LECTURE 9:

Authentication
&
Key Distribution

1

1

Where are we now?

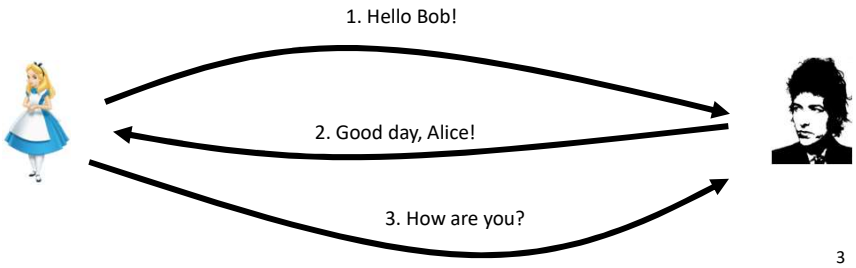
- We “know” a bit of the following:
  - Conventional (symmetric) cryptography
  - Hash functions and MACs
  - Public key (asymmetric) cryptography
    - Encryption
    - Signatures
    - Identification (Fiat-Shamir) + Zero Knowledge
- And now what?
  - Protocols (more “complicated” beasts)
    - **Authentication**
    - **Key Distribution**

2

2

Secure Protocols

- A ***protocol*** is a set of rules for exchanging *messages* between 2 or more ***entities/parties***
- A protocol has a number of ***rounds*** (>1) and a number of ***messages*** (>1)



3

3

Secure Protocols

- A **message** is a unit of information/data sent from one entity/party to another as part of a protocol
- A **round** is a basic unit of protocol time:
  1. *Wake up because of:*
     a) *Alarm clock*
     b) *Initial start or*
     c) *Receive message(s) from other(s)*
  2. *Compute something*
  3. *Send message(s) to others*
  4. *Repeat steps 2-3, if needed*
  5. *Wait for message(s) or sleep until alarm clock*

4

4

What's a *Secure Protocol*?

- When acting honestly, *entities=parties*=participants achieve the stated **goal** of the protocol, e.g.,:
  - A successfully authenticates to B, or B to A
  - A and B mutually authenticate each other
  - A and B exchange a fresh session key
- Adversary can try to defeat this goal
  - e.g., by successfully impersonating A in an authentication protocol with B

5

5

The Entities (2-Party Setting)

- **Alice** and **Bob**
  - want to mutually authenticate and/or share a key
- **Eve**, the adversary
  - passive or active
- More complex protocols may involve a **Trusted Third Party (TTP)**
  - 3rd party trusted by both Alice and Bob

6

6

Will focus on 2 types of protocols

- Authentication
- Key Distribution

7

7

Will focus on 2 types of protocols

- **Authentication**
- Key Distribution

8

8

Definitions

- **Entity** Authentication:
  - Confirm/validate that an entity is the one claimed

Entity Authentication has two types:

- **Unilateral** Authentication:
  - entity authentication that provides one entity with assurance of the other's identity, but not vice versa

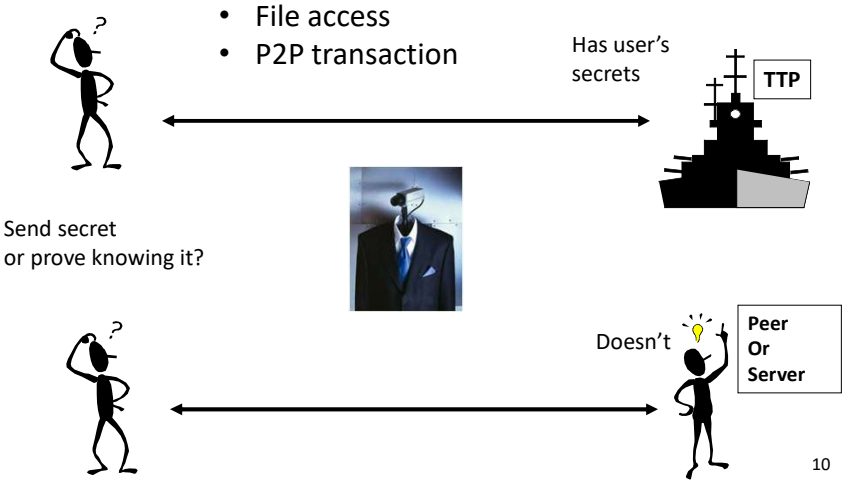
- **Mutual** Authentication:
  - entity authentication that provides both entities with assurance of each other's identity

