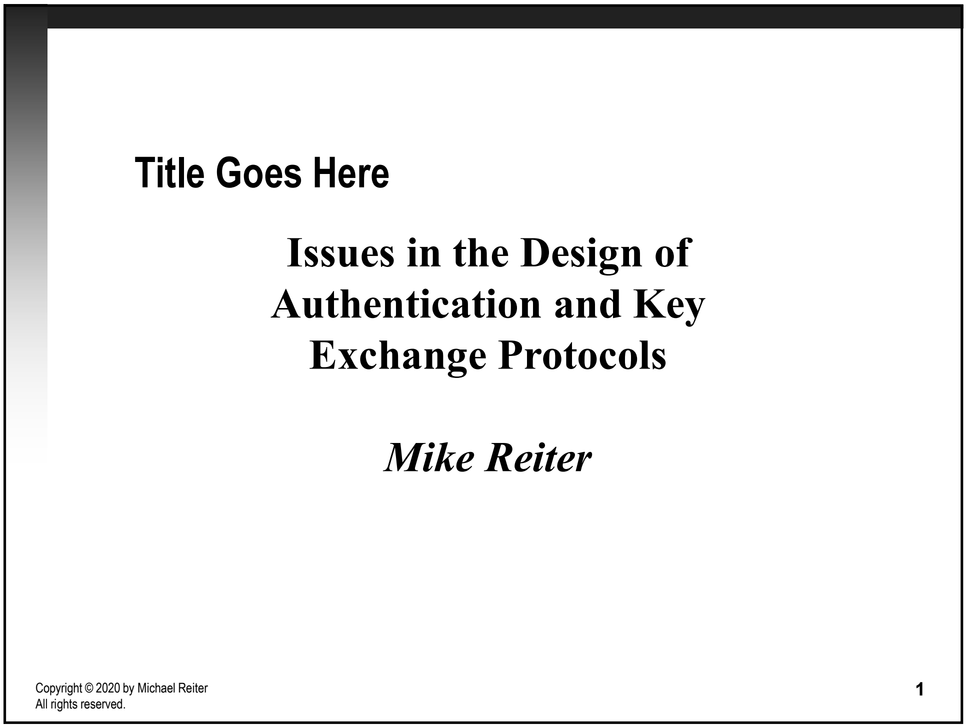
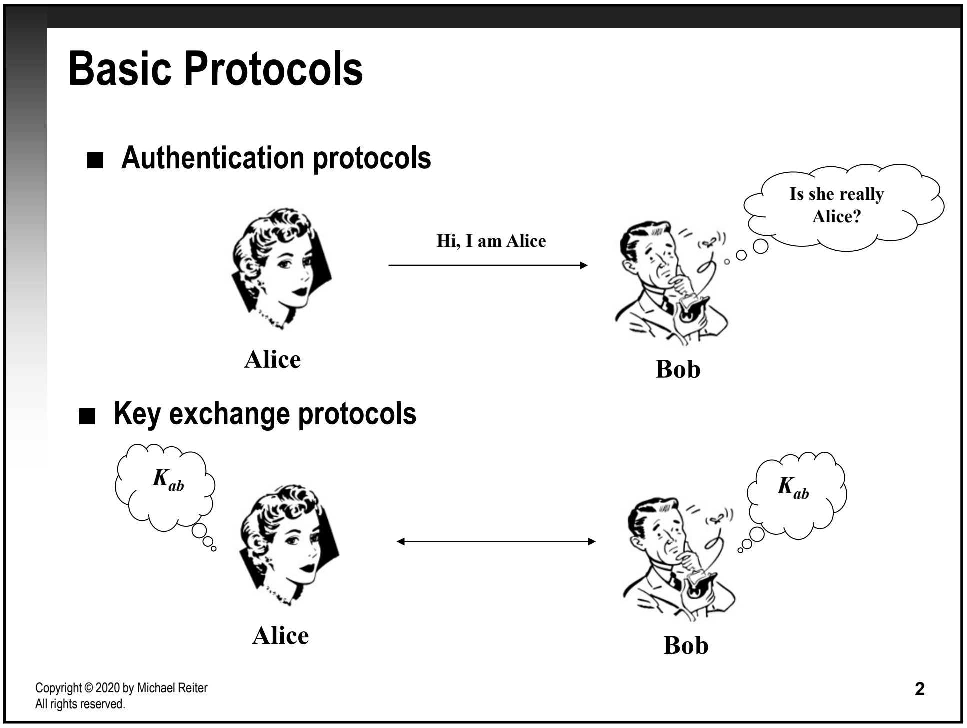


Title Goes Here

Issues in the Design of Authentication and Key Exchange Protocols

Mike Reiter

•1

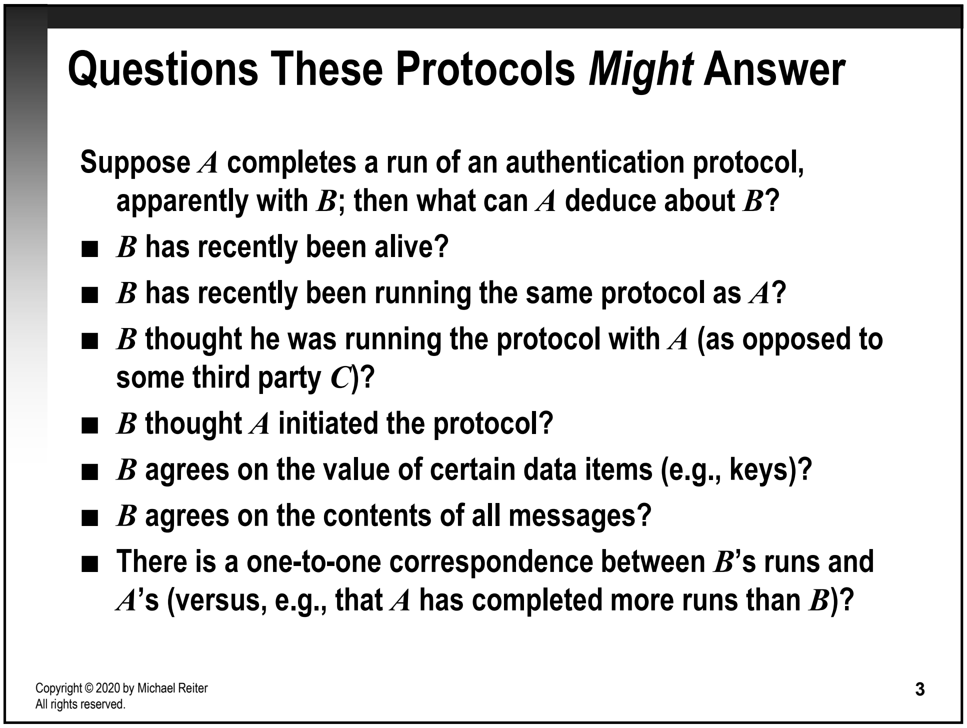


Basic Protocols

- Authentication protocols

- Key exchange protocols

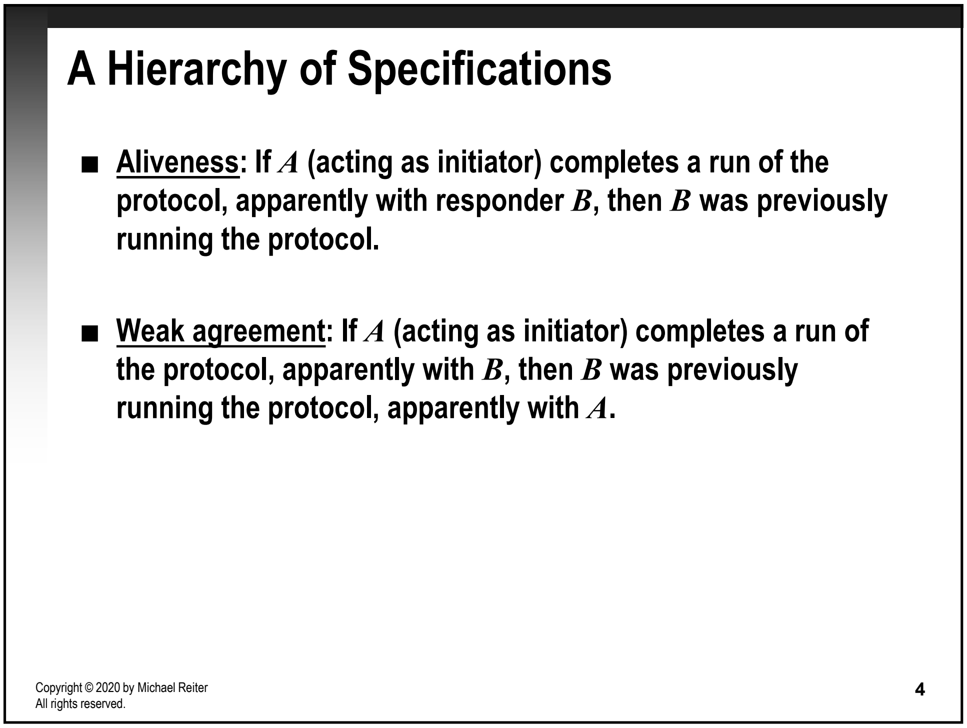
•2



Questions These Protocols *Might* Answer

Suppose *A* completes a run of an authentication protocol, apparently with *B*; then what can *A* deduce about *B*?

