CSE 484 / CSE M 584: Finish Hash Functions + Start Asymmetric Cryptography

Fall 2022

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

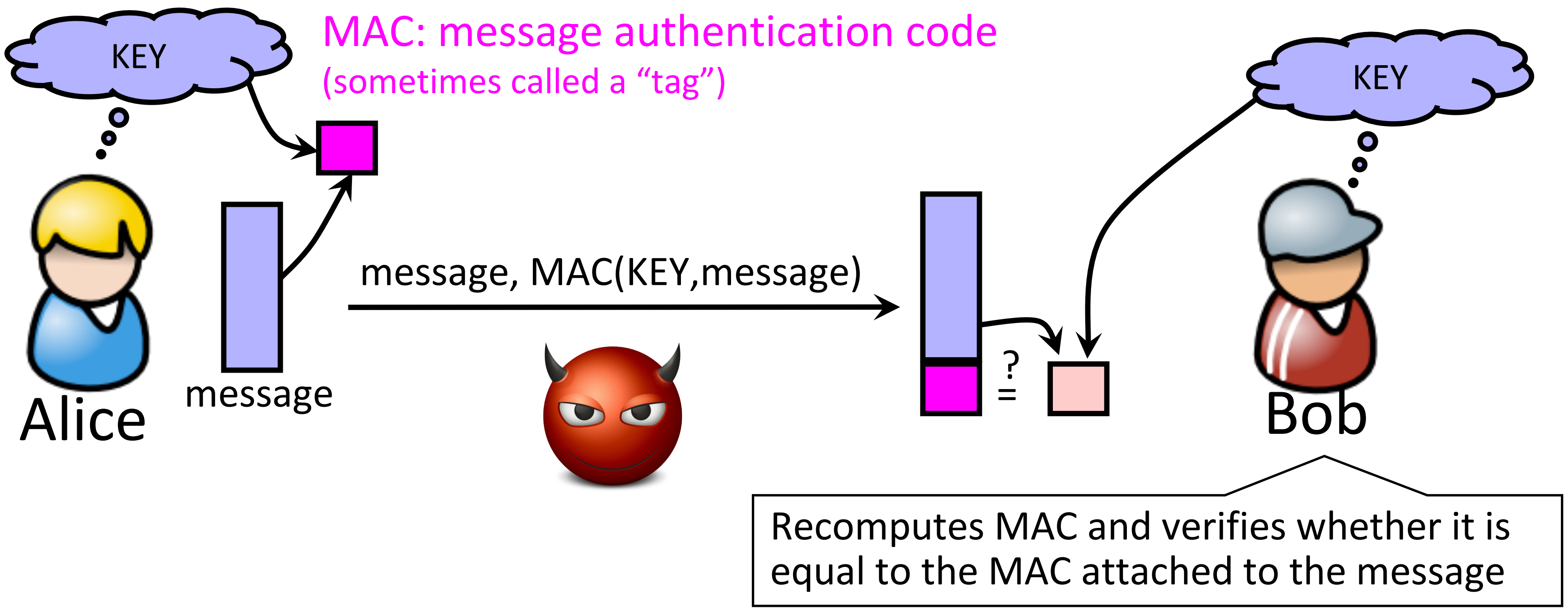
- Things due
  - Lab 1: Thursday
  - Homework 2: Next Friday
    - Individual assignment (no groups)
  - CSE 584M: Don't forget about weekly research readings

Hash Functions Review

- Map large domain to small range (e.g., range of all 160- or 256-bit values)
- Properties:
  - One-Wayness: Given an output (that was computed as the hash of a random domain element), hard to find a preimage
  - Collision Resistance: Hard to find two distinct inputs that map to same output
  - Weak Collision Resistance: Given a point in the domain and its hash in the range, hard to find a new domain element that maps to the same range element

Recall: Achieving Integrity

Message authentication schemes: A tool for protecting integrity.



Integrity and authentication: only someone who knows KEY can compute correct MAC for a given message.

