CSE 484 / CSE M 584: Asymmetric Cryptography

Fall 2023

Franziska (Franzi) Roesner
franzi@cs

UW Instruction Team: David Kohlbrenner, Yoshi Kohno, Franziska Roesner. Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Announcements

- Things due
  - Lab 1: Wednesday
  - Homework 2: Next Friday
    - Individual assignment (no groups)
  - CSE 584M: Don’t forget about weekly research readings
- Roadmap
  - Monday & Wednesday: Asymmetric crypto
  - Friday: Crypto in practice (on the web)
  - Next week: Web security

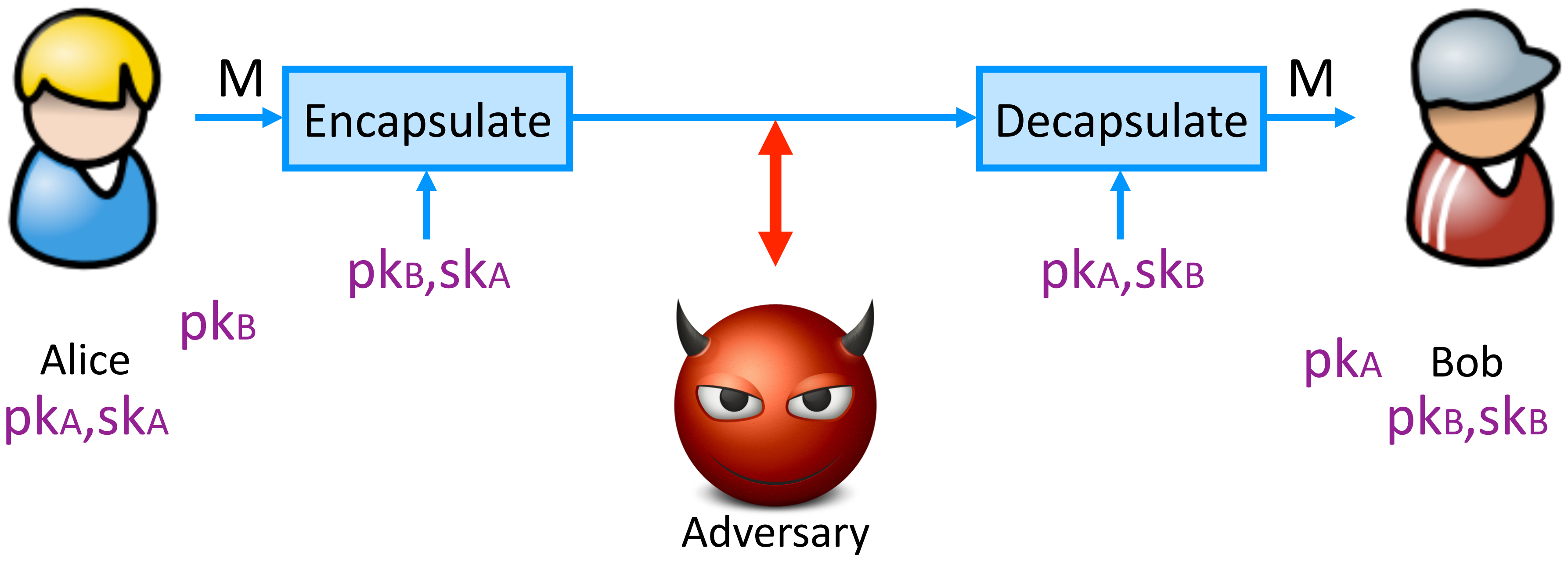
Flavors of Cryptography

- Symmetric cryptography
  - Both communicating parties have access to a shared random string K, called the key.

