Announcements

Midterm exam

- **Next Thursday**, in class (2-3:20pm)
  - Bring ***your photo ID*** with you

Next Tuesday: ***No Class!!***

- I am travelling to the security area PI meeting in DC
- Please use the time to prepare for the midterm
- We will merge the office hour in Thursday to the discussion sessions on Wednesday
  - So that it's convenient for your midterm preparation

LECTURE 9:

Authentication
&
Key Distribution

[lecture slides are adapted from previous slides by Prof. Gene Tsudik]

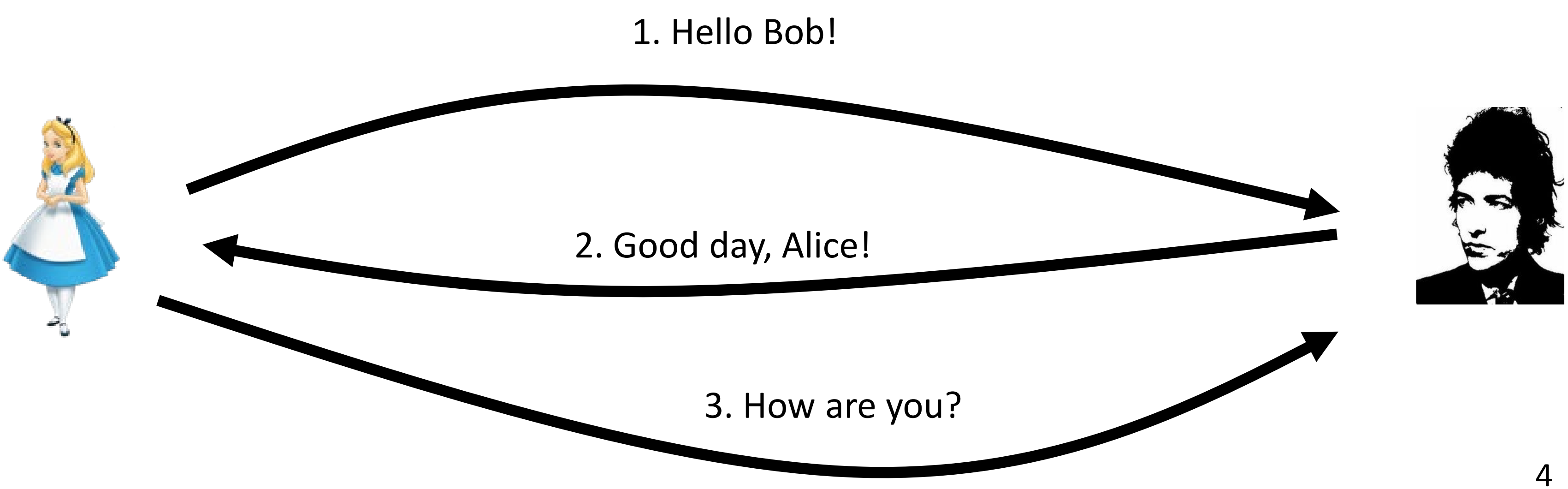
Where are we now?

- We "know" a bit of the following:
  - Conventional (symmetric) cryptography
  - Hash functions and MACs
  - Public key (asymmetric) cryptography
    - Encryption
    - Signatures
    - Identification (Fiat-Shamir) + Zero Knowledge
- And now what?
  - Protocols (more "complicated" beasts)
    - Authentication/Identification
    - Key Distribution

Secure Protocols

- A ***protocol*** is a set of rules for exchanging *messages* between 2 or more ***entities/parties***

- A protocol has a number of ***rounds*** (>1) and a number of ***messages*** (>1)



Secure Protocols

- A **message** is a unit of information/data sent from one entity/party to another as part of a protocol

- A **round** is a basic unit of protocol time:

1. *Wake up because of:*
    a) *Alarm clock*
    b) *Initial start or*
    c) *Receive message(s) from other(s)*
2. *Compute something*
3. *Send message(s) to others*
4. *Repeat steps 2-3, if needed*
5. *Wait for message(s) or sleep until alarm clock*

What’s a *Secure Protocol*?

