# Uses for Public Key Crypto

# Uses for Public Key Crypto

- Confidentiality
  - Transmitting data over insecure channel
  - Secure storage on insecure media
- Digital signature
  - Provides integrity and non-repudiation
  - No non-repudiation with symmetric keys

# Non-non-repudiation

- Alice orders 100 shares of stock from Bob
- Alice computes MAC using symmetric key
- Stock drops, Alice claims she did *not* order
- Can Bob prove that Alice placed the order?
- No! Bob also knows the symmetric key, so he could have forged the MAC
- Problem: Bob knows Alice placed the order, but he can't prove it

# Non-repudiation

- Alice orders 100 shares of stock from Bob
- Alice **signs** order with her private key
- Stock drops, Alice claims she did not order
- Can Bob prove that Alice placed the order?
- **Yes!** Alice's private key used to sign the order — only Alice knows her private key

# Public Key Notation

- Sign message $\mathrm{M}$ with Alice's private key: $[\mathrm{M}]_{\mathrm{Alice}}$
- Decrypt message $\mathrm{C}$ with Alice's private key: $[\mathrm{C}]_{\mathrm{Alice}}$
- Encrypt message $\mathrm{M}$ with Alice's public key: $\{\mathrm{M}\}_{\mathrm{Alice}}$
- Then

$$[\{\mathrm{M}\}_{\mathrm{Alice}}]_{\mathrm{Alice}} = \mathrm{M}, \quad \{[\mathrm{M}]_{\mathrm{Alice}}\}_{\mathrm{Alice}} = \mathrm{M}$$

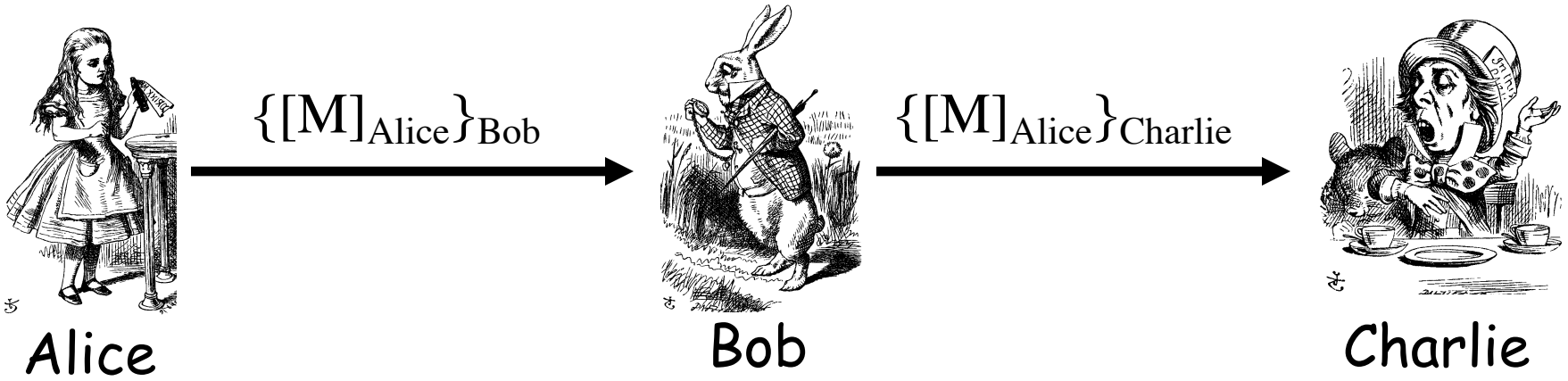
# Sign and Encrypt
# vs
# Encrypt and Sign

# Confidentiality and Non-repudiation?

- Suppose that we want confidentiality and integrity/non-repudiation
- Can public key crypto achieve both?
- Alice sends message to Bob
  - Sign and encrypt: $\{[M]_{Alice}\}_{Bob}$
  - Encrypt and sign: $[\{M\}_{Bob}]_{Alice}$
- Can the order possibly matter?

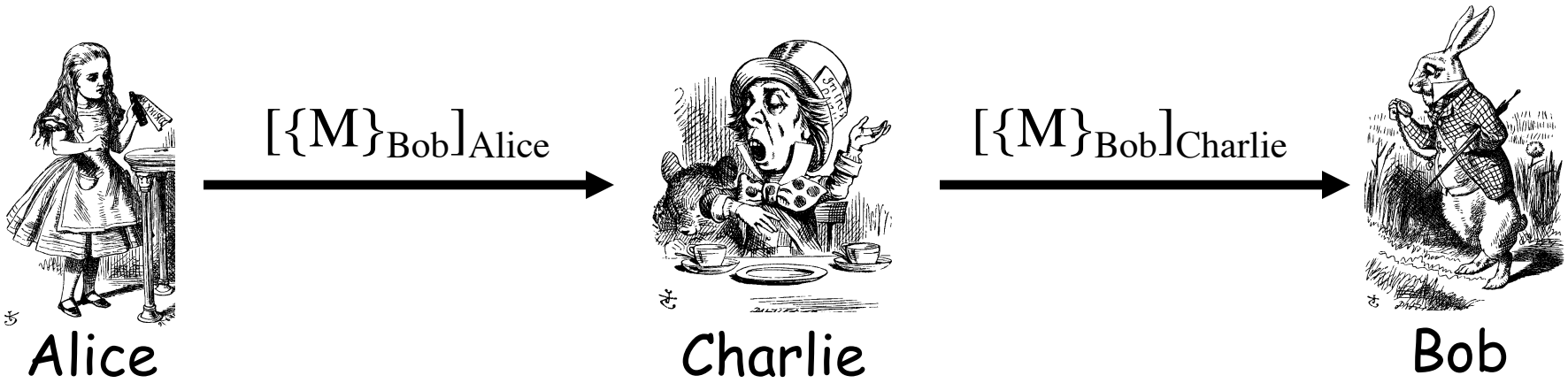
# Sign and Encrypt

- M = "I love you"



# Encrypt and Sign

- M = "My theory, which is mine…."




