CSE 484 / CSE M 584: Computer Security and Privacy

Cryptography

Autumn 2018

Tadayoshi (Yoshi) Kohno
yoshi@cs.Washington.edu

Thanks to Dan Boneh, Dieter Gollmann, Dan Halperin, Ada Lerner, John Manferdelli, John Mitchell, Franziska Roesner, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Admin

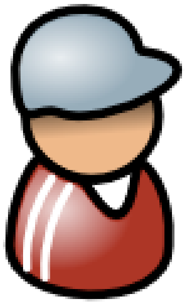
- Lab 1:
  - Due Oct 24, 4:30pm (Today!)

- TA Office Hours (especially for Lab 1): M 2:30, W 1:30, F 12
- My office hours (especially for crypto, research readings, administrivia, worksheet pick up): M 11:30

Challenge Question from Last Time

- Alice and Bob are both cryptographers, and they are talking on the phone. They want to randomly flip a coin. If they were together, in person, they would flip a real coin and see if it was Heads or Tails. But they are not together, in person, and they don't trust each other enough to have one of them flip a coin and tell the other person the answer.

- Using the techniques we've discussed so far in class, how can Alice and Bob effectively flip a random coin together, over the phone, such that they both trust the answer even though they don't trust each other?



Pick bit bA at random

C1 = Locked box with bA

Pick bit bB at random

C2 = Locked box with bB

Info to open first box C1

Now knows bA

Info to open second box C2

Now knows bB

Both compute random bit at bA xor bB

|| denotes concatenation

Pick bit bA at random

Pick RA as long random string

$C_1 = H(bA \,||\, RA)$ →

Pick bit bB at random

Pick RB as long random string

← $C_2 = H(bB \,||\, RB)$

Send bA || RA →

Verify that has of message equals C_1

← Send bB || RB

Verify that has of message equals C_2

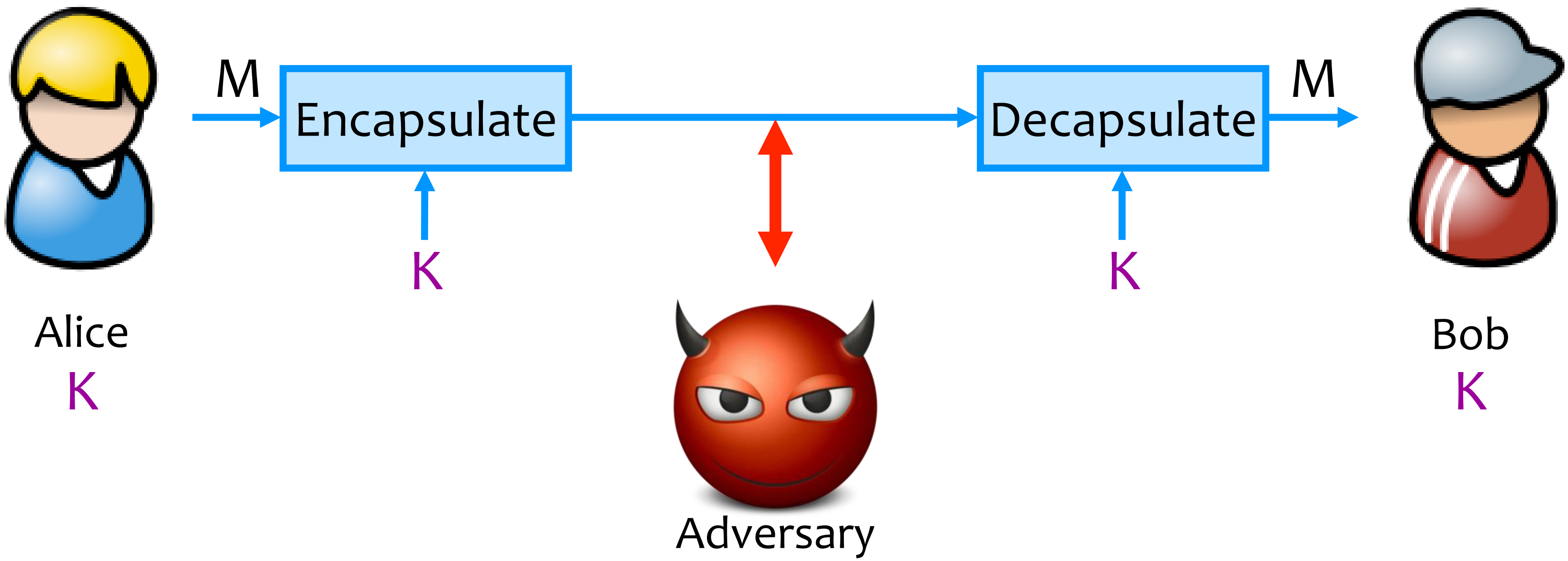
Both compute random bit at bA xor bB

Stepping Back: 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.