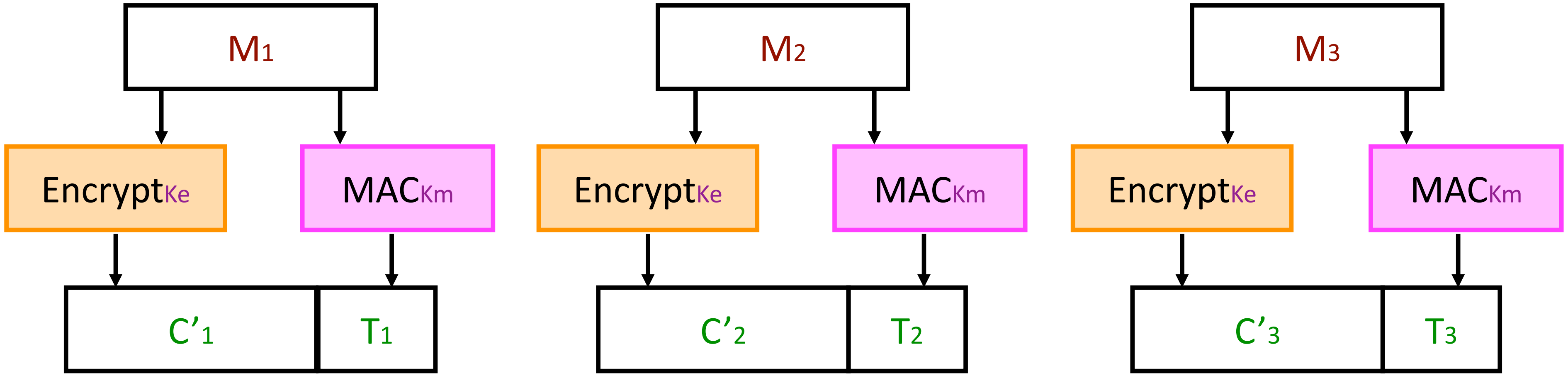
MAC with SHA3

- SHA3(Key || Message)
- Nice and simple ☺
- Previous hash functions couldn't quite be used in this way (see: length extension attack)
  - HMAC construction (FYI)

- Why not encryption? (Historical reasons)
  - Hashing is faster than block ciphers in software
  - Can easily replace one hash function with another
  - There used to be US export restrictions on encryption

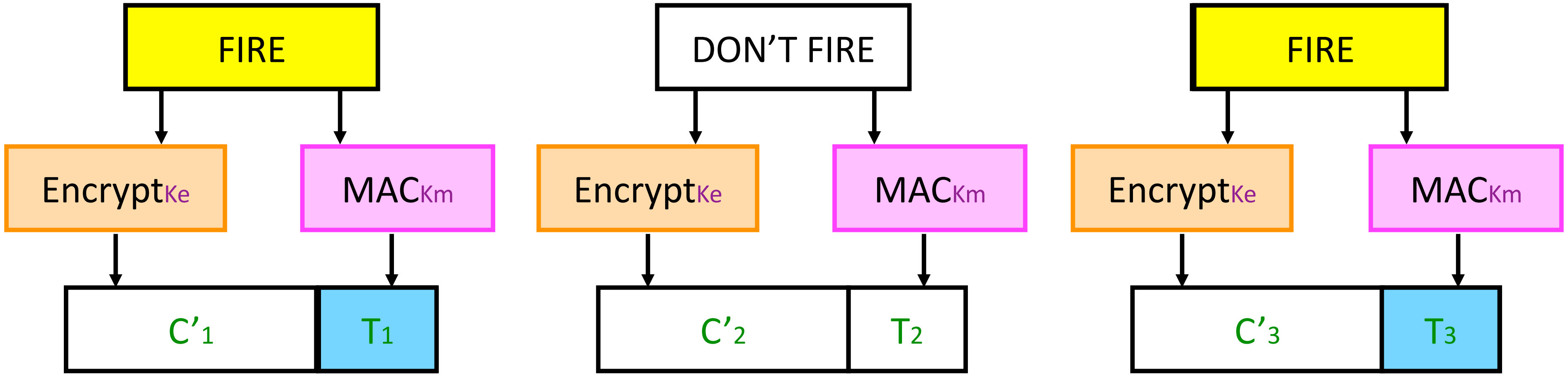
Authenticated Encryption

- What if we want both privacy and integrity?
- Natural approach: combine encryption scheme and a MAC.
- Does this work? → Canvas "quiz" time



