Lecture 23: Network Security
Public Key Cryptography
COMP 411, Fall 2022
Victoria Manfredi

WESLEYAN
UNIVERSITY

Today

1. **Network tools**
   - ssh and scp

2. **Public key cryptography**
   - overview
   - RSA algorithm

3. **Network security**
   - authentication

Network Tools

SSH AND SCP

SSH: remotely connecting to a device

E.g., connect to your virtual machine from your host device

On your VM, start an ssh server
- – sudo apt-get install openssh-server
- – sudo service ssh restart
- – (normally if you're connecting to a server you won't need to do)

Connect
- – ssh username@ip_address

```
vmanfredi@ ~ () $
> ssh vmanfred@129.133.177.177
The authenticity of host '129.133.177.177 (129.133.177.177)' can't be established.
ECDSA key fingerprint is SHA256:Tt5KAXugvAX3ZRgz9V54IeBTCYAFVG2iRRel4wobLpY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '129.133.177.177' (ECDSA) to the list of known hosts.
vmanfred@129.133.177.177's password:
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.2.0-27-generic x86_64)
```

scp: copy files to/from another device

E.g., copy files to your virtual machine from your host device

Runs over ssh, so ssh server should be running
- may also want to check ports open on device

Copy
- scp file_to_copy username@ip_address:~/path_to_put_file

```
vmanfredi@ ~ () $
> scp run_tls_scrape.sh vmanfred@129.133.177.177:~
vmanfred@129.133.177.177's password:
Permission denied, please try again.
vmanfred@129.133.177.177's password:
run_tls_scrape.sh
                      100%  825      1.1MB/s   00:00
```

Public Key Cryptography
OVERVIEW

Problem

Symmetric key crypto requires sender, receiver share secret
- Q: how to agree on key in first place (particularly if never met)?

Public key cryptography
- 2 parties communicate without shared secret known in advance
  - radically different approach!
- applications
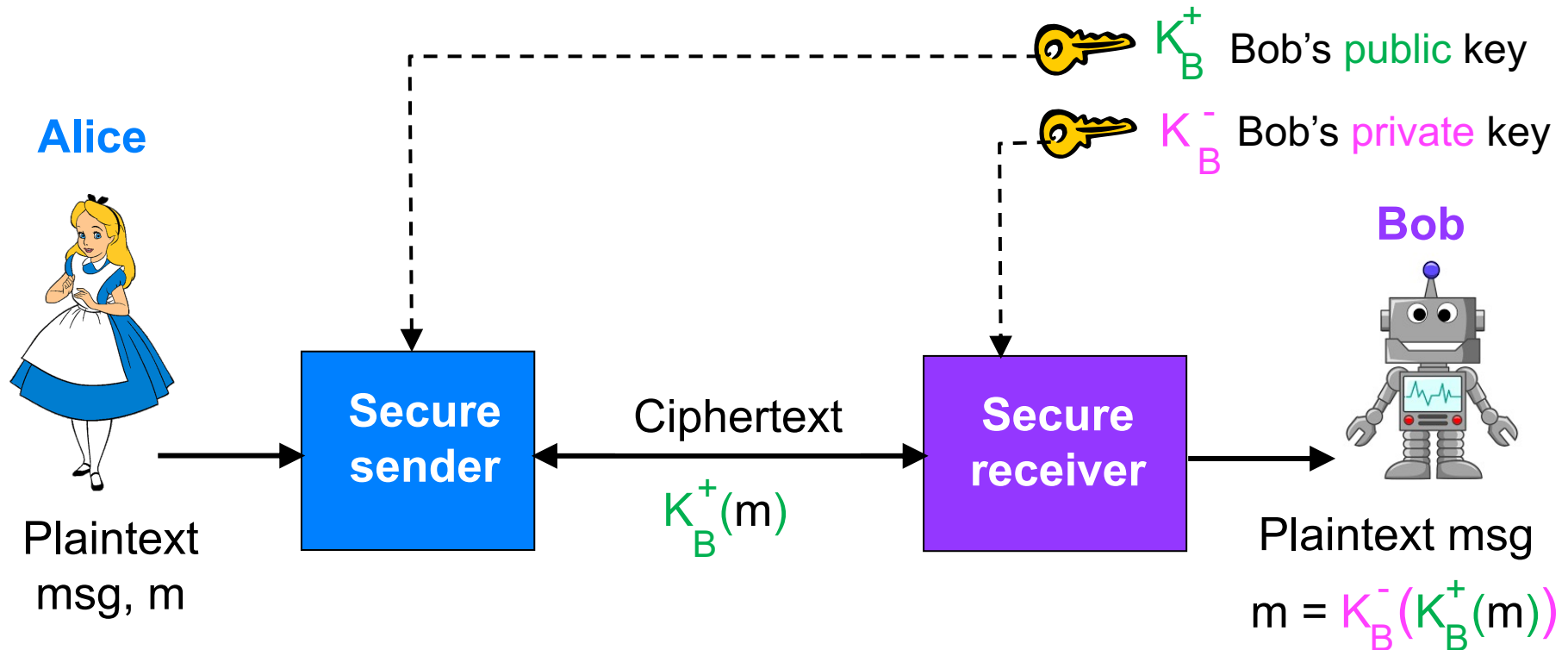 - encryption and decryption
  - digital signatures
  - key exchange