9

Purpose

Examples:

- Bank transactions, e.g., cash withdrawals
- Remote login
- File access
- P2P transaction



10

Basis for Authentication

- Something you **know**, such as PIN or password
- Something you **have:**
  - A secure token, e.g., that generates a one-time password
  - Key embedded in a "secure area" on a computer, in browser software, etc.
  - A smartcard, which may contain keys and can perform cryptographic operations on behalf of a user
- Something you **are** (a biometric)

11

11

Concrete Scenarios

- PIN-, PW-based schemes
  - Kerberos
- Biometric-based schemes
  - Iris/retina scanners
  - Thumbprint & hand/palmprint
  - Handwriting acceleration & pressure
- Secure token-based schemes
  - SecureID tokens, Duo mobile (sw version)
- Authentication protocols
  - Fiat-Shamir
  - Conventional- and public key-based (covered later)

12

12

Concrete Scenarios

- **PIN-, PW-based schemes**
  - Kerberos
- Biometric-based schemes
  - Iris/retina scanners
  - Thumbprint & hand/palmprint
  - Handwriting acceleration & pressure
- Secure token-based schemes
  - SecureID tokens
- Authentication protocols
  - Fiat-Shamir
  - Conventional- and public key-based (covered later)

13

13

Human Failings

- Humans are notoriously unreliable
- Human memory is very volatile storage



- What a human can remember:
  - PIN (no more than 6-8 digits)
  - Password (a word or a short phrase)
- Can a human do single-digit sums? Forget it …

14

14

Concrete Scenarios

- PIN-, PW-based schemes
  - Kerberos
- **Biometric-based schemes**
  - Iris/retina scanners
  - Thumbprint & hand/palmprint
  - Handwriting acceleration & pressure
- Secure token-based schemes
  - SecureID tokens
- Authentication protocols
  - Fiat-Shamir
  - Conventional- and public key-based (covered later)

15

15

Biometrics

- Retinal scanner, fingerprint reader, handprint reader, voiceprint, keystroke timing, signature (shape or pressure), etc.
- Accuracy:
  - False Acceptance Rate (False Positive)
  - False Rejection Rate (False Negative)
- Applicability
- Robustness

16

16

Fingerprints

- Vulnerability:
  - Dummy fingers
  - Lost fingers
- Applicability issues:
  - Not for people with high probability of damaged fingerprints (e.g., eczema)
  - Not for kids who are still growing
- Robustness issues:
  - Many potential noise sources: thermal and optical noise, temperature affecting skin condition …

17

17

Voice Recognition

- Single fixed phrase:
  - Can use tape recorder to fake
- Robustness issues:
  - Background noise
  - Colds, vocal cord damage/strain, laughing gas ☺
  - Use with public phones

18

18

Keystroke Timing

- Each person has a distinct typing timing and style
  - Hand/finger movements
- Applicability issues:
  - Best done for “local” authentication
    - Avoid network traffic delay

19

19

(Non-digital) Signatures

- Accuracy issues:
  - machines can not (yet) match human experts in recognizing shapes of signatures
- Can add information on acceleration and/or pressure
  - Signing on a special electronic tablet