Authenticated Encryption

- What if we want both privacy and integrity?
- Natural approach: combine encryption scheme and a MAC.
- But be careful!
  - Obvious approach: Encrypt-and-MAC
  - Problem: MAC is deterministic! same plaintext → same MAC

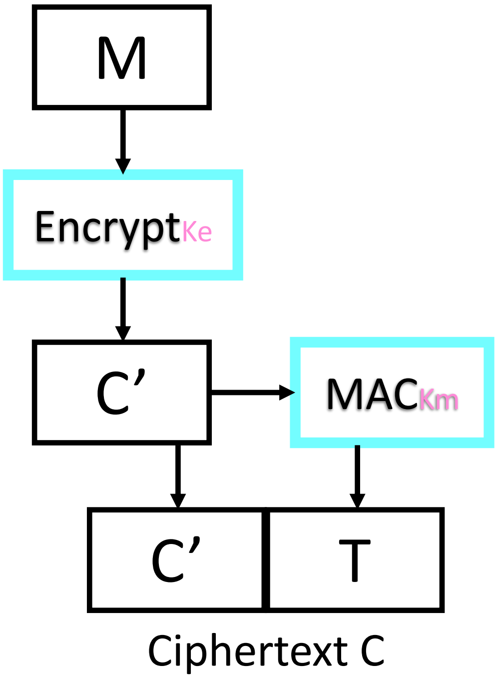


Authenticated Encryption

- Instead:

  Encrypt *then* MAC.

- (Not as good:
  MAC-then-Encrypt)



