Security: Principles & Symmetric Key Cryptography

CS 352, Lecture 18

http://www.cs.rutgers.edu/~sn624/352-S19

Srinivas Narayana

(heavily adapted from slides by Prof. Badri Nath and the textbook authors)

RUTGERS UNIVERSITY | NEW BRUNSWICK

Why Network Security?

- Malicious people share your network
  - People who want to snoop
  - People who want to destroy
  - People who want to corrupt
  - People who want to pretend
  - People who want to steal
- Problem made more severe as Internet becomes more commercialized
- Active and passive attacks

Key aspects of network security

confidentiality: only sender, intended receiver should “understand” message contents

- sender encrypts message
- receiver decrypts message

integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

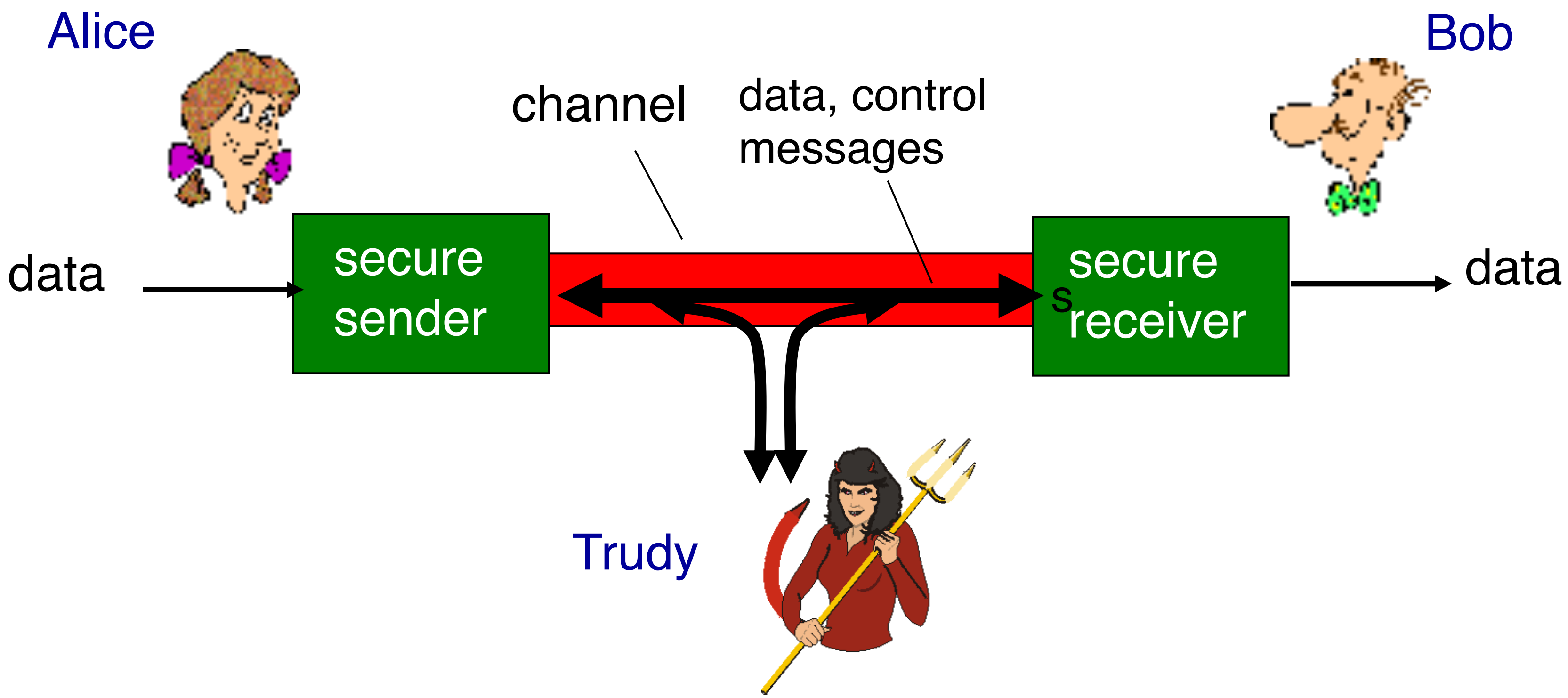
authentication: sender, receiver want to confirm identity of each other

non-repudiation: Once someone sends a message, or conducts a transaction, she can’t later deny the contents of that message

availability: services must be accessible and available to users

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob and Alice want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



Who might Bob and Alice be?

