Authentication Protocols and Key Establishment

Peter Sjödin

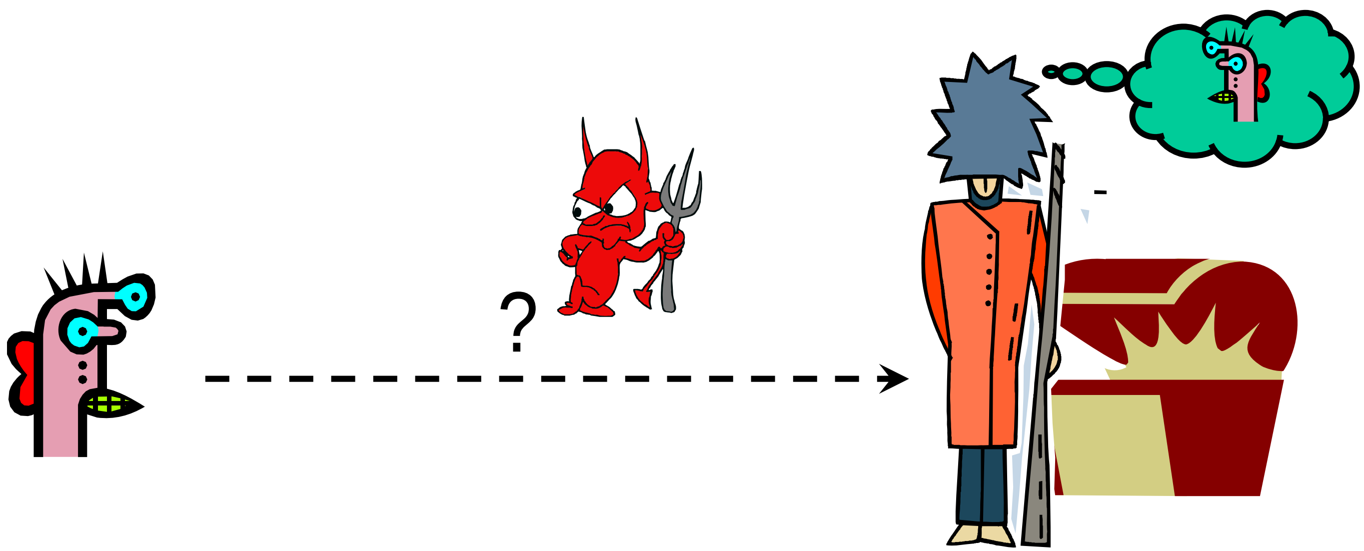
psj@kth.se

Based on material by Vitaly Shmatikov, Univ. of Texas, and by the previous course teachers

Authentication & handshakes

- Authentication & pitfalls
- Trusted intermediates
- Performance & randomness

Basic Problem

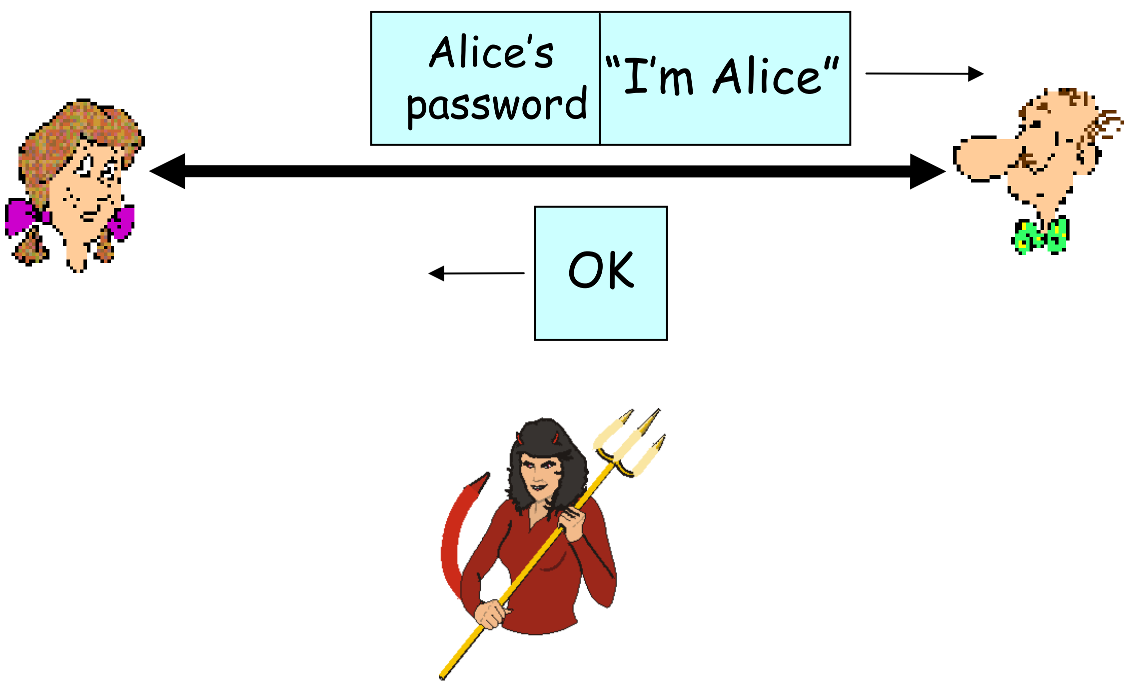


How do you prove to someone that
you are who you claim to be?

Any system with access control must solve this problem

Authentication: first attempt

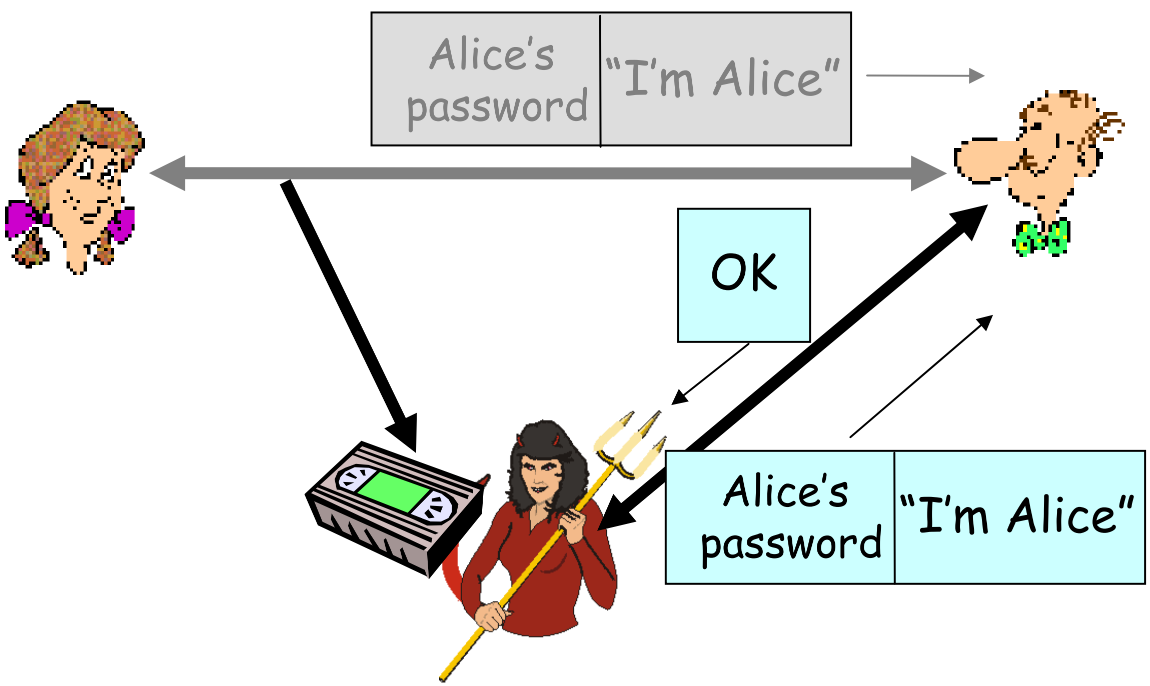
Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

