Asymmetric / Public Key Crypto

Introduction to Basic Cryptography

Dr. Ryan Riley

جامعة قطر
QATAR UNIVERSITY

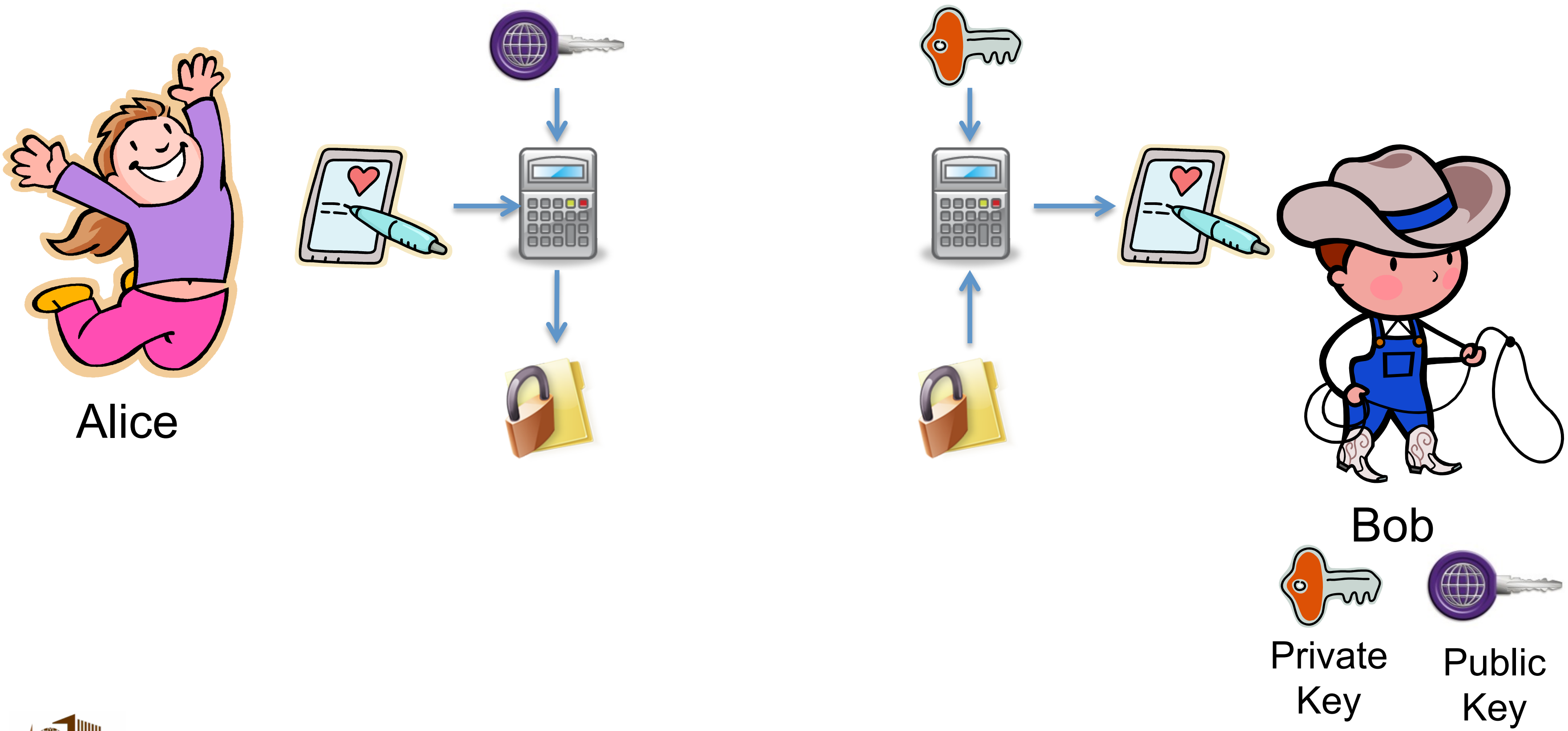
Introduction

- Public key crypto allows you encrypt with one key and have someone else decrypt the message *with a different key*
- This has two uses:
  - Confidentiality
    - Send secret messages to someone
  - Integrity:
    - Ensure something wasn't modified
    - Prove who created it

جامعة قطر
QATAR UNIVERSITY

Recall…

- A cryptographic technique where both parties in the communication use *different* keys



Public and Private Keys?

