رمزنگاری، امنیت اطلاعات و حریم خصوصی

ارائه: دکتر سیدعلی لاجوردی

بخش چهارم

Malleability

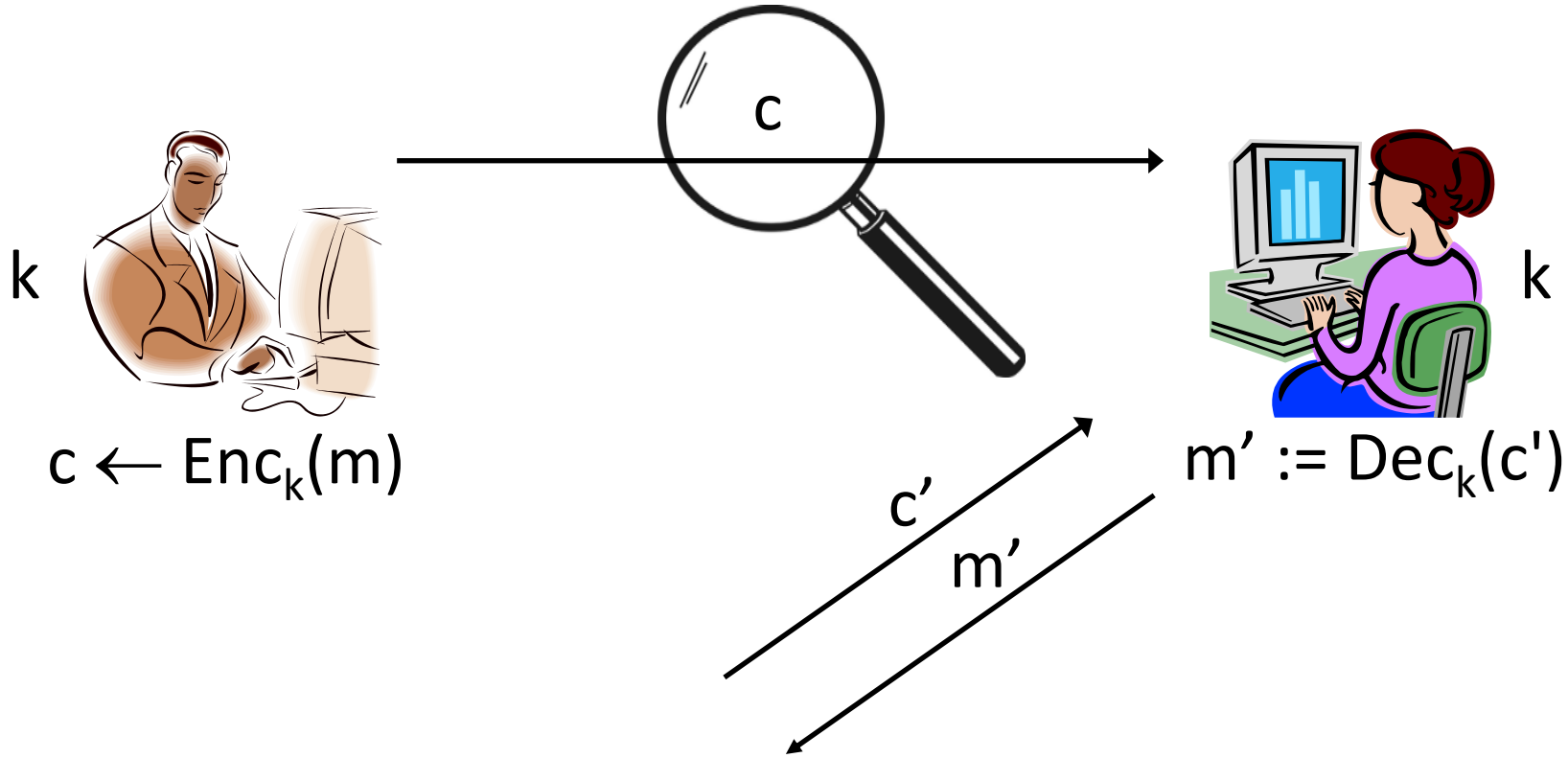
- (Informal:) A scheme is malleable if it is possible to modify a ciphertext and thereby cause a predictable change to the plaintext