Encrypt-then-MAC

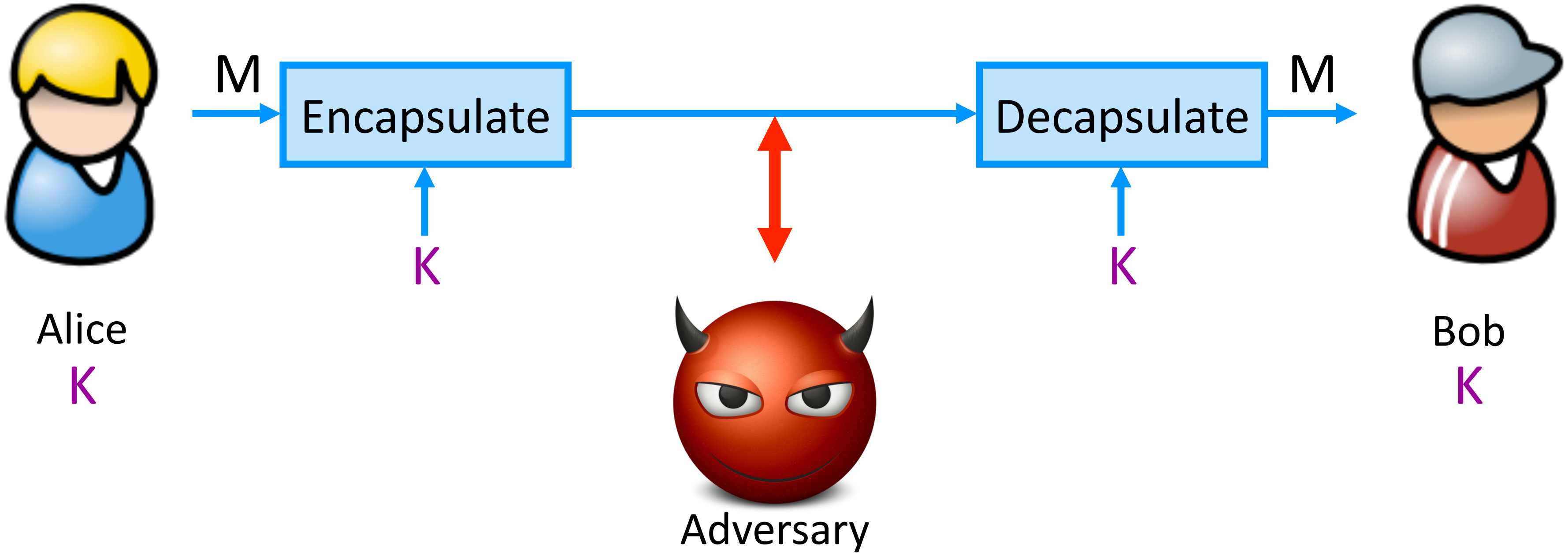
Stepping Back: Flavors of Cryptography

- 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.

