

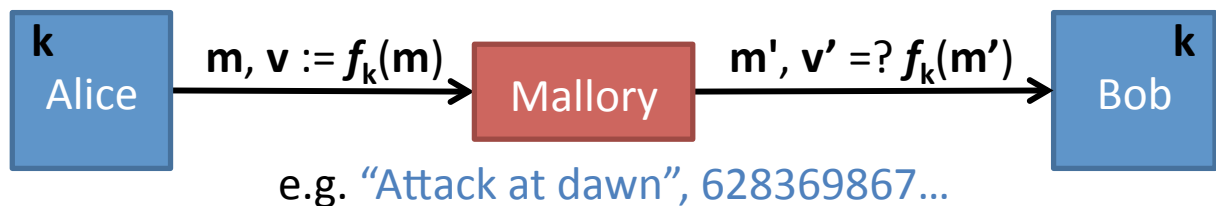
# Key Exchange and Key Management

## Review: Integrity

*Problem:* Sending a message over an **untrusted channel** without being changed

*Provably-secure solution:* **Random function**

*Practical solution:*



### **Pseudorandom function (PRF)**

Input: arbitrary-length  $k$

Output: fixed-length value

Secure if practically indistinguishable from a random function, unless know  $k$

*Real-world use:*

### **Message authentication codes (MACs)**

built on cryptographic hash functions

Popular example: **HMAC-SHA256<sub>k</sub>(m)**

[Cautions?!]

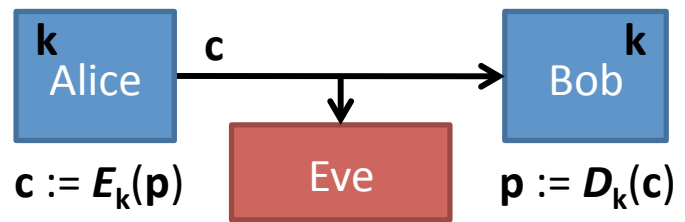
## Review: Confidentiality

*Problem:* Sending message in the presence of an **eavesdropper** without revealing it

*Provably-secure solution:* **One-time pad**

*Practical solution:*

**Pseudorandom generator (PRG)**



Input: fixed-length  $k$

Output: arbitrary-length stream

Secure if practically indistinguishable from a random stream, unless know  $k$

*Real-world use:*

**Stream ciphers** (can't reuse  $k$ )

Popular example: **AES-128** + **CTR mode**

**Block ciphers** (need **padding/IV**)

Popular example: **AES-128** + **CBC mode**

[Cautions?!]

## Building a **secure channel**

What if you want confidentiality and integrity at the same time?

**Encrypt, then add integrity,**  
not the other way around.  
(some reasons are subtle)

**Use separate keys** for  
confidentiality and integrity.

Need two shared keys,  
but only have one?  
That's what PRGs are for!

If there's a reverse (Bob to Alice)  
channel, use separate keys for that too

## Issue: How big should keys be?

Want prob. of guessing to be infinitesimal... but  
watch out for Moore's law – safe size gets  
1 bit larger every 18 months

128 bits usually safe for ciphers/PRGs

Need larger values for MACs/PRFs  
due to **birthday attack**

Often trouble if adversary can find  
any two messages with same MAC

Attack: Generate random values,  
look for coincidence.

Requires  $O(2^{|k|/2})$  time,  $O(2^{|k|/2})$  space.

For 128-bit output, takes  $2^{64}$  steps: doable!

[Puzzle: Do it in constant space?]

Upshot: Want output of MACs/PRFs  
to be twice as big as cipher keys  
e.g. use HMAC-SHA256 along side AES-128

## Exercise:

100 people all communicating with each other, how many keys are needed? (using symmetric key crypto)

# Issue: How do we get a shared key?

## Amazing fact:

Alice and Bob can have a public conversation to derive a shared key!