- Malleability can be dangerous!
  - E.g., encrypted bank transactions

- All the encryption schemes we have seen so far are malleable!
- E.g., the one-time pad...
  - Perfect secrecy does not imply non-malleability!
- Similar attacks (and sometimes others) on all the encryption schemes we have seen so far

Chosen-ciphertext attacks

- Models settings in which the attacker can influence what gets decrypted, and observe the effects
  - I.e., interact with the receiver (who decrypts) in addition to the sender (who encrypts)


k
c
k
c ← Enc_k(m)
m' := Dec_k(c')
c'
m'

Chosen-ciphertext attacks

- Models settings in which the attacker can influence what gets decrypted, and observe the effects
  - How to model?
- Allow attacker to submit ciphertexts of its choice* to the receiver, and learn the corresponding plaintext
  - In addition to being able to carry out a chosen-plaintext attack!

*With one restriction, described next

CCA-security

- In the definition of CCA-security, the attacker can obtain the decryption of any ciphertext of its choice (besides the challenge ciphertext)
  - Is this realistic?

- In the real world the attacker would not have access to a full decryption oracle, but might learn partial information about decrypted ciphertexts
  - In many such cases, submitting the challenge ciphertext would give no additional information

Chosen-ciphertext attacks and malleability

- If a scheme is malleable, then it cannot be CCA-secure
  - Modify c, submit modified ciphertext c’ to the decryption oracle and determine (information about) the original message based on the result

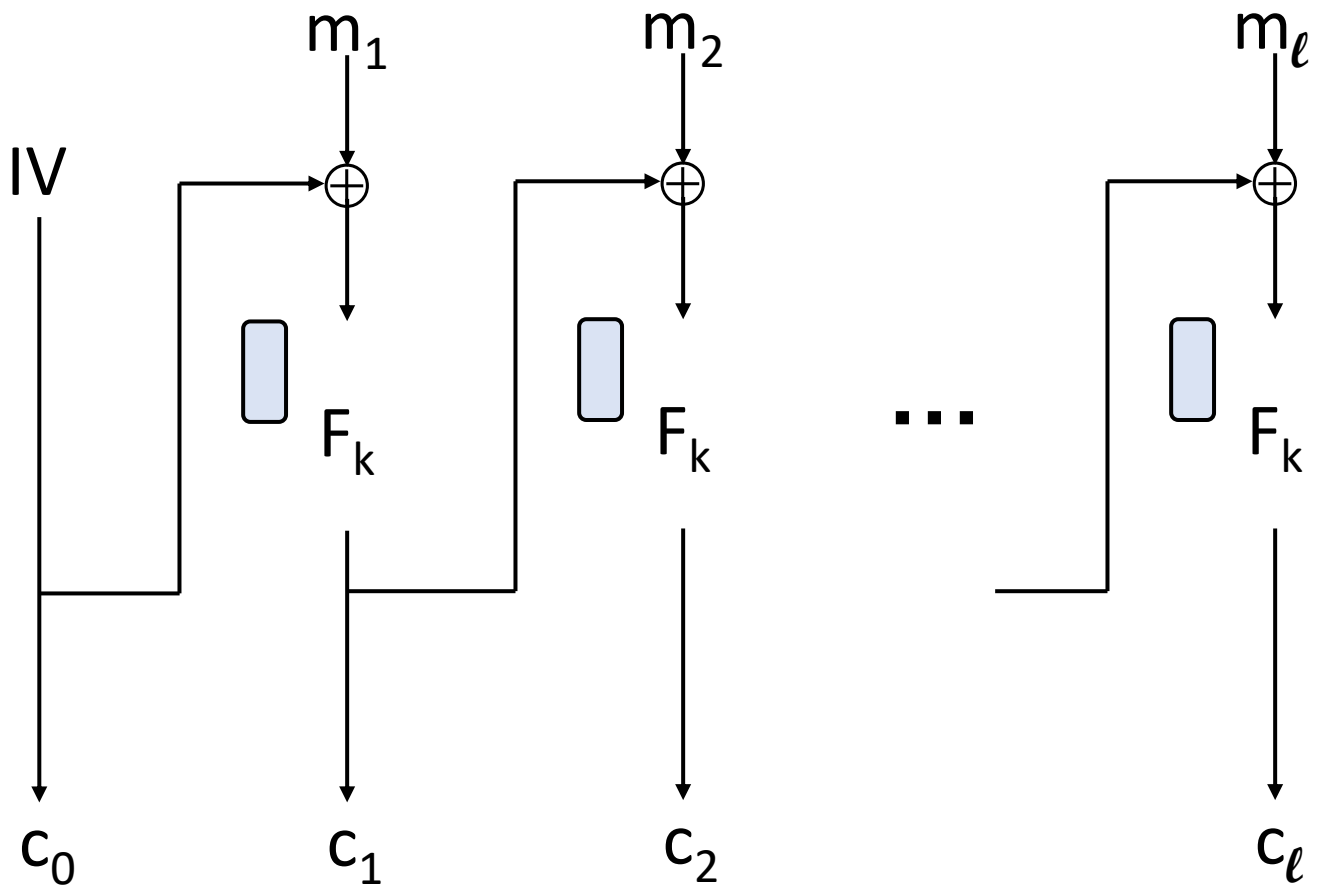
- CCA-security implies non-malleability
  - So we will focus on CCA-security