- When acting honestly, *entities=parties*=participants achieve the stated **goal** of the protocol, e.g.,:
  - A successfully authenticates to B, or B to A
  - A and B mutually authenticate each other
  - A and B exchange a fresh session key

- Adversary can try to defeat this goal
  - e.g., by successfully impersonating A in an authentication protocol with B

The Entities (2-Party Setting)

- **Alice** and **Bob**
  - want to mutually authenticate and/or share a key
- **Eve**, the adversary
  - passive or active
- More complex protocols may involve a **Trusted Third Party (TTP)**
  - 3rd party trusted by both Alice and Bob

Definitions

- **Entity** Authentication:
  - corroboration that an entity is the one claimed

Entity Authentication has two types:

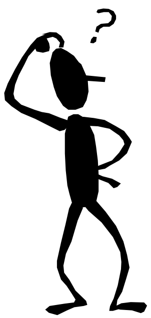
- **Unilateral** Authentication:
  - entity authentication: providing one entity with assurance of the other's identity, but not vice versa

- **Mutual** Authentication:
  - entity authentication which provides both entities with assurance of each other's identity

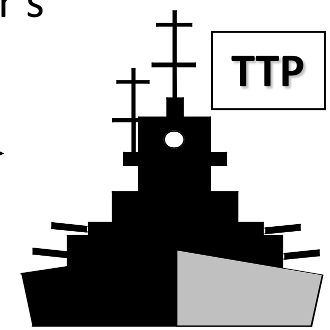
Purpose

Examples:

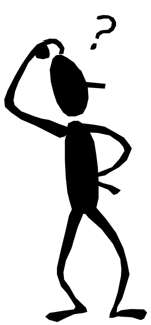
- Bank transactions, e.g., cash withdrawals
- Remote login
- File access
- P2P transaction



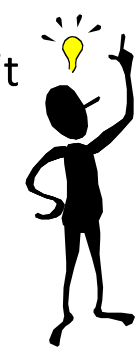
Has user's secrets



Send secret or prove knowing it?



Doesn't



Peer
Or
Server

Basis for Authentication

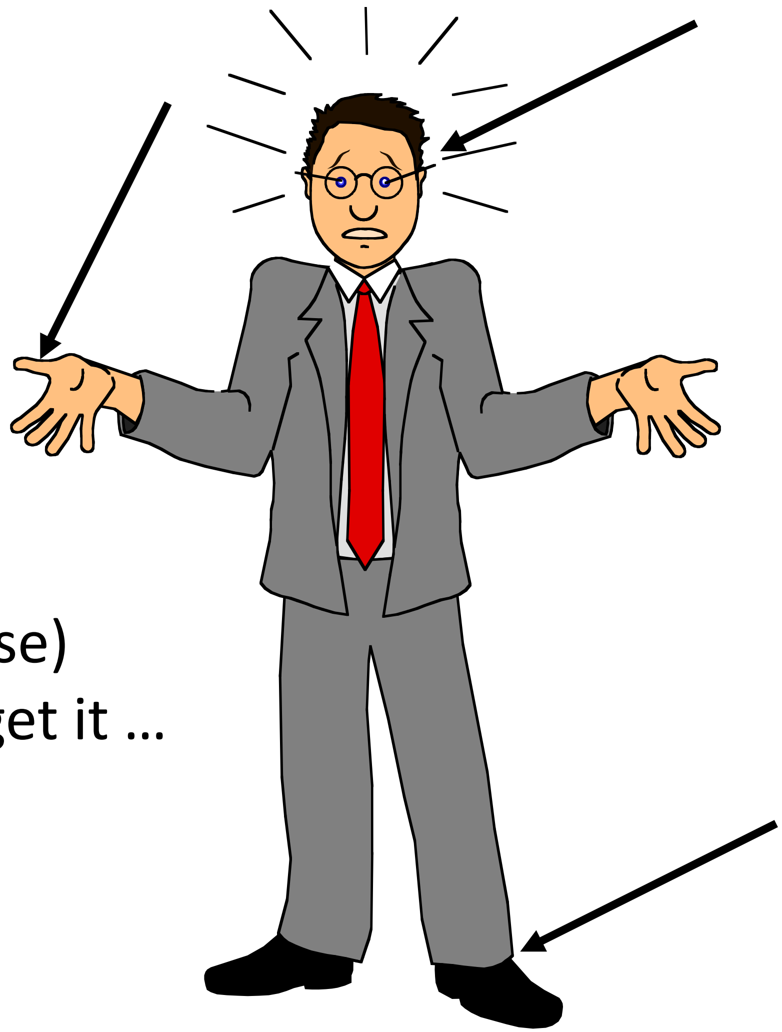
- Something you **know**, such as PIN or password
- Something you **have:**
  - A secure token, e.g., that generates a one-time password
  - Key embedded in a “secure area” on a computer, in browser software, etc.
  - A smartcard, which may contain keys and can perform cryptographic operations on behalf of a user
- Something you **are** (a biometric)