20

20

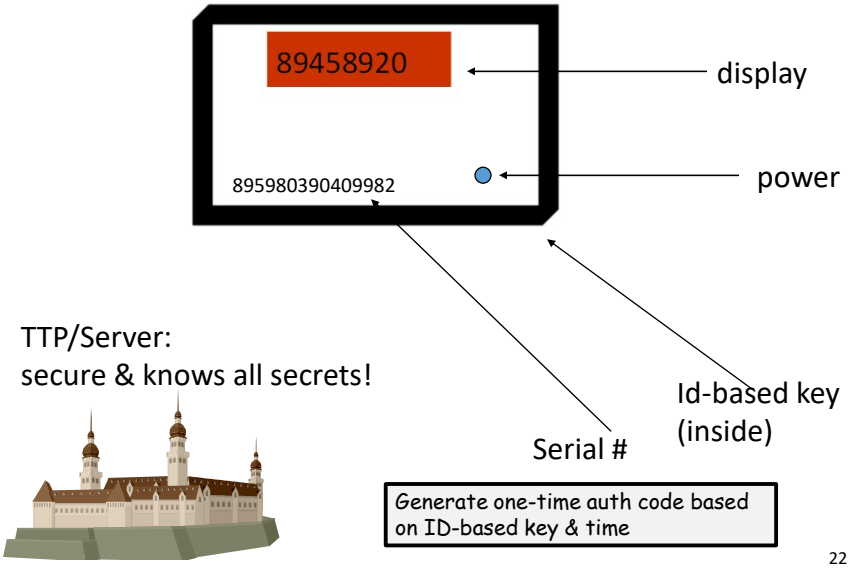
Concrete Scenarios

- PIN-, PW-based schemes
  - Kerberos
- Biometric-based schemes
  - Iris/retina scanners
  - Thumbprint & hand/palmprint
  - Handwriting acceleration & pressure
- **Secure token-based schemes**
  - SecureID tokens
- Authentication protocols
  - Fiat-Shamir
  - Conventional- and public key-based (covered later)

21

21

SecureID/Secure-Token

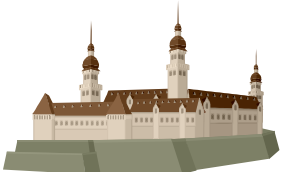


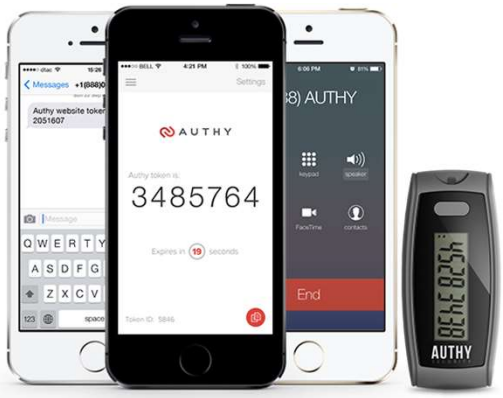
22

22

SecureID/Secure-Token

TTP/Server:
secure & knows all secrets!





Generate one-time auth code based on ID-based key & time

23

Concrete Scenarios

- PIN-, PW-based schemes
  - Kerberos
- Biometric-based schemes
  - Iris/retina scanners
  - Thumbprint & hand/palmprint
  - Handwriting acceleration & pressure
- Secure token-based schemes
  - SecureID tokens
- **Authentication protocols**
  - Fiat-Shamir
  - Conventional- and public key-based (detailed next)

24

Authentication (Protocols)

Since they communicate over a network, Bob cannot "see" Alice. So, Eve simply declares herself to be Alice

Protocol ap1.0: Alice says "I am Alice"









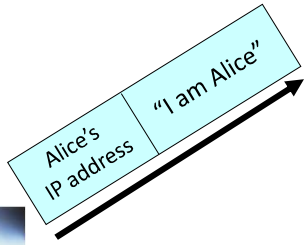
Authentication: Another Try

Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address





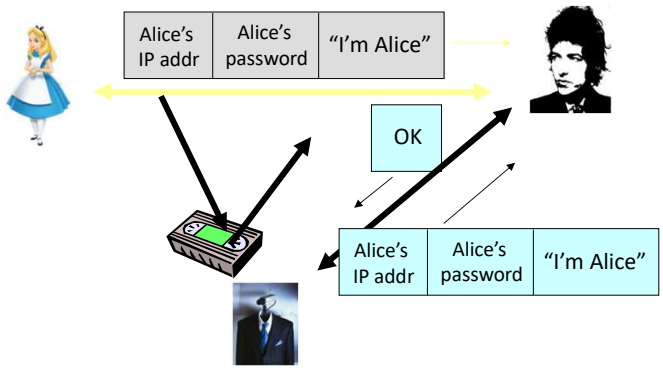




Eve can create a packet "spoofing" Alice's address

Authentication: Another Try

Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.