Padding-oracle attacks

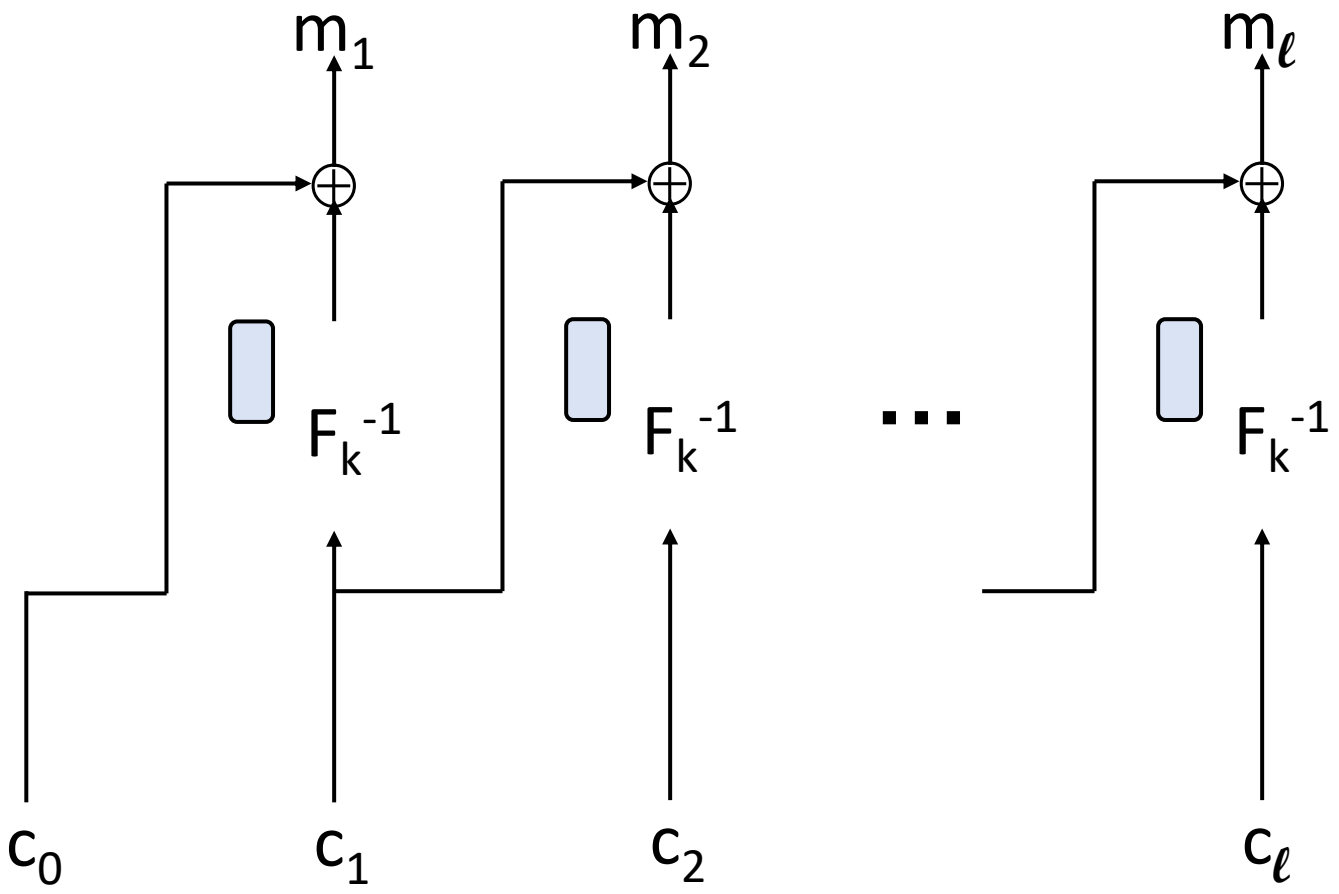
- We show a scenario where:
  - One bit about decrypted ciphertexts is leaked
  - The scenario occurs in the real world!
  - It can be exploited to learn the entire plaintext