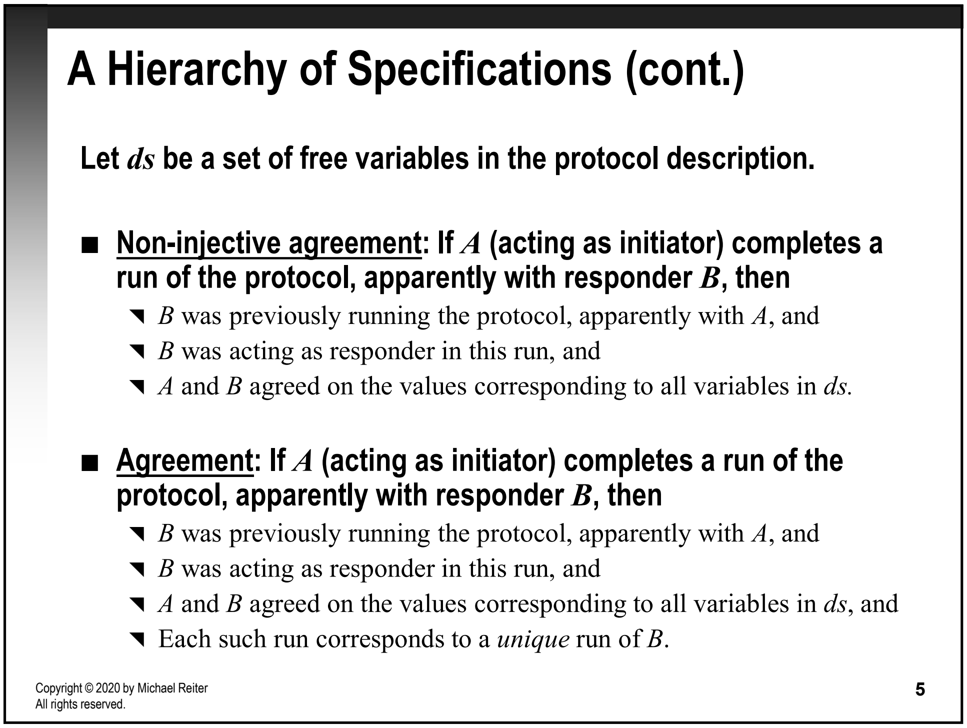
- *B* has recently been alive?
- *B* has recently been running the same protocol as *A*?
- *B* thought he was running the protocol with *A* (as opposed to some third party *C*)?
- *B* thought *A* initiated the protocol?
- *B* agrees on the value of certain data items (e.g., keys)?
- *B* agrees on the contents of all messages?
- There is a one-to-one correspondence between *B*'s runs and *A*'s (versus, e.g., that *A* has completed more runs than *B*)?



A Hierarchy of Specifications

- <u>Aliveness</u>: If *A* (acting as initiator) completes a run of the protocol, apparently with responder *B*, then *B* was previously running the protocol.

- <u>Weak agreement</u>: If *A* (acting as initiator) completes a run of the protocol, apparently with *B*, then *B* was previously running the protocol, apparently with *A*.

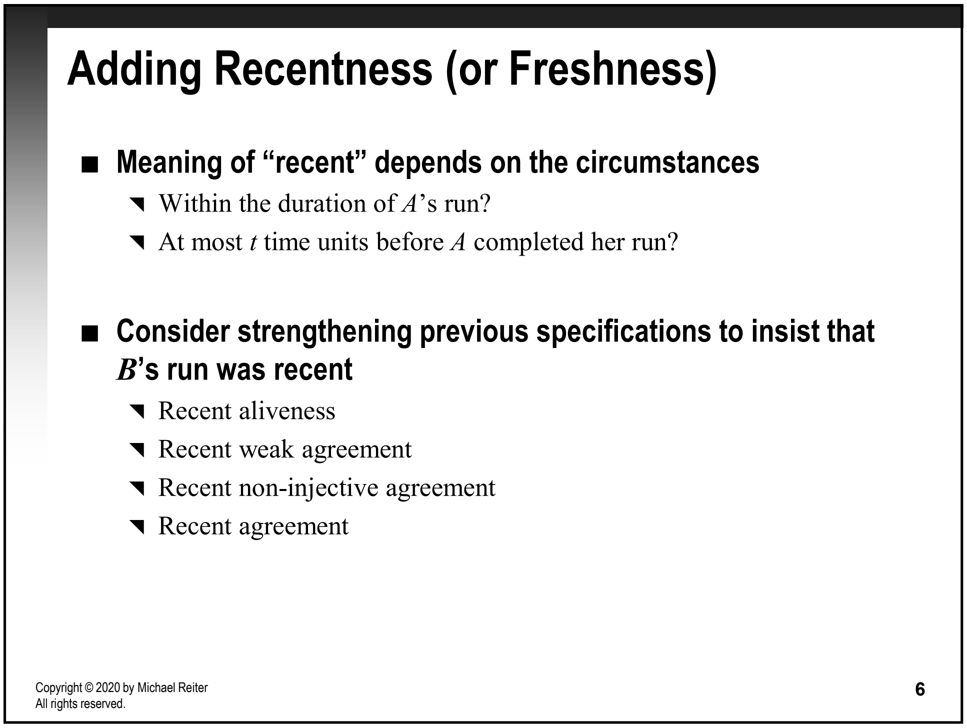


A Hierarchy of Specifications (cont.)

Let *ds* be a set of free variables in the protocol description.

- **<u>Non-injective agreement</u>: If *A* (acting as initiator) completes a run of the protocol, apparently with responder *B*, then**
  - *B* was previously running the protocol, apparently with *A*, and
  - *B* was acting as responder in this run, and
  - *A* and *B* agreed on the values corresponding to all variables in *ds.*
- **<u>Agreement</u>: If *A* (acting as initiator) completes a run of the protocol, apparently with responder *B*, then**
  - *B* was previously running the protocol, apparently with *A*, and
  - *B* was acting as responder in this run, and
  - *A* and *B* agreed on the values corresponding to all variables in *ds*, and
  - Each such run corresponds to a *unique* run of *B*.

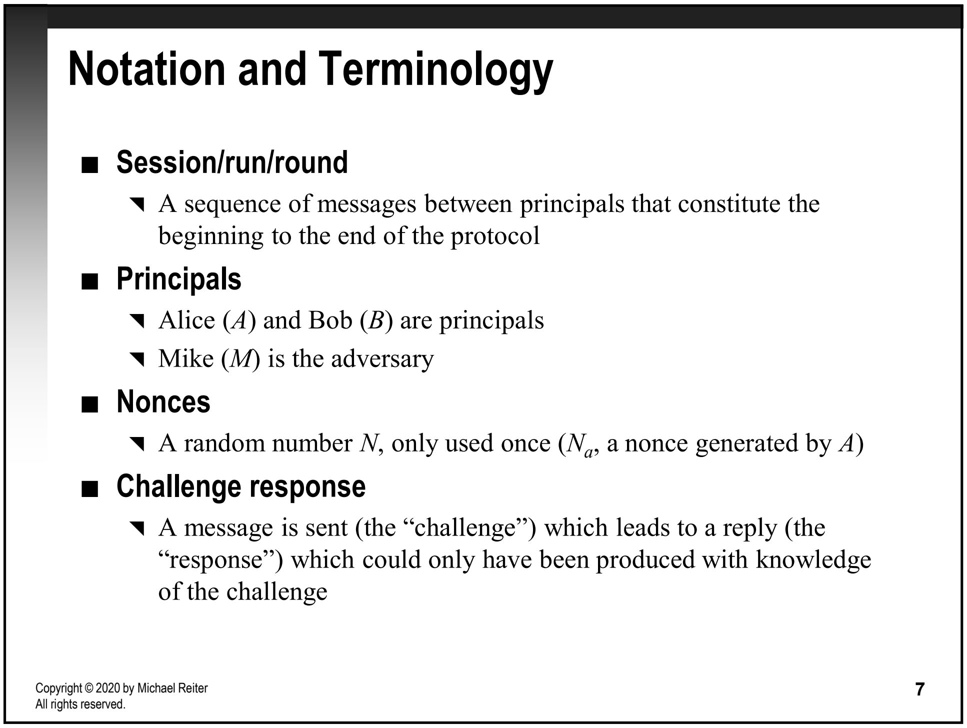
•5



Adding Recentness (or Freshness)

