Lecture 10

Protocols (Continued)

Chapters 9 and 11 in KPS

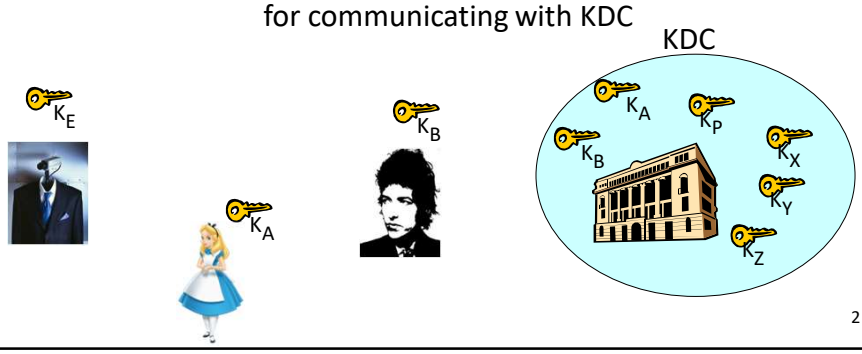
1

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

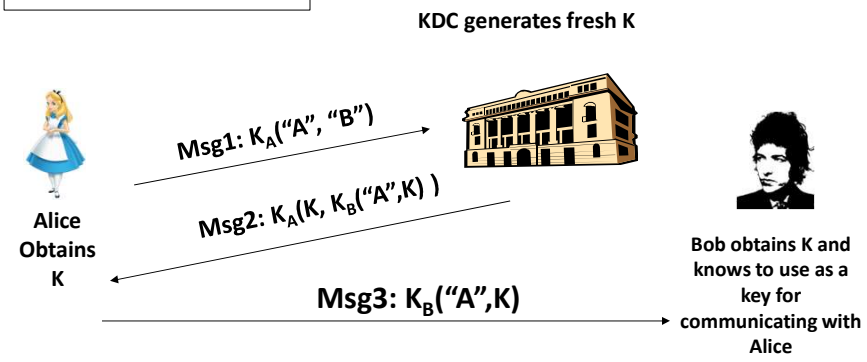
for communicating with KDC



2

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

K(X) = Encryption of X with key K



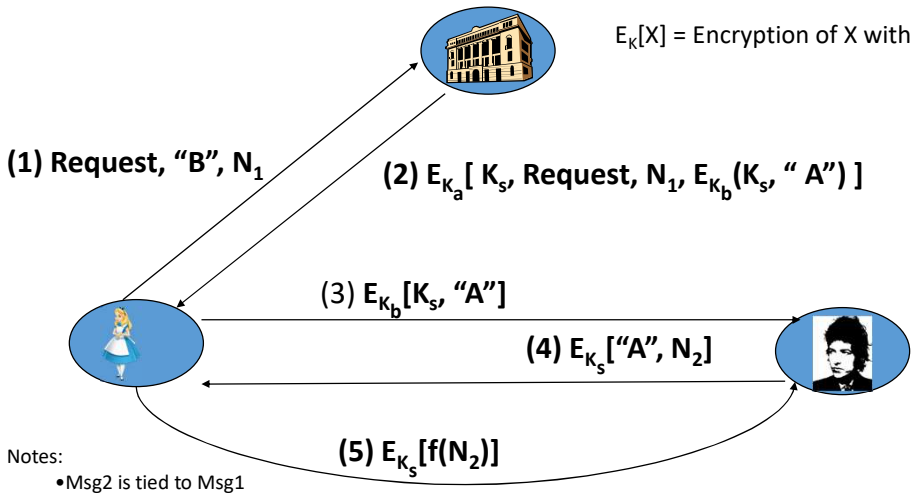
- Alice and Bob communicate using **K** as a short-term (*session) key* for encryption and/or data integrity
- Note:
  - Msg 1 is vuln to replay attack
  - Msg2 is not tied to Msg1 → Msg2 is vuln to replay attack
  - Msg3 is also vuln to replay attack
  - Bob and Alice don't authenticate each other!