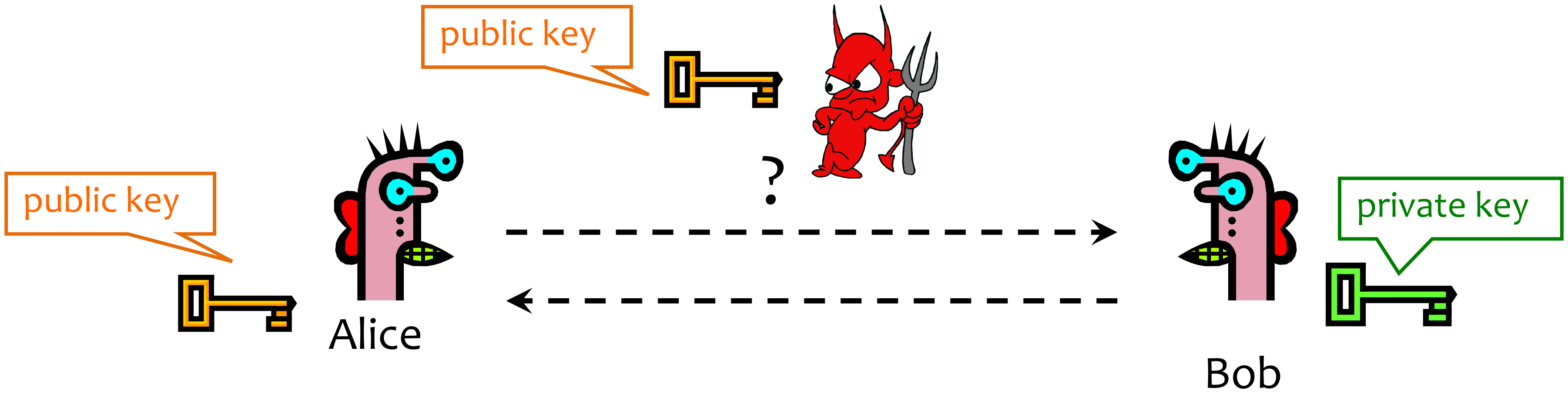
- Asymmetric cryptography
  - Each party creates a public key pk and a secret key sk.

Asymmetric Setting

Each party creates a public key pk and a secret key sk.



Public Key Crypto: Basic Problem



Ignore for now: How do we know it's REALLY Bob's??

Given: Everybody knows Bob's public key
Only Bob knows the corresponding private key

Goals: 1. Alice wants to send a secret message to Bob
2. Bob wants to authenticate themself

Applications of Public Key Crypto

- Encryption for confidentiality
  - Anyone can encrypt a message
    - With symmetric crypto, must know secret key to encrypt
  - Only someone who knows private key can decrypt
  - Key management is simpler (or at least different)
    - Secret is stored only at one site: good for open environments
- Digital signatures for authentication
  - Can "sign" a message with your private key
- Session key establishment
  - Exchange messages to create a secret session key
  - Then switch to symmetric cryptography (why?)