## Diffie-Hellman (D-H) key exchange

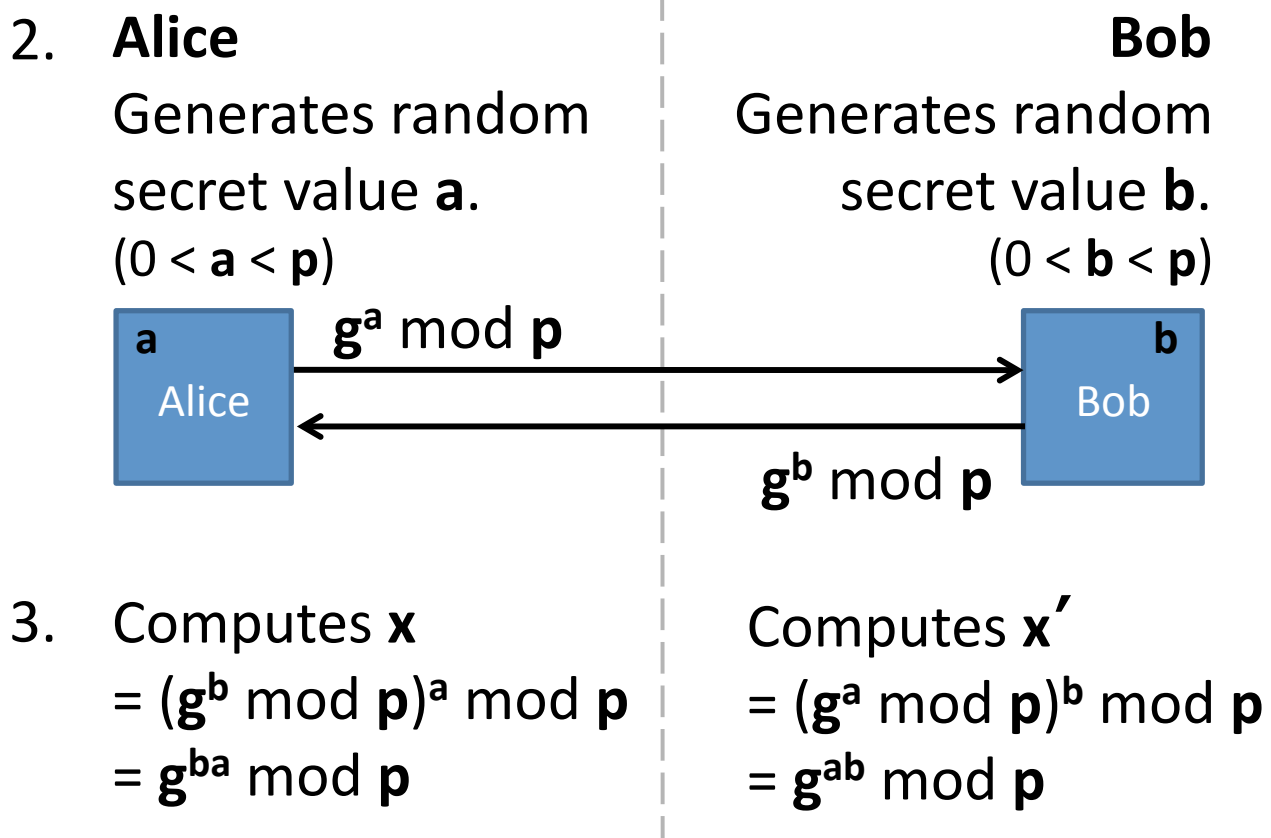
1976: Whit Diffie, Marty Hellman  
with ideas from Ralph Merkle  
(earlier, in secret, by Malcolm Williamson  
of British intelligence agency)

Relies on a mathematical hardness  
assumption called *discrete log problem*  
(a problem believed to be hard)

Cannot be use by itself.

## D-H protocol

1. Alice and Bob agree on public parameters  
(maybe in standards doc)  
**p**: a large “safe prime” s.t.  $(p-1)/2$  is also prime  
**g**: a square mod **p** (but not 1)