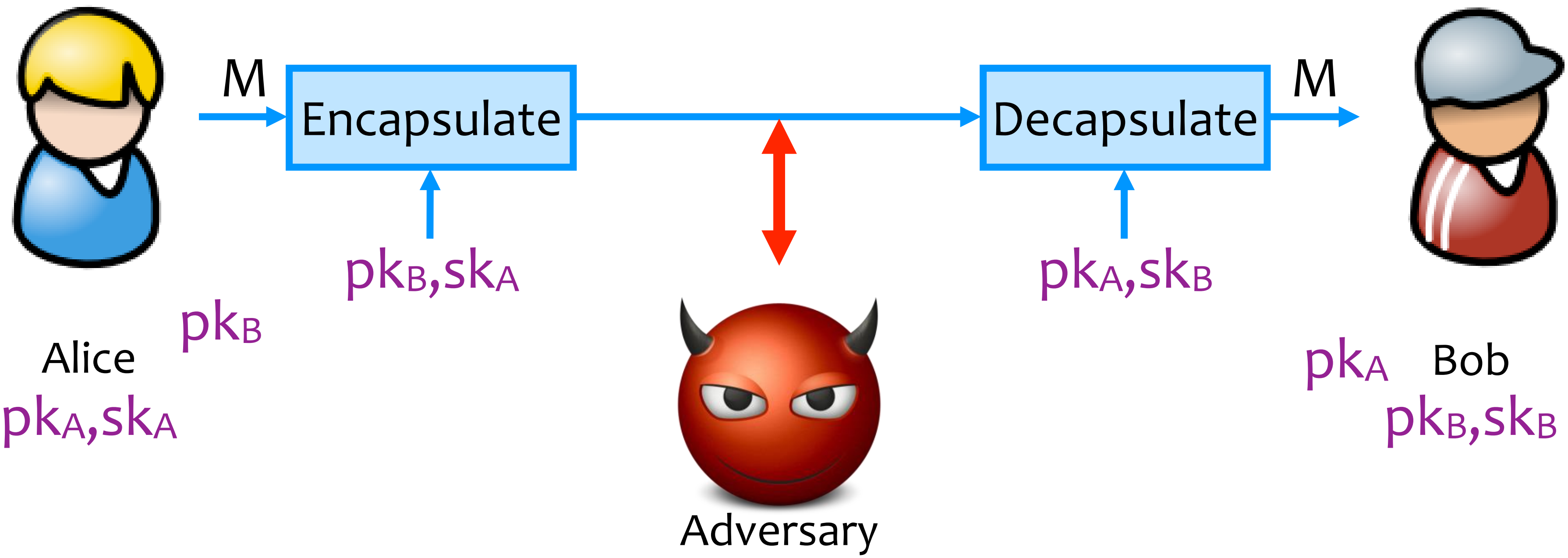
Symmetric Setting

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



Asymmetric Setting

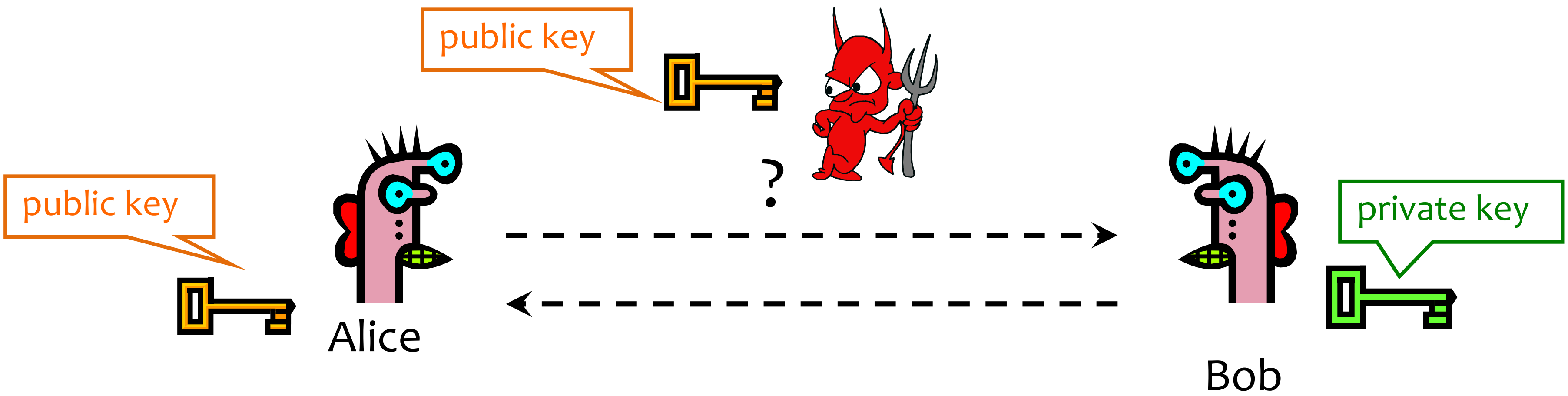
Each party creates a public key pk and a secret key sk.