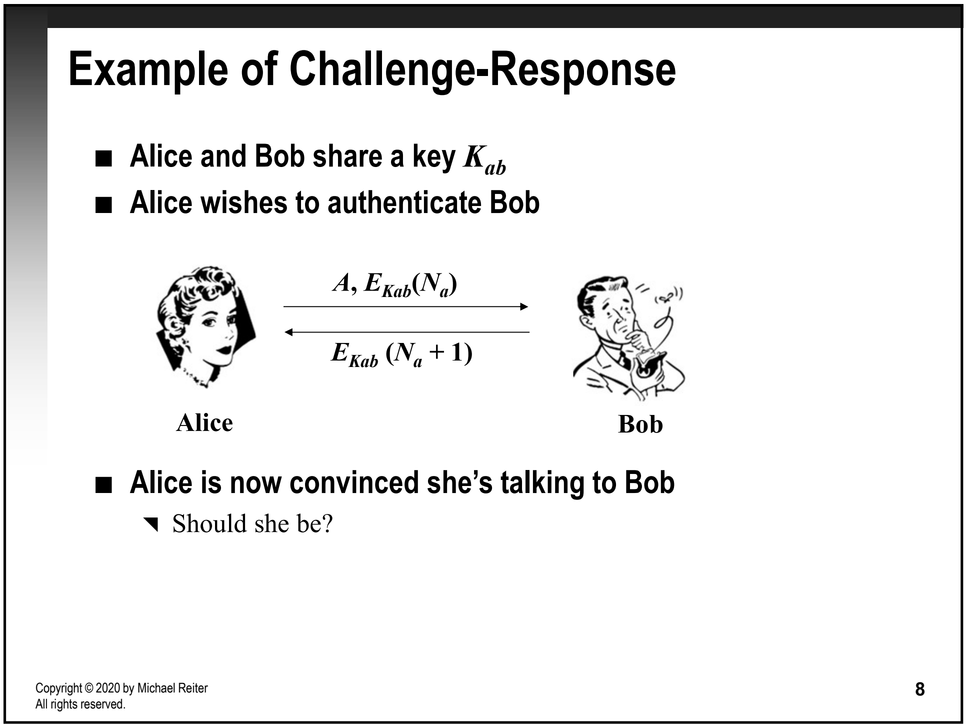
- **Meaning of "recent" depends on the circumstances**
  - Within the duration of *A*'s run?
  - At most *t* time units before *A* completed her run?
- **Consider strengthening previous specifications to insist that *B*'s run was recent**
  - Recent aliveness
  - Recent weak agreement
  - Recent non-injective agreement
  - Recent agreement

•6



Notation and Terminology

- **Session/run/round**
  - A sequence of messages between principals that constitute the beginning to the end of the protocol
- **Principals**
  - Alice (A) and Bob (B) are principals
  - Mike (M) is the adversary
- **Nonces**
  - A random number N, only used once (N_a, a nonce generated by A)
- **Challenge response**
  - A message is sent (the "challenge") which leads to a reply (the "response") which could only have been produced with knowledge of the challenge



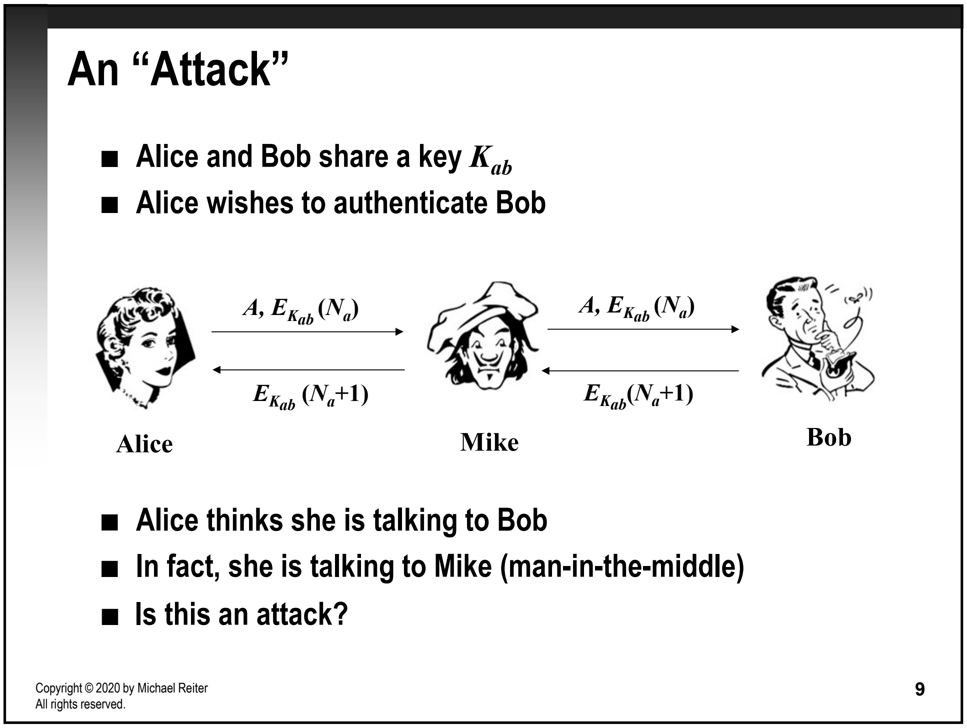
Example of Challenge-Response

- **Alice and Bob share a key K_{ab}**
- **Alice wishes to authenticate Bob**

$$A, E_{Kab}(N_a) \rightarrow$$

$$\leftarrow E_{Kab}(N_a + 1)$$

- **Alice is now convinced she's talking to Bob**
  - Should she be?



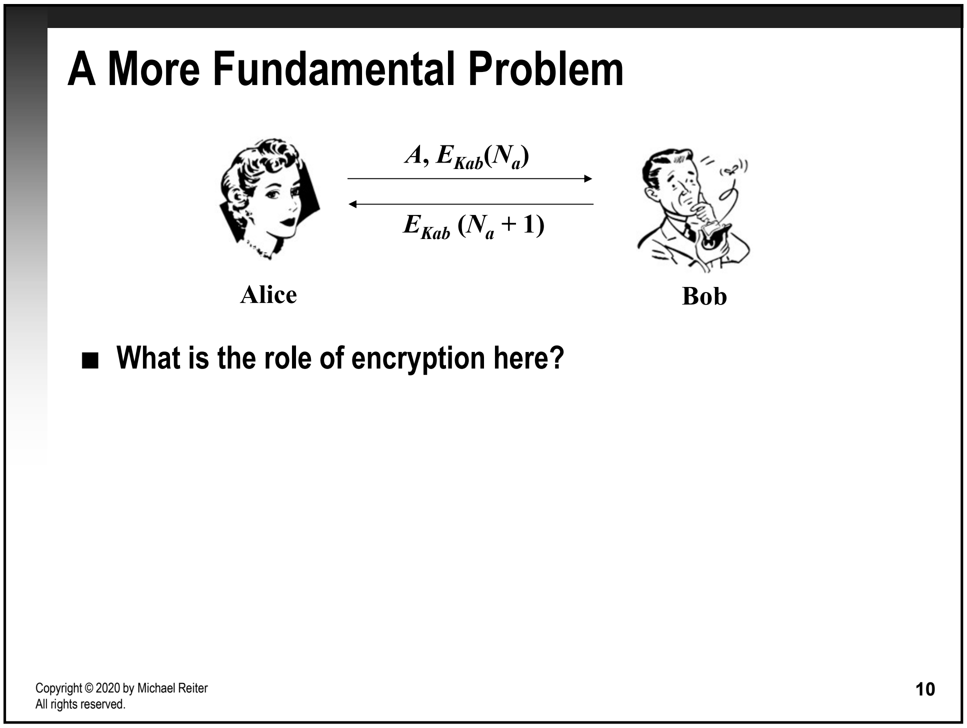
An “Attack”

- Alice and Bob share a key K_{ab}
- Alice wishes to authenticate Bob

$$\text{Alice} \underset{E_{K_{ab}}(N_a+1)}{\overset{A,\ E_{K_{ab}}(N_a)}{\rightleftarrows}} \text{Mike} \underset{E_{K_{ab}}(N_a+1)}{\overset{A,\ E_{K_{ab}}(N_a)}{\rightleftarrows}} \text{Bob}$$