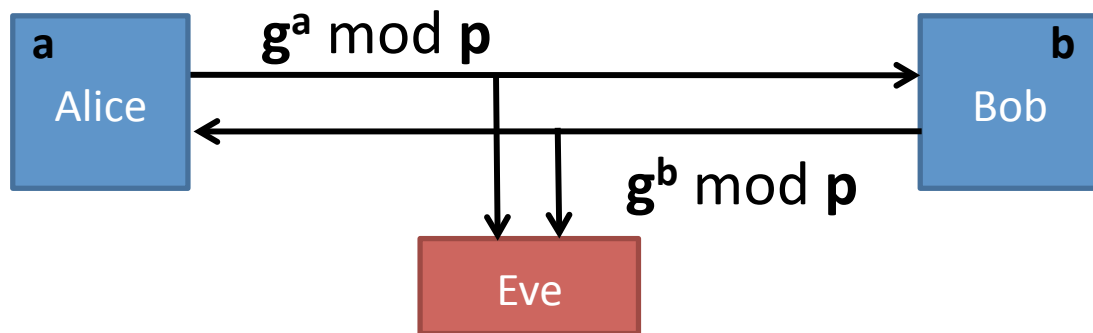
3. Computes **x**  
 $= (g^b \bmod p)^a \bmod p$   
 $= g^{ba} \bmod p$
- Computes **x'**  
 $= (g^a \bmod p)^b \bmod p$   
 $= g^{ab} \bmod p$

(Notice that  $x == x'$  )

Can use  $k := \text{HMAC}_0(x)$  as a shared key.



## Passive eavesdropping attack



Eve knows:  $p$ ,  $g$ ,  $g^a \bmod p$ ,  $g^b \bmod p$

Eve wants to compute  $x = g^{ab} \bmod p$

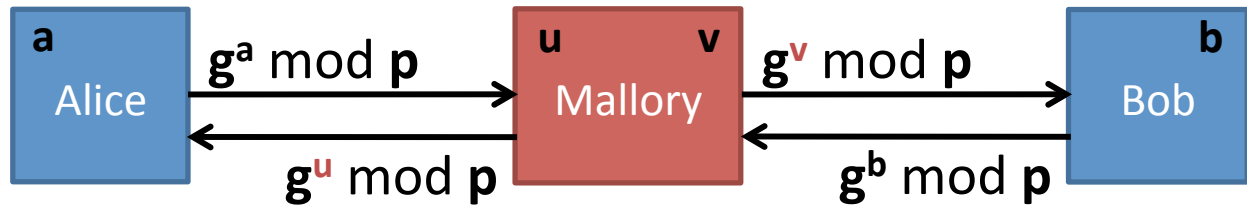
Best known approach:

Find  $a$  or  $b$ , then compute  $x$

Finding  $y$  given  $g^y \bmod p$  is an instance of the **discrete log problem**:  
No known efficient algorithm.

[What's D-H's big weakness?]

## Man-in-the-middle (MITM) attack



Alice does D-H exchange, *really with Mallory*, ends up with  $g^{au} \bmod p$

Bob does D-H exchange, *really with Mallory*, ends up with  $g^{bv} \bmod p$

Alice and Bob each think they are talking with the other, but really Mallory is between them and knows both secrets

*Bottom line:*

D-H gives you secure connection, but you don't know who's on the other end!

## Defending D-H against MITM attacks:

- Cross your fingers and hope there isn't an active adversary.
- Rely on out-of-band communication between users. [\[Examples?\]](#)
- Rely on physical contact to make sure there's no MITM. [\[Examples?\]](#)
- Integrate D-H with user authentication.  
If Alice is using a password to log in to Bob, leverage the password:  
Instead of a fixed  $g$ , derive  $g$  from the password – Mallory can't participate w/o knowing password.
- Use digital signatures. [\[More next week.\]](#)

## Exercise:

What happens if Alice uses the same  $a$  and  $g^a$  for all her communication with Bob?

# The hard part of crypto

## Key-management

### Principles:

0. Always remember,  
key management is the hard part!
1. Each key should have only one purpose.
2. Vulnerability of a key increases:
  - a. The more you use it.
  - b. The more places you store it.
  - c. The longer you have it.
3. Keep your keys far from the attacker.
4. Protect yourself against compromise of old keys.

Goal: **forward secrecy** — learning old key shouldn't help adversary learn new key.

[How can we get this?]

D-H doesn't quite deliver public-key encryption:

Alice and Bob agree on a secret key

El Gamal encryption directly supports encryption:

Bob's public key is  $B = g^b \bmod p$ ,  
private key is  $b$ .

Alice picks a random value  $r \in [0, p-2]$ ,  
forms ciphertext for message  $m$ :

$(g^r \bmod p, m \times B^r \bmod p)$  or  $(R, S)$

Bob computes  $R^{-b} \times S \bmod p$   
which is  $m \bmod p$ !

