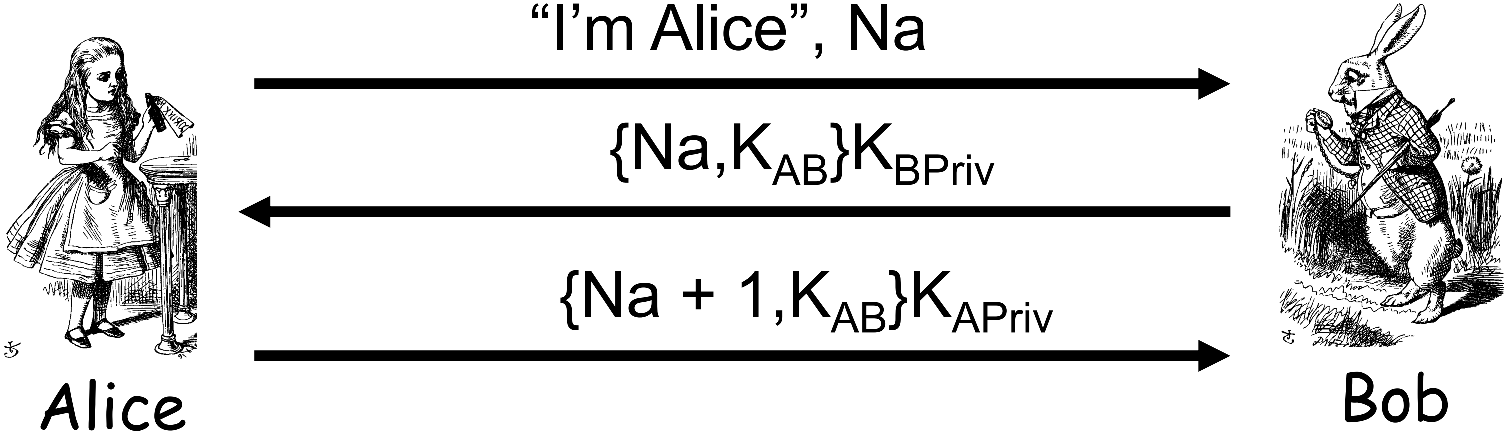
$\{Na + 1, K_{AB}\}K_{BPub}$

Alice

Bob

- Is this secure?
  - Alice is authenticated and session key is secure
  - Alice's "nonce", Na, useless to authenticate Bob
  - The key K_{AB} is acting as Bob's nonce to Alice
- No mutual authentication