Most widely used public key cryptography algorithms
- RSA: Rivest, Shamir, Adelson (1978)
- Diffie-Hellman (1976)

Public key cryptography at high level

Each user has its own public and private key pair

- K^+: public encryption key known to all
- K^-: private decryption key known only to receiver

Alice

Plaintext msg, m

Secure sender

Ciphertext

$K_B^+(m)$

Secure receiver

Bob

Plaintext msg

$m = K_B^-\big(K_B^+(m)\big)$

K_B^+ Bob's public key

K_B^- Bob's private key

Keys mathematically linked in special way

Requirements

1. Need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$K_B^-(K_B^+(m)) = m$$

2. Given public key K_B^+, should be impossible to compute private key K_B^-

Public Key Cryptography
RSA ALGORITHM

RSA: Rivest, Shamir, Adelson algorithm

Message

- bit pattern: can be uniquely represented by integer number
    - encrypting message is equivalent to encrypting number
- example
    - m = 10010001 uniquely represented by 145
    - encrypt m by encrypting 145 which gives new number, the ciphertext

Intuition

- security of RSA is based on hardness of prime factorization
    - easy to multiply 2 large prime numbers together, hard to factor result

Recall

- x mod n: remainder of x when divided by n

Creating public/private key pair using RSA

1. **Choose two large distinct prime numbers *p, q***
   – e.g., 1024 bits each

2. **Compute *n=pq, z=(p-1)(q-1)***

3. **Choose *e (<n)* such that no common factors with *z***
   – *e, z* are relatively prime

4. **Choose *d* such that *ed-1* is exactly divisible by *z***
   – *ed* mod *z = 1, so d = e⁻¹ mod z*

5. *Public key is (n,e). Private key is (n,d).*

$$K_B^+ \qquad\qquad K_B^-$$

Encryption and decryption

Given (n,e) and (n,d) as computed before

$$K_B^+ \qquad K_B^-$$

1. To encrypt message m (<n), compute

$$c = m^e \bmod n$$

2. To decrypt received bit pattern, c, compute

$$m = c^d \bmod n$$

Magic happens!

$$m = (\overbrace{m^e \bmod n}^{c})^d \bmod n$$

14

Why does RSA work?

Must show $c^d \bmod n = m$ where $c = m^e \bmod n$

$$c^d \bmod n = (m^e \bmod n)^d \bmod n \qquad \text{by substitution}$$
$$= m^{ed} \bmod n \qquad \text{by fact 1}$$
$$= m^{(ed \bmod z)} \bmod n \qquad \text{by fact 2}$$
$$= m^1 \bmod n \qquad \text{since } ed \bmod z = 1 \text{ by design}$$
$$= m$$

fact 1: $(a \bmod n)^d \bmod n \equiv a^d \bmod n$

fact 2: $a^b \bmod n = a^{(b \bmod z)} \bmod n$ where $n = pq$ and $z = (p-1)(q-1)$

Why is RSA secure?

Given Bob's public key (n,e)

– how hard is it to determine private key (n,d)?

(Relatively) easy

– compute $n=pq$ or $z=(p-1)(q-1)$

Hard

– find factors of $n=pq$ or $z=(p-1)(q-1)$ without knowing p or q

- 2^{1024} bit # x 2^{1024} bit #

– prime factorization takes exponential time

- no (non-quantum) efficient algorithm known

Example

1. Choose two large prime numbers *p=17, q=11*

2. Compute

   n = pq = 17x11 = 187

   z = (p-1)(q-1) = 16x10 = 160

3. Choose *e=7 (<n)* that no common factors with z

4. Choose *d* such that *ed-1* is exactly divisible by *z*

   (ed -1) / z = 1

   (7d-1) / 160 = 1

   7d = 161

   d = 23

5. *Public* key is *(n=187,e=7).* *Private* key is *(n=187,d=23).*

   K_B^+ K_B^-

Another example

Bob chooses *p=5, q=7*. Then *n=35, z=24*.
  e=5 (so *e, z* relatively prime).
  d=29 (so *ed-1* exactly divisible by z).

Encrypting 8-bit messages

Encrypt

bit pattern	m	m^e	$c = m^e \bmod n$
0000l010	12	248832	17

Decrypt

c	c^d	$m = c^d \bmod n$
17	481968572106750915091411825223071697	12

Another important property

The following property will be very useful later for signatures

$$K_B^-(K_B^+(m)) = \qquad m \qquad = \qquad K_B^+(K_B^-(m))$$

use public key
first, followed by
private key

use private key
first, followed by
public key

Result is the same!