Authentication: Playback Attack

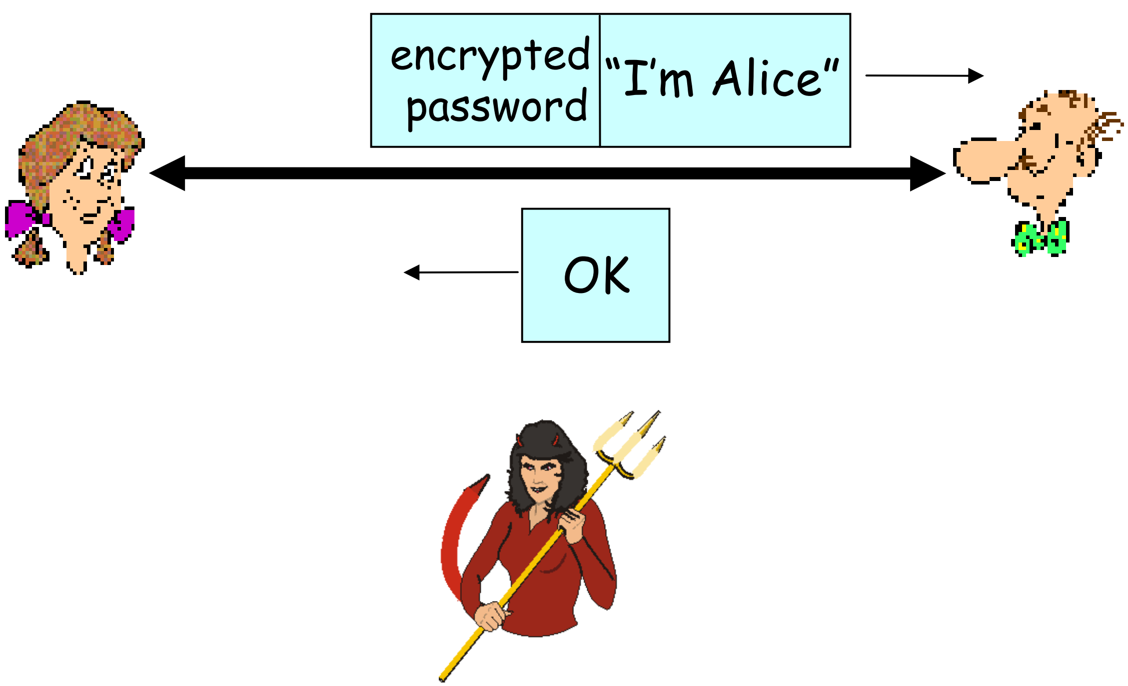
Alice says "I am Alice" and sends her secret password to "prove" it.



playback attack: Trudy records Alice's packet and later plays it back to Bob

Authentication: yet another try

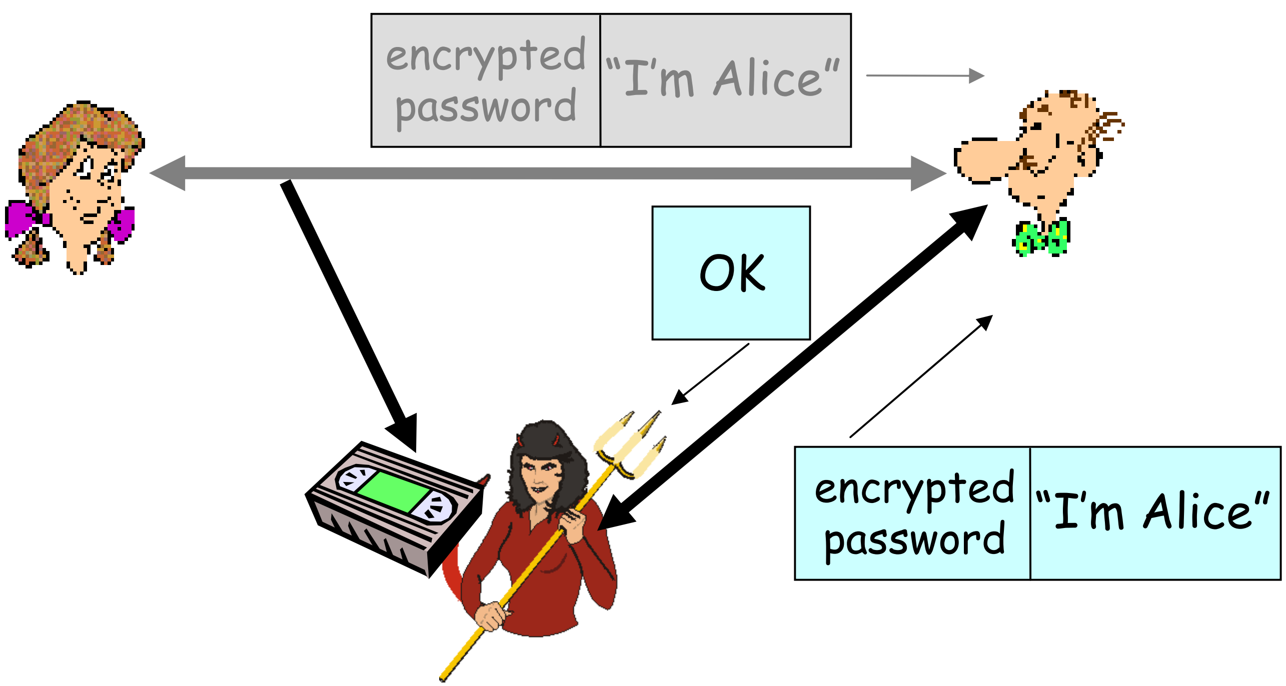
Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Failure scenario??

Authentication: another try

Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



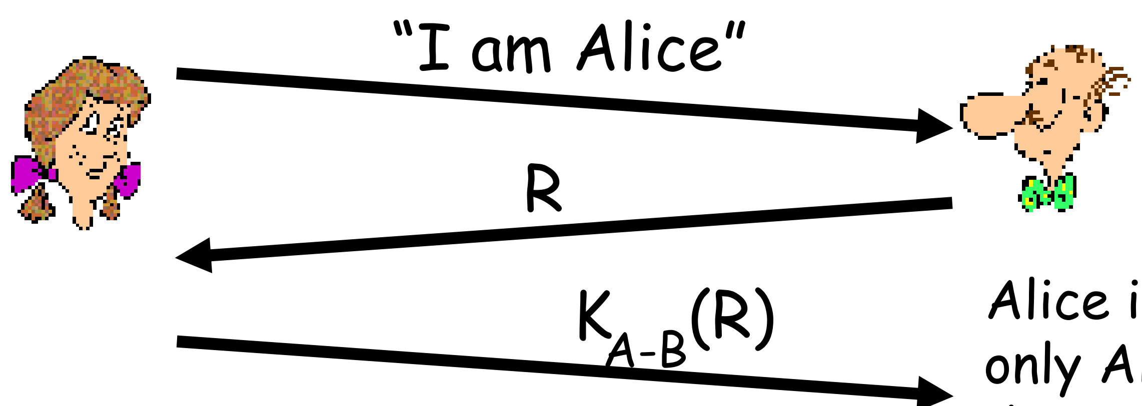
record and playback still works!