- Real humans ☺
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates
- other examples?

There are bad actors out there!

Q: What can a "bad actor" do?

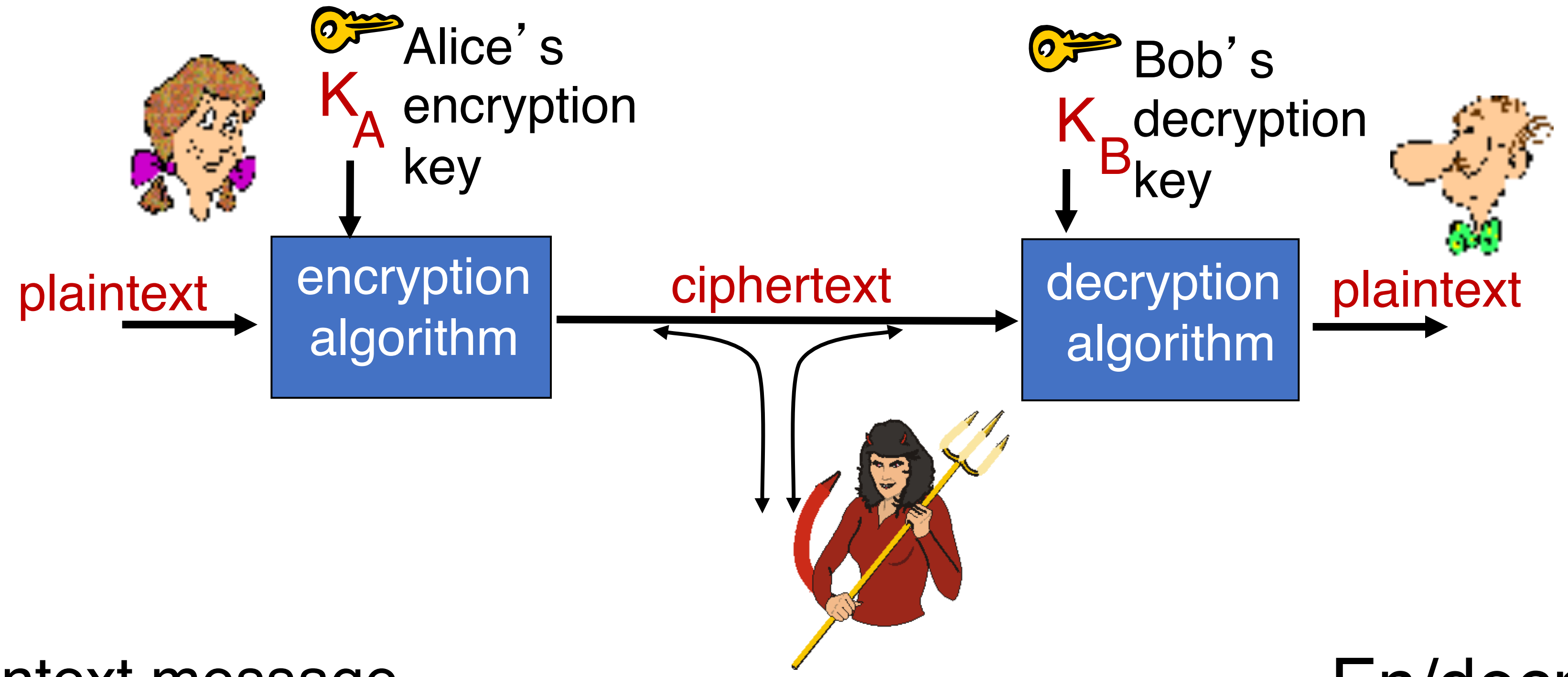
A: A lot!

- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting itself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

Cryptography

Preventing adversaries from reading private messages

Terminology of cryptography



m plaintext message

$c = K_A(m)$, $K_A(m)$ ciphertext, encrypted with key K_A

$m' = K_B(c)$, $K_B(c)$ decrypted plaintext with key K_B

Want: $m = K_B(K_A(m))$

Want: $K_A(m)$ to be uncorrelated with m

En/decryption algorithms are also called ciphers.

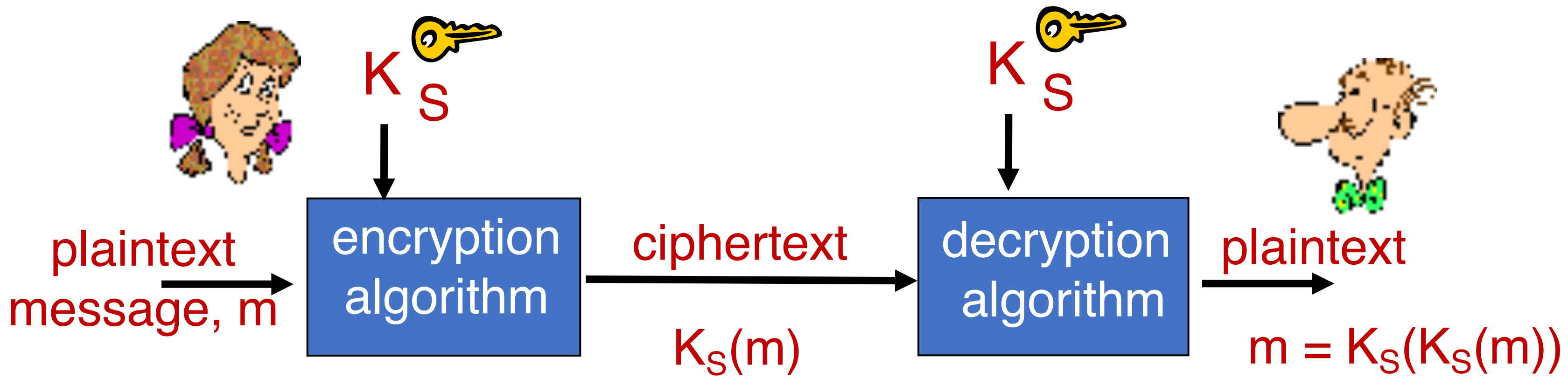
Cryptography: Algorithms and Keys

- Cryptography requires both an en-/decryption algorithm and “keys”
  - Key is a string known only to Alice and Bob, which controls how algorithm works

- Algorithm
  - Should be public and known to all
    - Inspires trust that the algorithm works

- Keys:
  - Should be long enough to prevent easy breaking of the encryption
  - Should be short enough to keep algorithm efficient
  - Typical key lengths: 56-bit, 128-bit, 256-bit, 512-bit

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: S

Q: how do Bob and Alice agree on key value?

Key techniques of symmetric key crypto:

Substitution and *Permutation*

Substitution-based ciphers

- monoalphabetic cipher: substitute one letter for another
- Ex: Caesar cipher: Each letter is replaced by a shift: succ(2), pred(3)
- More generally, map letters to other letters

plaintext: abcdefghijklmnopqrstuvwxyz

ciphertext: mnbvcxzasdfghjklpoiuytrewq

e.g.:

Plaintext: bob, i love you. alice

ciphertext: nkn, s gktc wky. mgsbc

Key: mapping from set of 26 letters to set of 26 letters

Problem: Easy to break by analyzing statistical properties of written language

Polyalphabetic cipher

- n substitution ciphers, $M_1, M_2, \ldots, M_n$
- cycling pattern:
  - e.g., n=4: M_1, M_3, M_4, M_3, M_2; M_1, M_3, M_4, M_3, M_2; ..
- for each new plaintext symbol, use subsequent substitution pattern in cyclic pattern
  - dog: d from M_1, o from M_3, g from M_4