Why can key application be re-ordered?

$$K_B^-(K_B^+(m)) \ = \ \ m \ = \ \ K_B^+(K_B^-(m))$$

Follows directly from modular arithmetic

$$(m^e \bmod n)^d \bmod n \equiv m^{ed} \bmod n$$
$$\equiv m^{de} \bmod n$$
$$\equiv (m^d \bmod n)^e \bmod n$$

RSA in practice

Exponentiation in RSA is computationally intensive
- e.g., symmetric key DES at least 100x faster than RSA
- use public key crypto to establish secure connection
  - then establish symmetric session key for encrypting data

Session key, K_S
- Bob and Alice use RSA to exchange symmetric key K_S
- once both have K_S, they use symmetric key cryptography

Generate ssh keys: ssh-keygen -t rsa

```
> ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/vmanfredi/.ssh/id_rsa): test_pair
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in test_pair.
Your public key has been saved in test_pair.pub.
The key fingerprint is:
SHA256:YJ28K/kCn+dEvBpnkEm6BdbHvxpzbKZ4uQgpznWbdlQ vmanfredi@vmanfredis-MacBook-Pro-2.local
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|     . + .       |
|    o = *        |
|   . = * oE      |
|    . = S..      |
|    .+ +.+ .     |
|  . =o=o0.*      |
| o o o=&=0       |
|  o  .***.       |
+----[SHA256]-----+
```

Public key

```
  1 ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDYoldG09vq5zNjE3iO7l4fDMiRpD1rVl+cv0nts8PBCxyvvNhdP6ZTZucc
    Jx6AqH5S+4l7BV6nayW7oJ35OBPXX3TUwJcGUto2nFFjkfqEw9+1LZPiLumZ9433X17aKJ6FqHAUClbyAzm6E1e+TZIeMu3V
    I/7qbJP0XtBX5LYoBdiLDdXziMqf3/rTcThUCllyf3zFLzl6u9hgPqMLo+1BTgSqRV1roK4P7yQZD1LLpzLxjqfUfCOMHhi8
    pbYh+X3J/hKZ28hRFO7mYHuUM9zGqykAU+Ew9i9WSaQF+5K81lmEyCHtKN5xVAdXp1bLYrr1SVWydmXb+VE3gM6dZraJ vma
    nfredi@vmanfredis-MacBook-Pro-2.local
~
~
"test_pair.pub" 1L, 422C written
```

vumanfredi@wesleyan.edu   23

Private key

```
 1 -----BEGIN RSA PRIVATE KEY-----
 2 MIIEpAIBAAKCAQEA2KJXRtPb6uczYxN4ju5eHwzIkaQ9a1ZfnL9J7bPDwQscr7zY
 3 XT+mU2bnHCcegKh+UvuJewVep2slu6Cd+TgT11901MCXBlLaNpxRY5H6hMPftS2T
 4 4i7pmfeN919e2iiehahwFApW8gM5uhNXvk2SHjLt1SP+6myT9F7QV+S2KAXYiw3V
 5 84jKn9/603E4VApZcn98xS85ervYYD6jC6PtQU4EqkVda6CuD+8kGQ9Sy6cy8Y6n
 6 1HwjjB4YvKW2Ifl9yf4SmdvIURTu5mB7lDPcxqspAFPhMPYvVkmkBfuSvNZZhMgh
 7 7SjecVQHV6dWy2K69UlVsnZl2/lRN4DOnWa2iQIDAQABAoIBAGJwQmB43LG9JWib
 8 7GhmgHZzhKBJlW807HV5pspQqV8LAZoJofedeKLlW5c7X2zvI5fpnOs94WkKEzdT
 9 IPWiOcHgKmKSsQ26kFXIamNobgHuT7UwZMaesp+4Edaai6tuUbpCc8tnd2K5fH3F
10 VFWxQfhfBBuaI7e6ZvDgNKP71ZoRV8b1HbiNA/JNyOdbVH2kqYFItxKGfovvpPyM
11 1xc0J/FHhthOgGPttVWX9d2K6yh6jq9WKOmIv2TaOo71TGn/RC5RgyDJTcJJ4PIV
12 s5iO9Vv6dV+aK6WHO8+45s/pXIkOLM+tqupylubwqy6hwSong4T0DAOU/B7bePec
13 3IOtAAECgYEA9SR5L+C2n7rbCxJlCYlIehXVpCATgk+fzJgrfTKMZLIZjrgIeBUJ
14 W+GAAFPqR0z7tKq6Jp0hISFWMTckAoPs+PzqHpG7P7daE6iNdu7TzupP1I8T3Ysq
15 B6K2OyGF1D//QBoyqTv5G26I3/5rN4Zm9GsDrTdo8+lTlWlQrsmUsAECgYEA4jqg
16 2YjGKame4x/zFSqdOZKhLZVqPiUbjoN1xL41fZP9M8Fu67aLAApaRbrkHnsUPRmS
17 T39YLo3orccvJb+3DVUO2g0KXhGO0gIbuqEFS1nmVrp+vWaSeqXz+g+cu8Ab/eRb
18 IaCH8u5fTXQmkG7TJuJfGFcCCBxTfCpttSi0hokCgYEAxxEWRBXj1zPimkwWtjbA
19 HzvJ4FyX2xMTvg24CxPYRBEIhgfWAMV8cxtcWWfLcJkIMT8OqTqh44hxuMeB03Ws
20 IskmySood2ZKBHq0Xec1IurNZtvFEvvmZorwFnZzedd6TLC5gQoNkQQirFqq8Ez5
21 H/Qi6S98z80ircr210knEAECgYBM9fL4bg4z6C9URu80GS4pgtdwIW9mOst4HQK4
22 bpjV4r11mO16JLx+xAbXx++I6wgEjSl3//NoywAH9kX0ypakY3ZM+bi4Lb+pKER0
23 pgieDLROdt1c44MbVE9+l0cTnBQpuEDEXM9C9pLXT5c69WjBxqrhJeBcD/7as7hk
24 s7d00QKBgQDUvFn2tq+A/4cDqkWZxAZEhp1cHaGvg4ARm8hzLwAEFiKct8kgk5gm
25 +7ur7r/y4QSj9+DSihd/TXnui2LTX8YmAUgBJUUQs7B9muo2Gt++QtwVeKVuvRC5
26 LOMblm6EKnxJqR+SZKLdGhzC2nt2KbsQoyXk7Ew7cAuUji473q1JGQ==
27 -----END RSA PRIVATE KEY-----
~
"test_pair" 27L, 1679C written
```
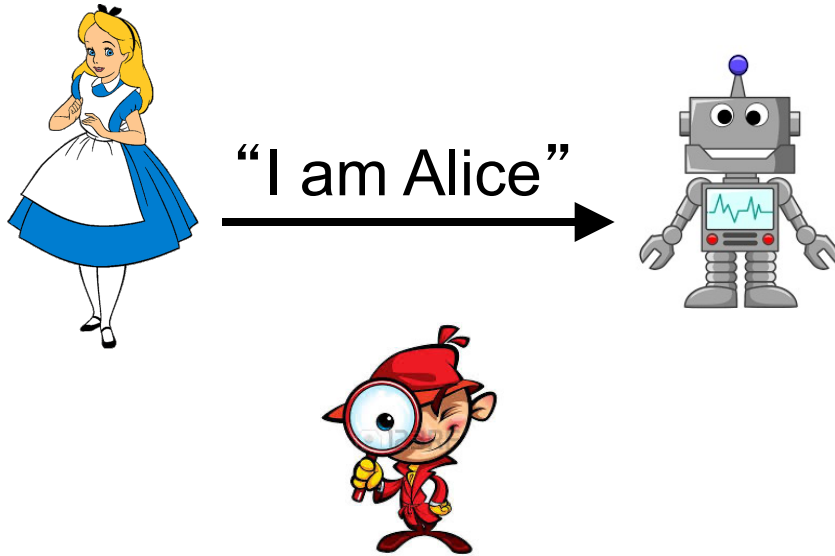
Network Security
AUTHENTICATION