Public Key Authentication and Session Key



"I'm Alice", Na

$\{Na, K_{AB}\}K_{BPriv}$

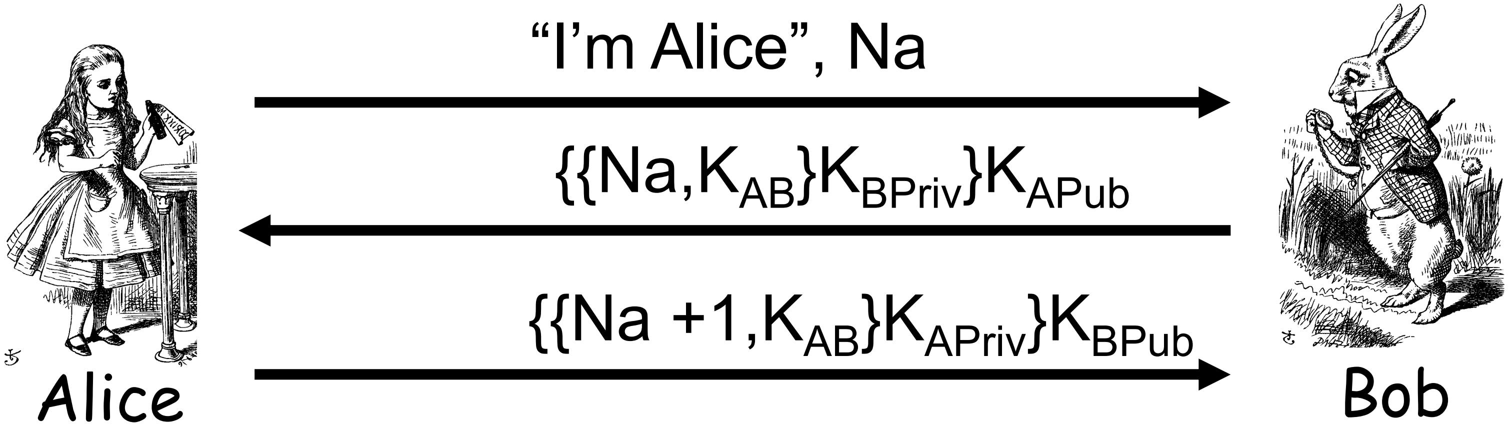
$\{Na + 1, K_{AB}\}K_{APriv}$

Alice

Bob

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

Public Key Authentication and Session Key



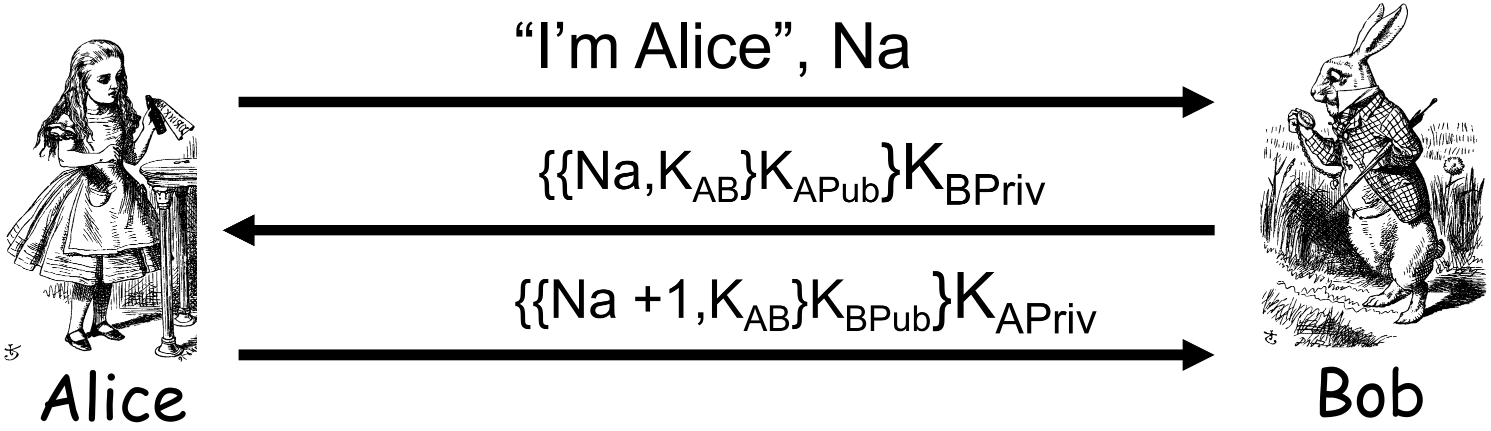
“I’m Alice”, Na

$\{\{Na, K_{AB}\}K_{BPriv}\}K_{APub}$

$\{\{Na +1, K_{AB}\}K_{APriv}\}K_{BPub}$

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

Public Key Authentication and Session Key



"I'm Alice", Na

$\{\{Na,K_{AB}\}K_{APub}\}K_{BPriv}$

$\{\{Na+1,K_{AB}\}K_{BPub}\}K_{APriv}$

- Is this secure?
- Seems to be OK
  - Anyone can see $\{Na,K_{AB}\}K_{APub}$ and $\{Na+1,K_{AB}\}K_{BPub}$

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)
  - 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?