Last time

- Internet Application Security and Privacy
  - Basics of cryptography
  - Symmetric-key encryption

www.eff.org
THE ELECTRONIC FRONTIER FOUNDATION
Protecting Rights in the Electronic Frontier
Cracking DES

This time

- Internet Application Security and Privacy
  - Public-key encryption
  - Integrity

Key exchange

- The hard part of symmetric ciphers is:
  - How do Alice and Bob share the secret key?
    - Meet in person; diplomatic courier
  - In general this is very hard
- Or, we invent new technology...

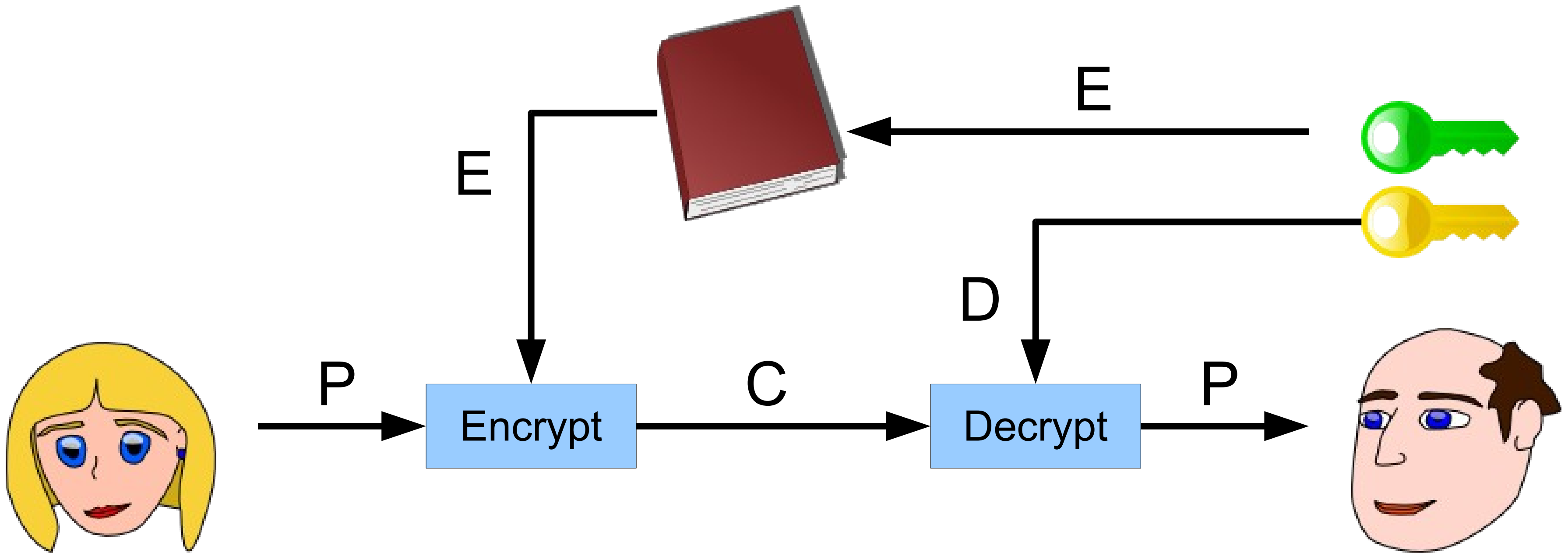
Public-key cryptography

- Invented (in public) in the 1970's.
  - Allows Alice to send a secret message to Bob without any prearranged shared secret!
  - In symmetric crypto, the same key “locks” the message as “unlocks” it.
  - In asymmetric (or “public-key”) crypto, there's one key for locking, and a different key for unlocking!
- Some common examples:
  - RSA, ElGamal, ECC