Authentication

Goal: Bob wants Alice to "prove" her identity to him
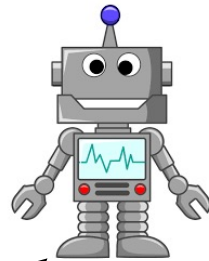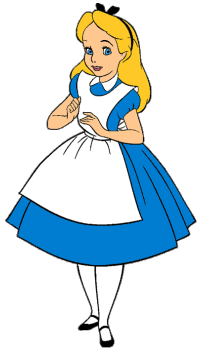
Protocol ap1.0: Alice says "I am Alice"

"I am Alice"

Failure scenario??

Authentication

Goal: Bob wants Alice to "prove" her identity to him
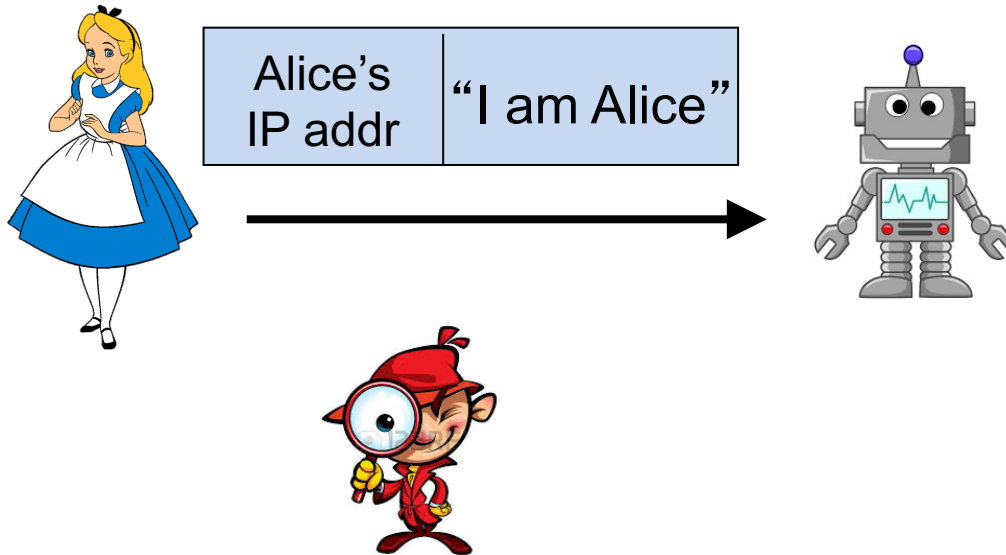
Protocol ap1.0: Alice says "I am Alice"

Failure scenario??