Concrete Scenarios

- PIN-, PW-, Biometric-based schemes
  - Kerberos
  - SecureID tokens
  - Iris/retina scanners
  - Thumbprint & hand/palmprint
  - Handwriting acceleration & pressure
- Public Key Identification Schemes:
  - Fiat-Shamir, etc.
- Authentication protocols
  - Conventional- and public key-based (covered later)

Human Failings

- Humans are notoriously unreliable
- Human memory is very volatile storage



- What a human can remember:
    - PIN (no more than 6-8 digits)
    - Password (a word or a short phrase)
- Can a human do single-digit sums? Forget it …

Biometrics

- Accuracy:
  - False Acceptance Rate (False Positive)
  - False Rejection Rate (False Negative)

- Retinal scanner, fingerprint reader, handprint reader, voiceprint, keystroke timing, signature (shape or pressure), etc.

Fingerprints

- Vulnerability:
  - Dummy fingers and dead fingers
  - Lost fingers
- Suitability and stability:
  - Not for people with high probability of damaged fingerprints (e.g., eczema)
  - Not for kids who are still growing
  - Other noise sources: thermal and optical noise, temperature affecting skin condition …

Voice Recognition

- Single fixed phrase:
  - Can use tape recorder to fake
- Stability:
  - Background noise
  - Colds, vocal cord damage/strain, laughing gas ☺
  - Use with public phones

Keystroke Timing

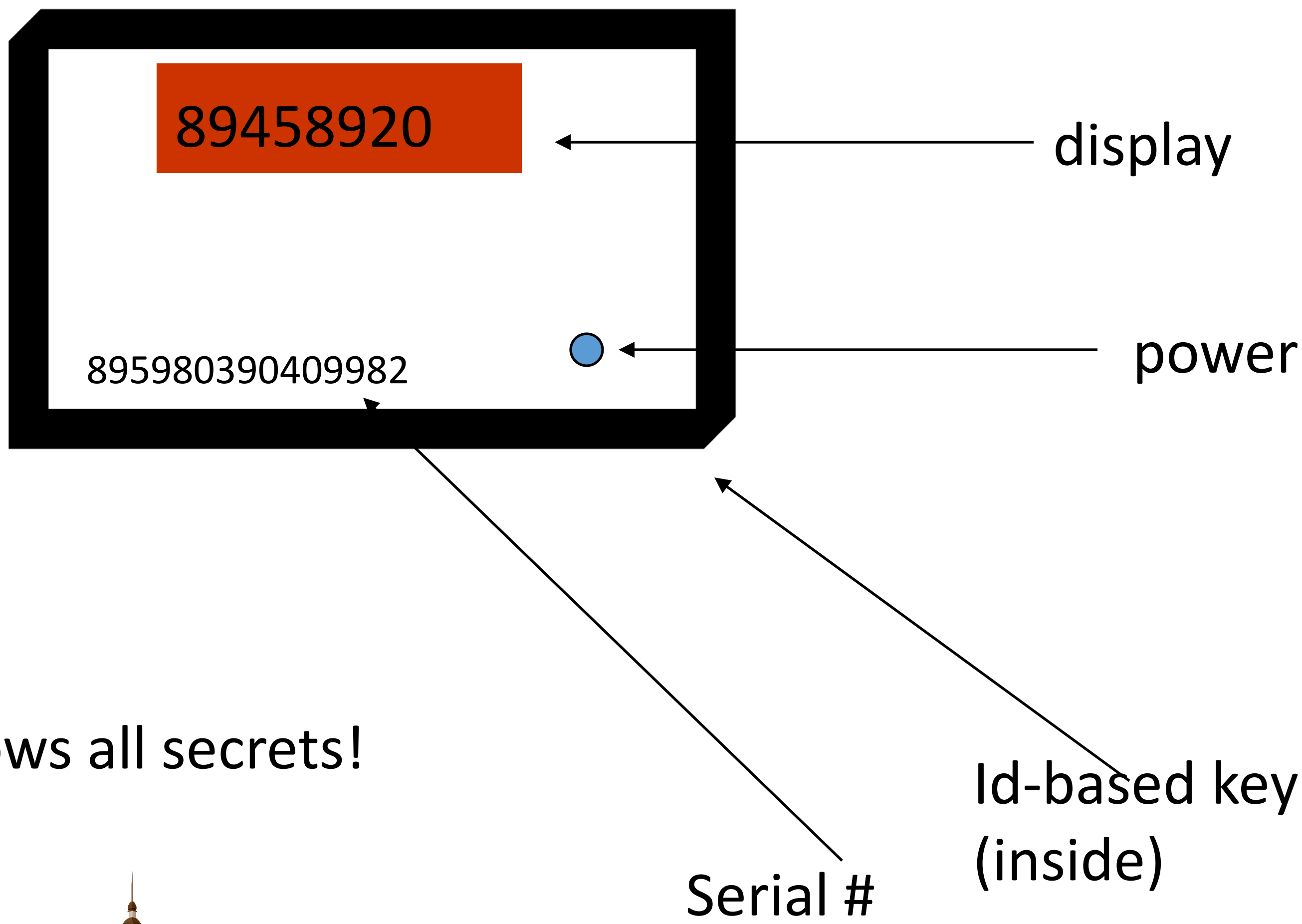
- Each person has a distinct typing timing and style
  - Hand/finger movements
- Suitability:
  - Best done for “local” authentication
    - Avoid network traffic delay