Public-key cryptography

- How does it work?
  - Bob gives everyone a copy of his public locking key. Alice uses it to lock (**encrypt**) a message, and sends the locked message to Bob.
  - Bob uses his private unlocking key to unlock (**decrypt**) the message.
    - Eve can't unlock it; she only has the locking key.
    - Neither can Alice!
- So with this, Alice just needs to know Bob's public key in order to send him secret messages
  - These public keys can be published in a directory somewhere

Public-key cryptography



Public Key Sizes

- Recall that if there are no shortcuts, Eve would have to try 2^{128} things in order to read a message encrypted with a 128-bit key.
- Unfortunately, all of the public-key methods we know do have shortcuts
  - Eve could read a message encrypted with a 128-bit RSA key with just 2^{33} work, which is easy!
  - If we want Eve to have to do 2^{128} work, we need to use a much longer public key.

Public Key Sizes

Comparison of key sizes for roughly equal strength

AES	RSA
80	1024
116	2048
128	2600
160	4500
256	14000

Hybrid Cryptography

- In addition to having longer keys, public-key crypto takes a long time to calculate (as compared to symmetric-key crypto)
  - Using public-key to encrypt large messages would be too slow, so we take a hybrid approach:
    - Pick a random 128-bit key for a symmetric-key cryptosystem
    - Encrypt the large message with that symmetric key (AES)
    - Encrypt the 128-bit key with a public-key cryptosystem
    - Send the symmetric-encrypted message and the public-encrypted key to Bob.
  - This hybrid approach is used for almost every cryptography application on the Internet today.

Is that all there is?

- It seems we've got this “sending secret messages” thing down pat. What else is there to do?
  - Even if we're safe from Eve reading our messages, there's still the matter of Mallory.
  - It turns out that even if our messages are encrypted, Mallory can sometimes modify them in transit!
  - Mallory won't necessarily know what the message says, but can still change it in an undetectable way.
    - e.g. bit-flipping attack on stream ciphers
  - This is counterintuitive, and often forgotten
    - The textbook even gets this wrong!
- How do we make sure that Bob gets the same message Alice sent?

Integrity components

- How do we tell if a message has changed in transit?
- Simplest answer: use a checksum.
  - For example, add up all the bytes of a message
  - The last digits of serial numbers (credit card, ISBN, etc.) are usually checksums.
  - Alice computes the checksum of the message, and sticks it at the end before encrypting it to Bob. When Bob receives the message and checksum, he verifies that the checksum is correct.

This doesn't work!

- With most checksum methods, Mallory can easily change the message in such a way that the checksum stays the same.

- We need a “cryptographic” checksum
- It should be hard for Mallory to find a second message with the same checksum as any given one.

Cryptographic Hash Functions

- These cryptographic checksums are called hash functions.
  - Common examples: MD5, SHA-1, SHA-256
- Hash functions generally have two properties:
  - One-way:
    - Given a hash value, it's hard to find a message which hashes to that value (a “preimage”).
  - Collision-resistant:
    - It's hard to find two messages which hash to the same value (a “collision”).

What is “hard”?

- For SHA-1, for example, it takes 2^{160} work to find a preimage, and 2^{80} work to find a collision.
  - Well, that's what we thought until last year.
  - It turns out finding collisions in SHA-1 may be easier than we thought.
- The difference is due to the well-known birthday paradox.

Cryptographic Hash Functions

- Hash functions are only useful when there is a secure way of sending the hash value.
  - For example, Bob can publish a hash of his public key on his business card.
  - Putting the whole key on there would be too big.
  - But Alice can download Bob's key from the Internet, hash it herself, and verify that the hash matches the one on Bob's card.

Cryptographic Hash Functions

- You can't just send an unencrypted message and hash to get integrity assurance
  - Even if you don't care about secrecy!
- Mallory can just change the message, and just compute the new hash value himself.

Recap

- Internet Application Security and Privacy
  - Public-key encryption
  - Integrity

Next time

- Internet Application Security and Privacy
  - Authentication
  - Security controls using cryptography
  - Link-layer security: WEP, WPA, WPA2