## **So Far: Foundations**

The Security Mindset

Message Integrity

Confidentiality

Key Exchange

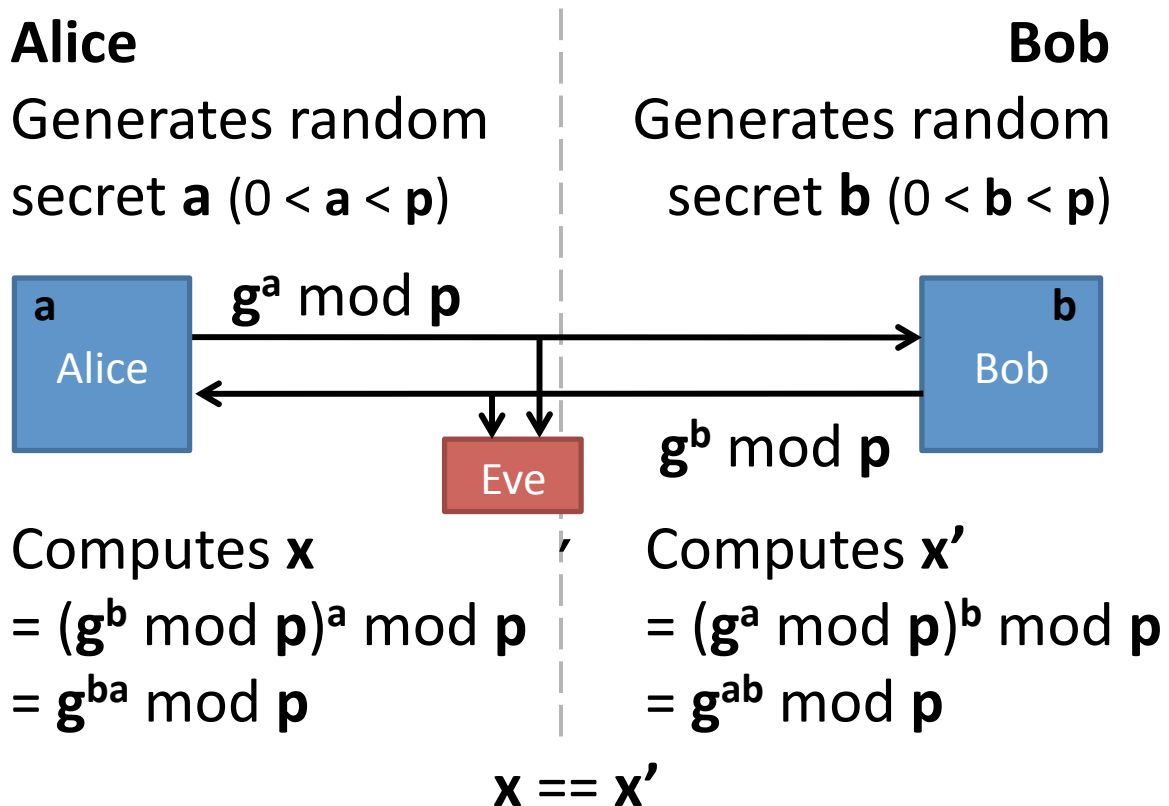
Building a Secure Channel

**Wednesday:**

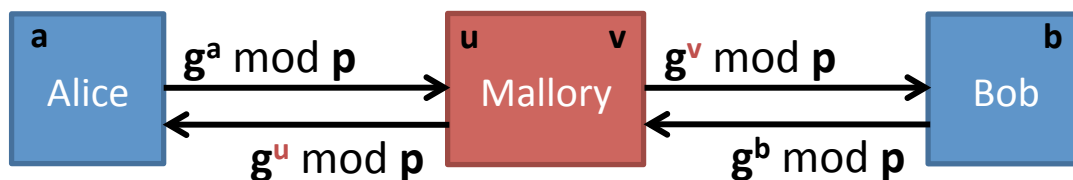
Public Key Crypto

## Review: Diffie-Hellman Key Exchange

Lets Alice and Bob **agree on a shared secret** value by having a public conversation



*Problem:* **Man-in-the-middle attacks**



Caution: D-H gives you a shared secret, but don't know who's on the other end!