Alice
Mike
Bob

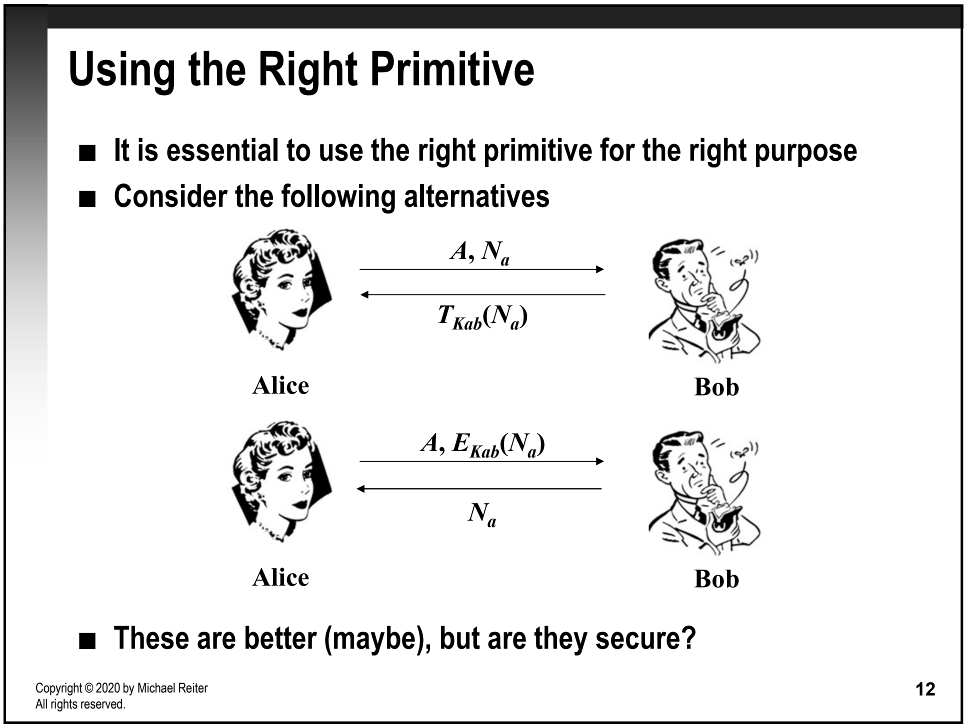
- Alice thinks she is talking to Bob
- In fact, she is talking to Mike (man-in-the-middle)
- Is this an attack?



A More Fundamental Problem

$$\text{Alice} \underset{E_{Kab}\ (N_a + 1)}{\overset{A,\ E_{Kab}(N_a)}{\rightleftarrows}} \text{Bob}$$

- What is the role of encryption here?

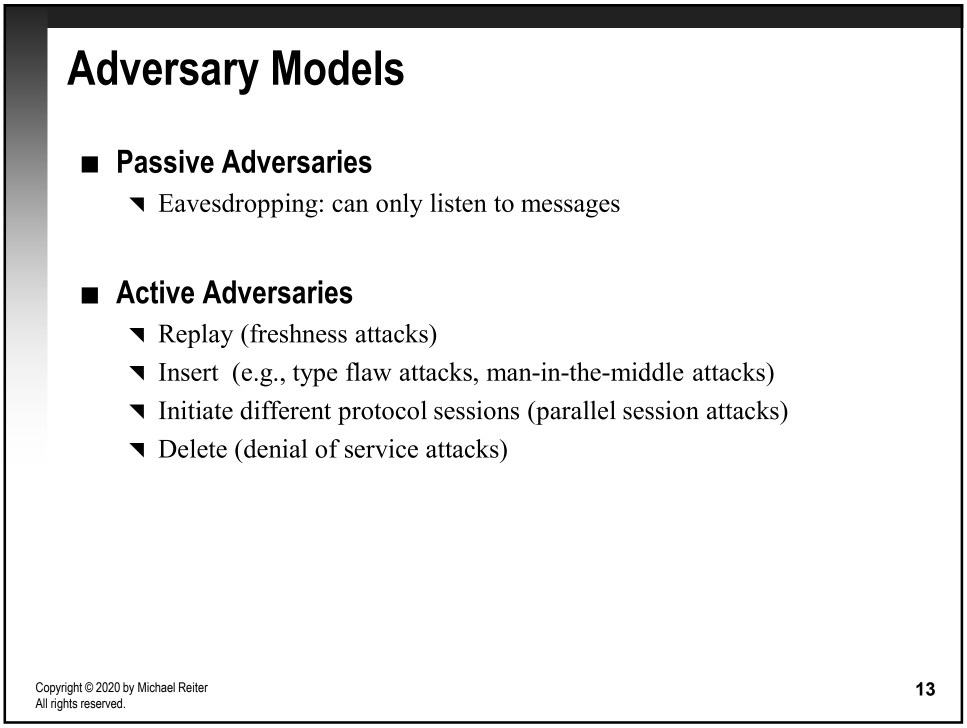


Using the Right Primitive

- It is essential to use the right primitive for the right purpose
- Consider the following alternatives

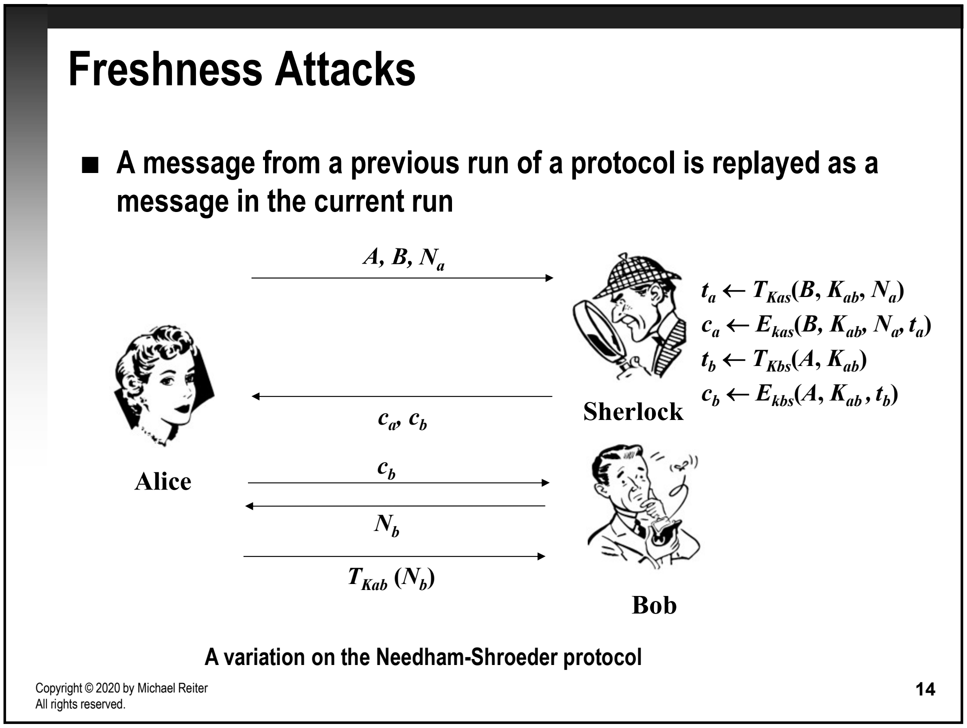
A, N_a →
← $T_{Kab}(N_a)$
Alice
Bob
$A, E_{Kab}(N_a)$ →
← N_a
Alice
Bob

- These are better (maybe), but are they secure?



Adversary Models

- Passive Adversaries
  - Eavesdropping: can only listen to messages
- Active Adversaries
  - Replay (freshness attacks)
  - Insert (e.g., type flaw attacks, man-in-the-middle attacks)
  - Initiate different protocol sessions (parallel session attacks)
  - Delete (denial of service attacks)



Freshness Attacks

- A message from a previous run of a protocol is replayed as a message in the current run

A, B, N_a

$t_a \leftarrow T_{Kas}(B, K_{ab}, N_a)$
$c_a \leftarrow E_{kas}(B, K_{ab}, N_a, t_a)$
$t_b \leftarrow T_{Kbs}(A, K_{ab})$
$c_b \leftarrow E_{kbs}(A, K_{ab}, t_b)$

c_a, c_b

c_b

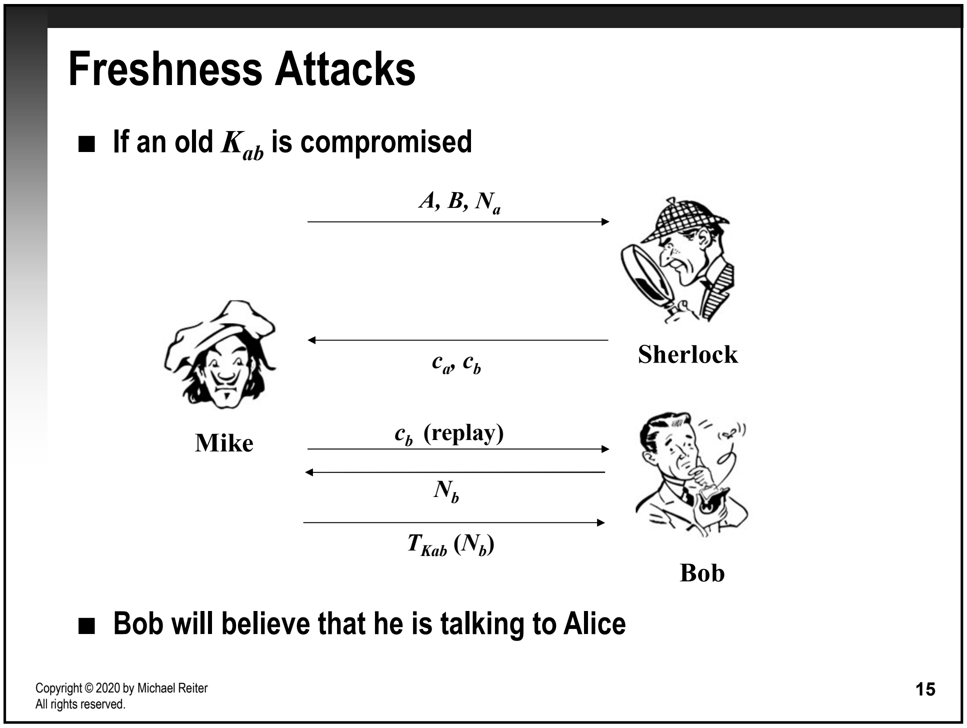
N_b

$T_{Kab}(N_b)$

A variation on the Needham-Shroeder protocol

14

•14



Freshness Attacks

- If an old K_{ab} is compromised

A, B, N_a

c_a, c_b

c_b (replay)

