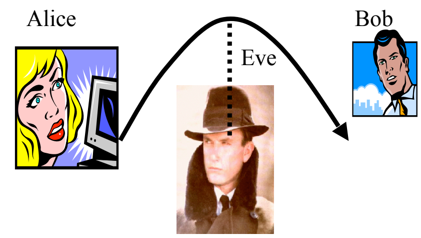
Cryptography



- Alice wants to send a secret message to Bob
  - but has no safe communication channel.
  - How to make sure that even if Eve intercepts the message, she will not be able to understand it?
- Applications:
  - Military (since before the Ancient Egypt)
  - eCommerce

Secret-key encryption

- Idea:
  - Alice uses a *secret* algorithm to encrypt her message *M* into an encrypted message *encr(M)*.
  - Bob knows the algorithm used by Alice for the encryption. Upon receiving *encr(M),* he can invert the process and recover *M*.
  - Eve does not know the encryption algorithm used by Alice, so it is difficult for her to decrypt *encr(M)*.
- Example: Caesar's cypher
  - Shift each letter by a fixed amount
  - Example: "Rome" → "Bywo" (shift = 10)
  - Problem: Too easy to break! How?

Better secret-key encryption

- Substitution cypher: map each letter to some other letter

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
↓ ↓
C N G R I V K L O Z J M U A Q X S P T B Y E H D F W

- Frequency attack: Given a long encrypted English text:
  - The most common letters probably correspond to E, T, 0...
  - Most common pairs of letters: TH, ER, IN...
  - Guess the rest
- Solution: Change the permutation frequently
  - Problem: It takes a big code book to remember all the permutations.
  - Enigma encryption machine

Problems with secret-key schemes

- Alice and Bob need to share knowledge of a key that nobody else knows
- If Alice can't meet Bob personally and they can't communicate on a safe channel, how can they agree on a key?
- Impossible? No! Diffie-Hellman key exchange protocol (1976).

Public-key cryptography

- Idea: Alice and Bob don't need to agree on a key
  - Bob has a public key e that is visible to everyone. People who want to send messages to Bob will use e to encrypt their message.
  - Bob also has a secret key d that nobody else knows (not even Alice)
  - To encrypt her message, Alice uses Bob's public key e to encrypt her message M into $encr(M)$. She doesn't need to know d.
  - Eve can intercept $encr(M)$, but without the knowledge of d, decrypting $encr(M)$ is very difficult
  - Bob can easily decrypt $encr(M)$ because he knows the secret key d

RSA cryptography system

- Rivest-Shamir-Adleman (1978)
- Bob chooses two large primes p and d chooses $e = p \cdot q$ to be the public key
- Define $\phi = (p-1)(q-1)$
- Bob chooses a private key d such that $3d \bmod \phi = 1$. (There are efficient algorithms to do so).
- Alice encodes her message as an integer M. She encrypts M as $encr(M) = M^3 \bmod e$
- Bob decrypts $encr(M)$ as follows:

$$\begin{aligned} decr(M) &= encr(M)^d \bmod e \\ &= (M^3 \bmod e)^d \bmod e \\ &= M^{3d} \bmod e \\ &= M \text{ (because of Fermat's Little theorem and Chinese Remainder Theorem)} \end{aligned}$$

- Nobody knows efficient algorithms to decrypt encr(M) without knowing the factors p and q of the public key e

RSA - Example

- Bob chooses primes $p = 17$, $q = 23$.
- $e = 17 * 23 = 391$ is Bob's public key
- $\phi = (17-1) * (23-1) = 352$
- Bob chooses his private key d so that $3d \bmod \phi = 1$. For example, $d = 235$.
- Suppose Alice wants to send M=24. She encrypts it as $encr(M) = M^3 \bmod e = 24^3 \bmod 391 = 139$
- If Eve sees $encr(M) = 139$, she can't easily recover $M = 24$ because she doesn't know p, q, or d.
- Upon receiving $encr(M) = 139$, Bob decrypts it as: $encr(M)^d \bmod e = 139^{235} \bmod 391 = 24$

Wow! That's a BIG number!

Fast modular exponentiation

How can I compute 139^{235} mod 391 ?

Not by computing 139^{235} and then taking mod 391!

(a * b) mod n = ((a mod n) * (b mod n)) mod n

Decompose 235 as a sum of powers of 2:

235 = 128 + 64 + 32 + 8 + 2 + 1

139^{235} mod 391 =

= (139^{128} * 139^{64} * 139^{32} * 139^{8} * 139^{2} * 139) mod 391

= [(139^{128} mod 391) * (139^{64} mod 391) * (139^{32} mod 391) * (139^{8} mod 391) * (139^{2} mod 391) * (139^{1} mod 391)] mod 391

Use a variant of the Power method you wrote:

139^{1} mod 391 = 139,

139^{2} mod 391 = (139*139) mod 391 = 162

139^{4} mod 391 = (162*162) mod 391 = 47

139^{8} mod 391 = (47*47) mod 391 = 254

139^{16} mod 391 = (254*254) mod 391 = 1

139^{32} mod 391 = (1*1) mod 391 = 1

139^{64} mod 391 = (1*1) mod 391 = 1

139^{128} mod 391 = (1*1) mod 391 = 1

Thus,

139^{235} mod 391 = (1 * 1 * 1 * 254 * 162 * 139) mod 391 = 24

RSA - Summary

- RSA relies completely on the fact that it is difficult to factorize large integers
- If Eve could factorize Bob's public key *e* into *p* * *q*, she could compute *□ = (p-1) (q-1),* and then find *d,* from which she could easily decrypt *encr(M)*.
- Nobody knows a polynomial-time algorithm to factorize large integers, so the message is safe
- Quantum computers can factorize large integers very quickly, but we don't know how to build them.