- Mathematically related keys that allow you to encrypt with one and decrypt with the other
  - Similar to the mathematics used in the Diffie-Hellman key exchange
- Every user has two keys: A public key and a private key
  - Public key: Not a secret. Anyone can have it
  - Private key: Secret. Only the owner can have it

Asymmetric Encryption

- Encryption with the public key
  - $C = E_{PUB\text{-}Alice}(M)$
  - $M = D_{PRIV\text{-}Alice}(C)$

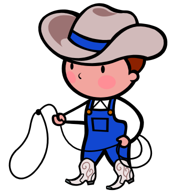
- Encryption with the private key
  - $C = E_{PRIV\text{-}Alice}(M)$
  - $M = D_{PUB\text{-}Alice}(C)$

- Other encryption/decryption pairs *don't work*

Public Key Crypto for Confidentiality

Alice

CT



Bob

- If Alice wants to send a message, M, to Bob…
  - She computes $CT = E_{PUB\text{-}Bob}(M)$ and sends it to Bob
  - Bob decrypts it by calculating $M = D_{PRIV\text{-}Bob}(C)$
- Who can perform the decryption?
  - Only Bob, with his private key
- Who can perform the encryption?
  - Anyone, because Bob's public key is public

Public Key Crypto for Confidentiality

- What if Bob wants to reply to Alice?
  - He should encrypt the message with *Alice's* public key
  - (Same way Alice sends a message to Bob)

Problem #1

- Public key cryptography is very slow
- Decryption speeds
  - AES-128: 100 MB/s
  - RSA-1024: 1 MB/s
- Using this for big files would be horrible

Problem #1 Solution

- We combine symmetric and asymmetric tools
- If Alice wants to send Bob a message, she…
  - Chooses a random symmetric key, k
  - Computes $CT=E_k (M)$ and sends it to Bob
  - Computes $CT_2 = E_{PUB\text{-}Bob}(k)$ and sends it to Bob
- Bob uses his private key to decrypt CT_2 into k and then uses k to decrypt CT and get the message
- Most cryptography on the internet is based, in part, on this concept

Problem #2

- Let's add Mallory, a malicious attacker who can *intercept and modify* messages
- Alice computes CT = $E_{PUB\text{-}Bob}(M)$ and sends it to Bob
  - Mallory intercepts it, throws it away
- Mallory computes $CT_{evil} = E_{PUB\text{-}Bob}(M_{evil})$ and sends it to Bob
  - Bob decrypts it, can't tell that it isn't from Alice

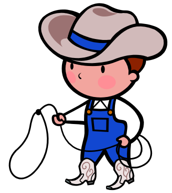
Problem Explained

- Our current technique provides confidentiality, but not integrity
  - Mallory couldn't read the message from Alice
  - Mallory replaced the message and Bob didn't know
- Solution?

Public Key Crypto for Integrity

DS

Alice



Bob

- If Alice wants to send a message, M, to Bob that proves it is from her
  - She computes DS = $E_{PRIV\text{-}Alice}(M)$ and sends it to Bob
  - Bob decrypts it by calculating $M = D_{PUB\text{-}Alice}(DS)$
- Who can perform the encryption?
  - Only Alice, with her private key
- Who can perform the decryption?
  - Anyone, because Alice's public key is public

Public Key Crypto for Integrity

- Bob knows the message is from Alice because *only Alice could have produced it*
- Notice this doesn't guarantee confidentiality
- We call this a *digital signature*
  - Alice is simply signing the message to prove it is from her

Speed Problem for Integrity

- What if you want to sign a large file?
  - This would be too slow
- Instead, sign a hash of the file

جامعة قطر
QATAR UNIVERSITY

RSA

- The first public key cryptosystem
- Invented by Rivest, Shamir, and Adleman
- Any bit size is ok
  - 512 was standard when it was released
  - 2048 or 4096 is standard now
- Based on prime numbers and factoring
  - The public key is the product of two primes
  - The private key is those two primes

RSA: Security

- How secure is this?
  - If factoring large numbers is easy, RSA is easy to break
  - If factoring large numbers is hard, RSA is hard to break
  - Right now we think factoring large numbers is hard, *but we can't prove it*
- A bruteforce attack is basically trying to factor the public key into two prime numbers

Note on Bit Size

- In symmetric key crypto, the *key size* is given in bits:
  - AES-128 means AES with a 128-bit key
  - 128-bits measures the keyspace (number of possible keys)
- In RSA asymmetric key crypto, the *prime number size* is given in bits:
  - RSA-2048 means RSA is using 2048-bit prime numbers to create the public and private keys
- Comparisons between symmetric and asymmetric security cannot be done based just on bit size

Summing Up

- Public key crypto involves two keys that are mathematically related
- Encrypting with one key requires decrypting with the other
- You need to be careful to make sure you know whether you are providing confidentiality, integrity, or both