Authentication by Nonce Challenge

Goal: avoid playback attack

Nonce: number (R) used only *once–in-a-lifetime*

Bob sends Alice a nonce R. Alice must return R, encrypted with shared secret key



Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice! Right?

Failures, drawbacks?

Authentication by Nonce Challenge

Goal: avoid playback attack

Nonce: number (R) used only *once-in-a-lifetime*

Bob sends Alice a no… urn R, encrypted with shared secret key

- Bob isn't authenticated
- If the key is derived from a password, Trudy can mount a dictionary attack
- Trudy can hijack connection after authentication, if she can send packets with Alice's source address!

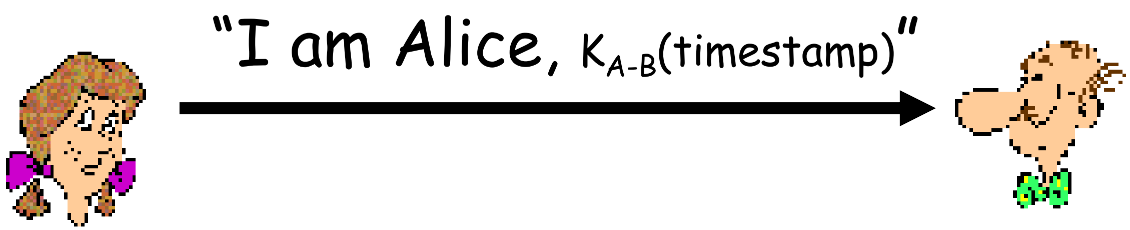
Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice! Right?

Failures, drawbacks?

Authentication: yet another try

Goal: avoid playback attack, efficiency

Alice encrypts a timestamp with shared secret key



Only Alice knows
key to encrypt
current time.

Failures, drawbacks?

Authentication: yet another try

Goal: avoid playback attack, efficiency

Alice encrypts a tim

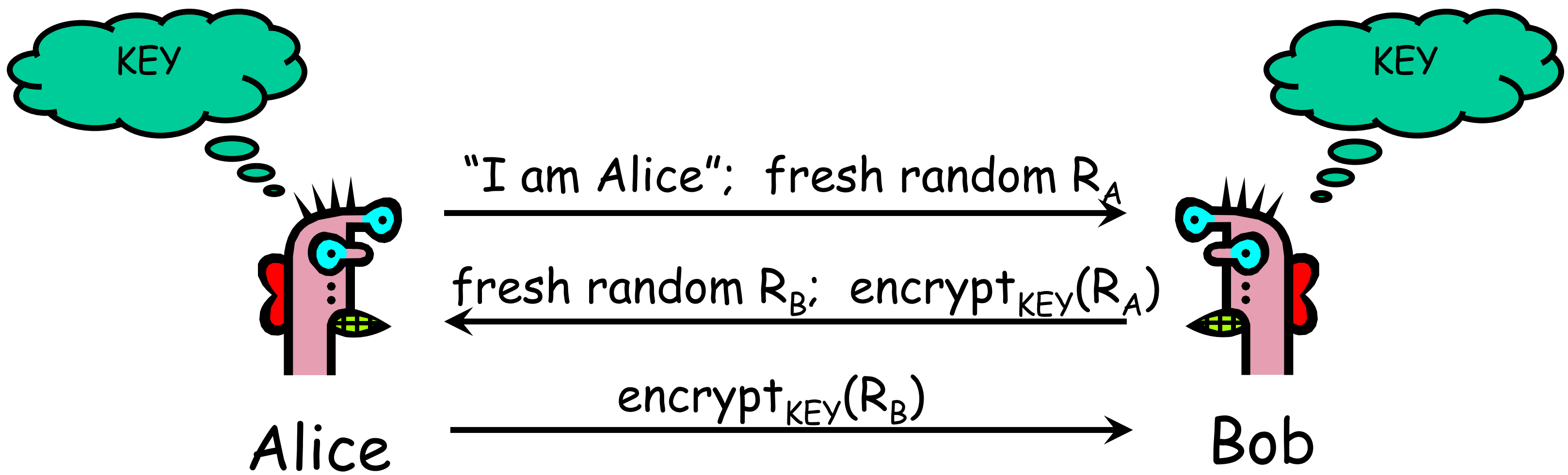
- Requires synchronization
- In practice, there must be some margins for time skew
  - During the time-skew window, there is room for play back attacks and for Trudy to impersonate Alice
- Bob is still not authenticated

“I

knows key to encrypt current time.

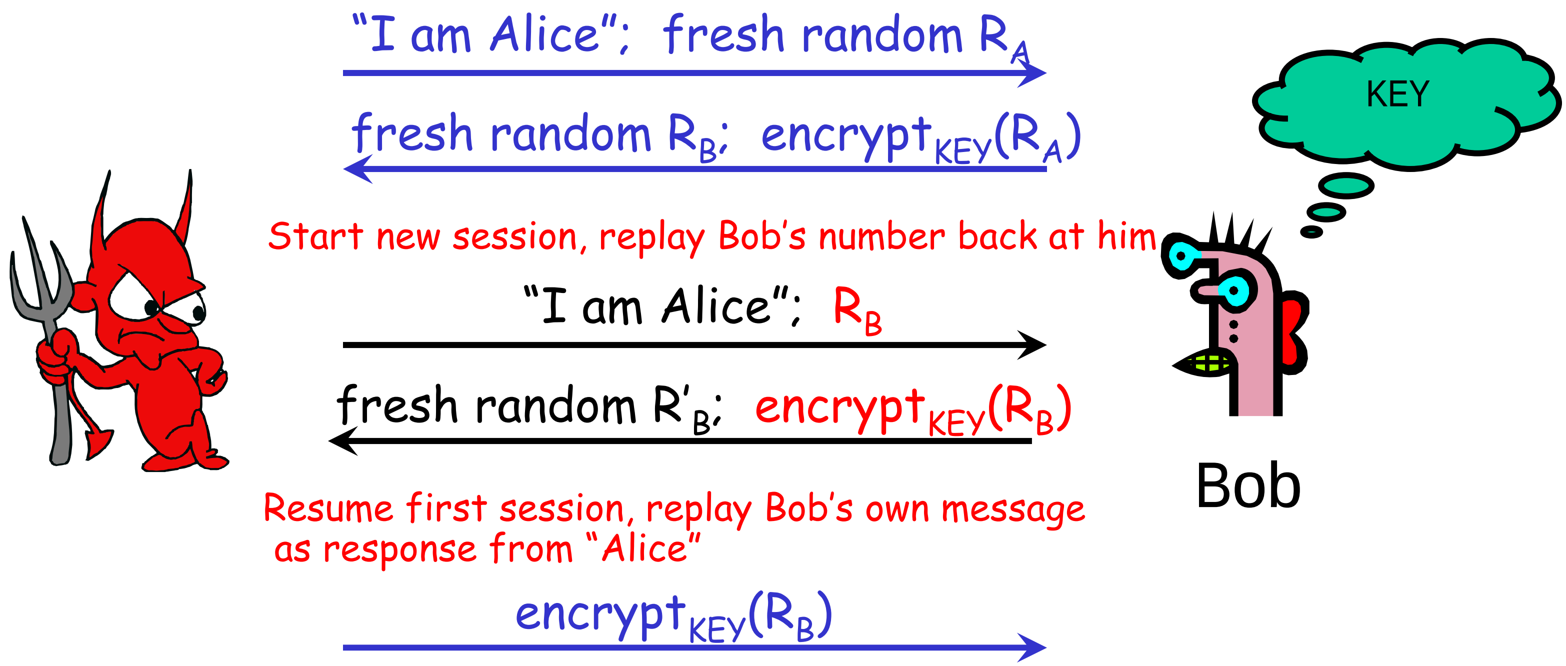
Failures, drawbacks?