In network, Bob cannot see Alice, so Trudy simply declares herself to be Alice

"I am Alice"

Authentication: another try

Goal: Bob wants Alice to "prove" her identity to him

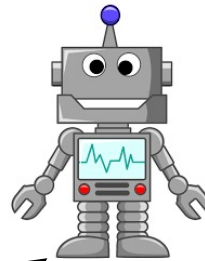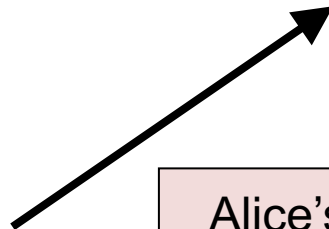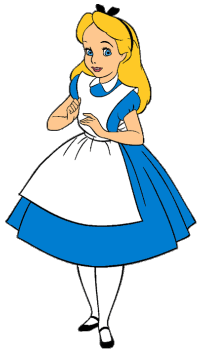Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address

Alice's IP addr | "I am Alice"

Failure scenario??

Authentication: another try

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap2.0: Alice says "I am Alice" in an IP packet
                  containing her source IP address
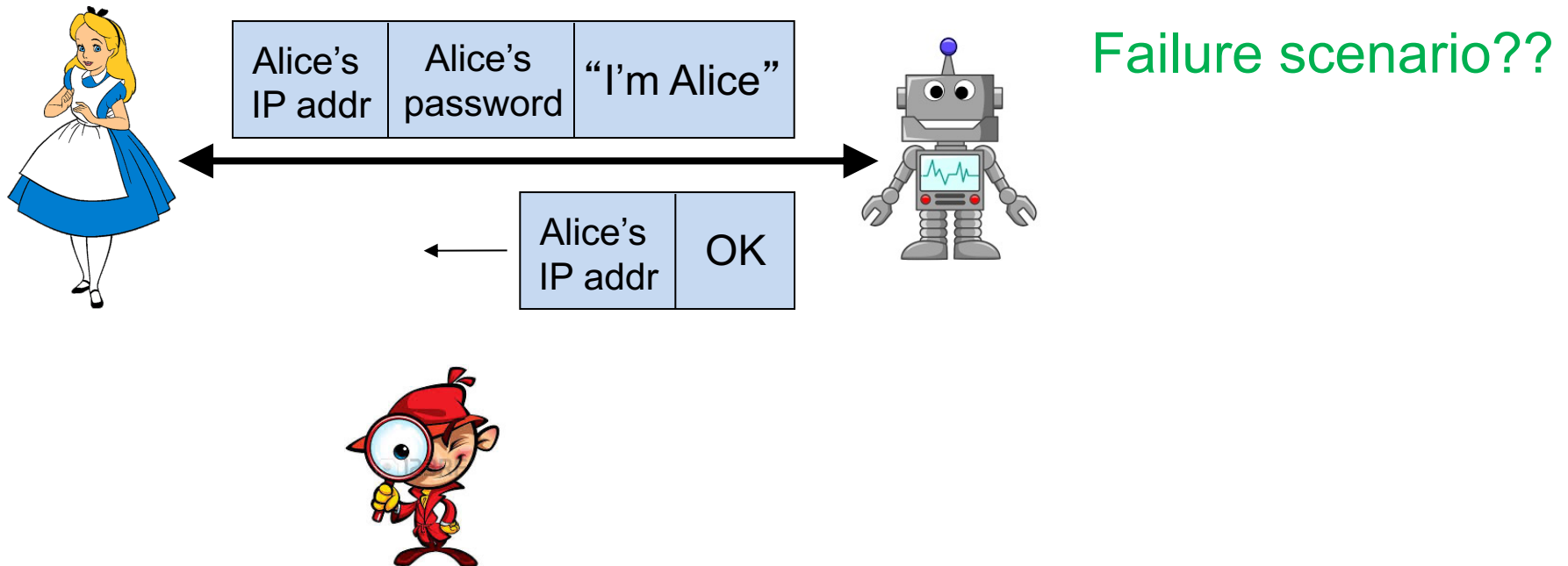
Failure scenario??
Trudy creates
pkt "spoofing"
Alice's addr

Alice's IP addr	"I am Alice"

Authentication: another try

Goal: Bob wants Alice to "prove" her identity to him

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

Alice's IP addr	Alice's password	"I'm Alice"

Alice's IP addr	OK

Failure scenario??

Authentication: another try

Goal: Bob wants Alice to "prove" her identity to him

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

Alice's IP addr	Alice's password	"I'm Alice"

Alice's IP addr	OK

Alice's IP addr	Alice's password	"I'm Alice"

Failure scenario??

Playback attack: Trudy records Alice's pkt, later plays it back to Bob

Authentication: yet another try

Goal: Bob wants Alice to "prove" her identity to him

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

Alice's IP addr	encrypted password	"I'm Alice"