(Non-digital) Signatures

- Machines can not (yet) match human experts in recognizing shapes of signatures

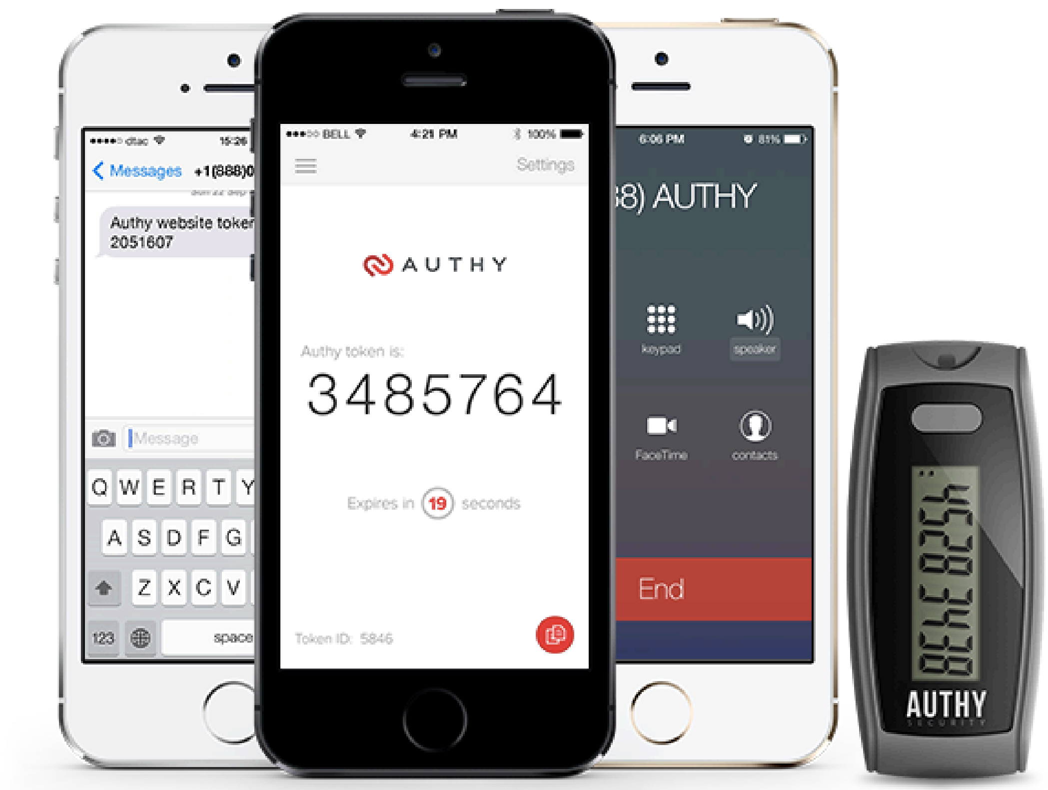
- Add information on acceleration and/or pressure
  - Signing on a special electronic tablet