- In this scenario, submitting the challenge ciphertext gives no additional information

CBC-mode encryption



CBC-mode decryption



Padding oracles

- Padding oracles are often present in, e.g., web applications

- Even if an error is not explicitly returned, an attacker might be able to detect differences in timing, behavior, etc. after decryption

Main idea of the attack

- Consider a two-block ciphertext IV, c
  - Encoded data = Fk-1(c) ⊕ IV
  - Goal is to learn the encoded data

- Main observation: If an attacker modifies (only) the ith byte of IV, this causes a predictable change to (only) the ith byte of the encoded data

CCA-security: a summary

- Chosen-ciphertext attacks are a significant, real-world threat
  - Modern encryption schemes are designed to be CCA-secure
- None of the schemes we have seen so far is CCA-secure
- We are going to consider an even stronger notion of security…

Authenticated encryption

Secrecy + integrity?

- We have shown primitives for achieving secrecy and integrity in the private-key setting

- What if we want to achieve both?
  - Against active attackers

Authenticated encryption

- An encryption scheme that achieves both secrecy and integrity
- Secrecy notion: CCA-security
- Integrity notion: unforgeability
  - Adversary cannot generate any ciphertext that decrypts to a previously unencrypted message
  - This is not implied by CCA-security

Generic constructions?

- “Generic” = modular construction based on any CPA-secure encryption scheme and any secure MAC

- We consider three natural choices
  - Encrypt and authenticate
  - Authenticate-then-encrypt
  - Encrypt-then-authenticate

Encrypt and authenticate

k1, k2

c, t

k1, k2

m

$c \leftarrow Enc_{k1}(m)$

$t = Mac_{k2}(m)$

$m = Dec_{k1}(c)$

$Vrfy_{k2}(m, t) = 1?$

Problems

- The tag t might leak information about m!
  - Nothing in the definition of security for a MAC implies that it hides information about m
  - So the combination may not even be EAV-secure

- If the MAC is deterministic (like CBC-MAC), then the tag leaks whether the same message is encrypted twice
  - I.e., the combination will not be CPA-secure

Problems

- Padding-oracle attack still works (if possible to distinguish padding failure from MAC failure)
  - So may not be CCA-secure

- Other counterexamples showing that it is not necessarily CCA-secure are also possible

Encrypt-then-authenticate

k1, k2

m

$c \leftarrow Enc_{k1}(m)$

$t = Mac_{k2}(c)$

c, t

k1, k2

$Vrfy_{k2}(c, t) = 1?$

$m = Dec_{k1}(c)$

Security?

- Theorem: If the underlying encryption scheme is CPA-secure and the MAC is secure (with unique tags) then encrypt-then-authenticate is a CCA-secure encryption scheme

- Encrypt-then-authenticate is the preferred generic approach for building an AE scheme

- Note: independent keys must be used!

