Unit 9: Cryptography

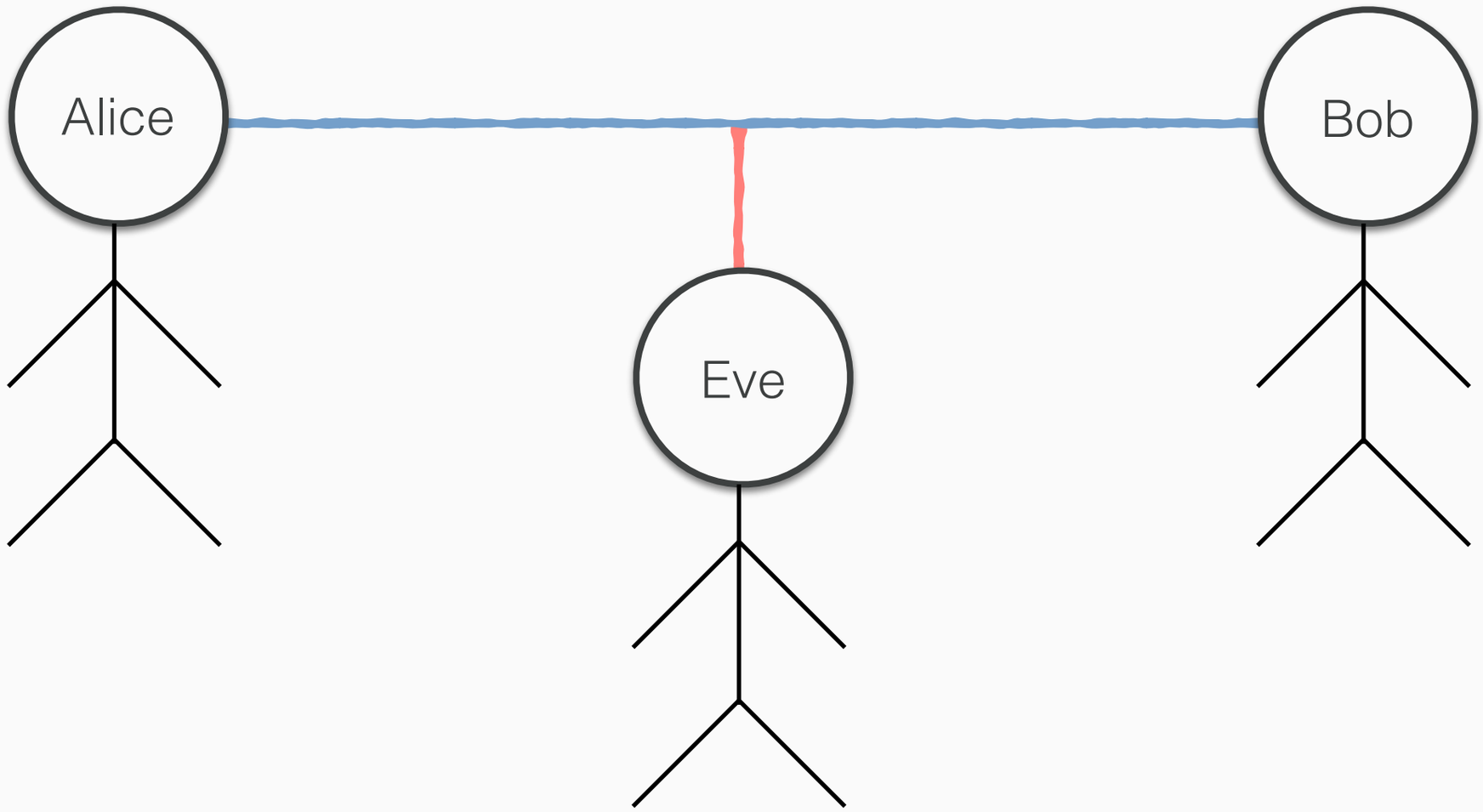
Dave Abel

April 13th, 2016

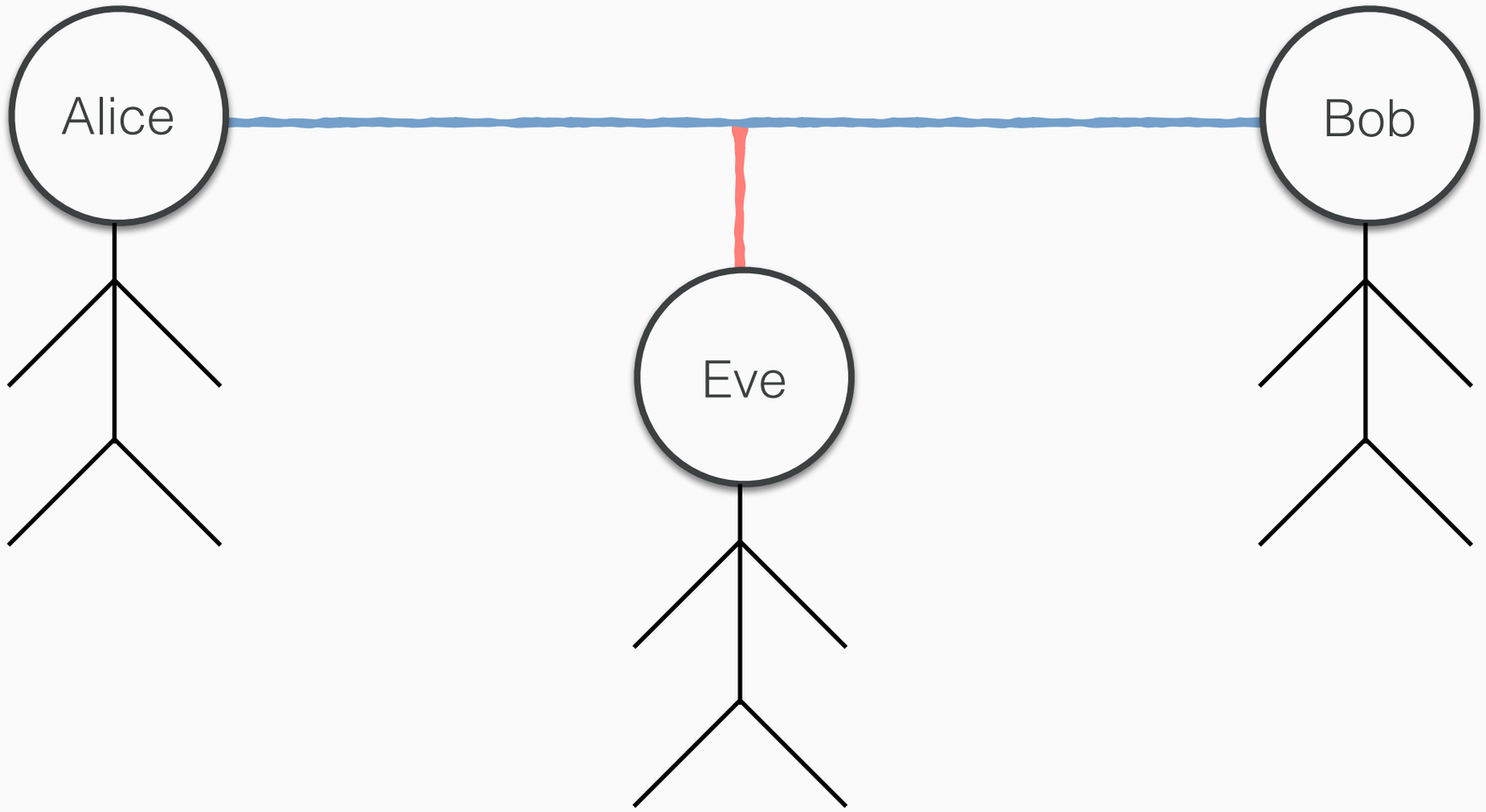
Outline For Today