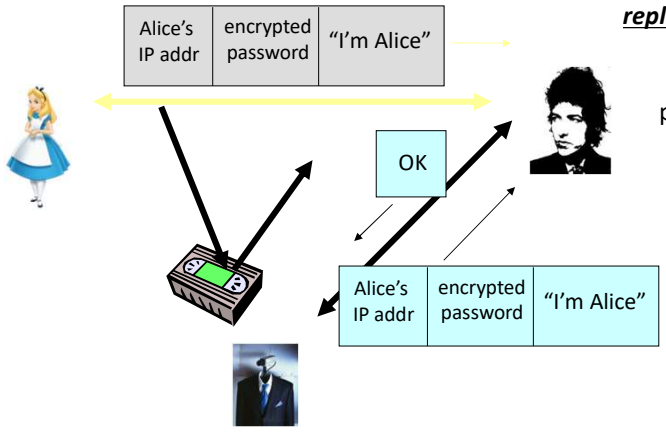


27

27

Authentication: Another Try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



replay attack: Eve records Alice's packet and later plays it back to Bob

28

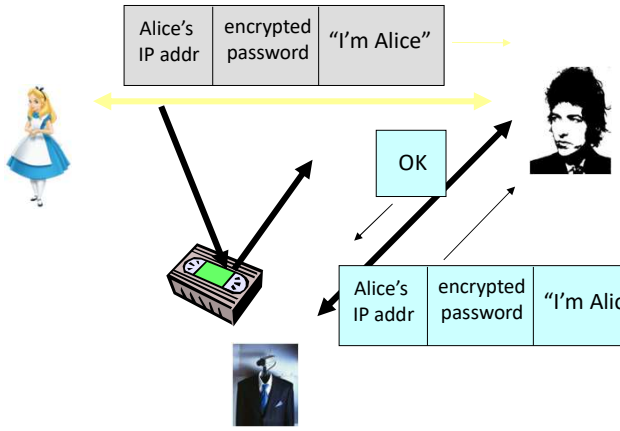
28

Authentication: Another Try

Protocol ap3.1: Alice says "I ar
encrypted secret p

Important! A common attack model when analyzing *protocol security*

replay attack: Eve records Alice's packet and later plays it back to Bob



Need sth that can't be replayed → sth only used once

29

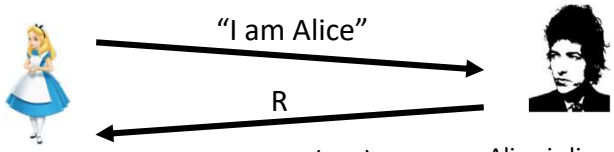
29

Authentication: Yet Another Try

Goal: avoid replay attack

Nonce: number used *once (R)*

ap4.0: to prove Alice "live", Bob sends Alice 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!

30

30

Authentication: ap5.0

ap4.0 requires shared symmetric key

• can we authenticate using public key?

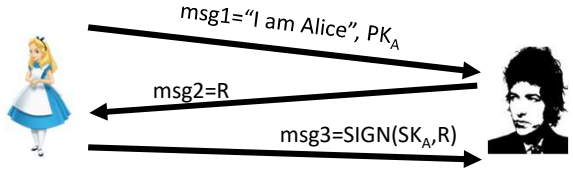
31

Authentication: ap5.0

ap4.0 requires shared symmetric key

• can we authenticate using public key?

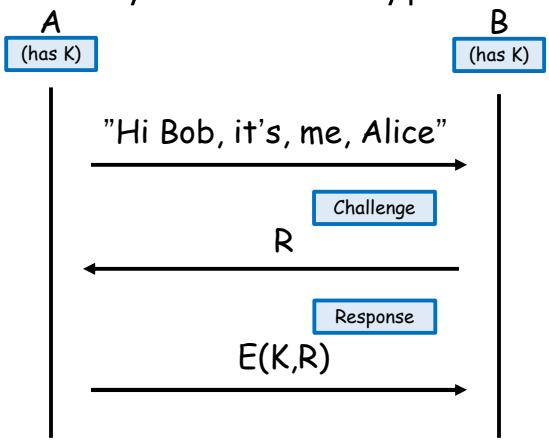
ap5.0: nonces and public key cryptography



Using PK_A, Bob verifies Alice's signature of R in msg3. Since R is fresh and only Alice can compute signatures using SK_A, Bob concludes that Alice is really there.