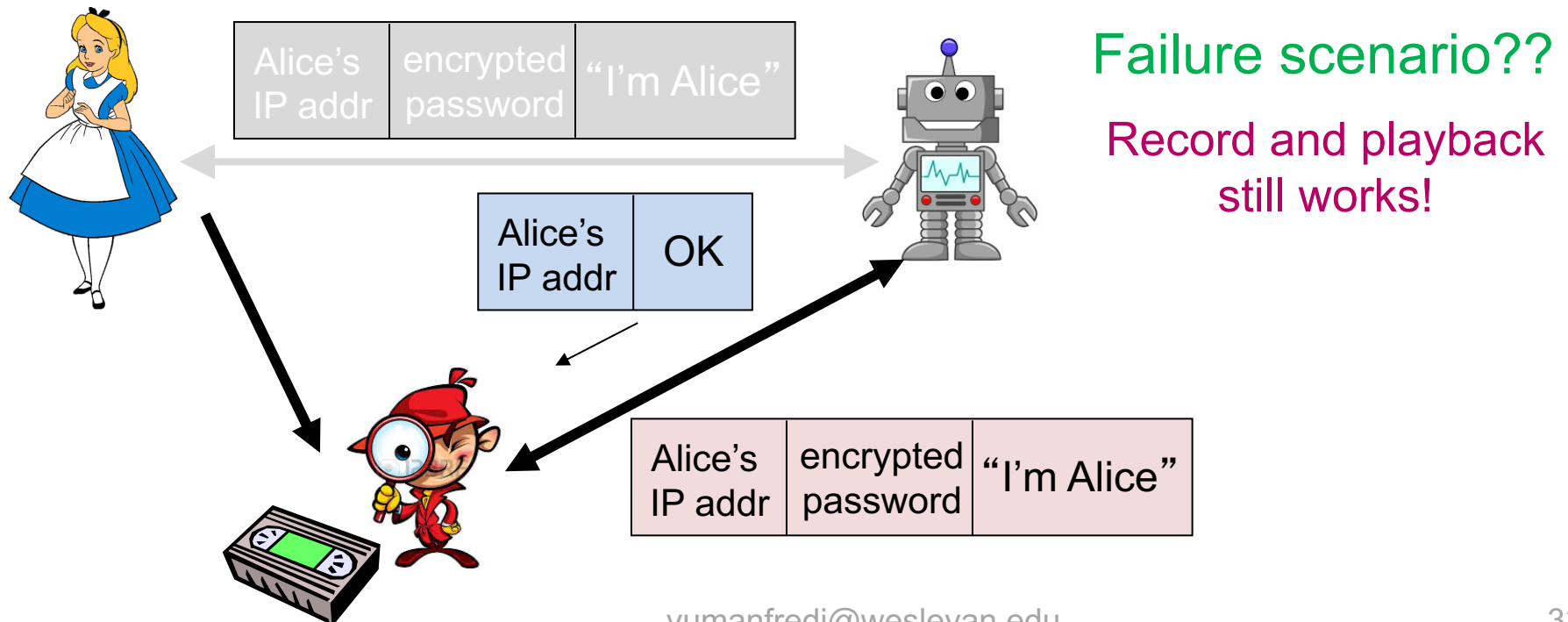| Alice's IP addr	OK

Failure scenario??

Authentication: yet another try

Goal: Bob wants Alice to "prove" her identity to him

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

Alice's IP addr	encrypted password	"I'm Alice"

Alice's IP addr	OK

Failure scenario??

Record and playback still works!

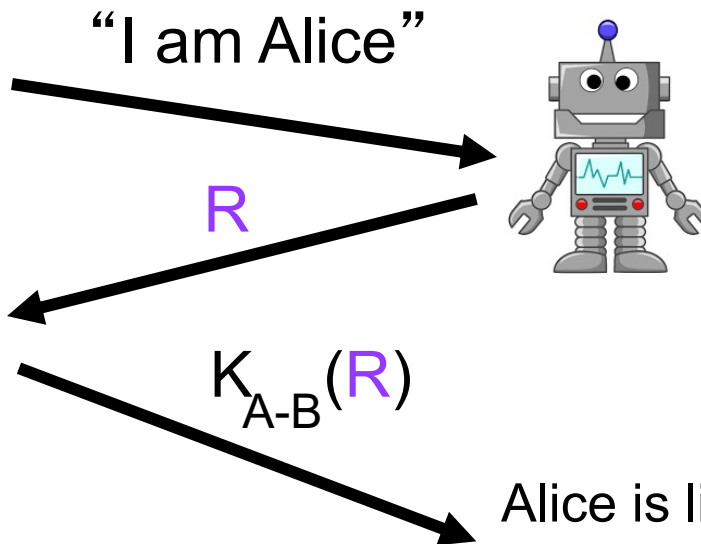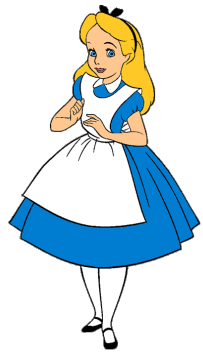| Alice's IP addr	encrypted password	"I'm Alice"

Authentication: yet another try

Goal: avoid playback attack

Nonce: number, R, used only once-in-a-lifetime

Protocol ap4.0: to prove Alice is "live", Bob sends Alice R. Alice must return R, encrypted with shared secret key

"I am Alice"