Flavors of Cryptography

- Symmetric cryptography
  - Both communicating parties have access to a shared random string K, called the key.
  - Challenge: How do you privately share a key?
- Asymmetric cryptography
  - Each party creates a public key pk and a secret key sk.
  - Challenge: How do you validate a public key?

Public Key Crypto: Basic Problem



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 himself

Applications of Public Key Crypto

- Session key establishment
  - Exchange messages to create a secret session key
  - Then switch to symmetric cryptography (why?)
- 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

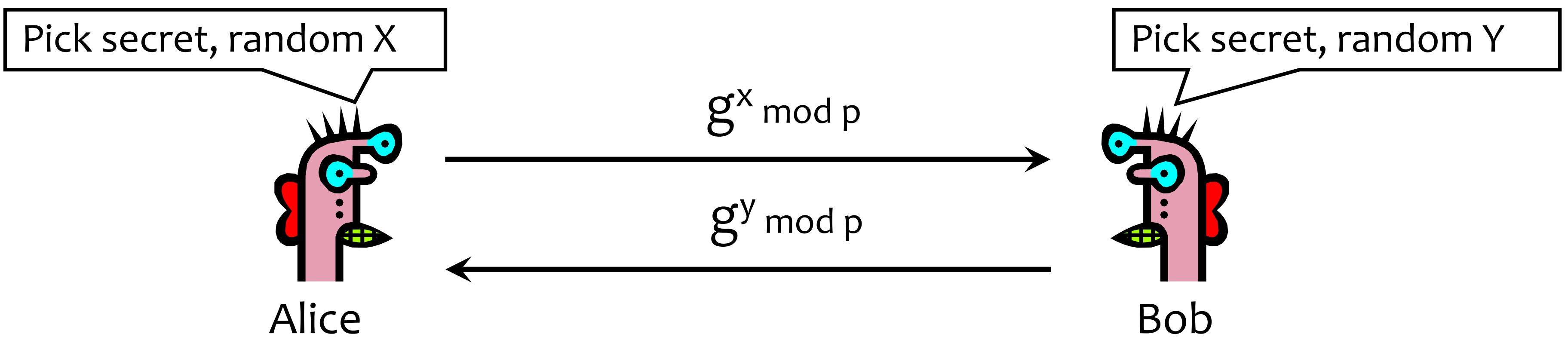
Session Key Establishment

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)
    - g=7 is a "generator" of Z_{11}^*
  - For p=11, g=3
    - 3^1 mod 11 = 3, 3^2 mod 11 = 9, 3^3 mod 11 = 5, …
    - Produces cyclic group {3,9,5,4,1} (order = 5)

Diffie-Hellman Protocol (1976)

- Alice and Bob never met and share no secrets
- Public info: p and g
  - p is a large prime, g is a **generator** of Z_p*
    - $Z_p^* = \{1, 2 \ldots p-1\}$; for all a in Z_p* there exists i s.t. $a=g^i \bmod p$
    - Modular arithmetic: numbers "wrap around" after they reach p



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

Why is Diffie-Hellman Secure?

- Discrete Logarithm (DL) problem:
  given g^x *mod p*, it's hard to extract x
  - There is no known efficient algorithm for doing this
  - This is not enough for Diffie-Hellman to be secure!
- Computational Diffie-Hellman (CDH) problem:
  given g^x *mod p* and g^y *mod p*, 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 *mod p* and g^y *mod p*, it's hard to tell the difference between g^{xy} *mod p* and g^r *mod p* where r is random

Properties of Diffie-Hellman

- 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*
  - Eavesdropper can't tell the difference between the established key and a random value
  - Often hash g^{xy} *mod p*, and use the hash as the key
  - Can use the new key for symmetric cryptography
- Diffie-Hellman protocol (by itself) does not provide authentication
  - Party in the middle attack (often called "man in the middle attack")

More on Diffie-Hellman Key Exchange

- **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**

Public Key Encryption

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