- *Encryption key:* n substitution ciphers, and the cyclic pattern

Permutation-based ciphers

- Instead of substituting letters in the plaintext, we change their order

A	N	D	R	E	W
1	4	2	5	3	6
t	h	i	s	i	s
a	m	e	s	s	a
g	e	i	w	o	u
l	d	l	i	k	e
t	o	e	n	c	r
y	p	t	n	o	w

Key = ANDREW
(used to define a permutation based on alphabet order)

Plaintext = thisisamessageiwould
liketoencryptnow

Ciphertext = tiihssaesmsagioewul
lkdietecdnrytopnw

Also possible to break by analyzing structure of language

Encryption in practice

- Most actual encryption algorithms use a complex combination of substitution and permutation
- Examples:
  - Data Encryption Standard (DES)
    - Multiple iterations of substitution and transposition using a 56-bit key
    - designed by IBM with input from the NSA
  - International Data Encryption Algorithm (IDEA)
    - uses a 128-bit key
  - Advanced Encryption Standard (AES)

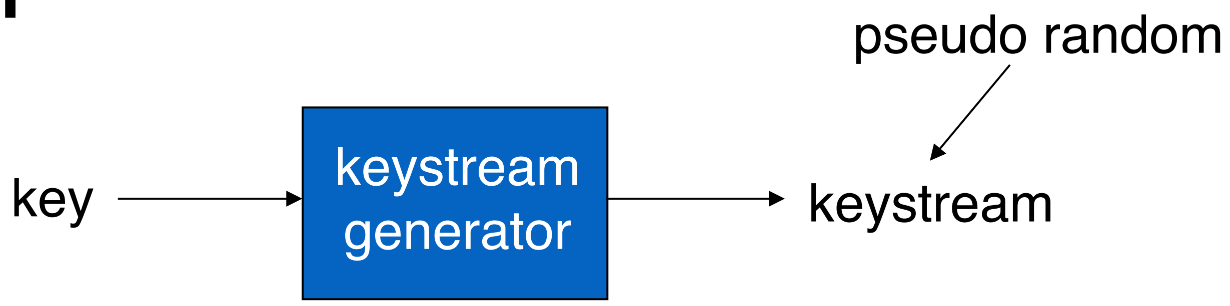
Stream and Block Ciphers

Two types of symmetric ciphers

- Stream ciphers
  - encrypt one bit at time

- Block ciphers
  - Break plaintext message in equal-size blocks
  - Encrypt each block as a unit

Stream Ciphers



- Combine each bit of keystream with bit of plaintext to get one bit of ciphertext
- m(i) = i^{th} bit of message
- ks(i) = i^{th} bit of keystream
- c(i) = i^{th} bit of ciphertext
- $c(i) = ks(i) \oplus m(i)$ ($\oplus$ = XOR)
- $m(i) = ks(i) \oplus c(i)$

Block ciphers

- Message to be encrypted is processed in blocks of k bits (e.g., 64-bit blocks).
- 1-to-1 mapping is used to map k-bit block of plaintext to k-bit block of ciphertext

Example with k=3:

input	output
000	110
001	111
010	101
011	100

input	output
100	011
101	010
110	000
111	001

What is the ciphertext for 010110001111 ?