R

$K_{A-B}(R)$

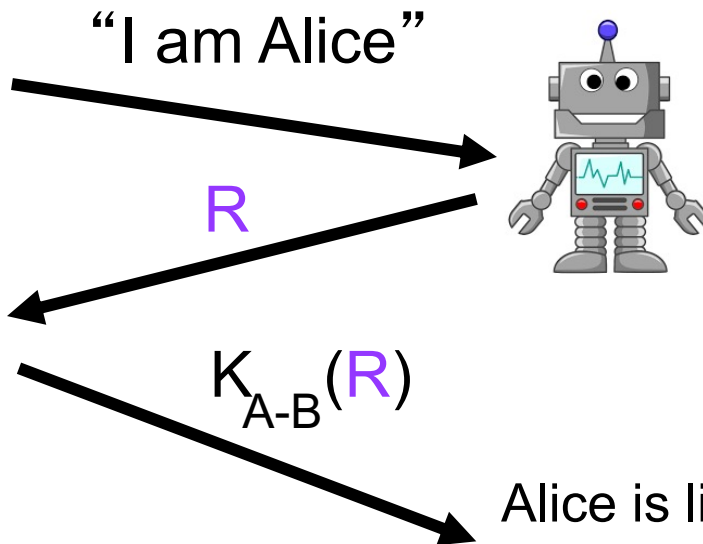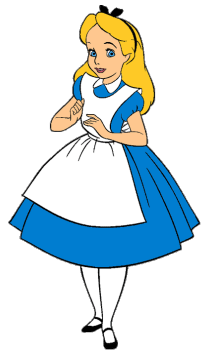Failure scenario??
Drawbacks?

Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

Authentication: yet another try

Goal: avoid playback attack

Nonce: number, R, used only once-in-a-lifetime

Protocol ap4.0: to prove Alice is "live", Bob sends Alice R. Alice must return R, encrypted with shared secret key
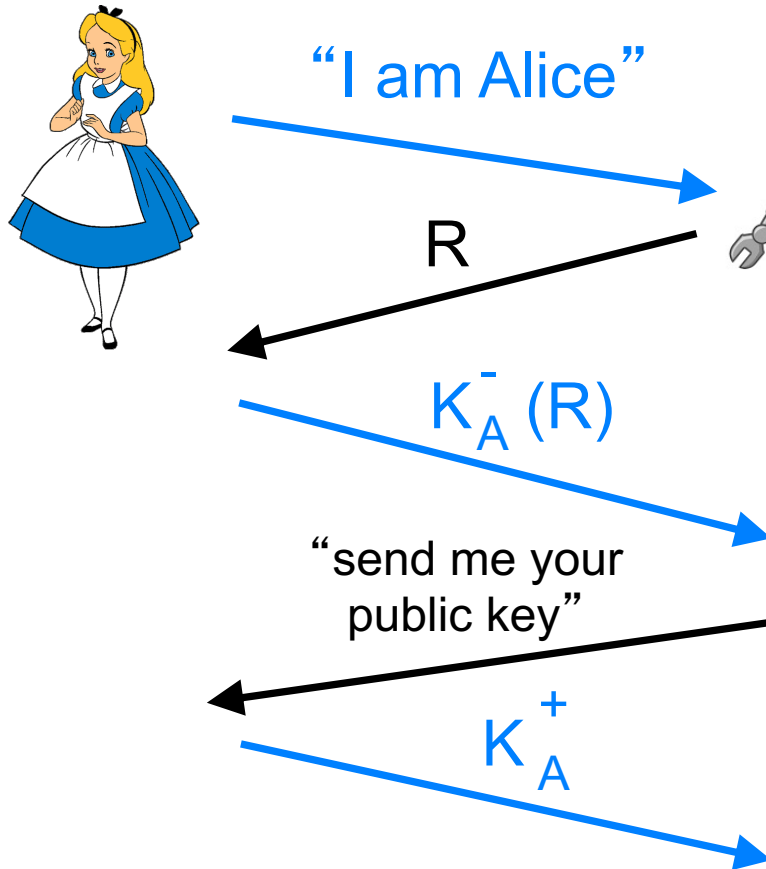

"I am Alice"

R

$K_{A-B}(R)$

Failure scenario??
Drawbacks?

Requires shared symmetric key. Can we authenticate using public key techniques?

Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

Authentication: a final try

Protocol ap5.0: use nonce and public key cryptography

"I am Alice"