- Modern Cryptography
  - One Way Functions
  - Diffie-Helman Key Exchange

Modern Cryptography

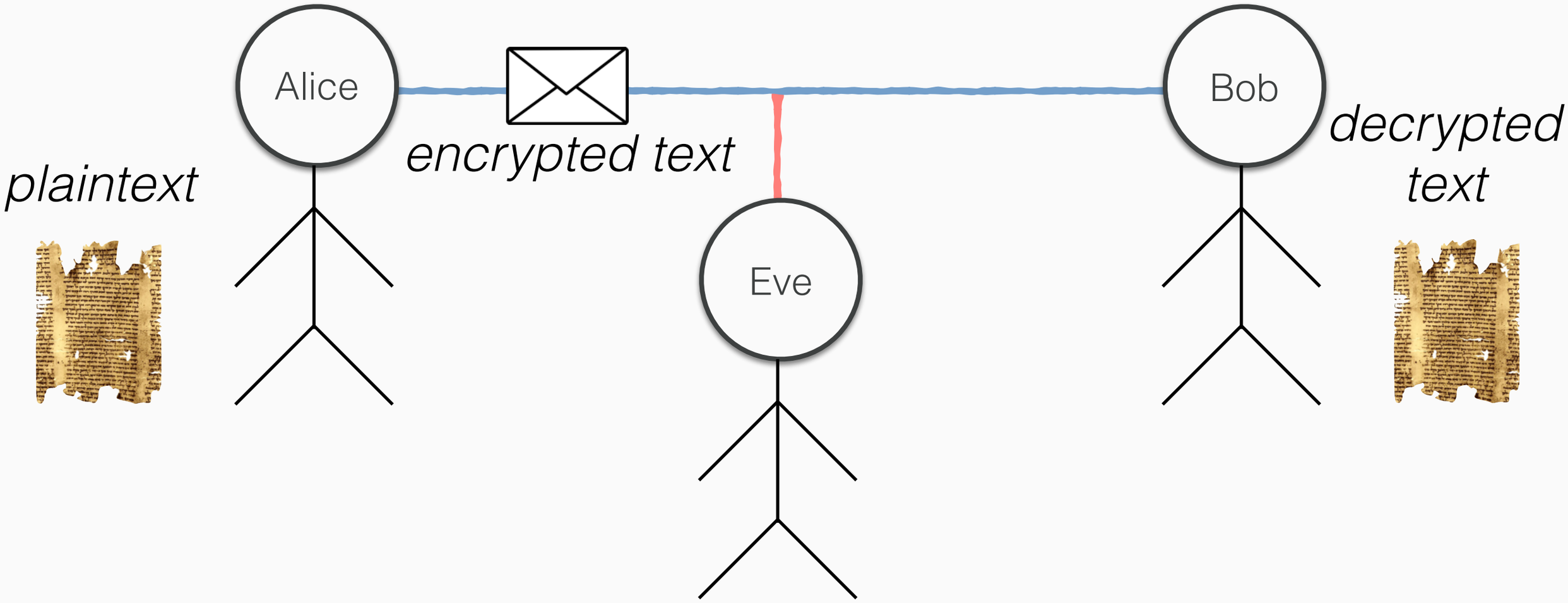


Modern Cryptography



Same problem, same terminology. More computation!

Modern Cryptography



Same problem, same terminology. More computation!