Block ciphers

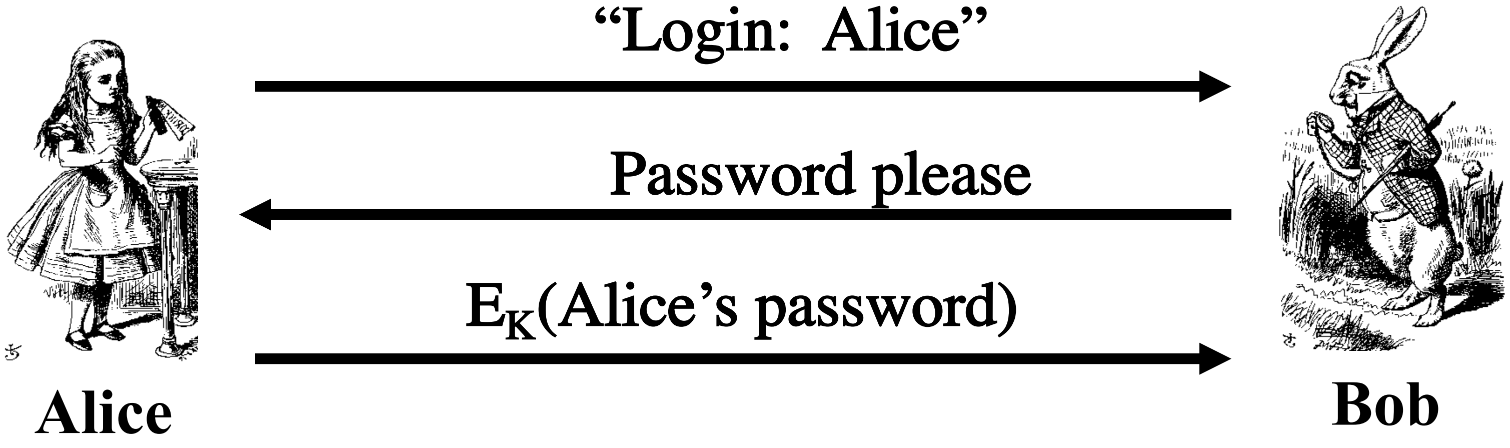
- How many possible mappings are there for k=3?
  - How many 3-bit inputs?
  - How many mappingsor permutations of the 3-bit inputs? 8!
  - Answer: 40,320 ; not very many!
- In general, 2^k! mappings; huge for k=64
- Problem:
  - Table approach requires table with 2^{64} entries, each entry with 64 bits
- Table too big: instead use function that simulates a randomly permuted table
- Many practical ciphers are block-based (DES, AES, …)

Problems with Symmetric Key Cryptography

Encryption using symmetric keys

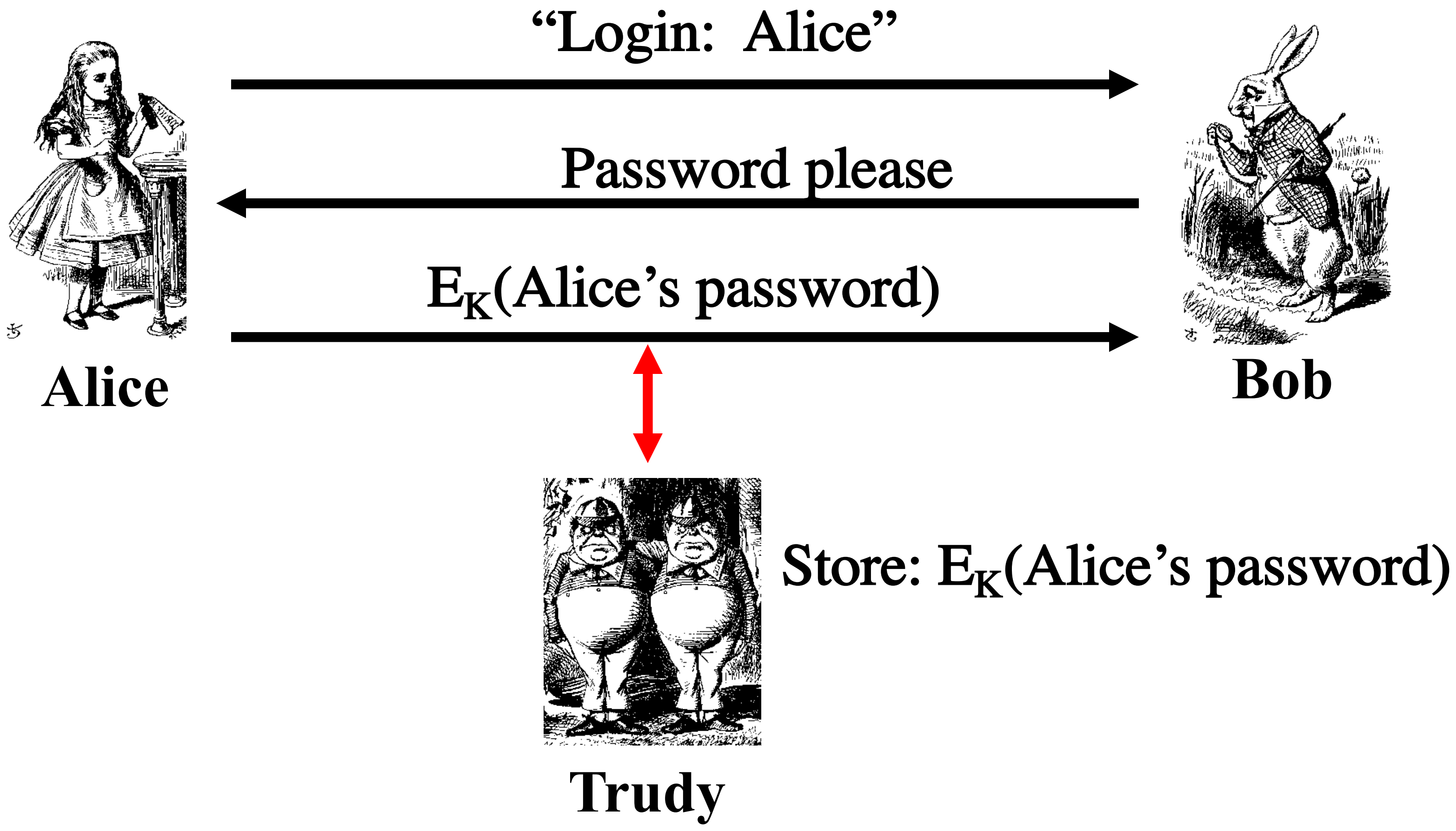
- Same key for encryption and decryption
- Efficient to implement: Often the same or very similar algorithm for encryption and decryption
- Achieves confidentiality
- Vulnerable to tampering
  - Bad guy can alter the encrypted message
- What about authentication?
- Vulnerable to replay attacks
  - Bad guy can steal the encrypted message and later present it on behalf of a legitimate user