- Note that Charlie cannot decrypt M

# Public Key Infrastructure

# Public Key Certificate

- Digital **certificate** contains name of user and user's public key (possibly other info too)
- It is ***signed*** by the issuer, a ***Certificate Authority*** (*CA*), such as VeriSign

  $M = (\text{Alice}, \text{Alice's public key}),\ S = [M]_{CA}$

  **Alice's Certificate** $= (M, S)$
- Signature on certificate is verified using *CA*'s public key

  Must verify that $M = \{S\}_{CA}$

# Certificate Authority

- Certificate authority (CA) is a trusted 3rd party (TTP) — creates and signs certificates
- Verify signature to verify **integrity** & identity of **owner of corresponding private key**
  - Does **not** verify the identity of the **sender** of certificate — certificates are public!
- Big problem if CA makes a mistake
  - CA once issued Microsoft cert. to someone else

# PKI

- Public Key Infrastructure (PKI): the stuff needed to securely use public key crypto
  - Key generation and management
  - Certificate authority (CA) or authorities
  - Certificate revocation lists (CRLs), etc.
- No general standard for PKI
- We mention 3 generic "trust models"
  - We only discuss the CA (or CAs)

# PKI Trust Models

- Monopoly model
  - One universally trusted organization is the *CA* for the known universe
  - Big problems if *CA* is ever compromised
  - Who will act as *CA* ???
    - System is useless if you don't trust the *CA*!

# PKI Trust Models

- Oligarchy
  - Multiple (as in, "a few") trusted CAs
  - This approach is used in browsers today
  - Browser may have 80 or more CA certificates, just to verify certificates!
  - User can decide which CA or CAs to trust

# PKI Trust Models

- Anarchy model
  - Everyone is a CA…
  - Users must decide who to trust
  - Places a significant burden on users.
- Why is it anarchy?
  - Suppose certificate is signed by Frank and you don't know Frank, but you do trust Bob and Bob says Alice is trustworthy and Alice vouches for Frank. Should you accept the certificate?

# Confidentiality in the Real World

# Symmetric Key vs Public Key

- Symmetric key
  - **Speed**
  - No public key infrastructure (PKI) needed (but have to generate/distribute keys)
- Public Key
  - **Signatures** (non-repudiation)
  - No ***shared*** secret (but, do have to get private keys to the right user…)

# Notation Reminder

- Public key notation
  - Sign M with Alice's private key

    $[M]_{Alice}$
  - Encrypt M with Alice's public key

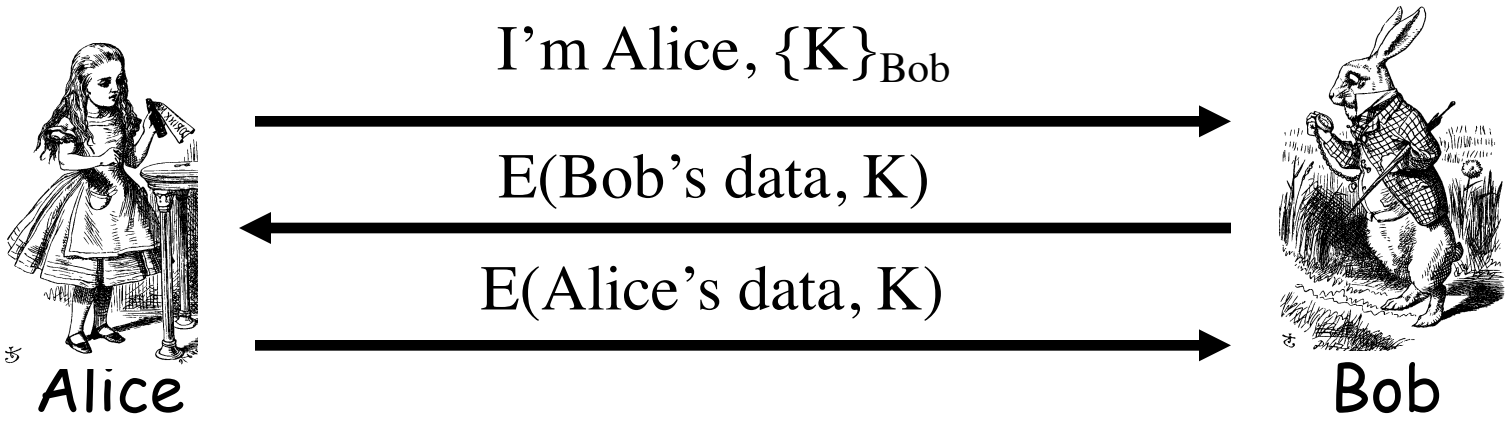
    $\{M\}_{Alice}$
- Symmetric key notation
  - Encrypt P with symmetric key K

    $C = E(P,K)$
  - Decrypt C with symmetric key K

    $P = D(C,K)$

# Real World Confidentiality

- **Hybrid cryptosystem**
  - Public key crypto to establish a key
  - Symmetric key crypto to encrypt data…



# Midterm Topics

- Python
  - Grammas
  - Modules
  - Functions, Lamda Functions, List Comprehensions
  - Data structure
    - List, tuple, set, dictionary
  - Closure, class
  - Build-ins

# Midterm Topics

- Database
  - SQL
- Security
  - Basic concepts
  - Symmetric key crypto
  - Public key crypto