Direct constructions

- Other, more-efficient constructions have been proposed and are an active area of research and standardization

- E.g., GCM, CCM, OCB, …

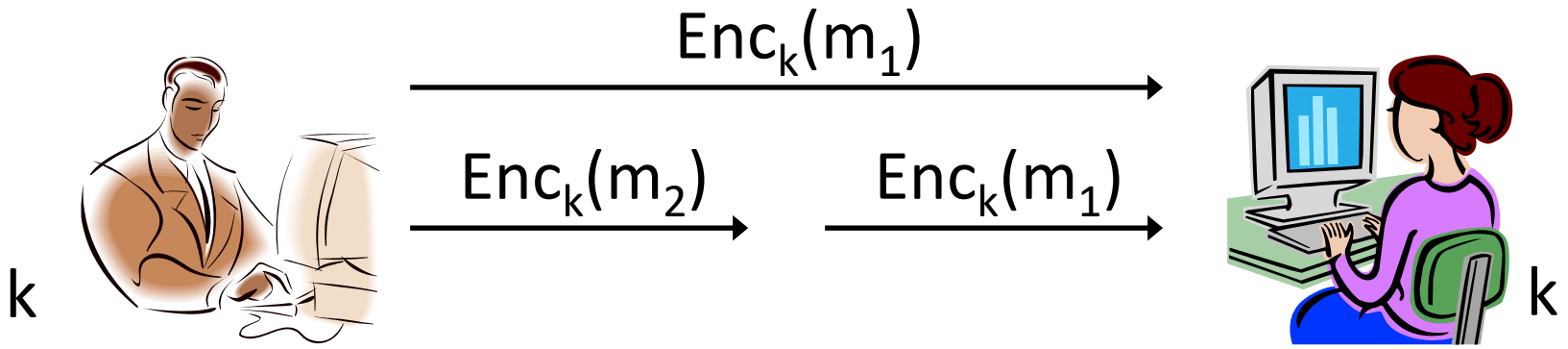
Secure sessions

Secure sessions?

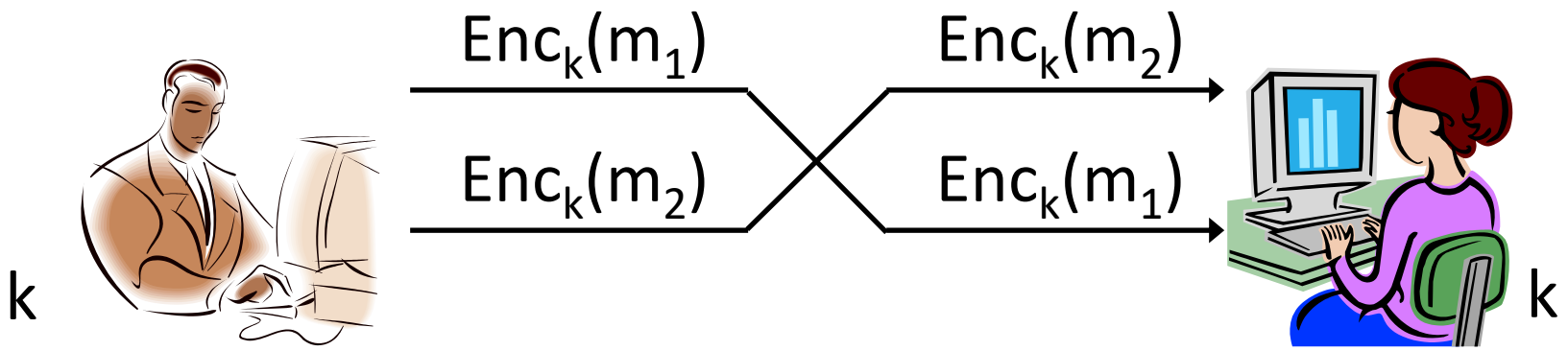
- Consider parties who wish to communicate securely over the course of a session
  - “Securely” = secrecy and integrity
  - “Session” = period of time during which the parties maintain state

- Use authenticated encryption…?