SecureID/Secure-Token



TTP/Server:
secure & knows all secrets!

SecureID/Secure Token

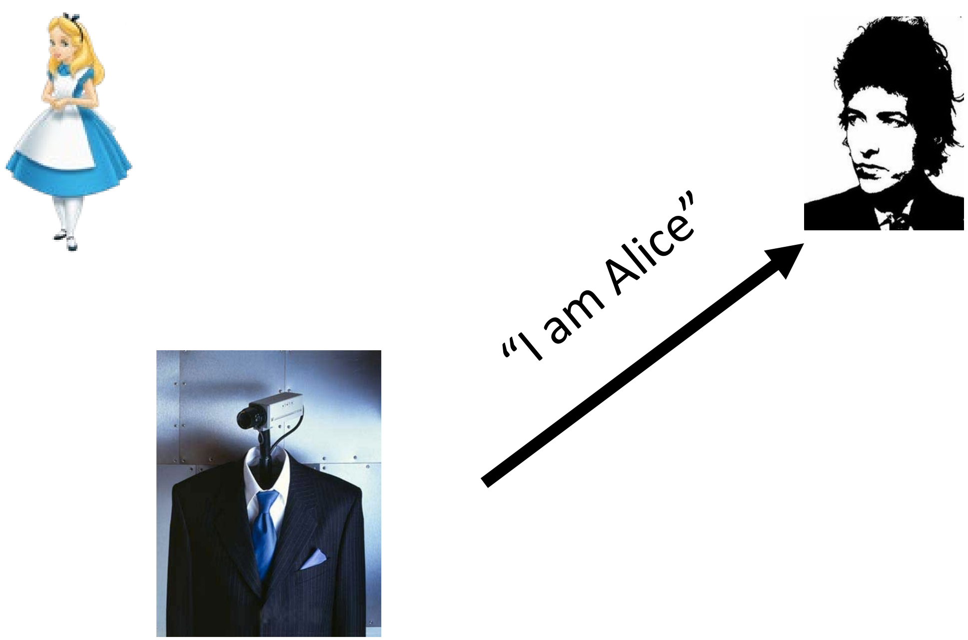


TTP/Server:
secure & knows all secrets!

Authentication (Protocols)

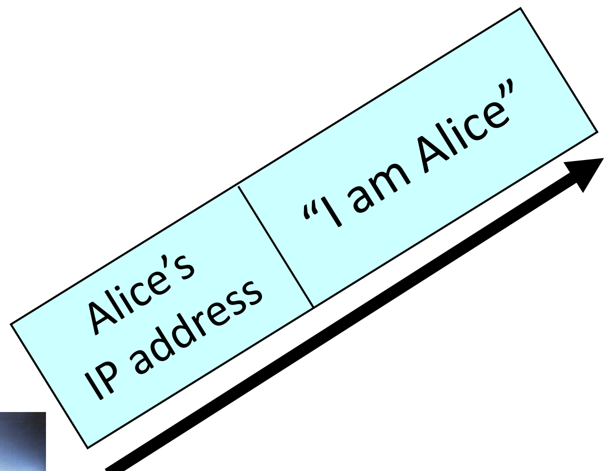
Since they communicate over a network, Bob cannot "see" Alice.
So, Eve simply declares herself to be Alice

<u>Protocol ap1.0:</u> Alice says "I am Alice"