3

3

A Typical Key Distribution Scenario

$E_K[X]$ = Encryption of X with K



Notes:

- Msg2 is tied to Msg1
  → Msg 2 fresh/new, not replayed
- Bob authenticates Alice
- Msg3 is vuln to replay attack
- Msg1 may not from Alice (KDC doesn't authenticate Alice)
- Alice DOES NOT authenticate Bob

4

4

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

5

5

Public Key Distribution

General schemes:

- Public announcement (e.g., in a newsgroup or email message)
  - Can be forged
- Publicly available directory
  - Can be tampered with
- Public-key certificates (PKCs) issued by trusted off-line Certification Authorities (CAs)

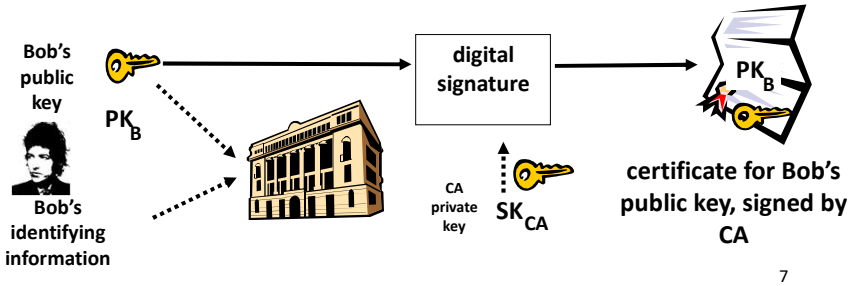
6

6

Certification Authorities

- Certification authority (CA): trusted, highly secure (physically and electronically) component
- Issues public key certificates; each binds a public key to a specific entity
- Each entity (user, host, etc.) registers its public key with CA.
  - Bob provides "proof of identity" to CA.
  - CA creates public key certificate binding Bob's ID/name to this public key.
  - Certificate containing Bob's public key is signed by CA:

CA says: "this is Bob's public key"

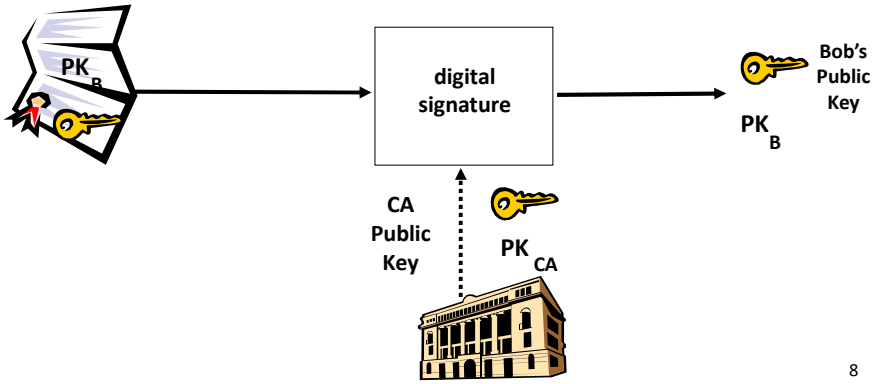


7

7

Certification Authority

- **When Alice wants to get Bob's public key:**
  - **Get Bob's certificate (from Bob or elsewhere)**
  - **Using CA's public key verify the signature on Bob's certificate**
  - **Check for expiration**
  - **Check for revocation (we'll talk about this later)**
  - **Use Bob's public key**