Mutual Authentication With Symmetric Encryption



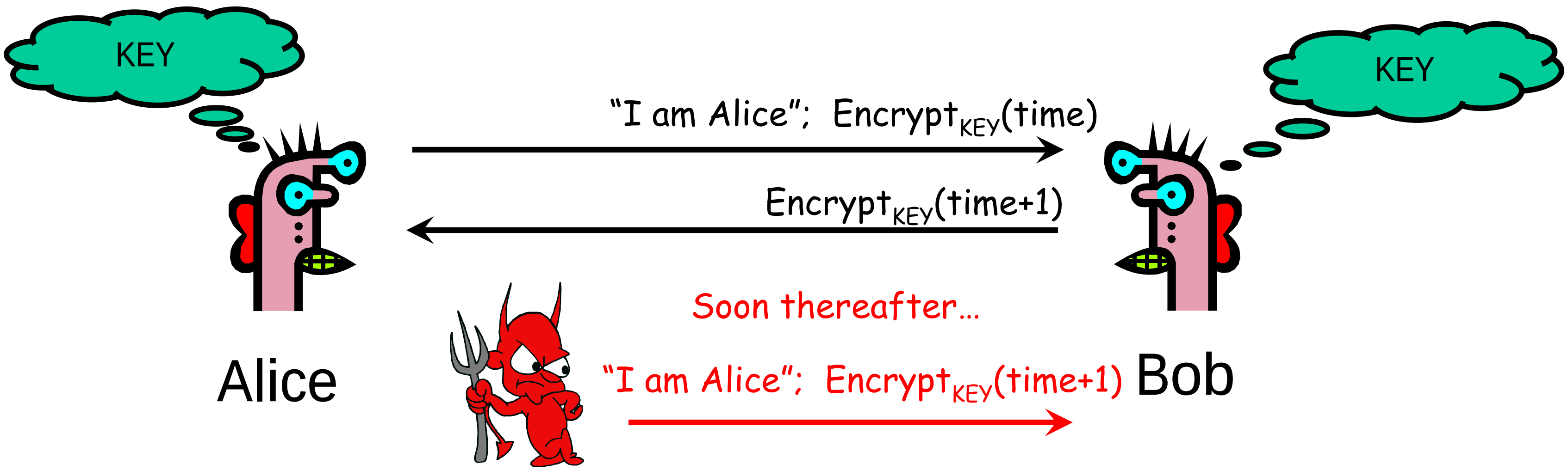
- <u>Mutual</u> authentication: Bob to Alice and Alice to Bob
- Bob's reasoning: I must be talking to Alice because…
  - Person who correctly encrypted R_B is someone who knows KEY… Only Alice knows KEY… Alice must have encrypted R_B… Because R_B is fresh, Alice can only know R_B if she received my message

Reflection Attack



- Bob's reasoning: I must be talking to Alice because…
  - Person who correctly encrypted R_B is someone who knows KEY… Only Alice knows KEY… No! Bob himself knows KEY, too!
- Security often fails because of flawed reasoning

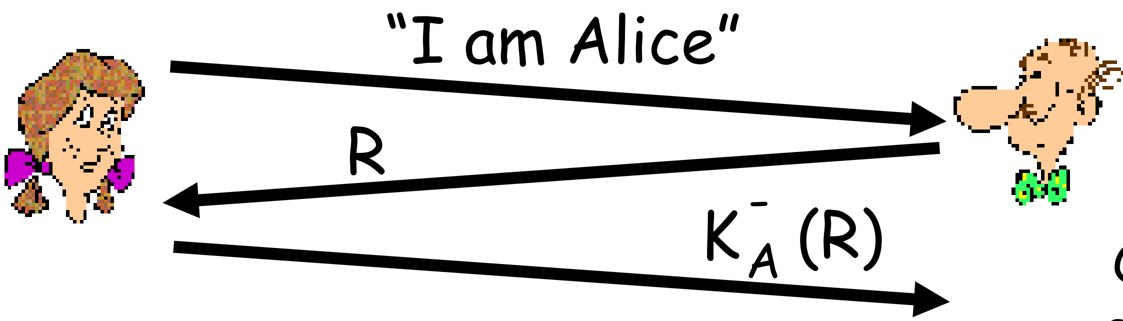
Timestamp Reflection



- Problem: same key for Alice and Bob
  - Attacker can get Bob to encrypt using Alice's key
- Problem: messages don't include intended recipient
- Problem: Bob doesn't remember his own messages

Authentication with Public Key

Use nonce, public key cryptography



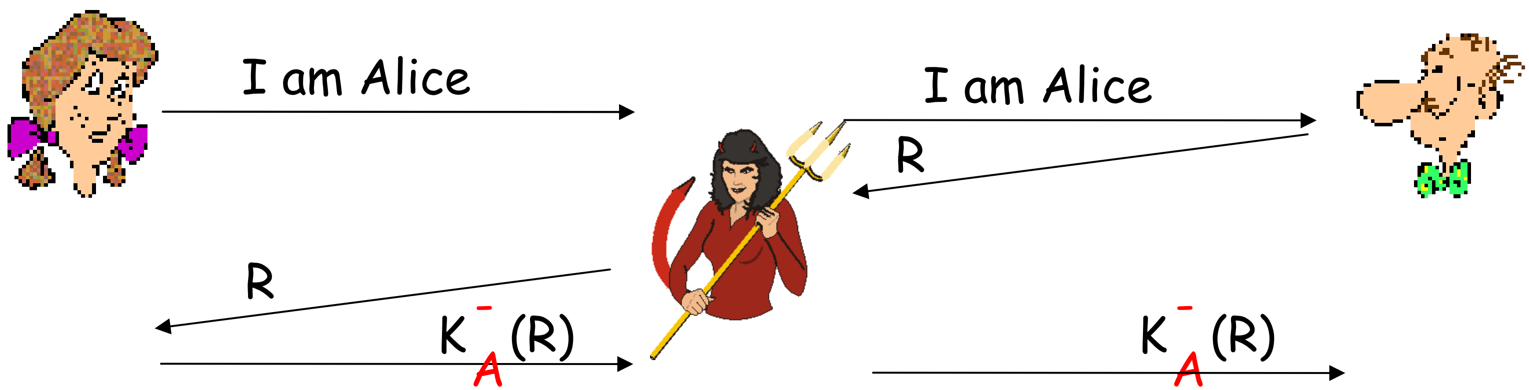
Bob checks that

$$K_A^+(K_A^-(R)) = R$$

Only Alice could have encrypted R that way!