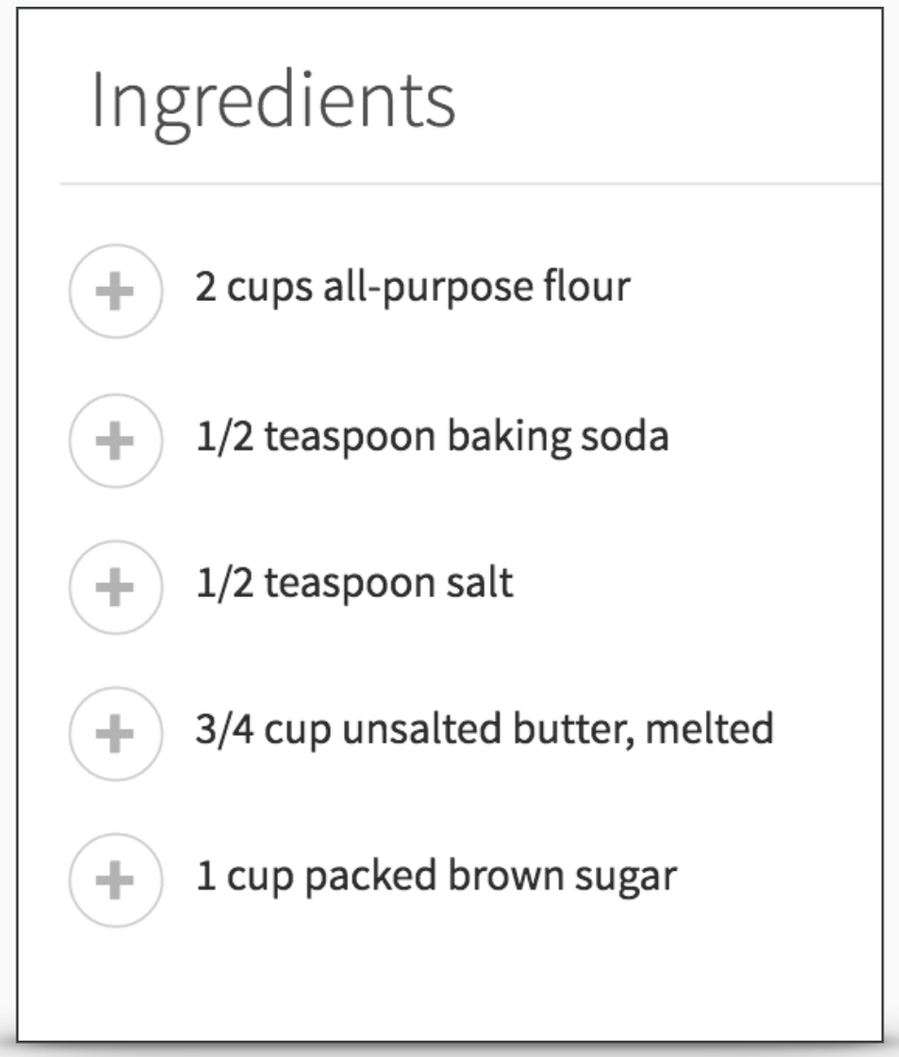
Modern Cryptography

We can use the tools of computation to be confident Eve cannot read our encrypted messages.

Modern Cryptography: Outline

- One Way Functions
- Public Key Cryptography
- Randomness

One Way Function: Intuition



One Way Function: Intuition

One Way Function: Intuition

Easy!

One Way Function: Intuition

????

One Way Function: Intuition

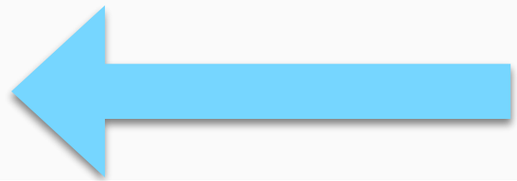
Hard

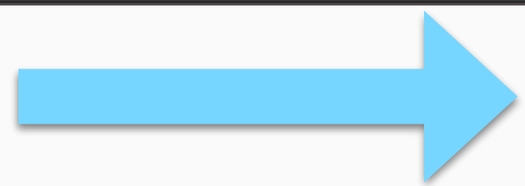
One Way Function: Intuition

Hard

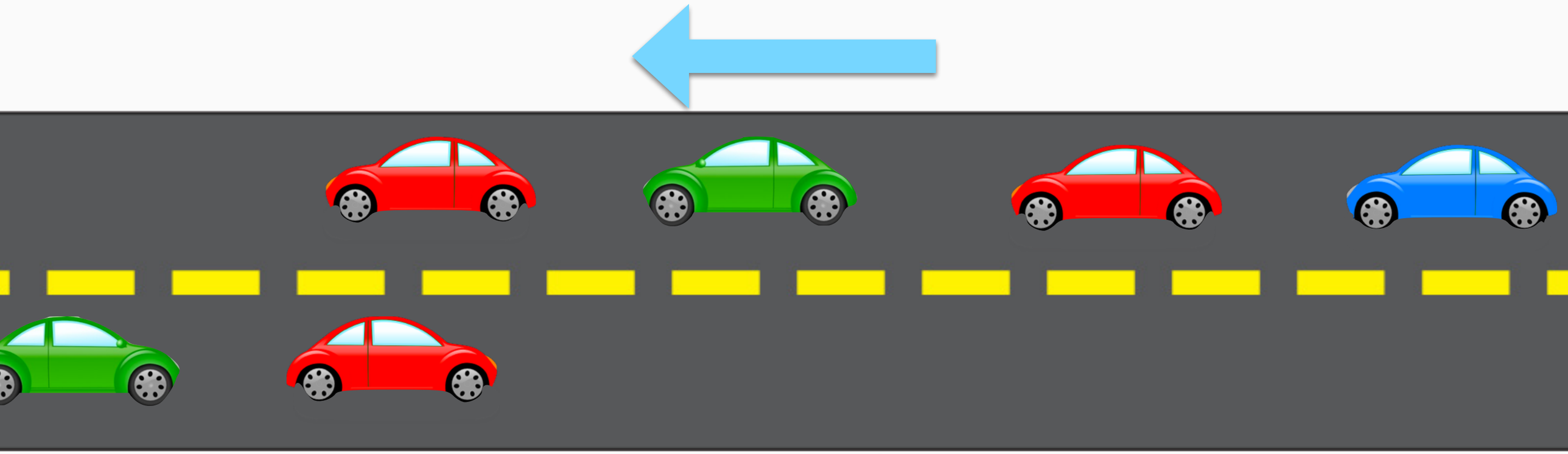
Q: are there *functions* that are easy to compute in one direction, but hard to compute in the other direction?