Session Key Establishment

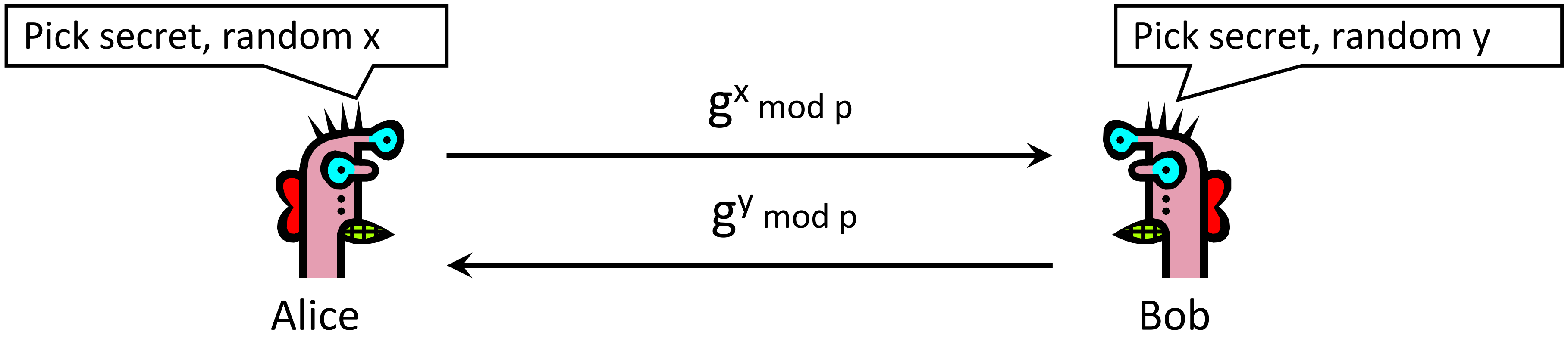
Modular Arithmetic

- Given g and prime p, compute: g^1 mod p, g^2 mod p, … g^{100} mod p
  - For p=11, g=10
    - 10^1 mod 11 = 10, 10^2 mod 11 = 1, 10^3 mod 11 = 10, …
    - Produces cyclic group {10, 1} (order=2)
  - For p=11, g=7
    - 7^1 mod 11 = 7, 7^2 mod 11 = 5, 7^3 mod 11 = 2, …
    - Produces cyclic group {7,5,2,3,10,4,6,9,8,1} (order = 10)
      - Numbers "wrap around" after they reach p
    - g=7 is a "generator" of Z_{11}^*

Diffie-Hellman Protocol (1976)

Diffie-Hellman Protocol (1976)

- Alice and Bob never met and share no secrets
- <u>Public</u> info: p and g
  - p is a large prime, g is a **generator** of Z_p^*
    - $Z_p^* = \{1, 2 \dots p-1\}$; a is in Z_p^* if there is an i such that $a = g^i \bmod p$



Compute $k = (g^y)^x = g^{xy} \bmod p$

Compute $k = (g^x)^y = g^{xy} \bmod p$

Example Diffie Hellman Computation

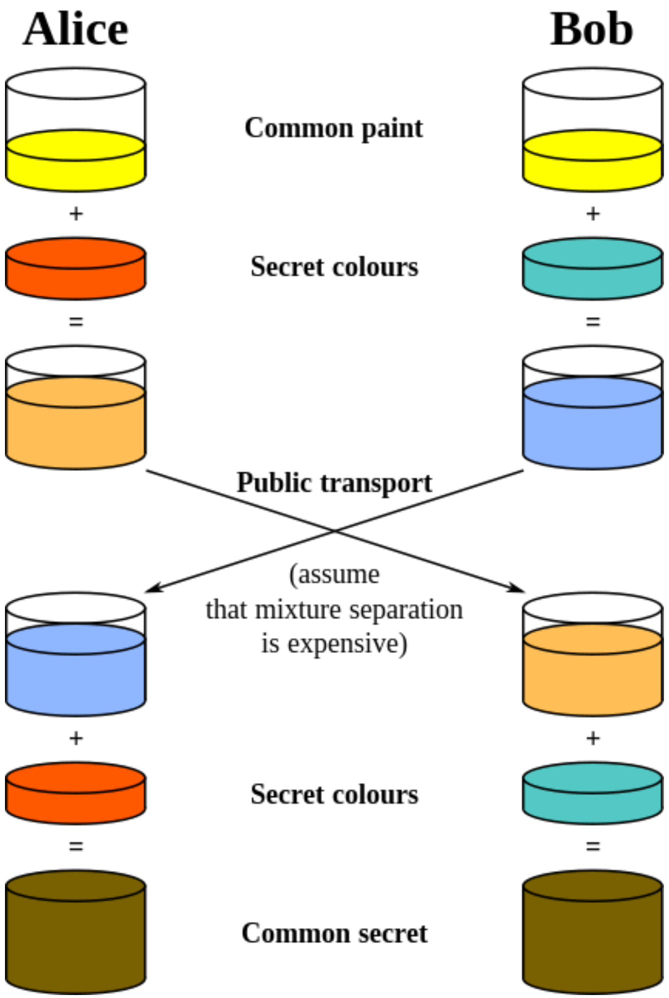
- PUBLIC
  - p = 11
  - g = 2
  - (g is a generator for group mod p)

- Alice: x=9, sends 6 (g^x mod p = 2^9 mod 11 = 6)
- Bob: y=4, send 5 (g^y mod p = 2^4 mod 11 = 5)

- A compute: 5^x mod 11 (5^9 mod 11 = 9)
- B compute 6^y mod 11 (6^4 mod 11 = 9)
- Both get 9

- All computations modulo 11

Diffie-Hellman: Conceptually



Common paint: p and g

Secret colors: x and y

Send over public transport:

g^x mod p

g^y mod p

Common secret: g^{xy} mod p

[from Wikipedia]

Why is Diffie-Hellman Secure?