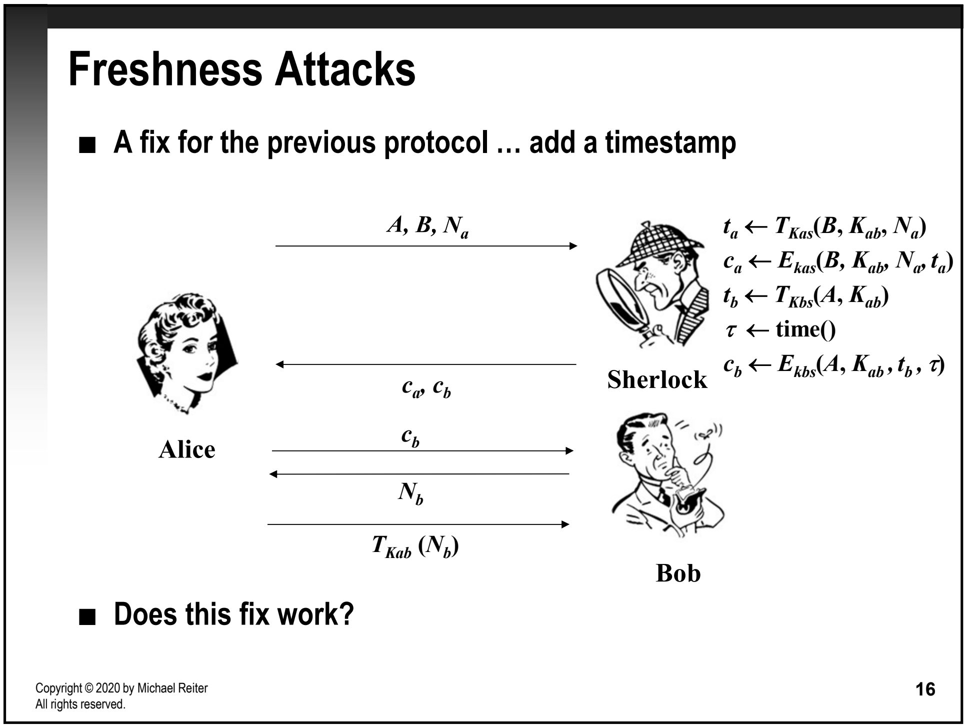
N_b

$T_{Kab}(N_b)$

- Bob will believe that he is talking to Alice

15

•15



Freshness Attacks

- A fix for the previous protocol … add a timestamp

Alice → Sherlock: A, B, N_a

$t_a \leftarrow T_{Kas}(B, K_{ab}, N_a)$
$c_a \leftarrow E_{kas}(B, K_{ab}, N_a, t_a)$
$t_b \leftarrow T_{Kbs}(A, K_{ab})$
$\tau \leftarrow \text{time}()$
$c_b \leftarrow E_{kbs}(A, K_{ab}, t_b, \tau)$

Sherlock → Alice: c_a, c_b

Alice → Bob: c_b

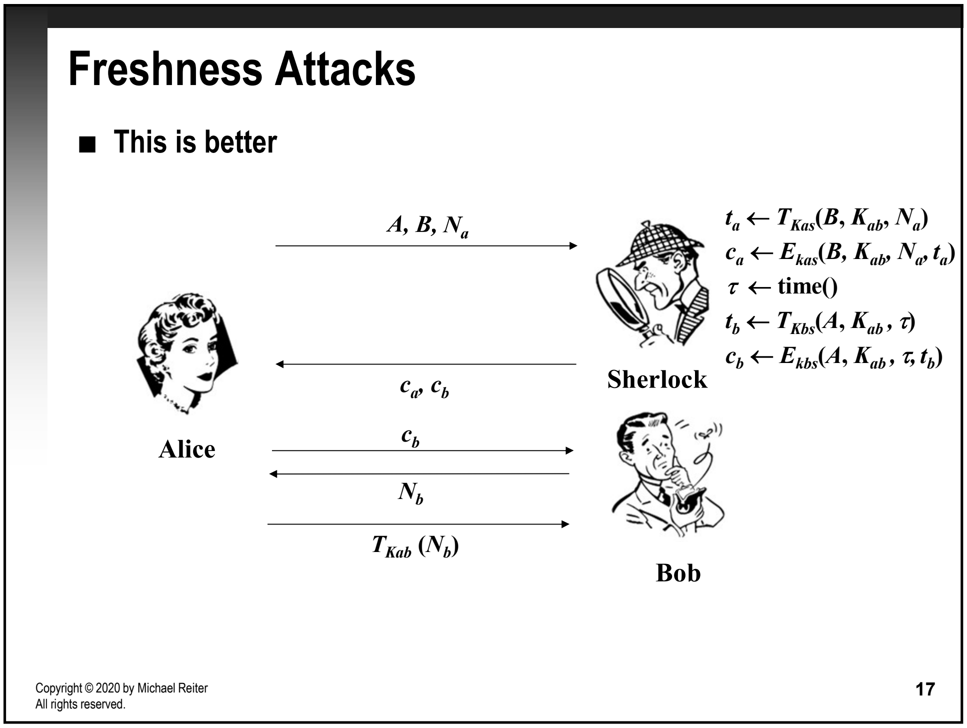
Bob → Alice: N_b

Alice → Bob: $T_{Kab}(N_b)$

- Does this fix work?

16

•16



Freshness Attacks

- This is better

Alice → Sherlock: A, B, N_a

$t_a \leftarrow T_{Kas}(B, K_{ab}, N_a)$
$c_a \leftarrow E_{kas}(B, K_{ab}, N_a, t_a)$
$\tau \leftarrow \text{time}()$
$t_b \leftarrow T_{Kbs}(A, K_{ab}, \tau)$
$c_b \leftarrow E_{kbs}(A, K_{ab}, \tau, t_b)$