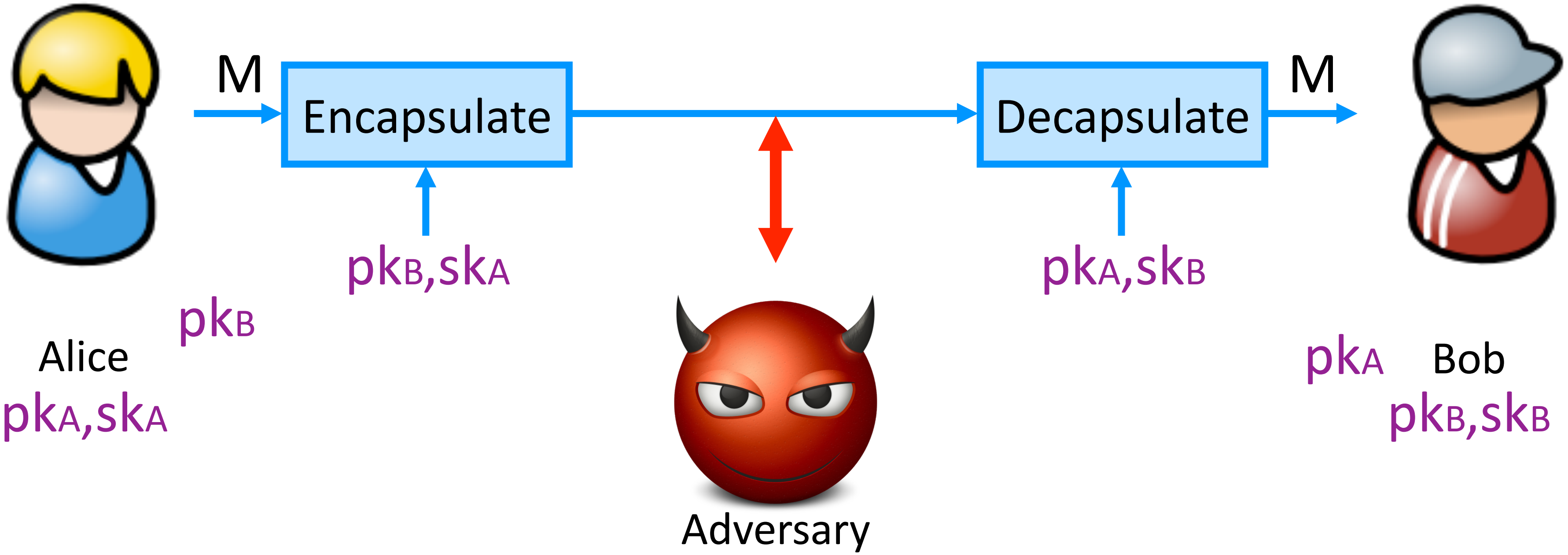
Symmetric Setting

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

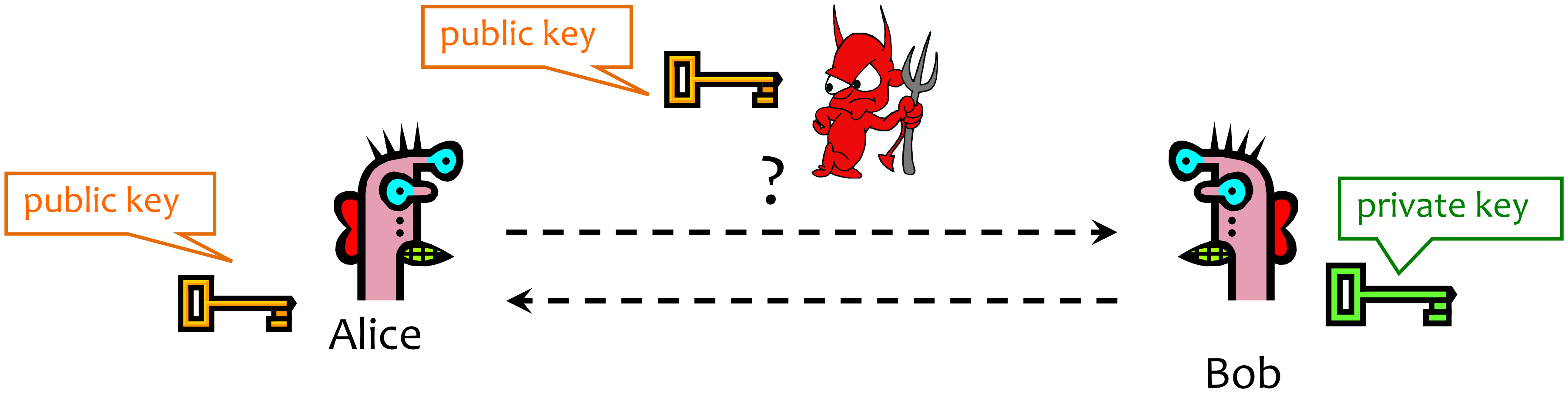


Asymmetric Setting

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



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?)

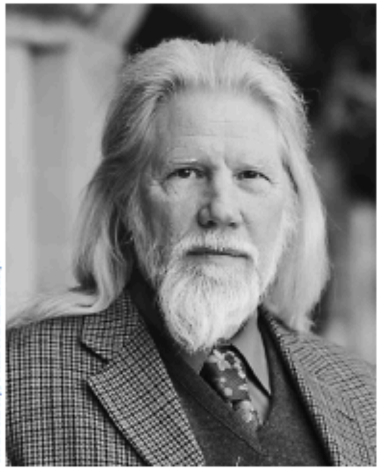
Session Key Establishment

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}^*

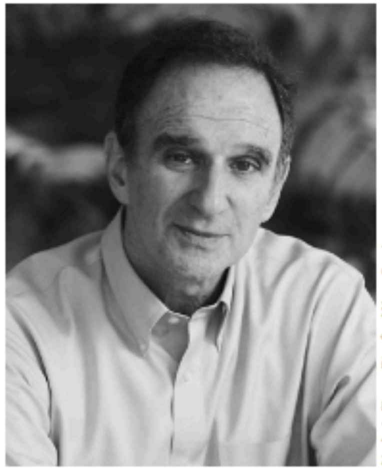
Diffie-Hellman Protocol (1976)

Diffie and Hellman Receive 2015 Turing Award



Rod Searcey/Stanford University.