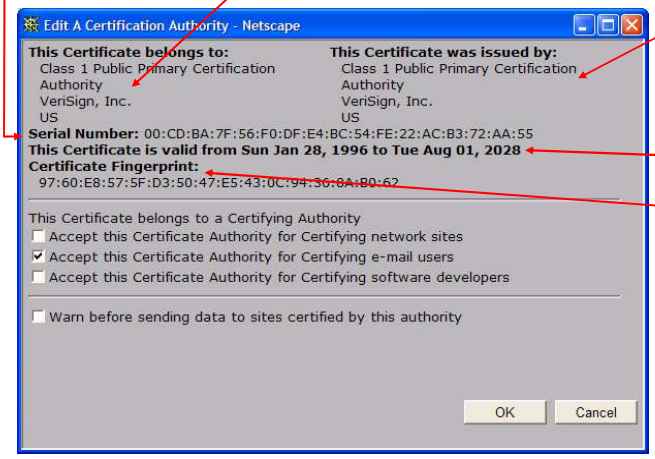


8

8

A Certificate Contains

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

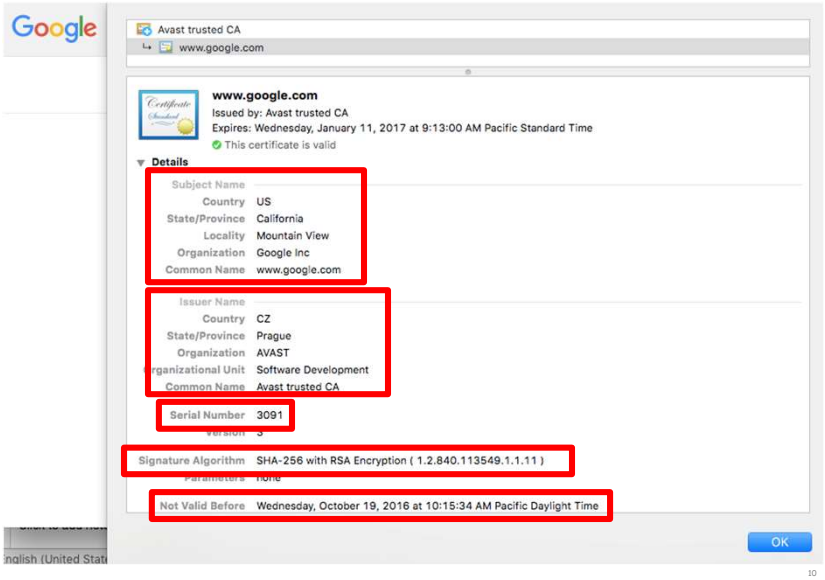


- info about certificate issuer
- valid dates
- digital signature by issuer

9

9

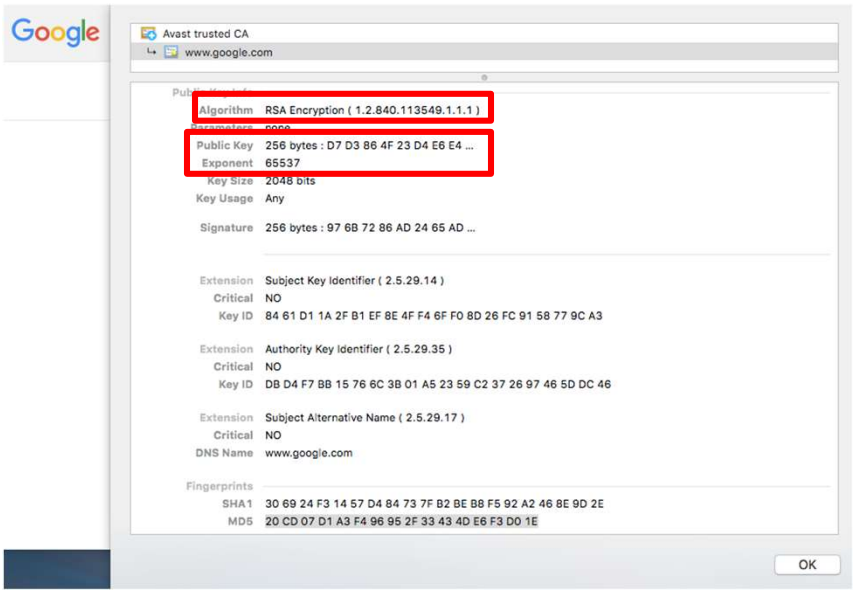
A Sample Certificate (1/2)



10

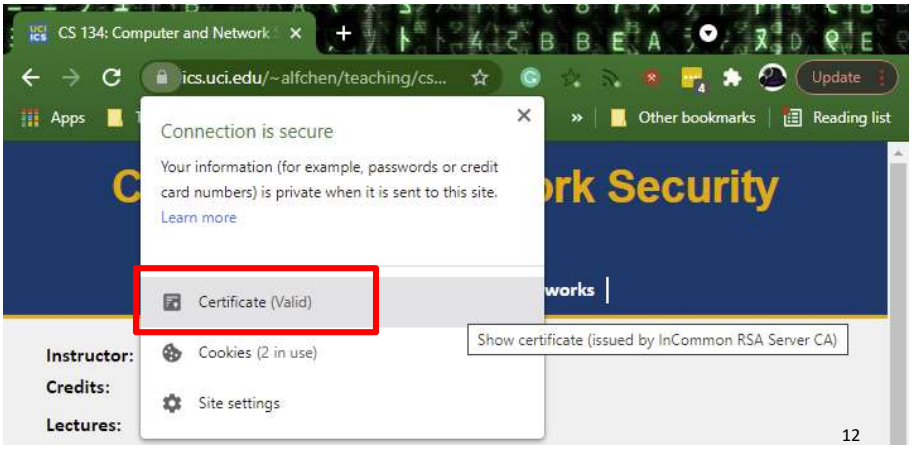
10

A Sample Certificate (2/2)



11

Open your browser to check it yourself!