Replay attack

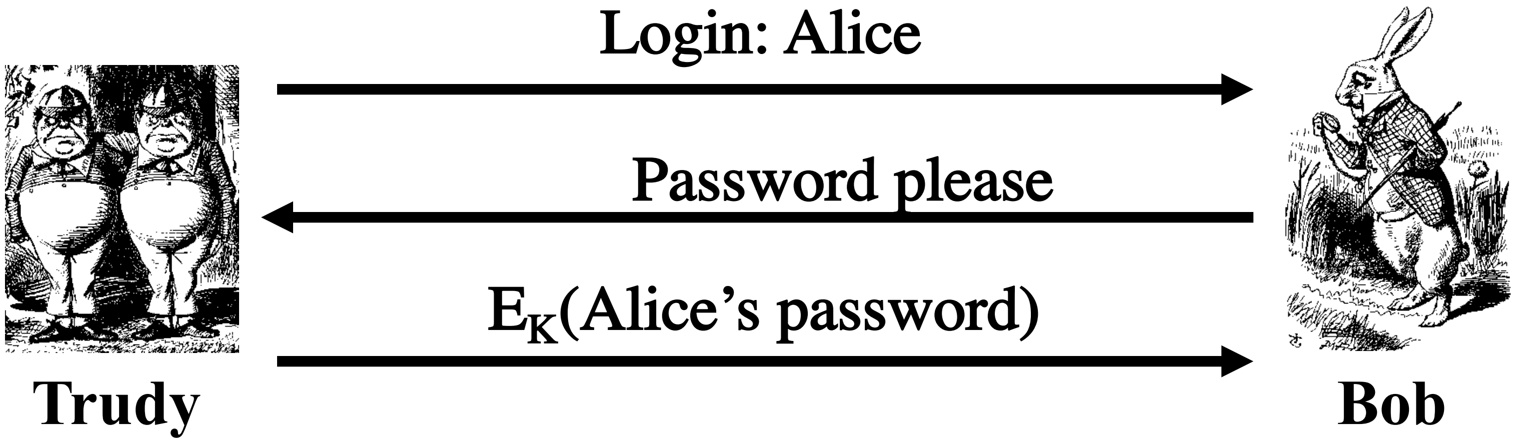


- Alice's password is encrypted
  - From both Bob and attackers
  - If Bob is trusted, he can decrypt using the same key
- But still subject to replay attack