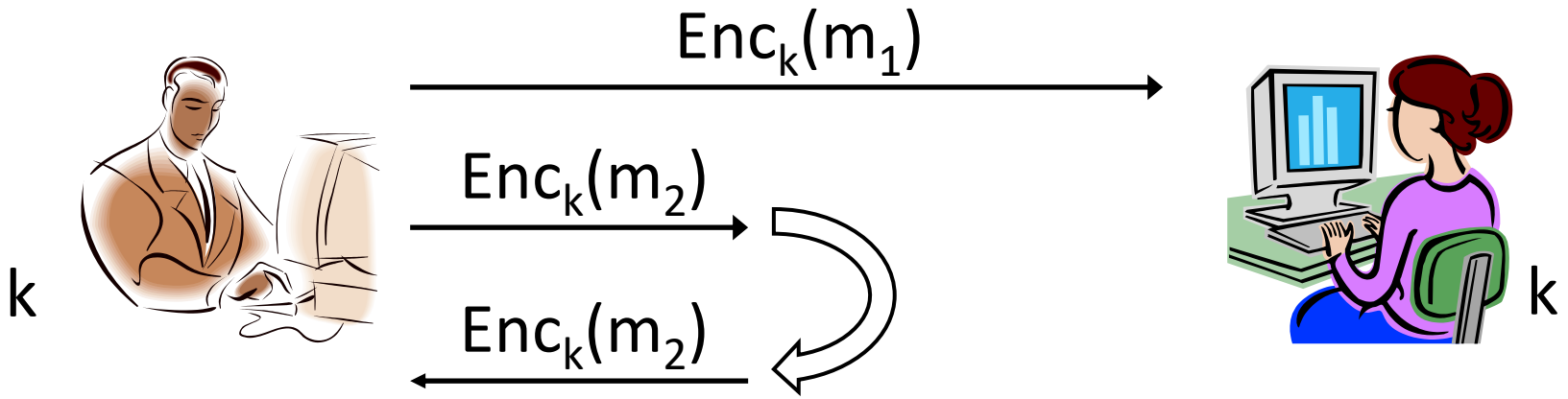
Replay attack



Re-ordering attack

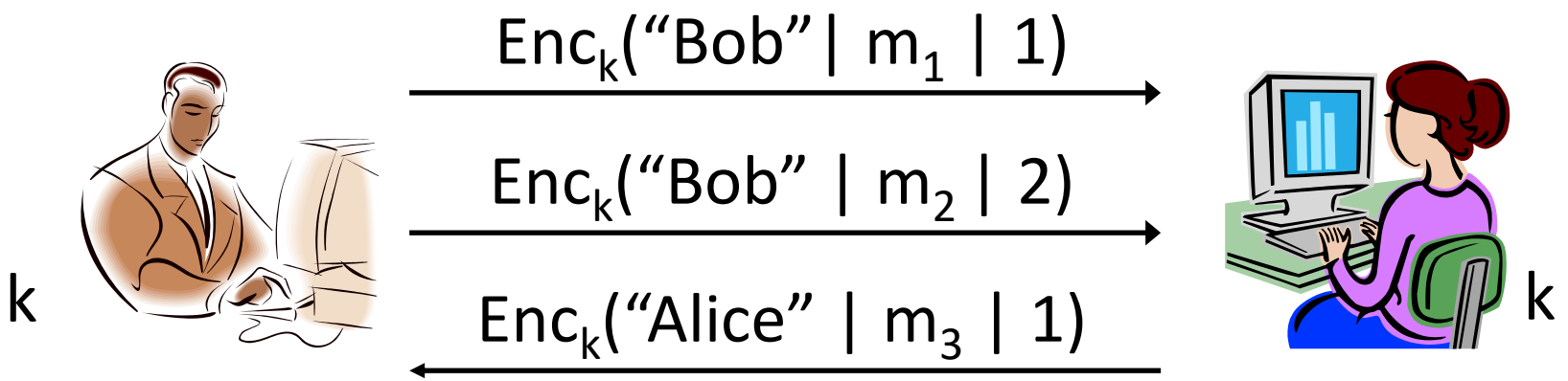


Reflection attack



Secure sessions

- These attacks (and others) can be prevented using counters/sequence numbers and identifiers


Enc_k("Bob"| m_1 | 1)
Enc_k("Bob" | m_2 | 2)
Enc_k("Alice" | m_3 | 1)
k
k

Secure sessions

- These attacks (and others) can be prevented using counters and identifiers
  - Can also use a directionality bit in place of identifiers