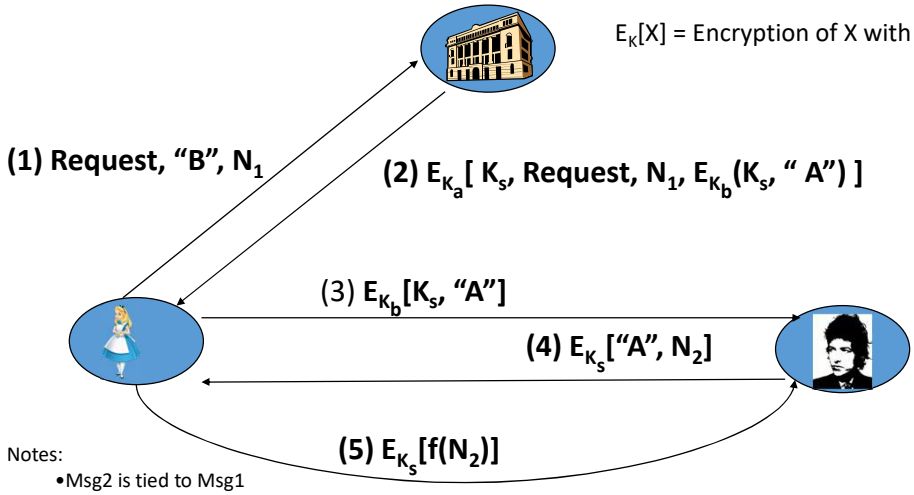
12

Back to Protocols

13

13

A Typical Key Distribution Scenario



Notes:

- Msg2 is tied to Msg1
  → Msg 2 fresh/new, not replayed
- Bob authenticates Alice
- Msg3 is vuln to replay attack
- Msg1 may not from Alice (KDC doesn't authenticate Alice)
- Alice DOES NOT authenticate Bob

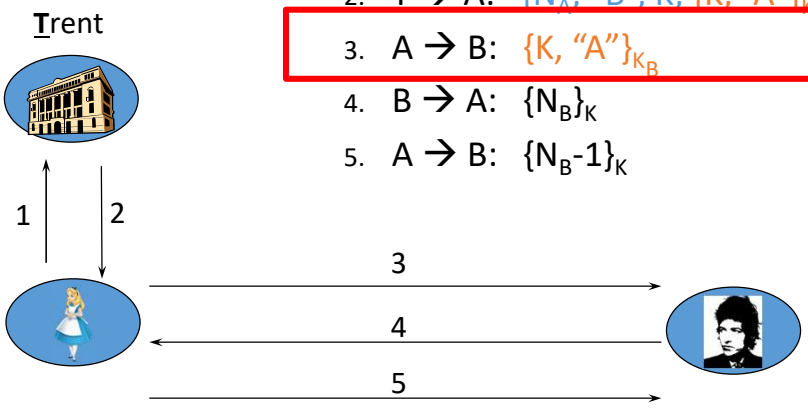
14

14

Needham-Schroeder Protocol (1978): First Distributed Security Protocol

$\{X\}_K$ = Encryption of X with key K

1. A → T: "A", "B", N_A
2. T → A: $\{N_A$, "B", K, $\{K$, "A"$\}_{K_B}\}_{K_A}$
3. A → B: $\{K$, "A"$\}_{K_B}$
4. B → A: $\{N_B\}_K$
5. A → B: $\{N_B\text{-}1\}_K$



15

Security?

Denning-Sacco Attack: suppose Eve recorded **an old** protocol session for which she somehow knows the session key K':

1. A → T: "A", "B", N_A
2. T → A: $\{N_A$, "B", K', $\{K'$, "A"$\}_{K_B}\}_{K_A}$
3. A → B: $\{K'$, "A"$\}_{K_B}$

At a later time:

3. E → B: $\{K'$, "A"$\}_{K_B}$
4. B → E: $\{N_B\}_{K'}$
5. E → B: $\{N_B\text{-}1\}_{K'}$

16

Fixing the Attack

- Bob has no guarantees about freshness of the message in step 3 → vuln to replay attack!
- Eve exploits this to impersonate Alice to Bob - old session keys are useful.
- Can be fixed by adding timestamps:
  - Limits usefulness of old session keys
  - Eve's attack becomes:

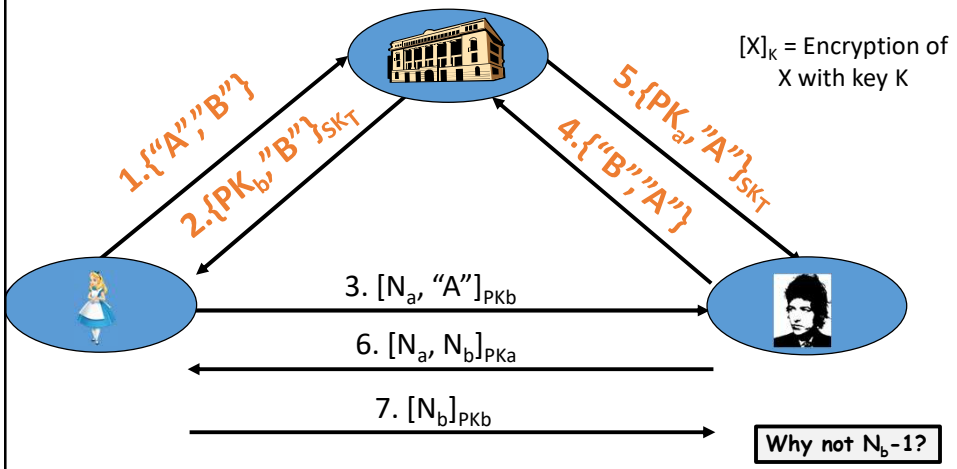
3: E → B: $\{K', \mathbf{T'}, A\}_{K_B}$