32

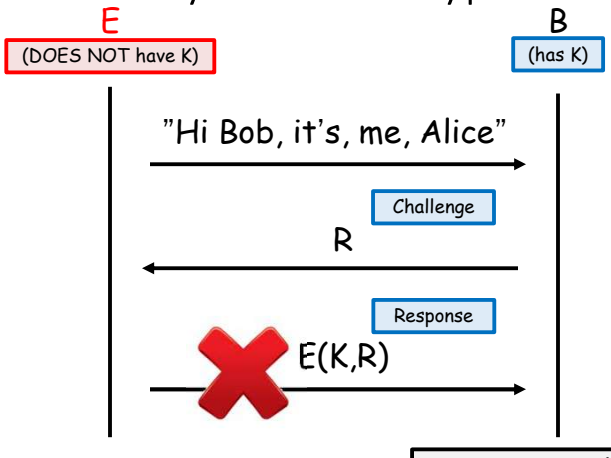
The Protocol (Nonces) based on symmetric crypto



Can a more active attack break this authentication protocol?

33

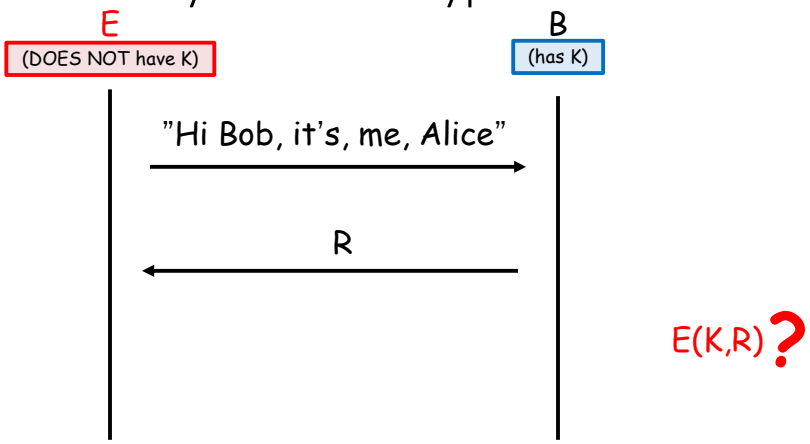
The Protocol (Nonces) based on symmetric crypto



To impersonate Alice, Eve needs to get the correct E(K, R).

34

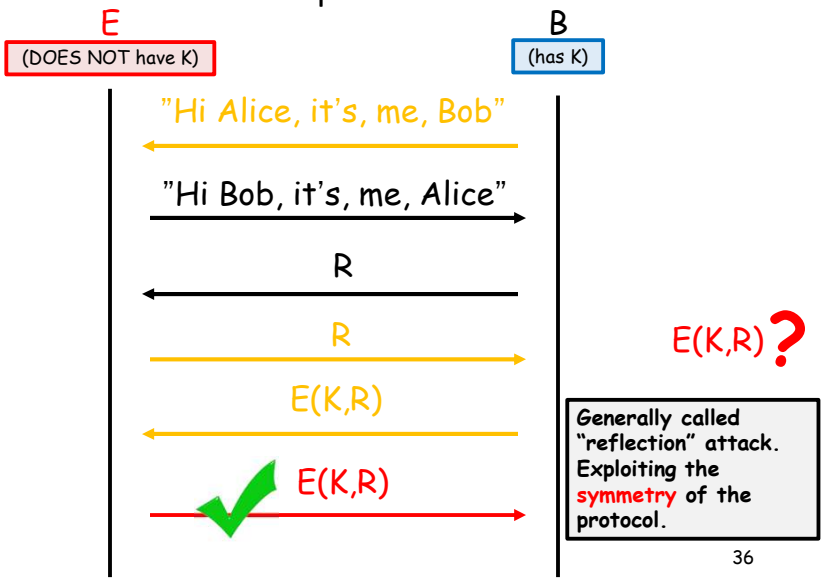
The Protocol (Nonces) based on symmetric crypto