Replay attack

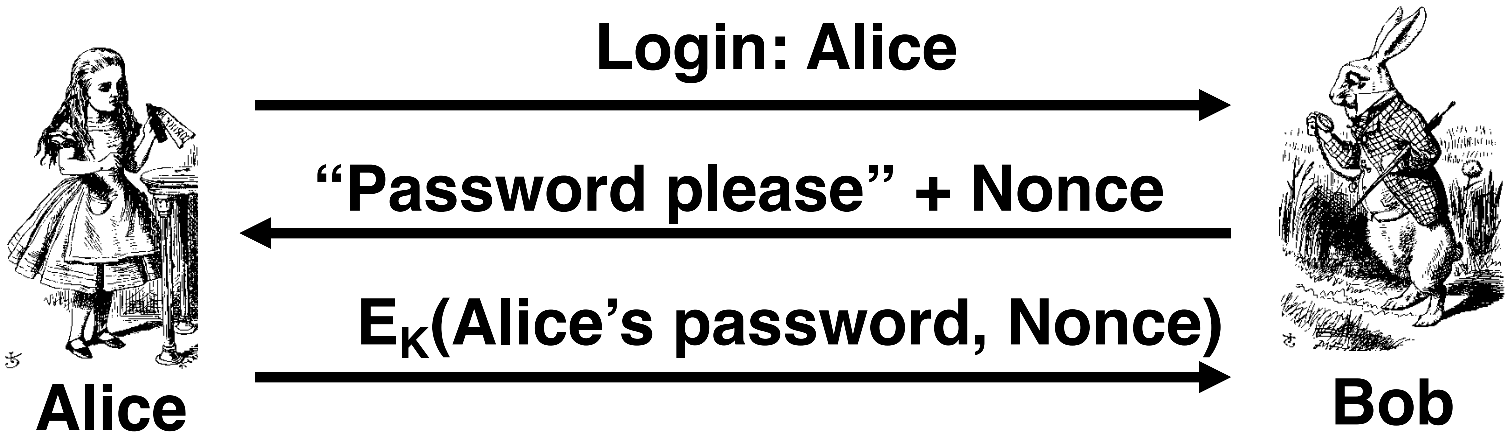


Replay attack



- This is a **replay** attack
- How can we prevent a replay?
- By adding a NONCE value; Number used once only
- Use a temporary random number!

Challenge-Response



- Nonce is the challenge
- The encrypted msg is the response
- Nonce changed every time
- Prevents replay, ensures freshness
- Even if Trudy steals the encrypted Nonce value, it won’t work in the next login (nonce changed)