attack is now thwarted because T' is stale

17

17

PK-based Needham-Schroeder Protocol



- PK_A: Alice's public key, PK_B: Bob's public key
- SK_T: TTP's secret (private) key used for signing
- Everyone knows TTP's public key PK_T
- $CERT_B$ = Message 2, $CERT_A$ = Message 5

18

Vulnerability to replay attacks

- 1, 2, 4, 5: Delivery of public key
- Does not guarantee freshness of the public key

How to solve it?

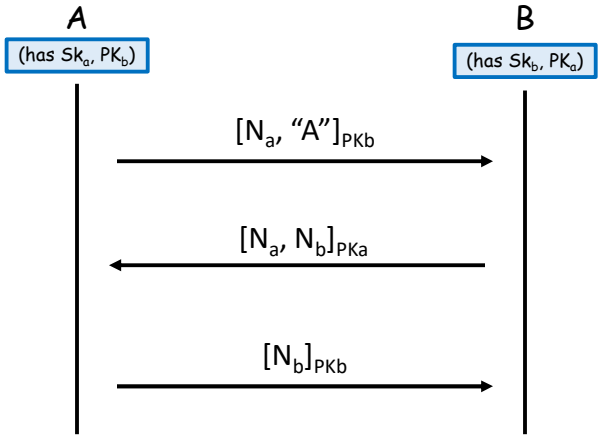
- Timestamp in messages 2 and 5 or nonces in messages 1&2 and 4&5
- Public Key Certificate: assign expiration time/data to each certificate (messages 2 and 5)

19

19

Lowe's Attack (Impersonation by Interleaving)

Original protocol

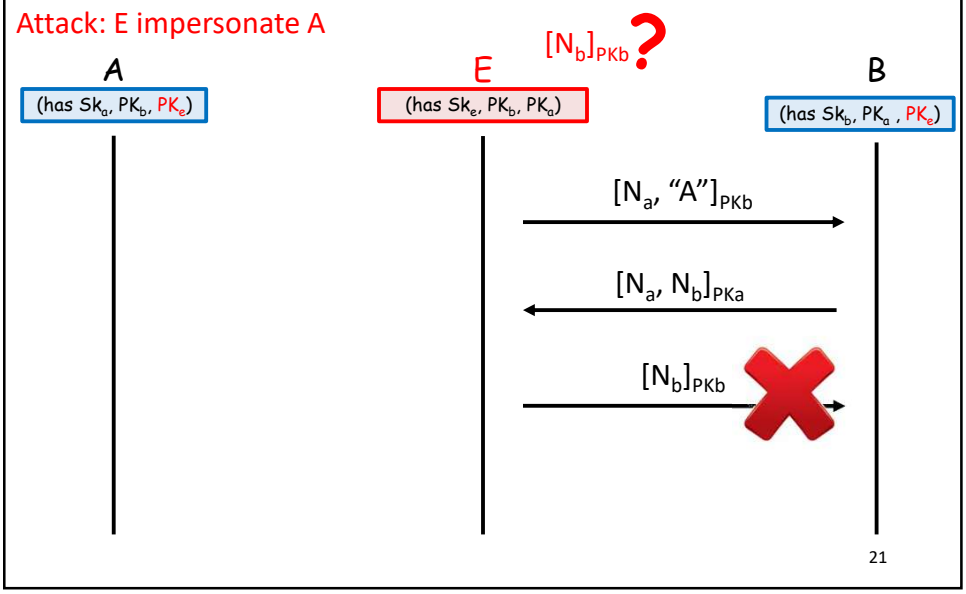


20

20

Lowe's Attack
(Impersonation by Interleaving)