35

35

A more active attacker breaks the auth protocol



36

36

The fix: Breaking the symmetry

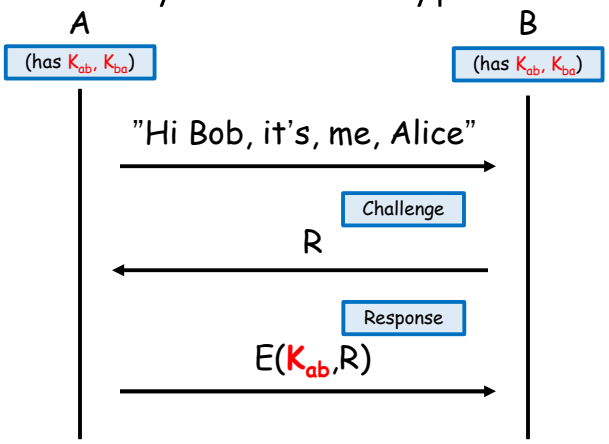
1. A → B: "Hi Bob, it's, me, Alice"
2. B → A: R
3. A → B: E(K_{ab},R)

- K_{ab} is only used in A→B direction and a different key (K_{ba}) is used in B→A direction

37

37

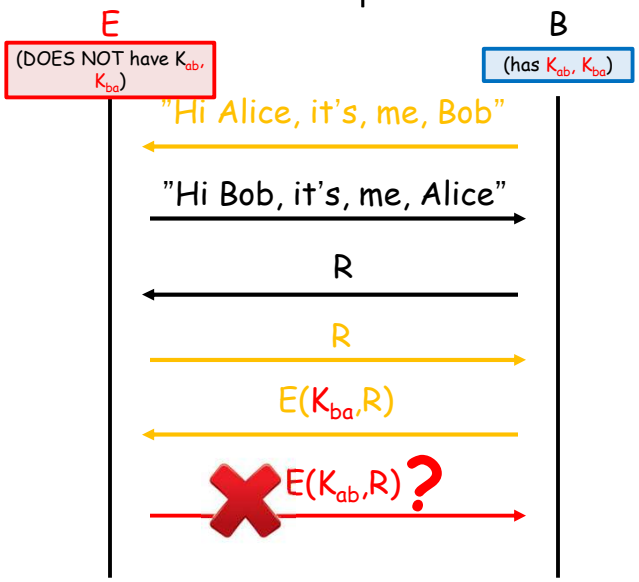
The Protocol (Nonces) based on symmetric crypto



38

38

Active attacker can no longer breaks the auth protocol



39

39

Another fix: Breaking the symmetry

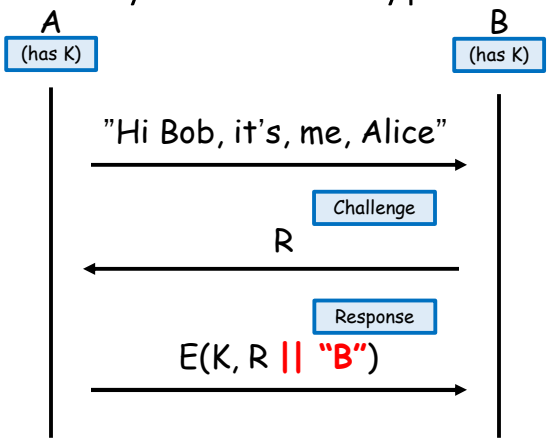
1. A → B: "Hi Bob, it's, me, Alice"
2. B → A: R
3. A → B: E(K, R || **"B"**)

- Use the same K in both directions but include **explicit direction identifier** in msg

40

40

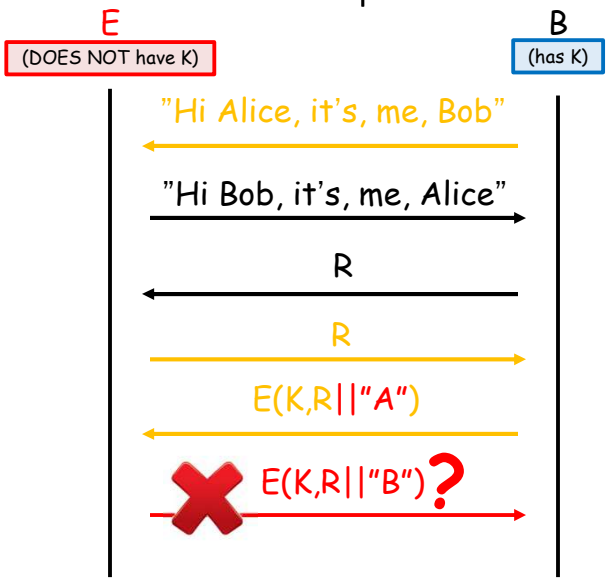
The Protocol (Nonces) based on symmetric crypto



41

41

Active attacker can no longer breaks the auth protocol



42

42

Variants of nonce to defend against replay attacks?