Man-in-the-middle attack

Man (woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)

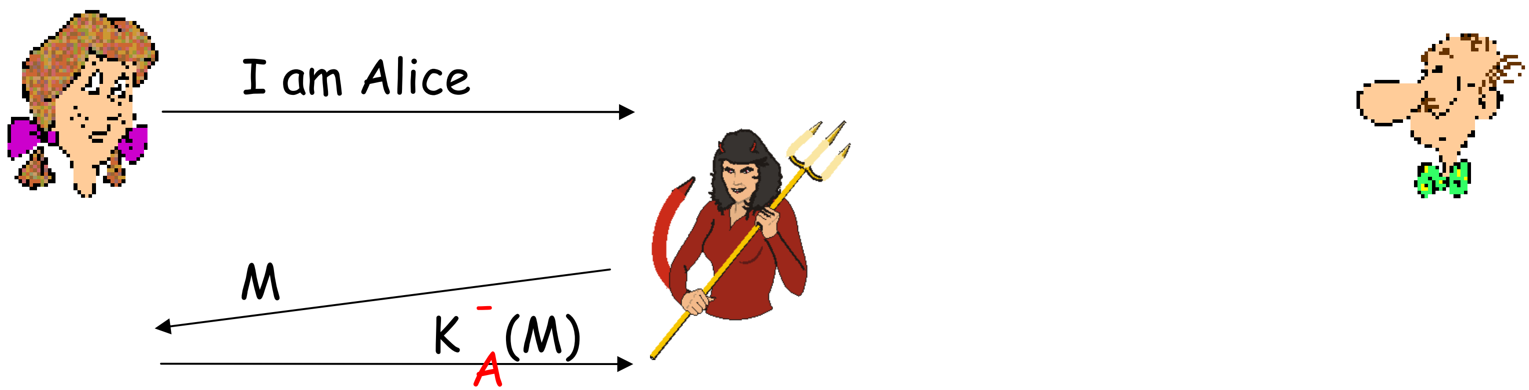


Difficult to detect:

- Trudy receives all messages as well
- Replay attacks
  - Must not allow Trudy to abuse the authentication to create bogus messages!

Encryption Tricking

Encryption tricking: Trudy tricks Alice to encrypt something with her private key



- Alice can be tricked into providing:
  - Digital signature
  - Message decryption
  - …
- Key reuse should be avoided
- Data should have structure
  - (so Alice would know what kind of data she signs, encrypts, etc)

Authentication & handshakes

- Authentication & pitfalls
- Trusted intermediates
- Performance & randomness

Trusted Intermediaries

Symmetric key problem:

- How do two entities establish shared secret key over network?

Solution:

- trusted key distribution center (KDC) acting as intermediary between entities

Public key problem:

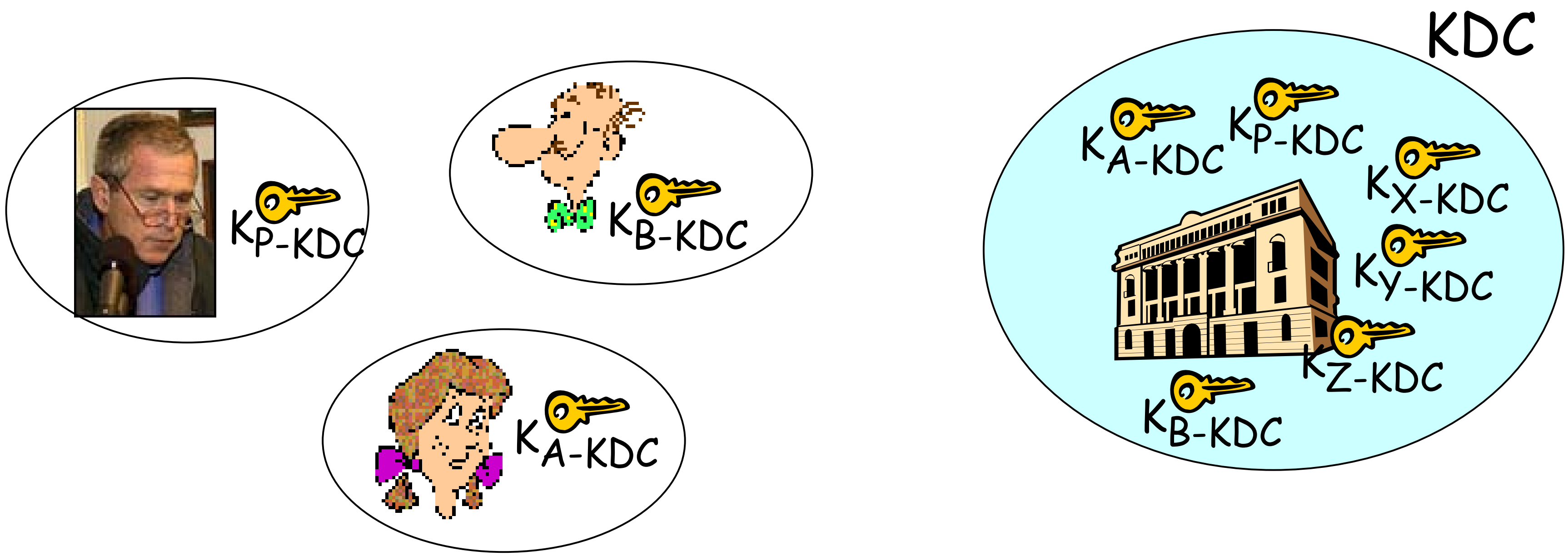
- When Alice obtains Bob's public key (from web site, e-mail, diskette), how does she know it is Bob's public key, not Trudy's?

Solution:

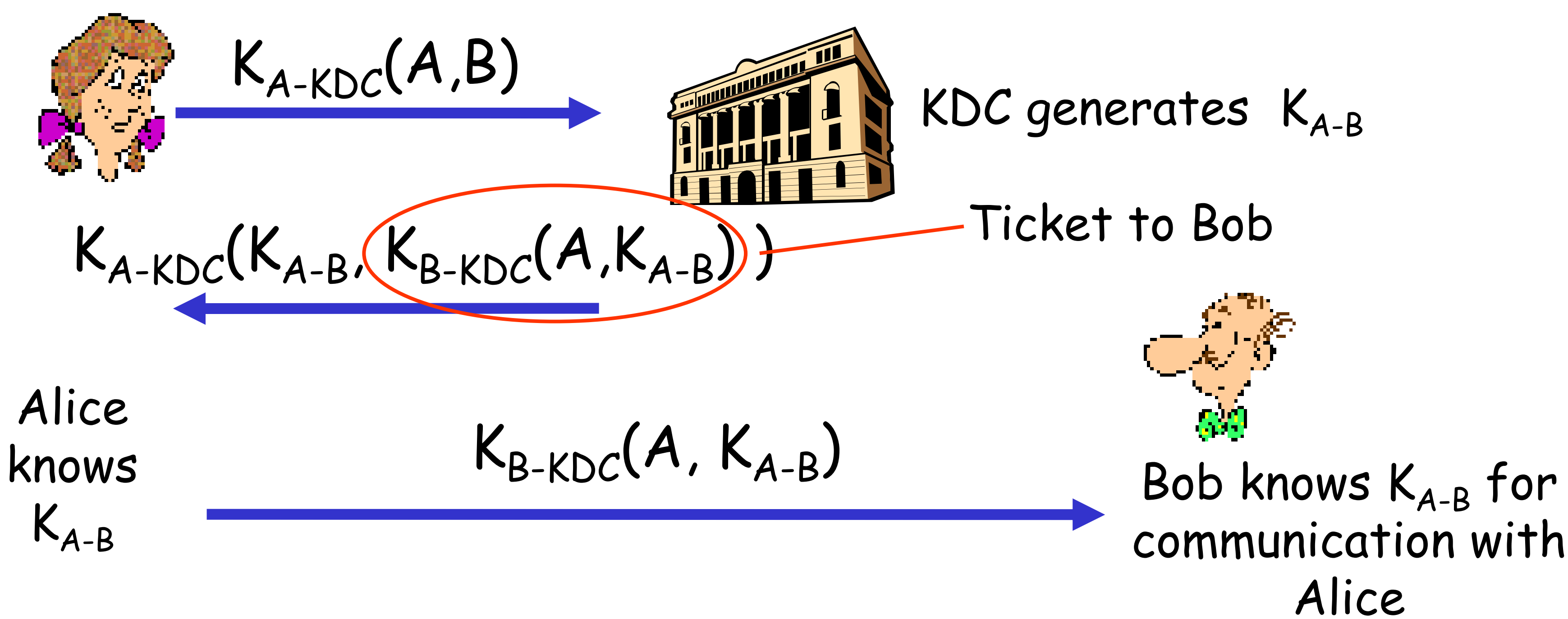
- trusted certification authority (CA)

Key Distribution Center (KDC)

- Alice, Bob need shared symmetric key.
- KDC: server shares different secret key with *each* registered user (many users)
- Alice, Bob know own symmetric keys, $K_{A\text{-}KDC}$ $K_{B\text{-}KDC}$, for communicating with KDC.