Whitfield Diffie

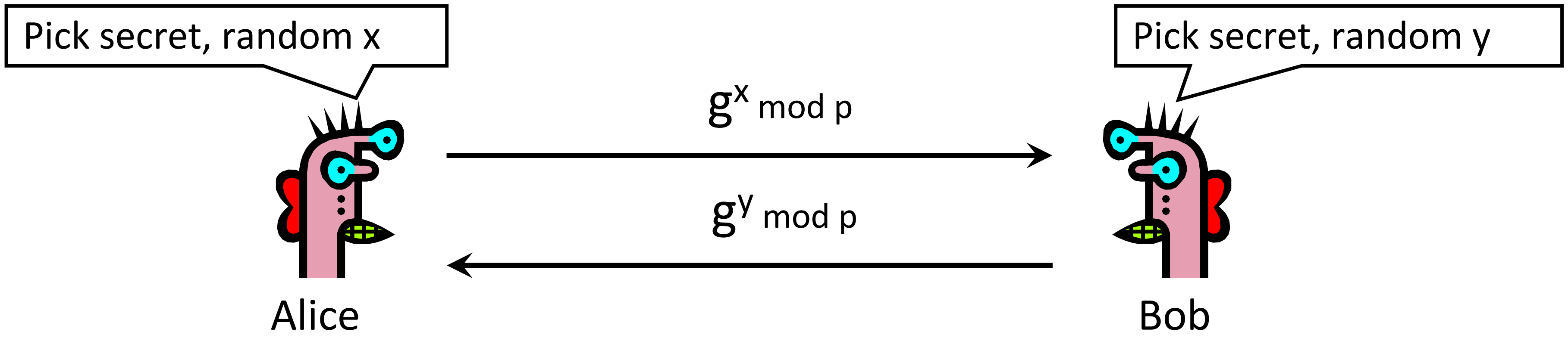


Linda A. Cicero/Stanford News Service.

Martin E. Hellman

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^*
    - $Z_p^* = \{1, 2 \ldots p\text{-}1\}$; 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

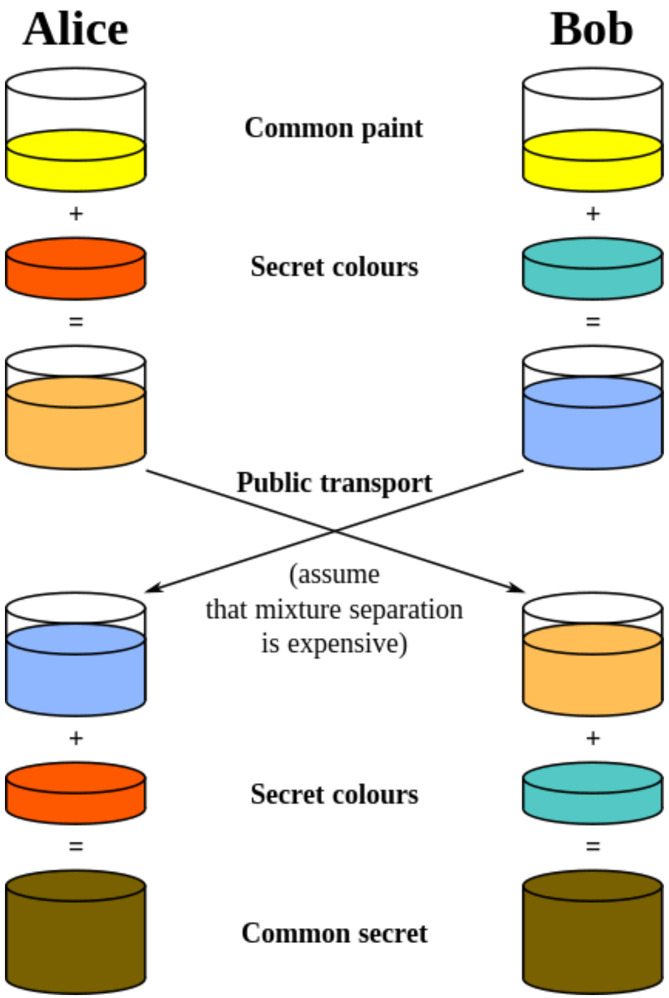
- PUBLIC
  - p = 11
  - g = 2
  - (g is a generator for group mod p)

- 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:
$g^x \bmod p$
$g^y \bmod p$

Common secret: $g^{xy} \bmod p$

[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 efficient algorithm for doing this
  - This is not 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 $g^{xy} \bmod p$, 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