Sherlock → Alice: c_a, c_b

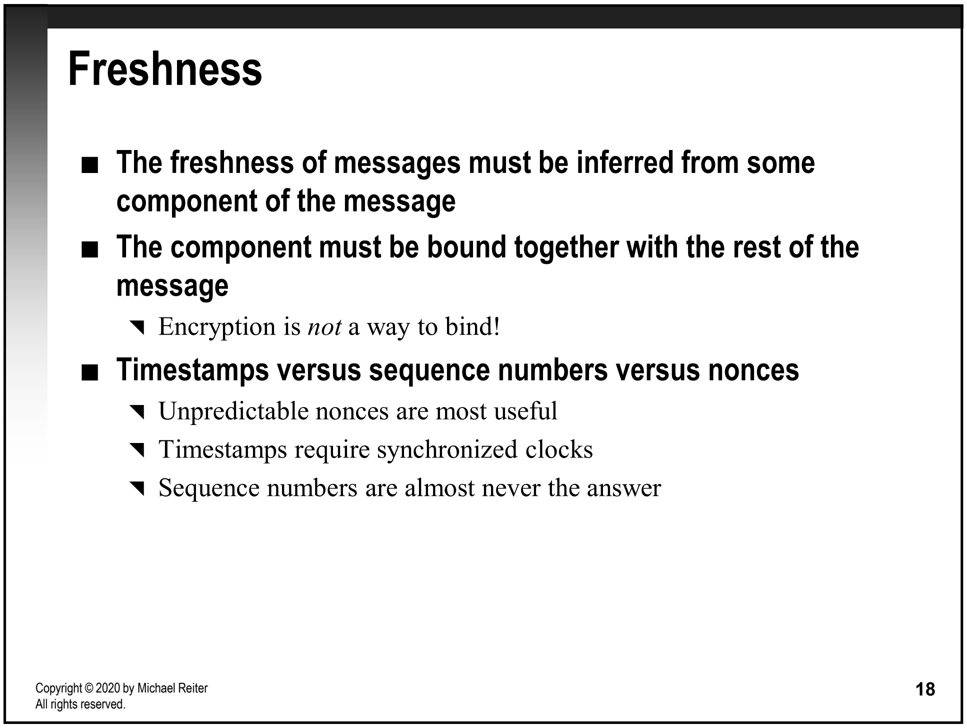
Alice → Bob: c_b

Bob → Alice: N_b

Alice → Bob: $T_{Kab}(N_b)$

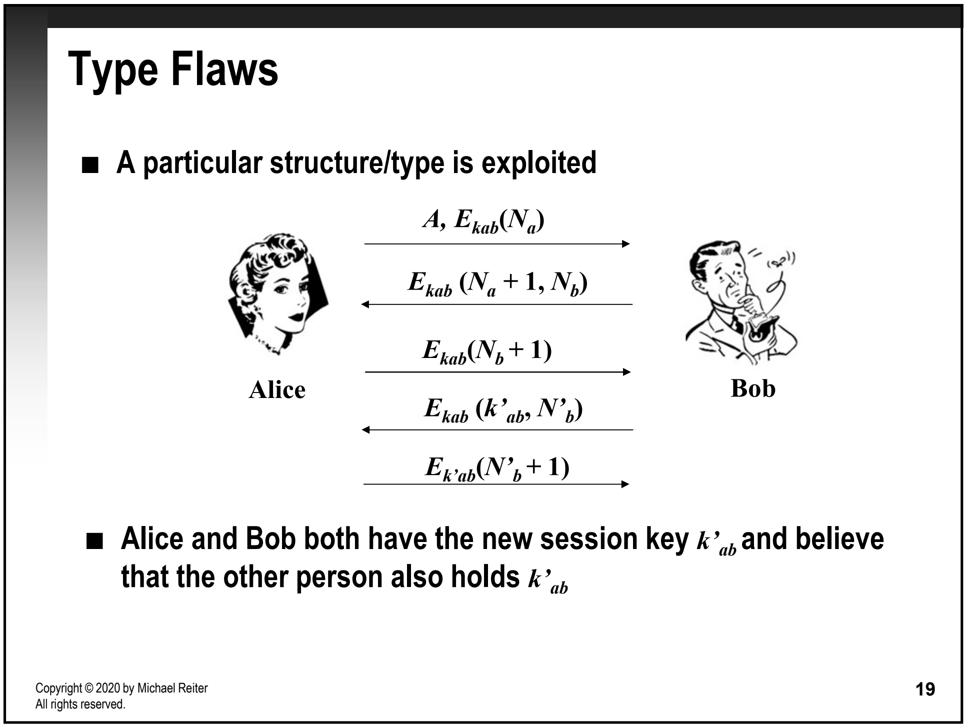
17

•17



Freshness

- The freshness of messages must be inferred from some component of the message
- The component must be bound together with the rest of the message
  - Encryption is *not* a way to bind!
- Timestamps versus sequence numbers versus nonces
  - Unpredictable nonces are most useful
  - Timestamps require synchronized clocks
  - Sequence numbers are almost never the answer



Type Flaws

- A particular structure/type is exploited

$A, E_{kab}(N_a)$ →

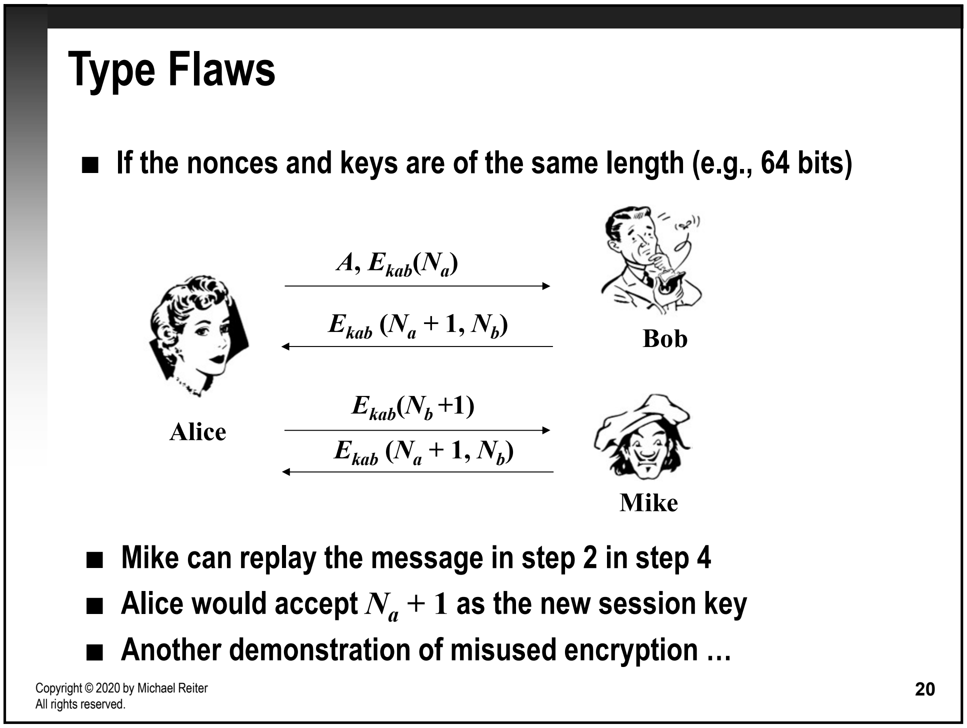
← $E_{kab}\ (N_a + 1, N_b)$

$E_{kab}(N_b + 1)$ →

← $E_{kab}\ (k'_{ab}, N'_b)$

$E_{k'ab}(N'_b + 1)$ →

- Alice and Bob both have the new session key k'_{ab} and believe that the other person also holds k'_{ab}



Type Flaws

- If the nonces and keys are of the same length (e.g., 64 bits)

$A, E_{kab}(N_a)$

$E_{kab}\ (N_a + 1, N_b)$

Bob

Alice

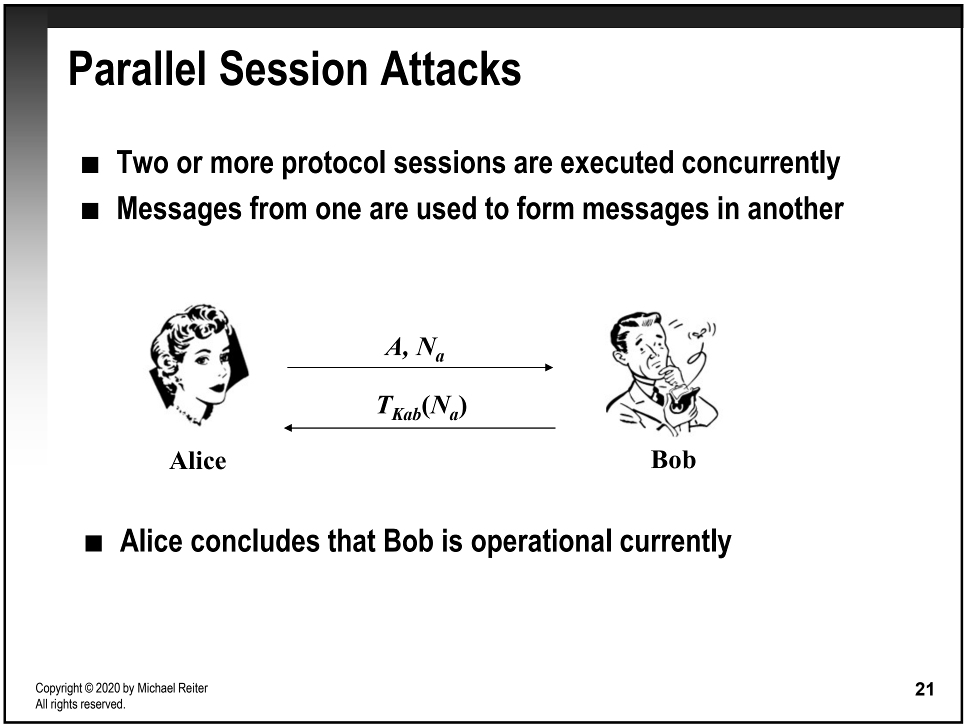
$E_{kab}(N_b + 1)$

$E_{kab}\ (N_a + 1, N_b)$

Mike

- Mike can replay the message in step 2 in step 4
- Alice would accept $N_a + 1$ as the new session key
- Another demonstration of misused encryption …

•20



Parallel Session Attacks

- Two or more protocol sessions are executed concurrently
- Messages from one are used to form messages in another