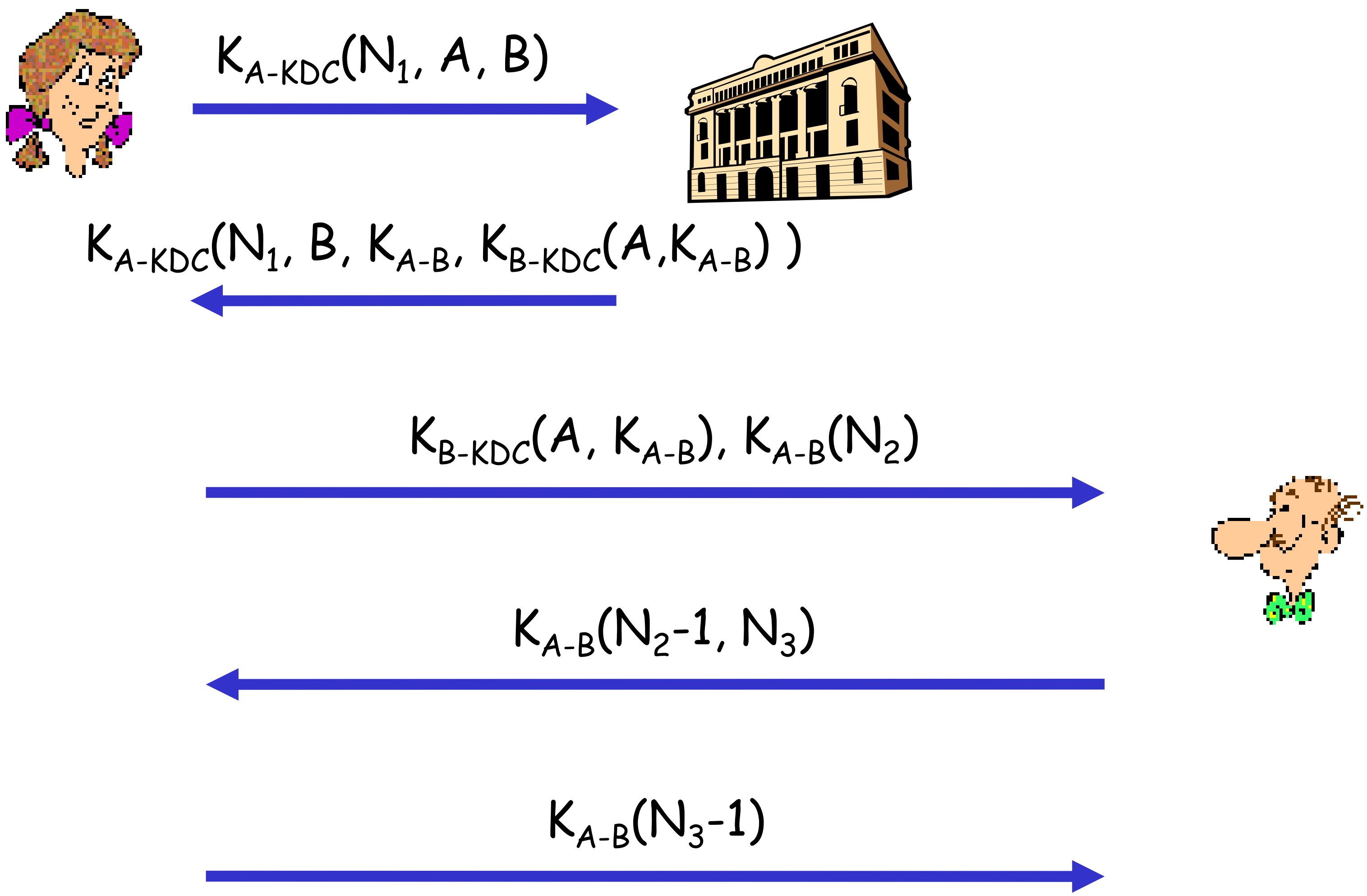


Key Distribution Center (KDC)