R

$K_A^-(R)$
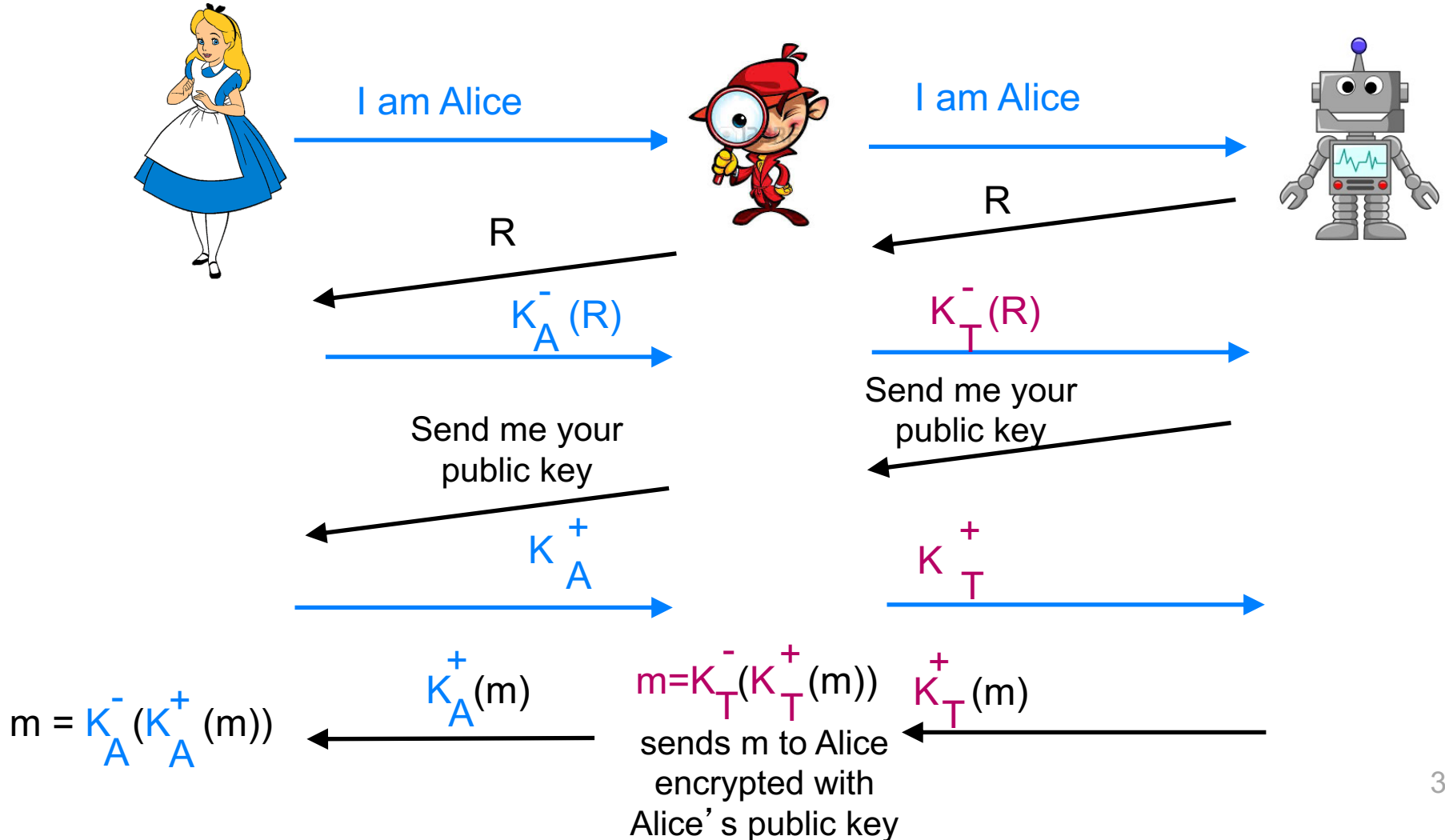
"send me your public key"

K_A^+

Bob computes

$$K_A^+(K_A^-(R)) = R$$

and knows only Alice could have the private key that encrypted R such that this holds true
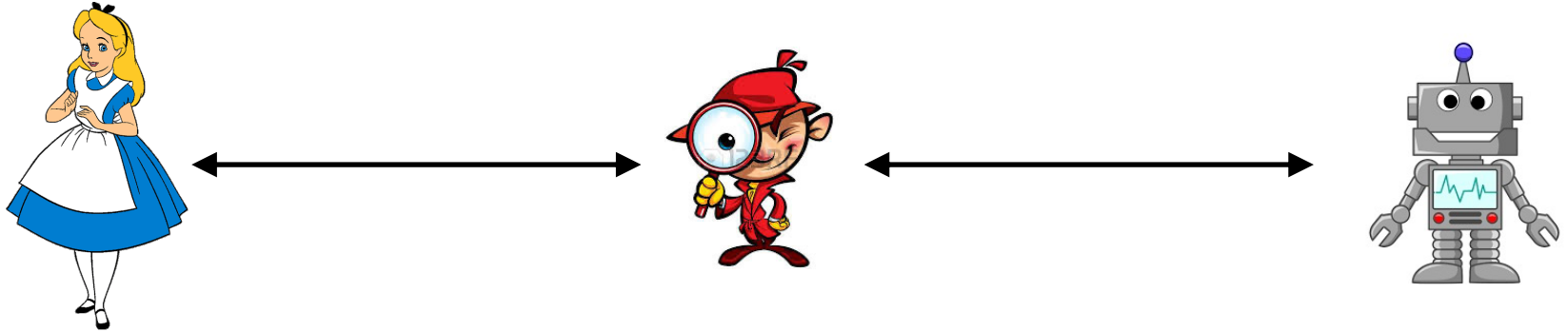
ap5.0 security hole

Man-in-the-middle Attack

– Trudy poses as Alice (to Bob) and as Bob (to Alice)

I am Alice

I am Alice

R

R

$K_A^-(R)$

$K_T^-(R)$

Send me your
public key

Send me your
public key

K_A^+

K_T^+

$K_A^+(m)$

$m = K_T^-(K_T^+(m))$

$K_T^+(m)$

$m = K_A^-(K_A^+(m))$

sends m to Alice
encrypted with
Alice's public key

ap5.0 security hole

Man-in-the-middle Attack

– Trudy poses as Alice (to Bob) and as Bob (to Alice)

Difficult to detect

– Bob receives everything that Alice sends, and vice versa
  • e.g., so Bob, Alice can meet one week later and recall conversation!
– problem is that Trudy receives all messages as well!