Authentication: Another Try

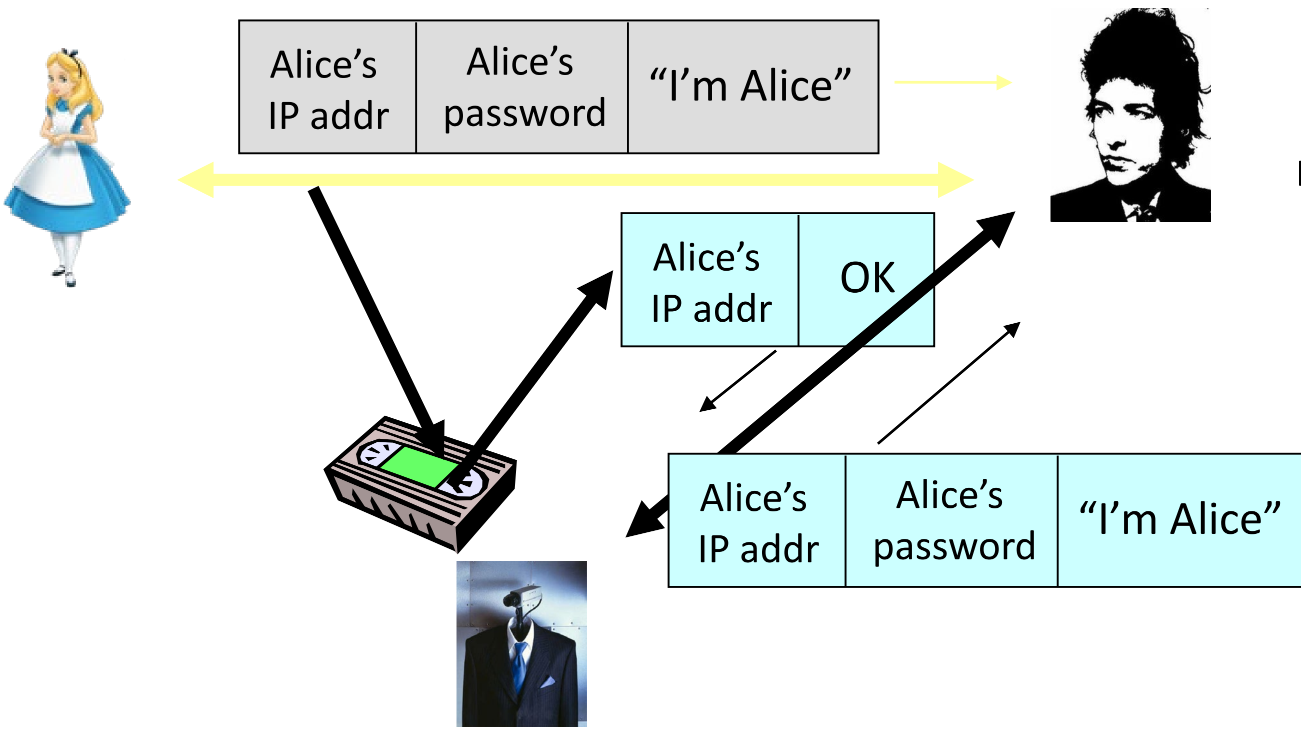
Protocol ap2.0: Alice says “I am Alice” in an IP packet containing her source IP address