Attack: E impersonate A

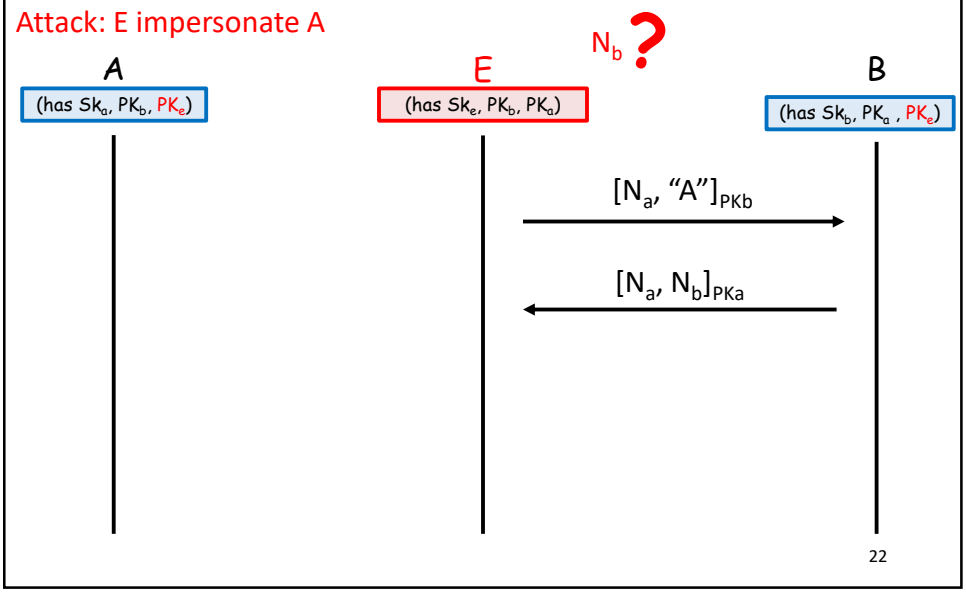


21

21

Lowe's Attack
(Impersonation by Interleaving)

Attack: E impersonate A

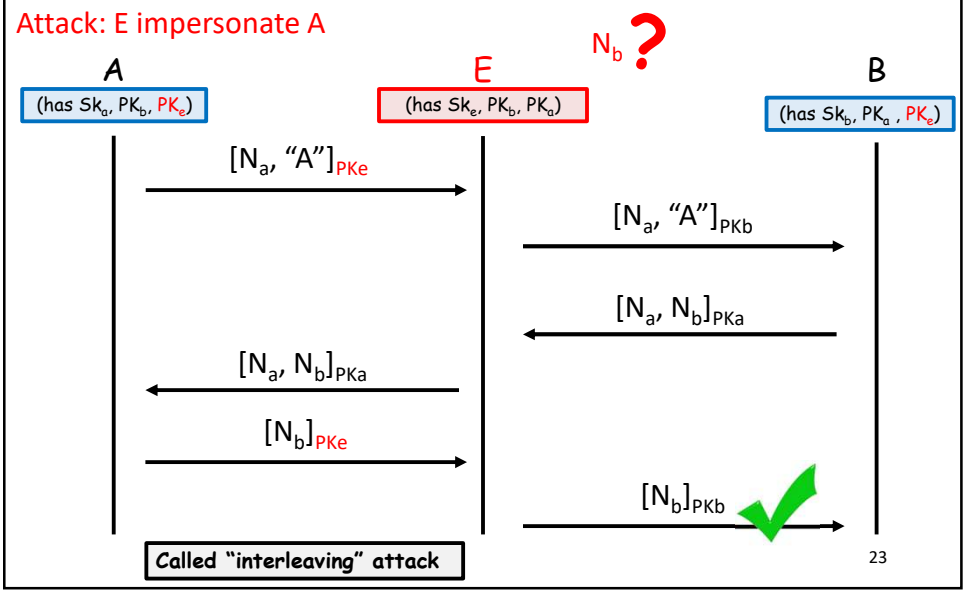


22

22

Lowe's Attack (Impersonation by Interleaving)

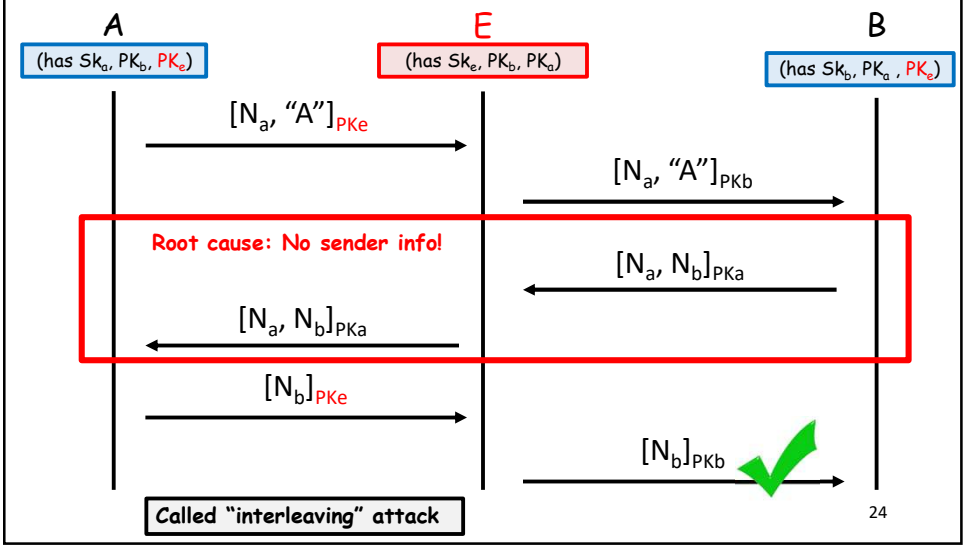
Attack: E impersonate A



23

Lowe's Attack (Impersonation by Interleaving)