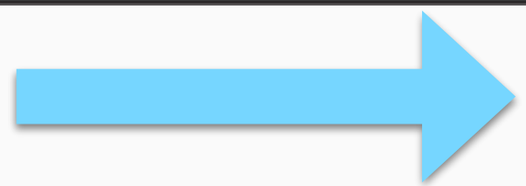
Direction & Functions



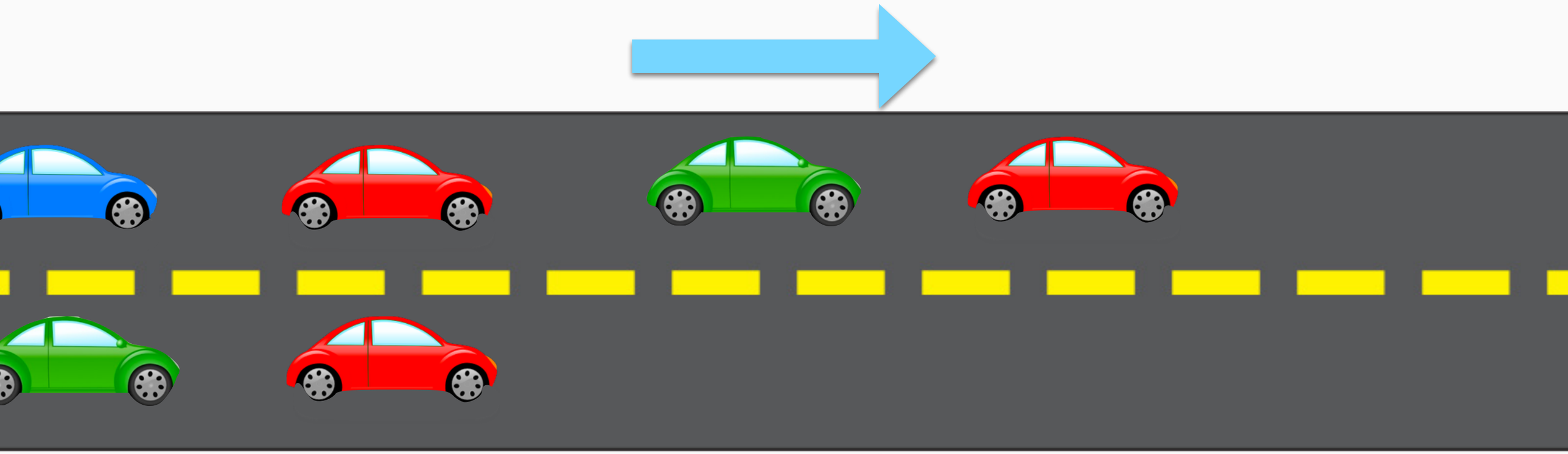


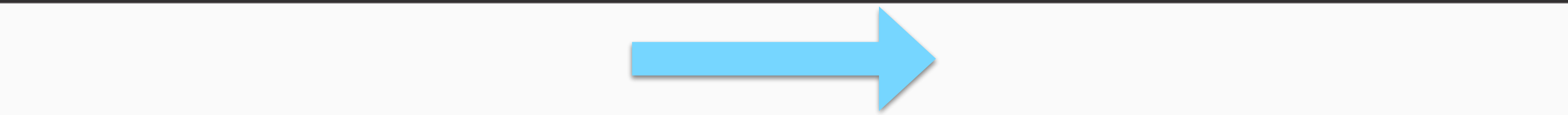
Direction & Functions