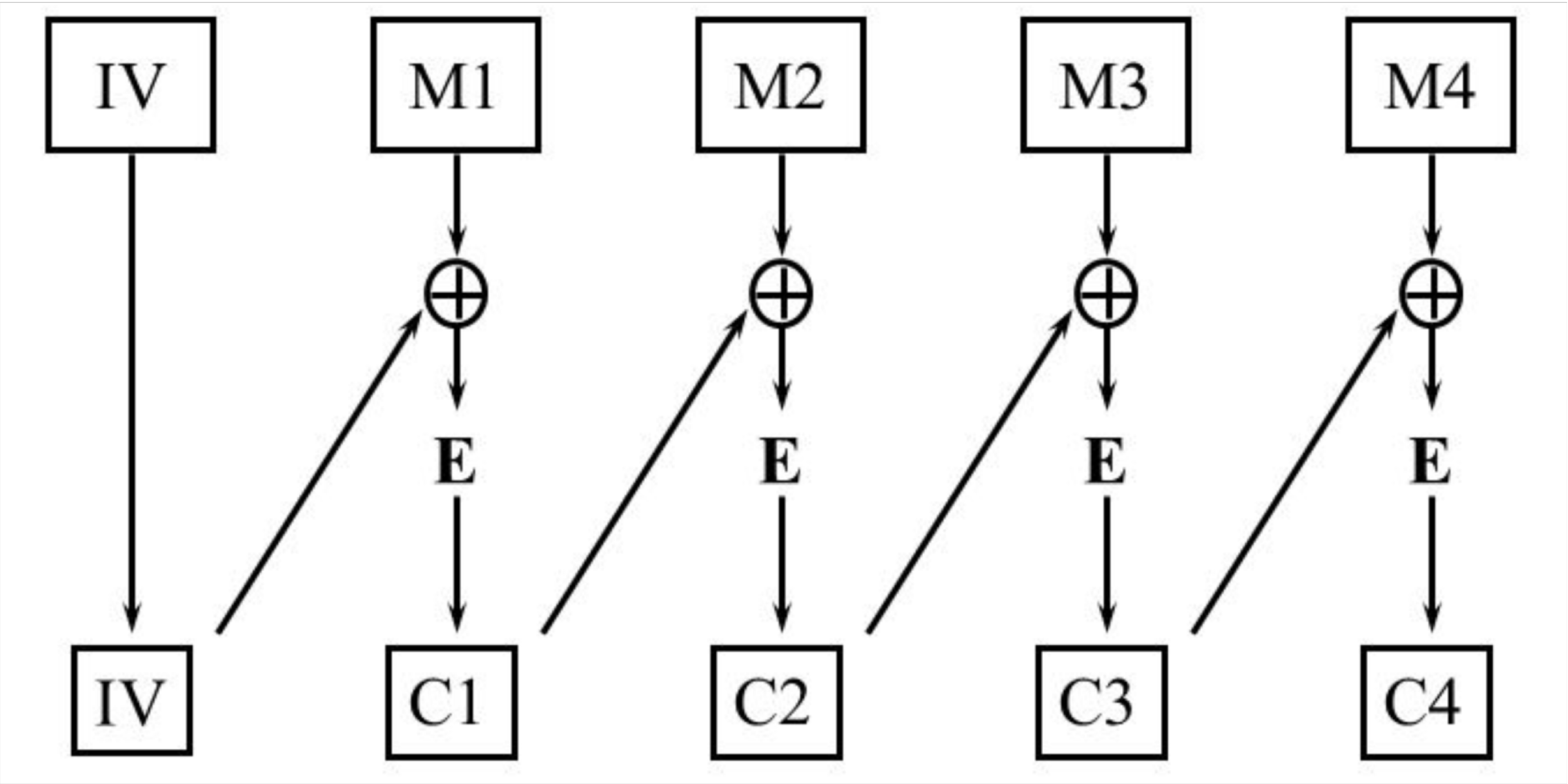
General problems with repeated ciphertext

- Block ciphers take chunks of info (ex: 64-bit) to other chunks
- Previous example: Passwords can be replayed
- But more generally, easy to guess parts of payload with repeated plaintext
  - Example: HTTP/1.1
  - Then use those parts of a message to guess other parts of the payload
  - … and so on

Generalizing nonces for all messages

- Nonces can be sent as plain text
- Can we use a nonce on *every* message to prevent replay?
  - Yes!
  - Send nonce, $E_k(\text{message} \oplus \text{nonce})$ to transfer message
- But very inefficient: Double bits for every message
- Use a method to generate nonces automatically
- Cipher block chaining: use the previous ciphertext as a nonce for the next plain text block
- First block randomized using Initialization Vector (IV)

Cipher block chaining: Encryption



Exercise: how would decryption work?

How to agree on a shared secret key?

- In reality: two parties may meet in person or communicate “out of band” to exchange shared key
- But communicating parties may never meet in person
  - Example: An online retailer and customer
  - Much more common for a network ☺
- What if the shared secret is stolen?
  - All secret communications can now be decrypted and are visible
- Is there a way to communicate securely without worrying about secure key exchange?

Next lecture: Public key cryptography

RC4 Stream Cipher

- RC4 is a popular stream cipher
  - Extensively analyzed and considered good
  - Key can be from 1 to 256 bytes
  - Used in WEP for 802.11
  - Can be used in SSL