Eve can create
a packet “spoofing”
Alice’s address

Authentication: Another Try

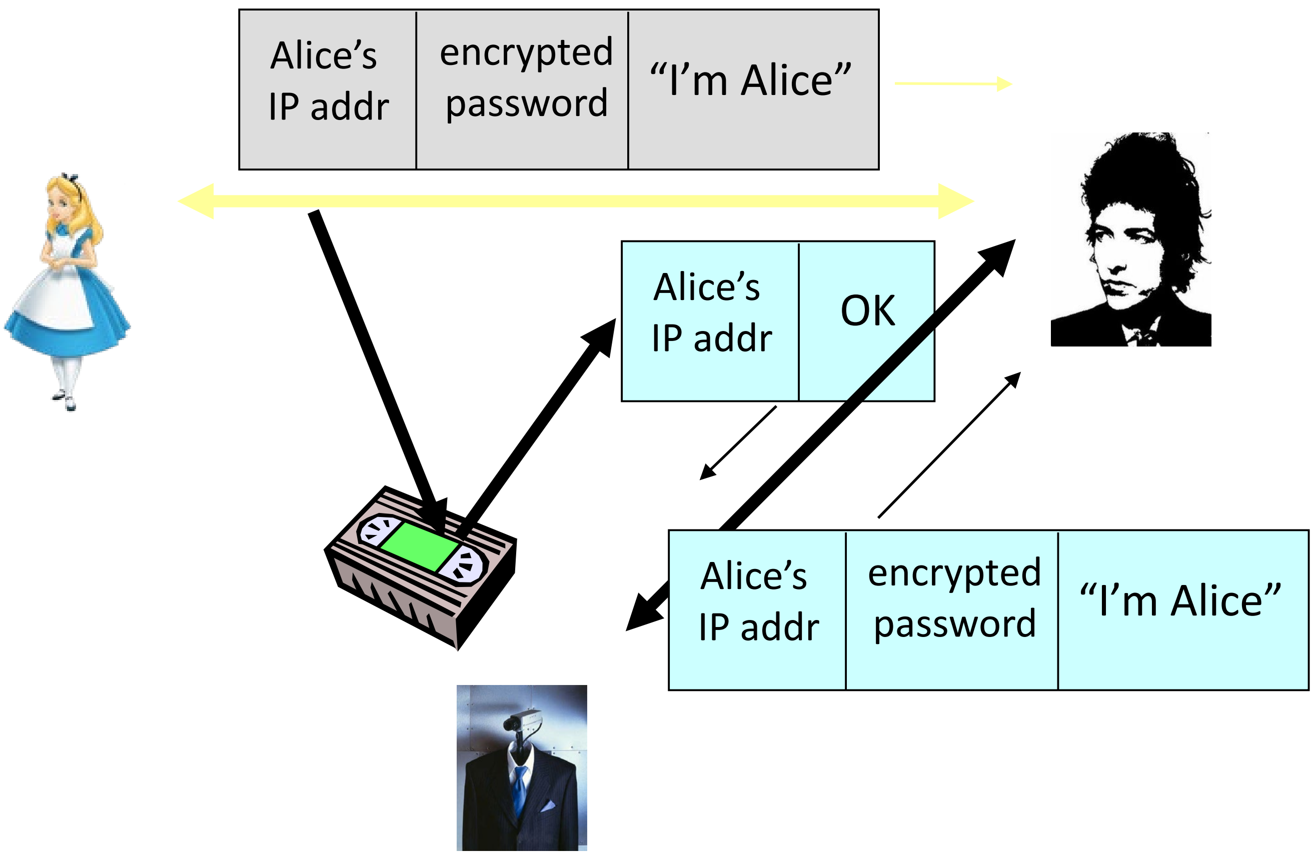
<u>Protocol ap3.0:</u> Alice says "I am Alice" and sends her secret password to "prove" it.



playback attack: Eve records Alice's packet and later plays it back to Bob

Authentication: Another Try

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



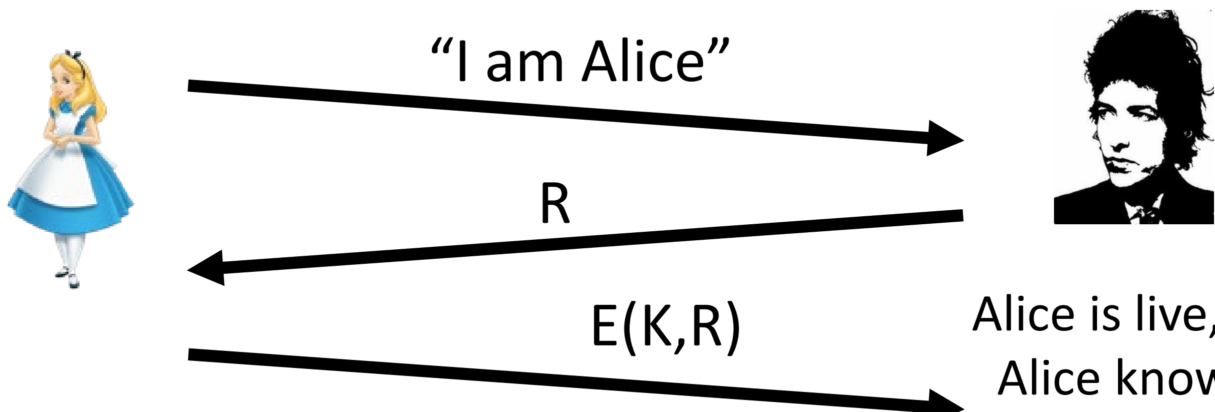
record and playback still works!