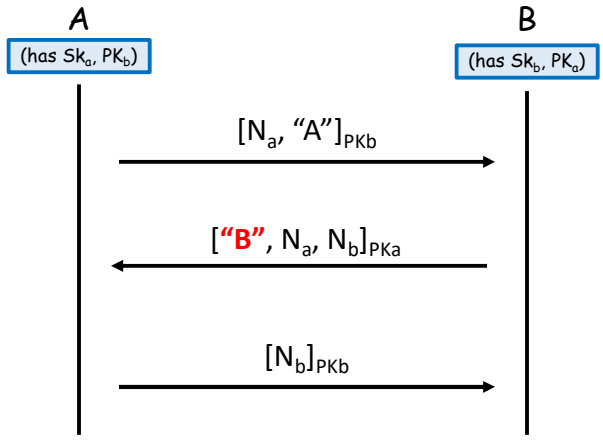
Attack: E impersonate A



24

Lowe's Attack (Impersonation by Interleaving)

The fix

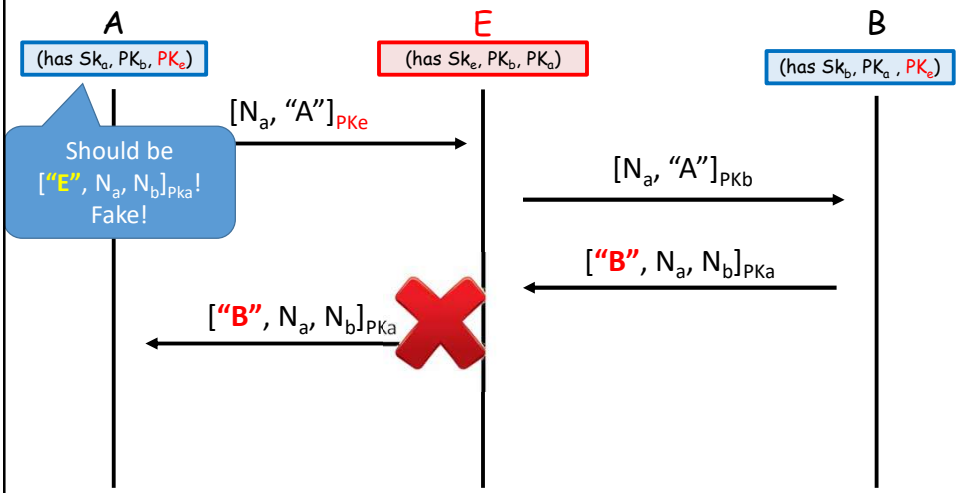


25

25

Lowe's Attack (Impersonation by Interleaving)

Defeat Lowe's attack



26

26

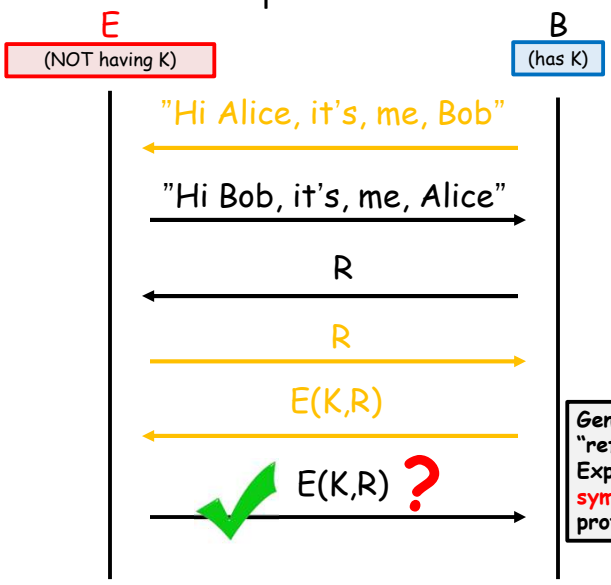
More practice on

refection attack & interleaving attack

27

27

Active attacker breaks the auth protocol



28

28

Reflection Attack and a Fix

- Original Protocol
  1. $A \rightarrow B:$ r_A
  2. $B \rightarrow A:$ $\{ r_A, r_B \}_K$
  3. $A \rightarrow B:$ r_B