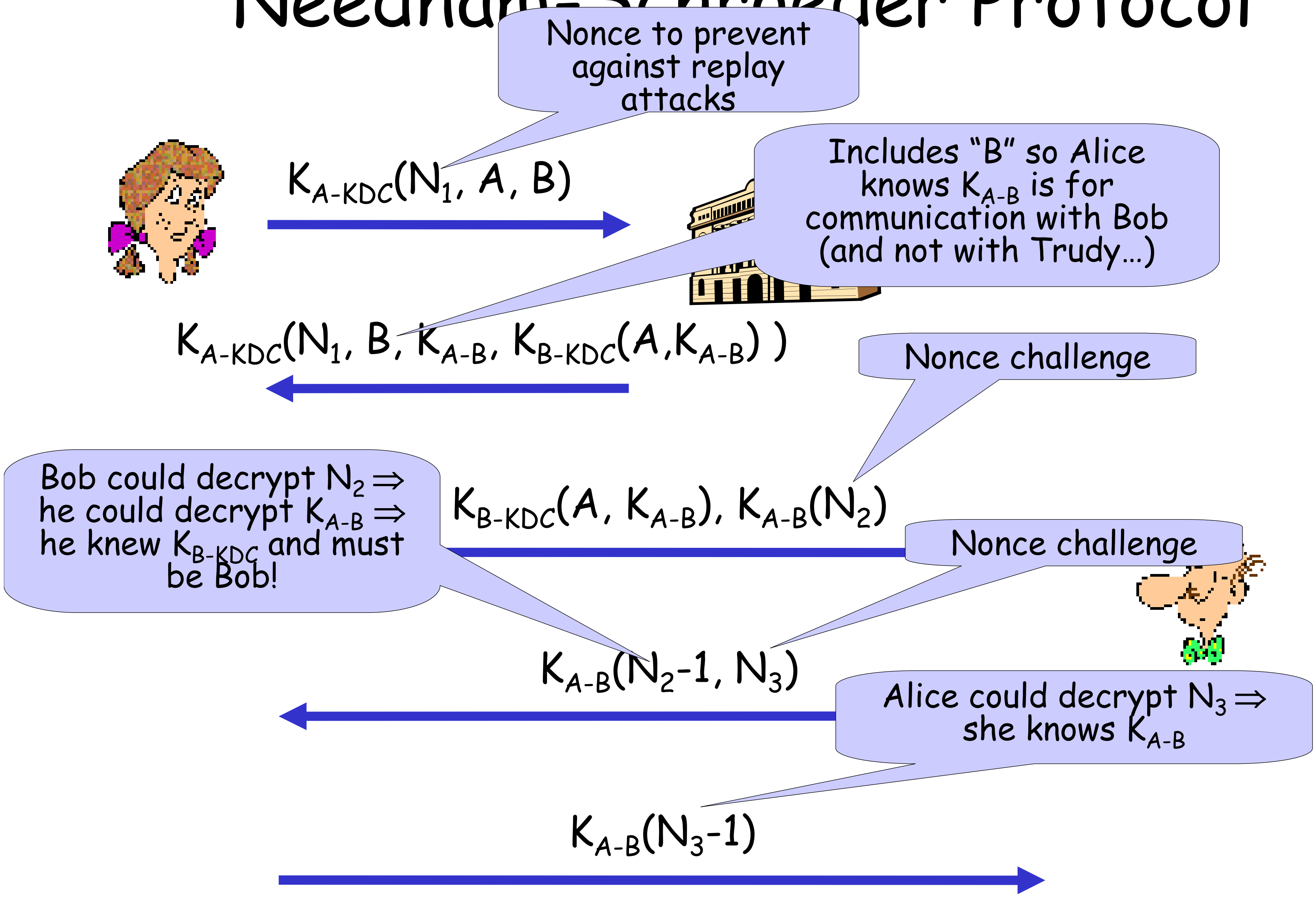


Alice and Bob can communicate using K_{A-B} as *session key* for shared symmetric encryption

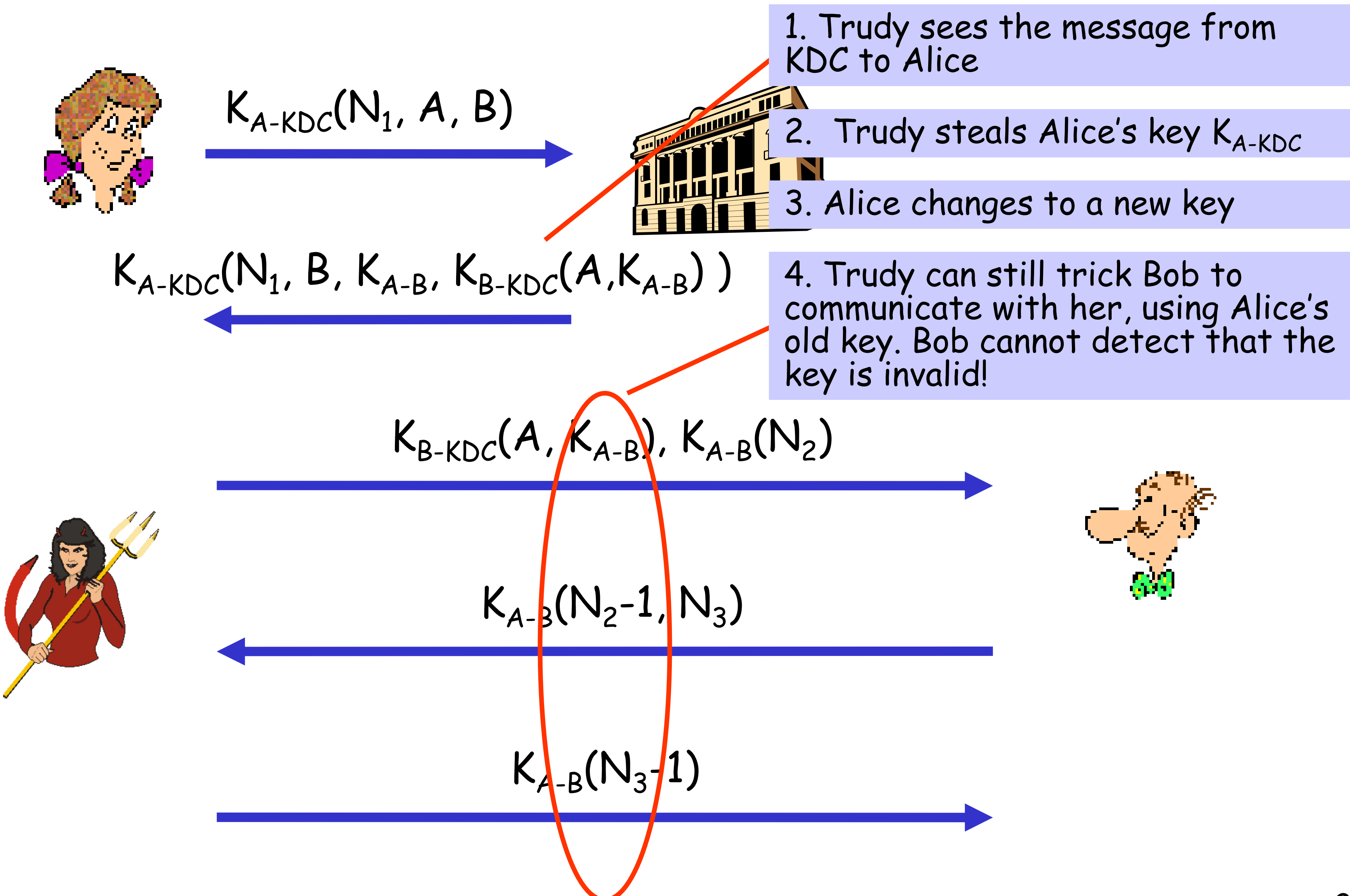
Needham-Schroeder Protocol



Needham-Schroeder Protocol



Ticket Invalidation Problem

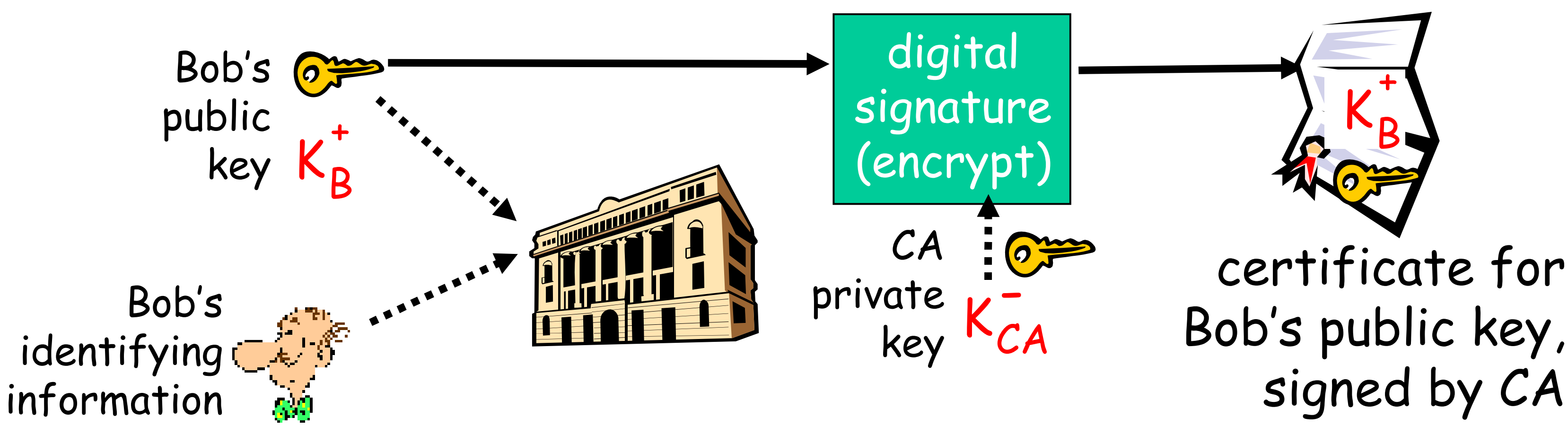


Key Distribution Center (KDC)