A, N_a

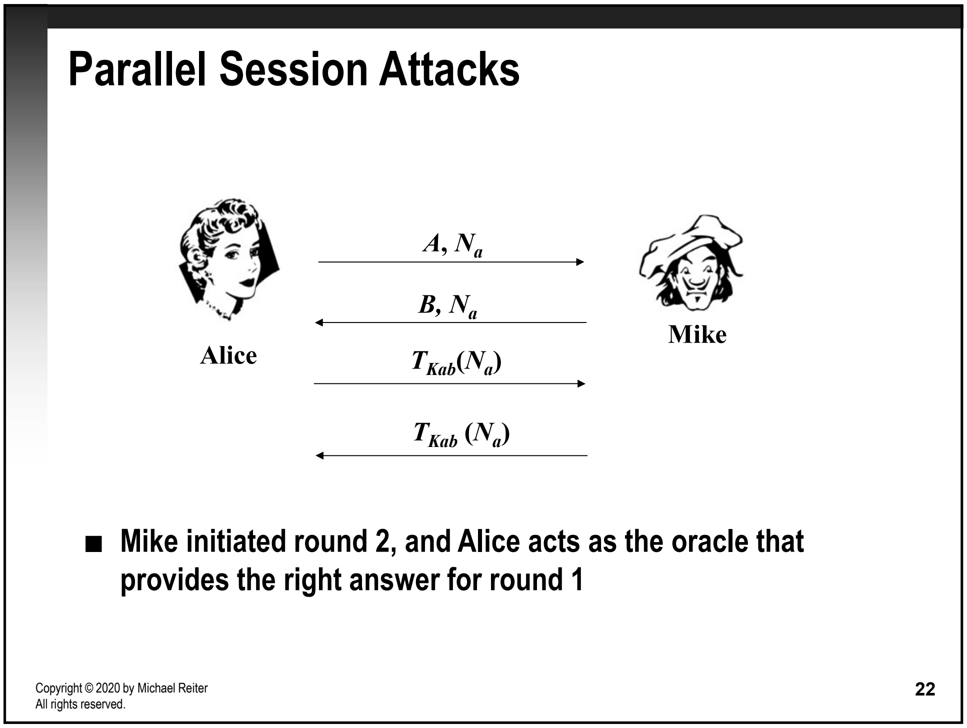
$T_{Kab}(N_a)$

Alice

Bob

- Alice concludes that Bob is operational currently

•21



Parallel Session Attacks

Alice

Mike

A, N_a →

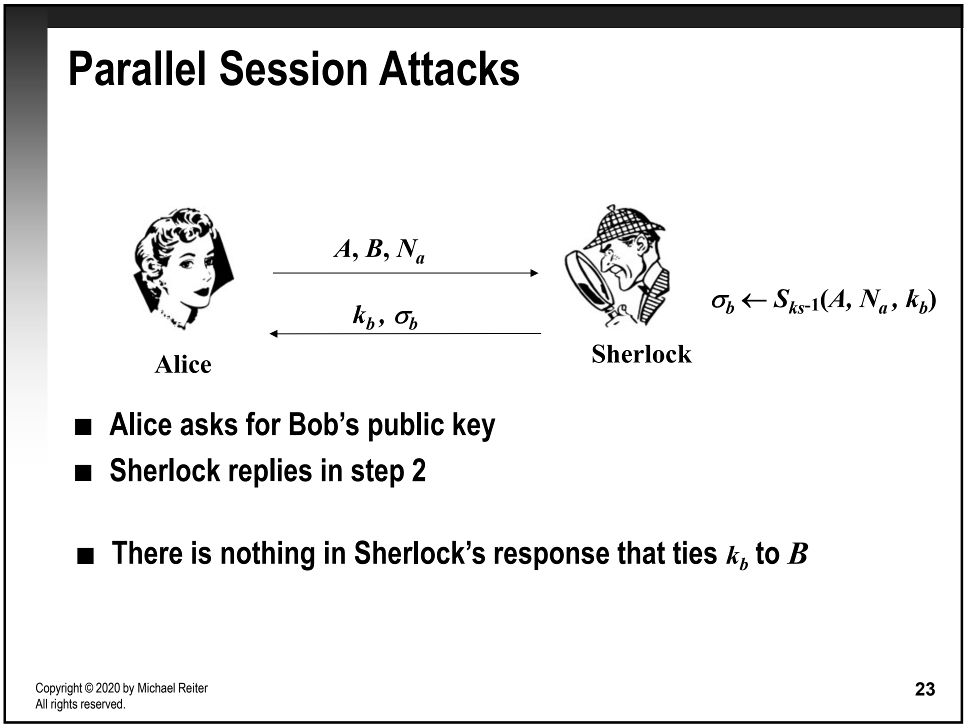
← B, N_a

$T_{Kab}(N_a)$ →

← $T_{Kab}\ (N_a)$

- Mike initiated round 2, and Alice acts as the oracle that provides the right answer for round 1

•22



Parallel Session Attacks

Alice

Sherlock

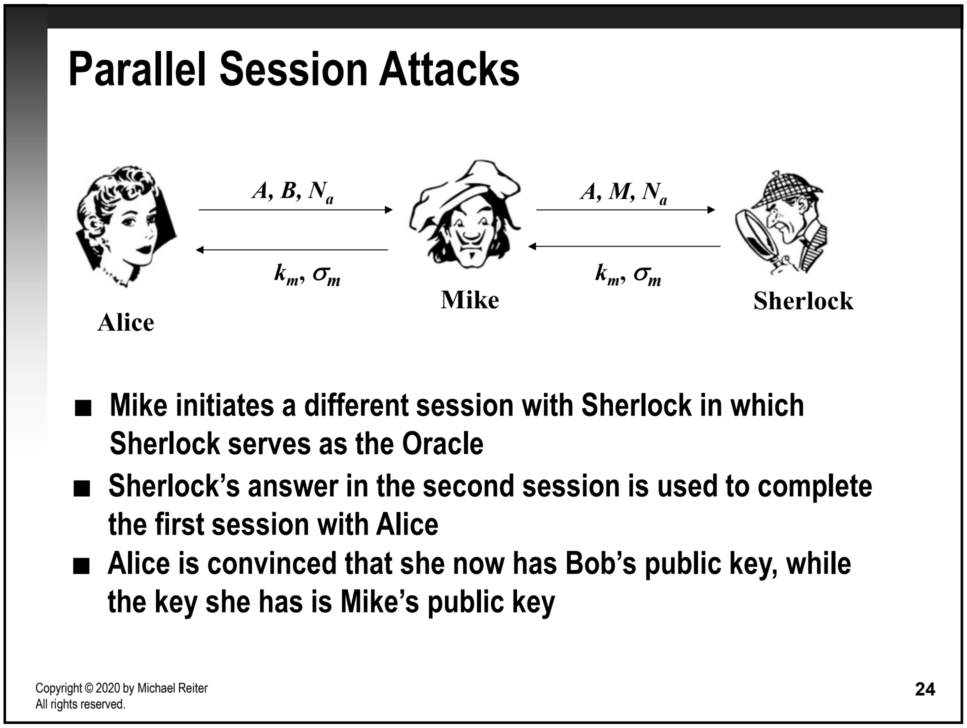
A, B, N_a →

← k_b, σ_b

$\sigma_b \leftarrow S_{ks^{-1}}(A, N_a, k_b)$

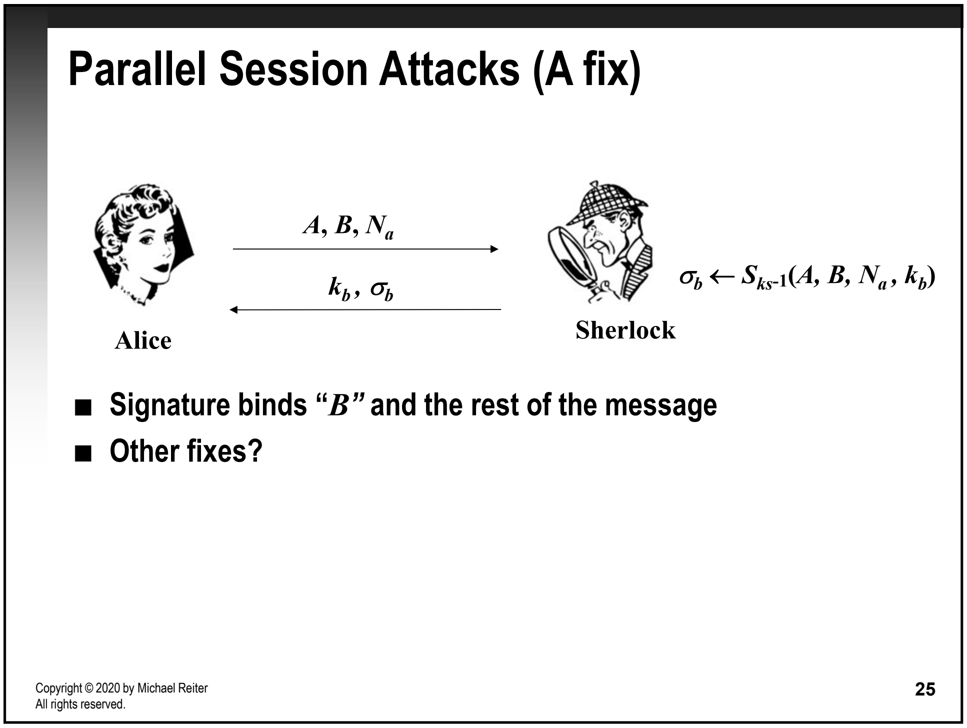
- Alice asks for Bob’s public key
- Sherlock replies in step 2
- There is nothing in Sherlock’s response that ties k_b to B