- Attack
  1. $A \rightarrow E:$ r_A
  2. $E \rightarrow A:$ r_A : Starting a new session
  3. $A \rightarrow E:$ $\{ r_A, r_A' \}_K$: Reply to (2)
  2. $E \rightarrow A:$ $\{ r_A, r_A' \}_K$: Reply to (1)
  3. $A \rightarrow E:$ r_A'

  Solutions?
- Use 2 different uni-directional keys k" (A→B) and k' (B→A)
- Remove symmetry (direction, msg identifiers)

29

Interleaving Attacks

- Protocol for Mutual Authentication
  1. $A \rightarrow B:$ $A, r_A,$
  2. $B \rightarrow A:$ $r_B, \{ r_B, r_A, A \}_{SK_B}$
  3. $A \rightarrow B:$ $r_A', \{ r_A', r_B, B \}_{SK_A}$

- Attack (E impersonates A):
  1. $E \rightarrow B:$ A, r_A
  2. $B \rightarrow E:$ $r_B, \{ r_B, r_A, A \}_{SK_B}$

  1. $E \rightarrow A:$ B, r_B
  2. $A \rightarrow E:$ $r_A', \{ r_A', r_B, B \}_{SK_A}$

  3. $E \rightarrow B:$ $r_A', \{ r_A', r_B, B \}_{SK_A}$

- Attack due to symmetric messages (2), (3)

30

Lessons learned?

- Designing **secure** protocols is hard. There are **many** documented failures in the literature.
- Good protocols are already standardized (e.g., ISO 9798, X.509, …) – use them!
  - In other words, don't invent your own!
- The problem of verifying (proving) protocol security gets much harder as protocols get more complex: more parties, messages and rounds.

31

31

If interested to learn further, read this paper:

"Programming Satan's Computer"
by R. Anderson and R. Needham

available at:

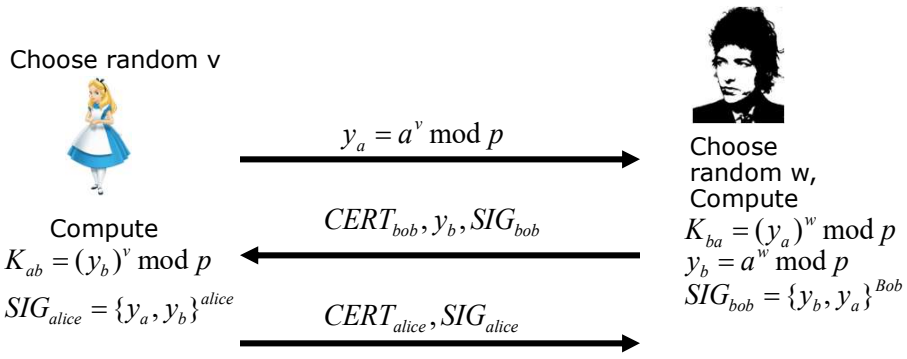
http://www.cl.cam.ac.uk/~rja14/Papers/satan.pdf

32

32

Secure Protocol Examples

Authenticated Public-Key-based Key Exchange (Station-to-Station or STS Protocol)



x.509 Authentication & Key Distribution Protocols

One-msg
A→B



$$\{1, t_a, r_a, B, other_a, [K_{ab}]_{PK_B}\}_{SK_A}$$



Two-msg
A→B



$$\{2, t_a, r_a, B, other_a, [K_{ab}]_{PK_B}\}_{SK_A}$$

$$\{2, t_b, r_b, A, r_a, other_b, [K_{ba}]_{PK_A}\}_{SK_B}$$



Three-msg
A←→B



$$\{3, t_a, r_a, B, other_a, [K_{ab}]_{PK_B}\}_{SK_A}$$

$$\{3, t_b, r_b, A, r_a, other_b, [K_{ba}]_{PK_A}\}_{SK_B}$$

$$\{3, r_b\}_{SK_A}$$



35

35

PK-based Denning-Sacco Attack



$Cert_A = \{PK_A, A\}_{SK_T}$
$Cert_B = \{PK_B, B\}_{SK_T}$
$Cert_C = \{PK_C, C\}_{SK_T}$

1. A, B

2. $Cert_A$, $Cert_B$



3. $Cert_A, Cert_B, [\{K_{AB}, T_A\}_{SK_A}]_{PK_B}$



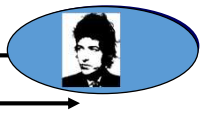
4. Secure communication with K_{AB}

Thinks she is talking to A

Bob impersonates Alice

C

3'. $Cert_A, Cert_C, [\{K_{AB}, T_A\}_{SK_A}]_{PK_C}$



4'. Secure communication with K_{AB}

36

36