- Many subtle problems to consider
  - How to prevent key reuse by attacker?
  - What kind of nonce do we use *(more later)*

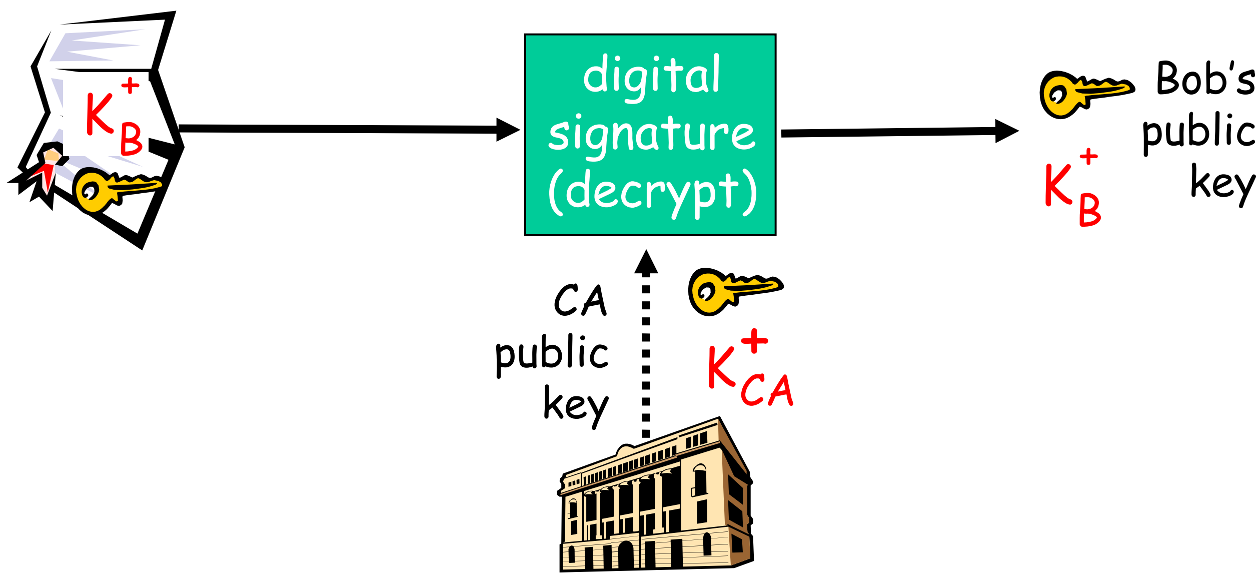
Certification Authorities

- Certification authority (CA): binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA
    - CA says "this is E's public key"



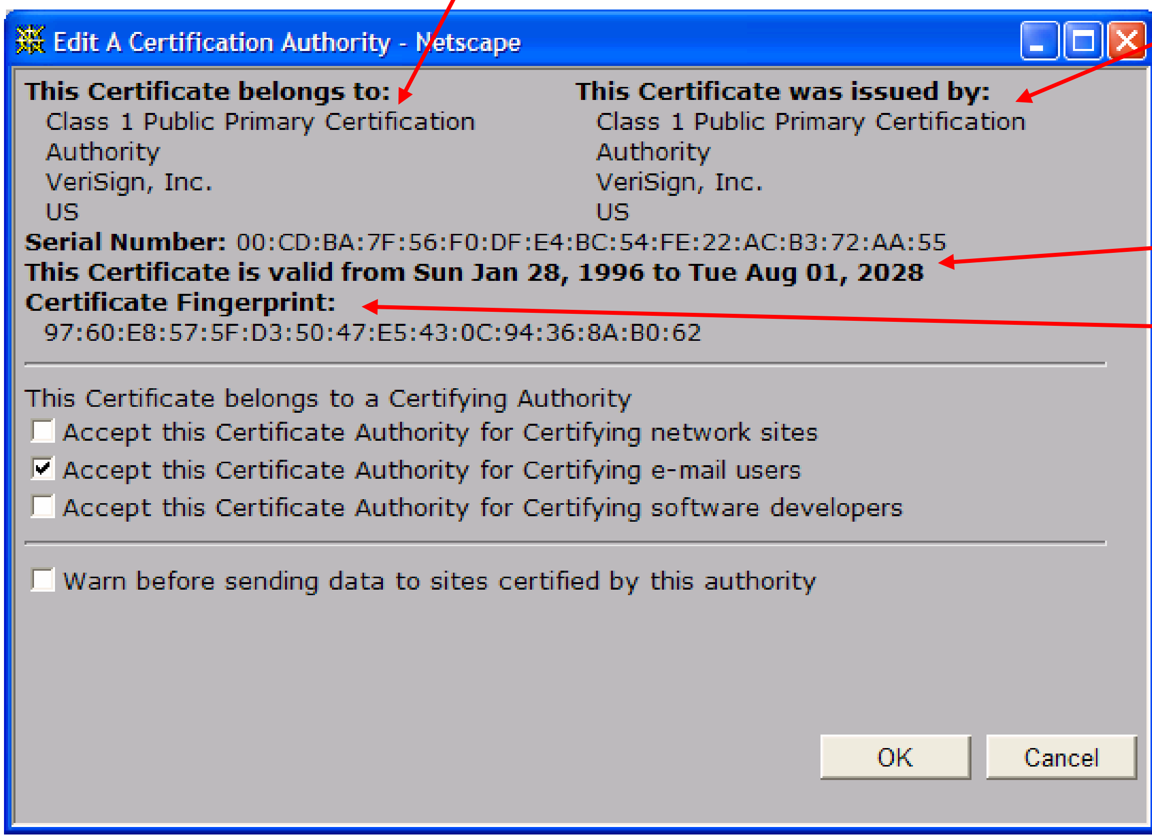
Certification Authorities

- When Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key



A certificate contains:

- Serial number (unique to issuer)
- info about certificate owner, including algorithm and key value itself (not shown)



- info about certificate issuer
- valid dates
- digital signature by issuer

Certificates & CAs

- No prior arrangement needed with peer
- CA does not need to be online for verification
  - Peers can distribute their own certificates
  - Only need to have received CA's public key securly at some point in the past
  - Note: Online CA required to support revocation!

Authentication & handshakes

- Authentication & pitfalls
- Trusted intermediates
- Performance & randomness

Performance and randomness

- Performance issues force us to encrypt only as much as is absolutely necessary
  - Encryption and decryption times can differ
    - RSA 2048 decrypt = 60 * RSA 2048 encrypt
  - Public key cryptos often orders of magnitude more expensive than symmetric crypto
- Message exchanges are also very expensive
  - ~100ms RTT to the US
  - Limit number of messages, piggybacking
    - Sometimes leads to weaknesses!

Performance and randomness

- Randomness is very important!
  - True randomness is hard to achieve
    - Radioactive decay
    - Source of white noise
  - Pseudo-randomness = pseudo-security
    - Need good seeds to pseudrandom generators
    - Seeds must not be exposed!
    - “Normal” rand()-functions not random enough!
- Who controls source of randomness?

Nonces – Numbers used once

- Nonces injects "noise" into protocols
  - Sequence numbers
  - Time stamps
  - Large random numbers
- Sequence numbers and time stamps can be attacked by guessing and may repeat
  - If you know N_X, you can guess N_{X+1}
  - Setting back clock/lost state

Summary

- Like cryptographic algorithms, the design of an authentication algorithm requires a lot of skill and care – *don't do this at home!*
- Using KDCs and CAs allows communication with unrelated peers – trusted intermediate
- Performance considerations impact on the design of authentication algorithms
- Randomness is both tricky to achieve and important for the result