- Discrete Logarithm (DL) problem:
  given g^x *mod p*, it's hard to extract x
  - There is no known <u>efficient</u> algorithm for doing this
  - This is <u>not</u> enough for Diffie-Hellman to be secure!
- Computational Diffie-Hellman (CDH) problem:
  given g^x and g^y, it's hard to compute g^{xy} *mod p*
  - … unless you know x or y, in which case it's easy
- Decisional Diffie-Hellman (DDH) problem:
  given g^x and g^y, it's hard to tell the difference between
  g^{xy} *mod p* and g^r *mod p* where r is random

Diffie-Hellman Caveats (1)

- Assuming DDH problem is hard (depends on choice of parameters!), Diffie-Hellman protocol is a secure key establishment protocol against passive attackers
  - Common recommendation:
    - Choose p=2q+1, where q is also a large prime
    - Choose g that generates a subgroup of order q in Z_p*
    - DDH is hard in this group
  - Eavesdropper can't tell the difference between the established key and a random value
  - In practice, often hash g^{xy} *mod p*, and use the hash as the key
  - Can use the new key for symmetric cryptography

Diffie-Hellman Caveats (2)

- Diffie-Hellman protocol (by itself) does not provide authentication (against <u>active</u> attackers)
  - Person in the middle attack (aka "man in the middle attack")

Diffie-Hellman Key Exchange Today

- Important Note:
  - We have discussed discrete logs modulo integers
  - Significant advantages in using **elliptic curve groups**
    - Groups with some similar mathematical properties (i.e., are “groups”) but have better security and performance (size) properties
    - Today’s de-facto standard

Stepping Back: Asymmetric Crypto

- We've just seen session key establishment
  - Can then use shared key for symmetric crypto
- Next: public key encryption
  - For confidentiality
- Then: digital signatures
  - For authenticity

Requirements for Public Key Encryption

- Key generation: computationally easy to generate a pair (public key PK, private key SK)
- Encryption: given plaintext M and public key PK, easy to compute ciphertext $C=E_{PK}(M)$
- Decryption: given ciphertext $C=E_{PK}(M)$ and private key SK, easy to compute plaintext M
  - Infeasible to learn anything about M from C without SK
  - Trapdoor function: Decrypt(SK,Encrypt(PK,M))=M

Some Number Theory Facts

- Euler totient function $\varphi(n)$ $(n \geq 1)$ is the number of integers in the $[1,n]$ interval that are relatively prime to n
  - Two numbers are relatively prime if their greatest common divisor (gcd) is 1
  - Easy to compute for primes: $\varphi(p) = p-1$
  - Note that $\varphi(ab) = \varphi(a)\,\varphi(b)$ if a & b are relatively prime

RSA Cryptosystem [Rivest, Shamir, Adleman 1977]

- Key generation:
  - Generate large primes p, q
    - Say, 2048 bits each (need primality testing, too)
  - Compute **n**=pq and φ(**n**)=(p-1)(q-1)
  - Choose small **e**, relatively prime to $\varphi(n)$
    - Typically, **e=3** or **e=2^{16}+1=65537**
  - Compute unique **d** such that ed $\equiv$ 1 mod $\varphi(n)$
    - Modular inverse: d $\equiv$ e^{-1} mod $\varphi(n)$
  - Public key = (e,n); private key = (d,n)
- Encryption of m: c = m^e mod n
- Decryption of c: c^d mod n = $(m^e)^d$ mod n = m

How to compute?

- Extended Euclidian algorithm
- Wolfram Alpha ☺
- Brute force for small values