Authentication: Yet Another Try

<u>Goal:</u> avoid playback attack

<u>Nonce:</u> number used *once (R)*

<u>ap4.0:</u> to prove Alice "live", Bob sends Alice nonce, R. Alice must return R, encrypted with shared secret key



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

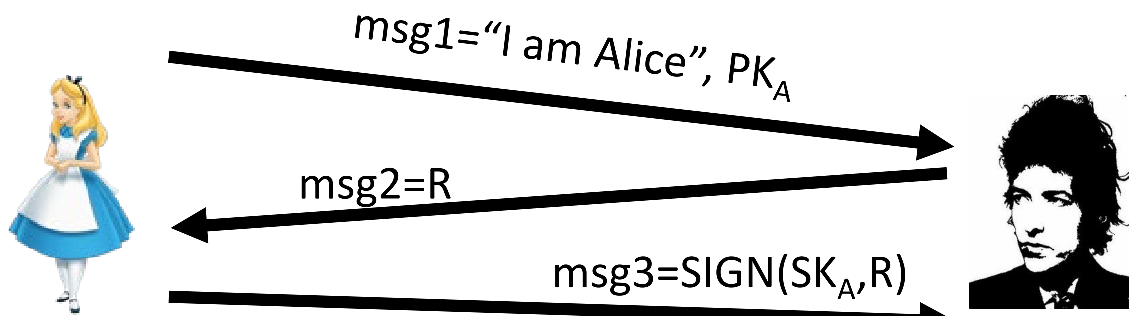
- K may be derived from Alice's password …
- This protocol works if Bob never authenticates to Alice using K

Authentication: ap5.0

ap4.0 requires shared symmetric key

- can we authenticate using public key?

<u>ap5.0:</u> nonces and public key cryptography



Using PK_A, Bob verifies Alice's signature of R in msg3. Since R is fresh and only Alice can compute signatures using SK_A, Bob concludes that Alice is really there.

The Protocol (Nonces)

1.	A → B:	"Hi Bob, it's, me, Alice"	
2.	B → A:	R	(challenge)
3.	A → B:	E(K, R||B)	(response)

Why not simply send E(K,R) in last message?

The Protocol (what if?)

1. B → A (Eve): “Hi Alice, it’s me Bob”

1.Eve → B: ”Hi Bob, it’s, me, Alice“

2.B → A (Eve): R (challenge)

2. Eve → B: R

3. B → Eve: E(K,R)

3. Eve → B: E(K,R) (response)

The Protocol (Nonces)

1. A → B: "Hi Bob, it's, me, Alice"

2. B → A: R

3. A → B: E(Kab,R) or
   E(K, R||B)

- Kab is only used in A→B direction and a different key (Kba) is used in B→A direction
- Alternatively, can use the same K in both directions but include explicit direction identifier in msg

The Protocol (Sequence #s)

1. A → B: "Hi Bob, it's, me, Alice"

2. B → A: S_b (challenge)

increment S_b

1. A → B: $E(K, S_b || B)$ (response)

- No PRNG needed
- Both A and B must remember S_b
- What if S_b wraps around?

Time-Stamps

Including a date/time-stamp in message allows recipient to check for freshness (as long as time-stamp is protected by cryptographic means).

1. A → B: $E(K, TIME_A \,||\, B)$

This results in fewer protocol messages

But requires synchronized clocks…
(Similar to the SecureID scenario)

Key Distribution and Management

- Conventional (Secret) key distribution
- Public key distribution

Trusted Intermediaries

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 intermediary between entities

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)

Key Distribution Center (KDC)

- Responsible for distributing keys to pairs of users (hosts, processes, applications)

- Each user must share a unique master key with the KDC
  - Use this key to communicate with KDC to get a temporary *session* key for establishing a secure "session" with another user/program/host/entity
  - Each master key is distributed (agreed upon) in some off-line fashion (in person, by snail-mail, etc.)

Key Distribution Center (KDC) aka Trusted Third Part (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