•23



Parallel Session Attacks

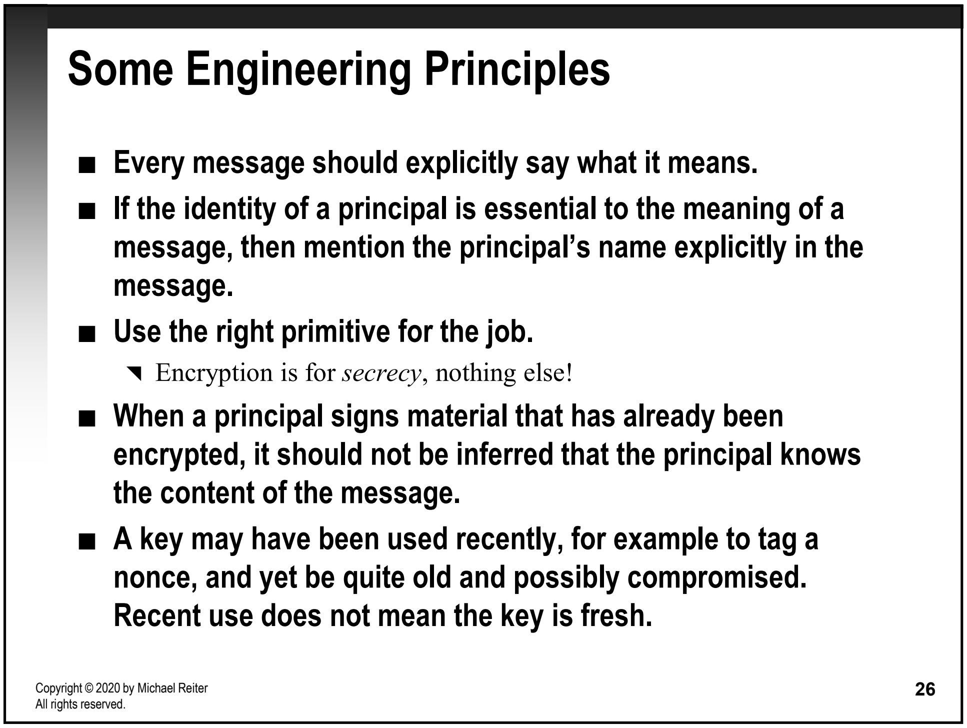
- Mike initiates a different session with Sherlock in which Sherlock serves as the Oracle
- Sherlock's answer in the second session is used to complete the first session with Alice
- Alice is convinced that she now has Bob's public key, while the key she has is Mike's public key



Parallel Session Attacks (A fix)

$\sigma_b \leftarrow S_{ks^{-1}}(A, B, N_a, k_b)$

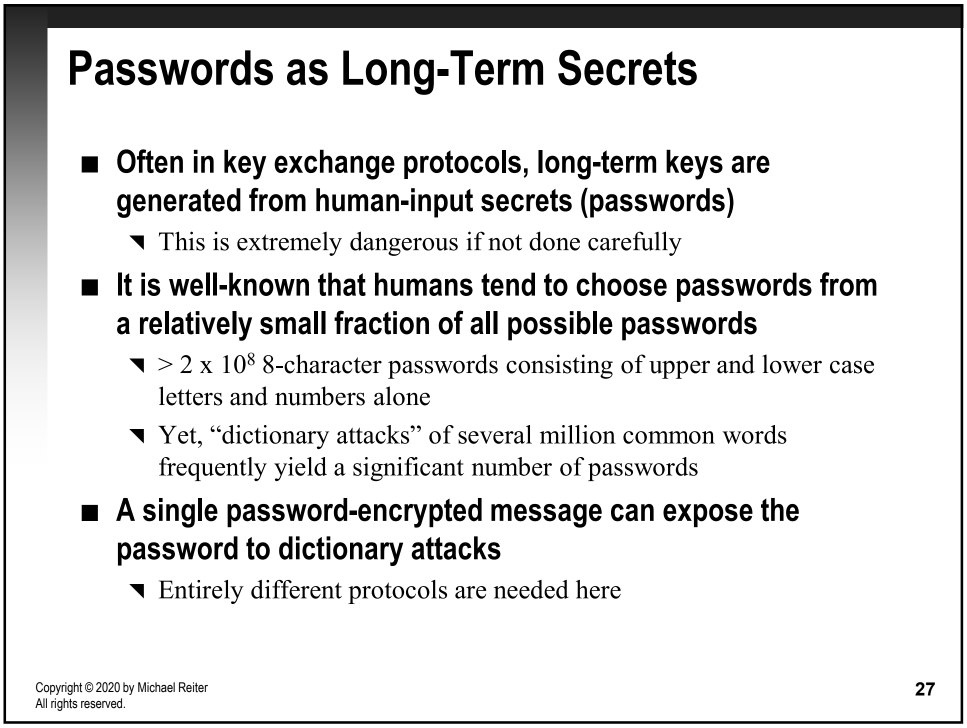
- Signature binds "B" and the rest of the message
- Other fixes?



Some Engineering Principles

- Every message should explicitly say what it means.
- If the identity of a principal is essential to the meaning of a message, then mention the principal’s name explicitly in the message.
- Use the right primitive for the job.
  - Encryption is for *secrecy*, nothing else!
- When a principal signs material that has already been encrypted, it should not be inferred that the principal knows the content of the message.
- A key may have been used recently, for example to tag a nonce, and yet be quite old and possibly compromised. Recent use does not mean the key is fresh.

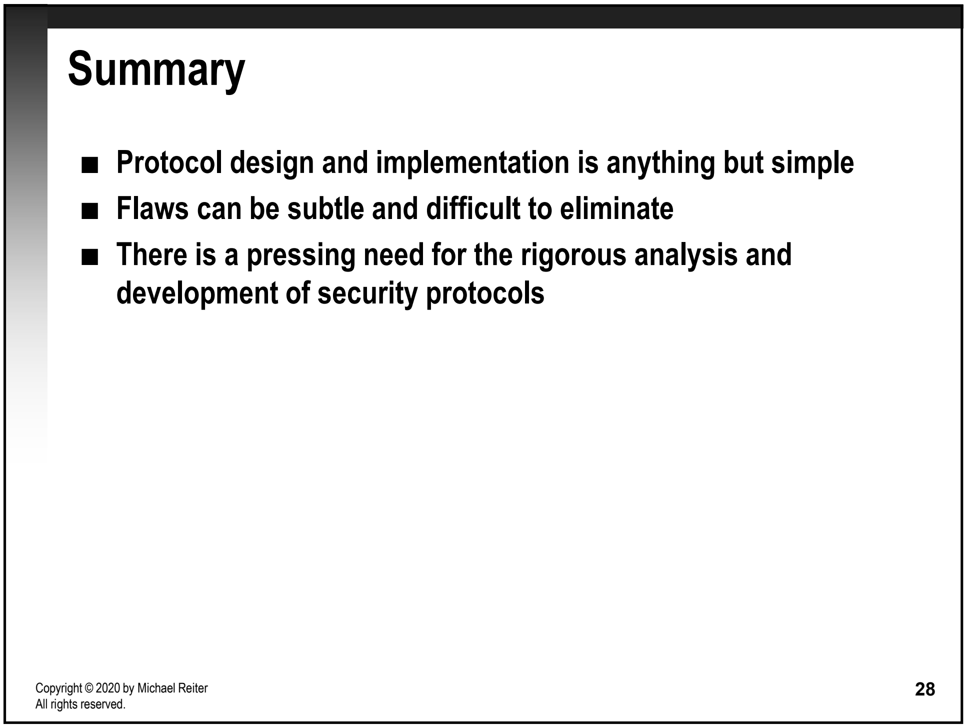
•26



Passwords as Long-Term Secrets

- Often in key exchange protocols, long-term keys are generated from human-input secrets (passwords)
  - This is extremely dangerous if not done carefully
- It is well-known that humans tend to choose passwords from a relatively small fraction of all possible passwords
  - $> 2 \times 10^8$ 8-character passwords consisting of upper and lower case letters and numbers alone
  - Yet, “dictionary attacks” of several million common words frequently yield a significant number of passwords
- A single password-encrypted message can expose the password to dictionary attacks
  - Entirely different protocols are needed here

•27



Summary

- Protocol design and implementation is anything but simple
- Flaws can be subtle and difficult to eliminate
- There is a pressing need for the rigorous analysis and development of security protocols