43

43

The Protocol (Sequence #s)

1. A → B: "Hi Bob, it's, me, Alice"

2. B → A: S_b (challenge)

A and B both remember S_b

In future sessions:

1. A → B: E(K, S_b||B) (response)

S_b++ on both A and B sides

- No random number generator needed
- Both A and B must remember S_b
- What if S_b wraps around?

44

44

Time-Stamps

Including a date/time-stamp in message allows recipient to check for freshness (as long as time-stamp is protected by cryptographic means).

1. A → B: $E(K, TIME_A \,||\, B)$

This results in fewer protocol messages

But requires synchronized clocks…

45

45

Will focus on 2 types of protocols

- Authentication
- **Key Distribution**
  - Fundamental security assumption for both symmetric & asymmetric crypto we learned before
  - Need to be built on top of authentication --- need to ensure that we distribute the secret to the right entity, not Eve

46

46

Key Distribution and Management

- Conventional (Secret) key distribution
- Public key distribution

47

47

The Entities (2-Party Setting)

- **Alice** and **Bob**
  - want to mutually authenticate and/or share a key
- **Eve**, the adversary
  - passive or active
- More complex protocols may involve a **Trusted Third Party (TTP)**
  - 3rd party trusted by both Alice and Bob

48

48

Trusted Intermediaries, or Trusted 3rd Party (TTP)

Symmetric Key Problem:

- How do two entities establish shared secret key over a distance (i.e., over a network)?

Solution:

- Mutually trusted **on-line** key distribution center (KDC) acts as TTP

Public Key Problem:

- When Alice gets Bob's public key (from a web site, email, disk, bboard), how does she know it is really Bob's?

Solution:

- Trusted **off-line** certification authority (CA) as TTP

49

49

Trusted Intermediaries, or Trusted 3rd Party (TTP)

Symmetric Key Problem:

- How do two entities establish shared secret key over a distance (i.e., over a network)?

Solution:

- Mutually trusted **on-line** key distribution center (KDC) acts as TTP

Public Key Problem:

- When Alice gets Bob's public key (from a web site, email, disk, bboard), how does she know it is really Bob's?

Solution:

- Trusted **off-line** certification authority (CA) as TTP

50

50

Key Distribution Center (KDC)

- Responsible for distributing keys to pairs of users (hosts, processes, applications)
- Each user must share a unique **master key** with the KDC
  - Use this key to communicate with KDC to get a temporary *session* key for establishing a secure "session" with another user/program/host/entity
  - Each master key is distributed (agreed upon) in some **off-line** fashion (in person, by snail-mail, etc.)

51

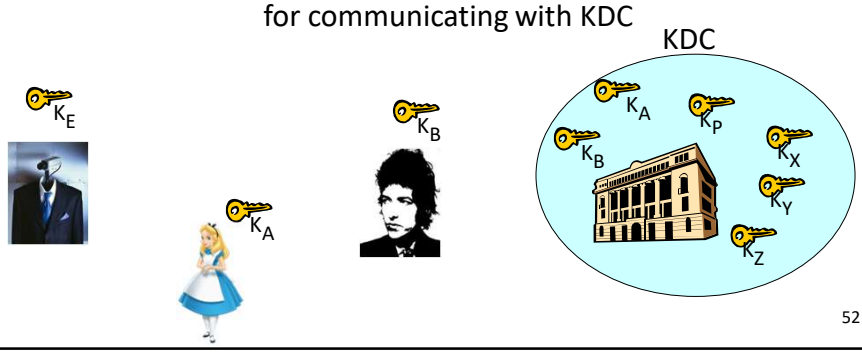
51

Key Distribution Center (KDC) as Trusted Third Party (TTP)

- Alice and Bob need to share a key
- KDC shares different master key with *each* registered user (many users)
- Alice and Bob know their own master keys:

K_A and K_B

for communicating with KDC



52

52