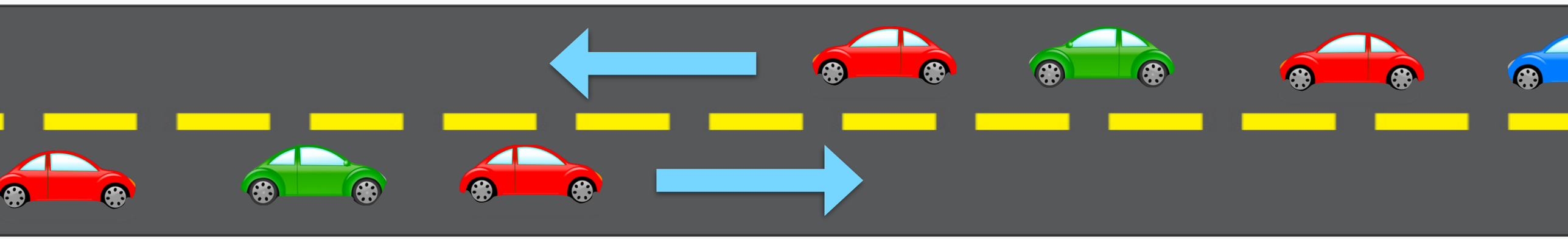
Direction & Functions





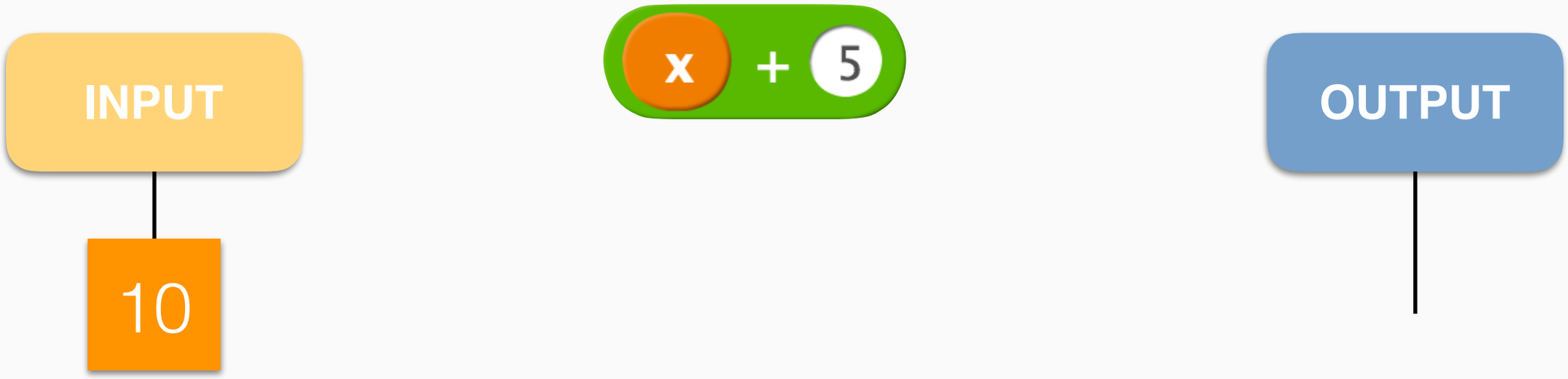
Direction & Functions

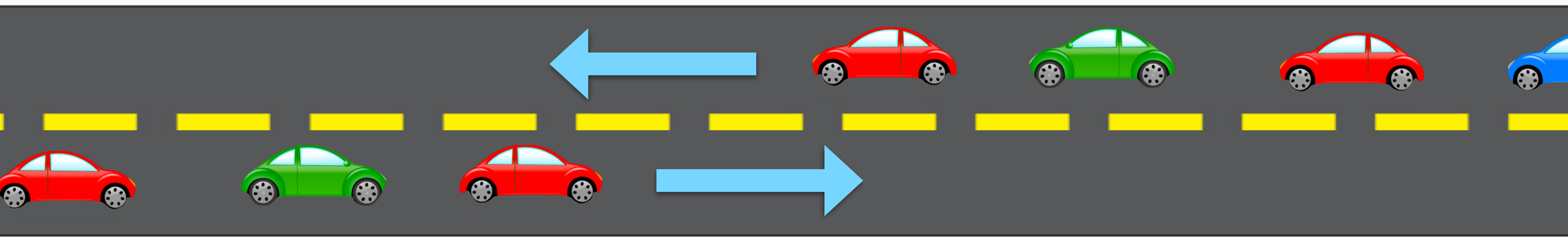
Addition Function:



Direction & Functions

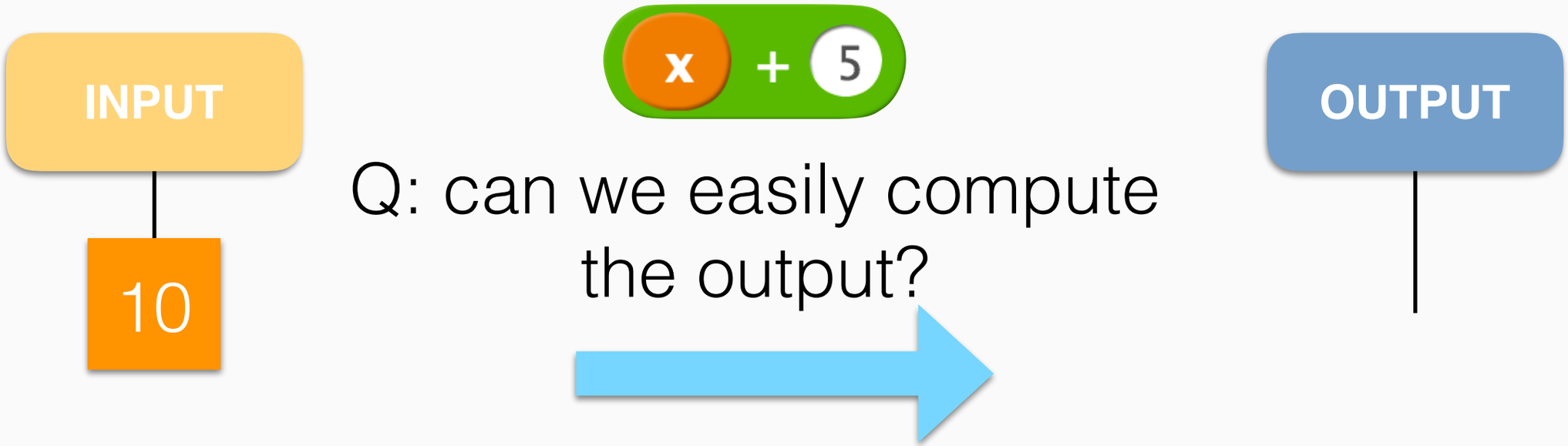
Addition Function:



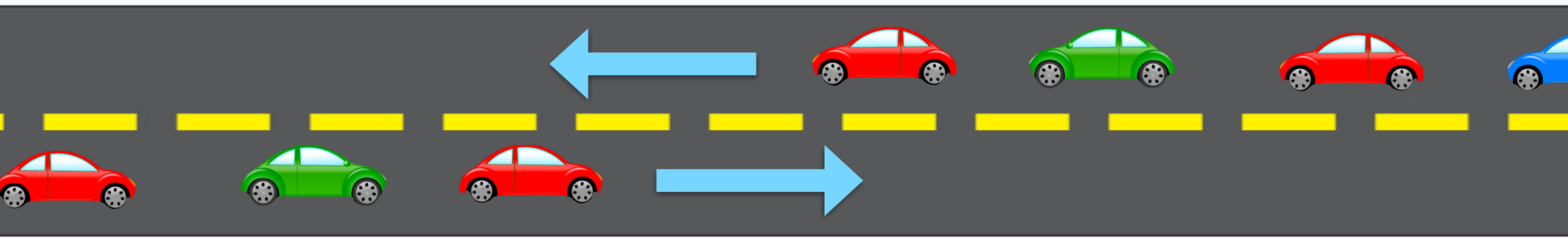


Direction & Functions

Addition Function:

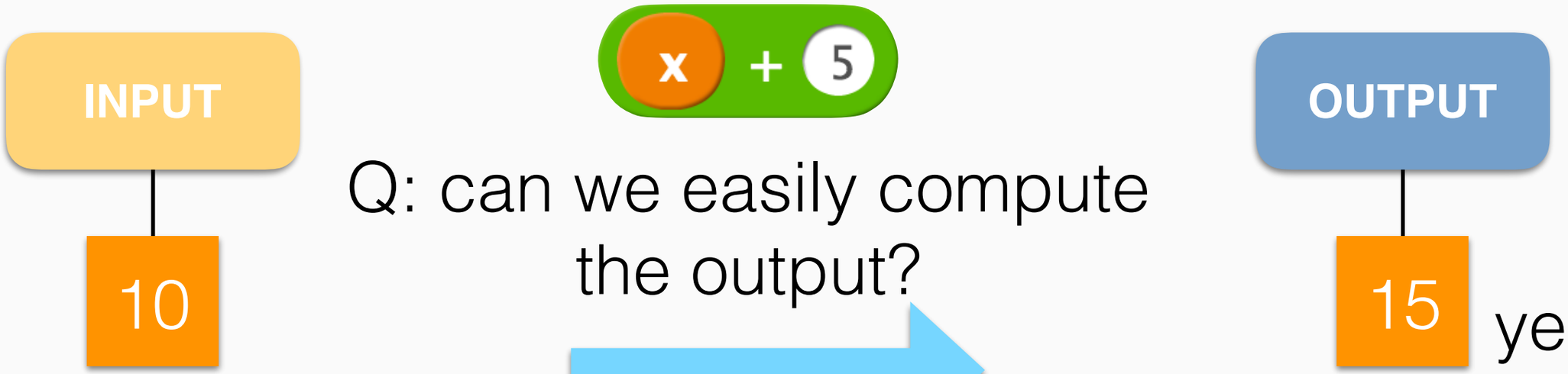


Q: can we easily compute the output?



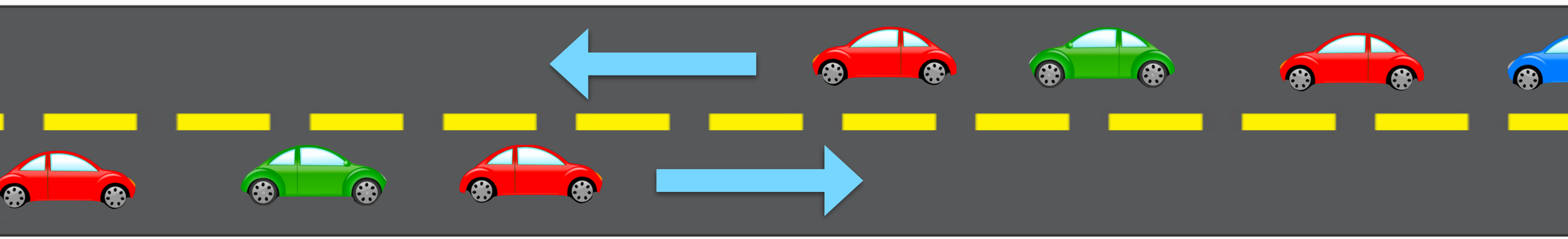
Direction & Functions

Addition Function:



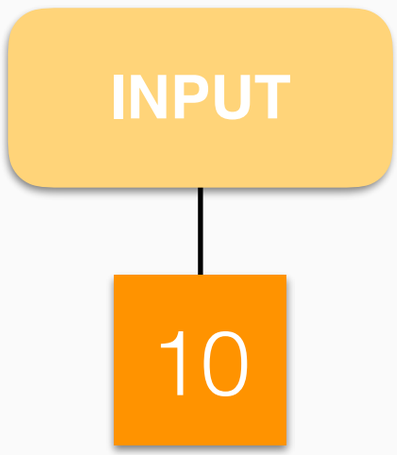
Q: can we easily compute the output?

yes!

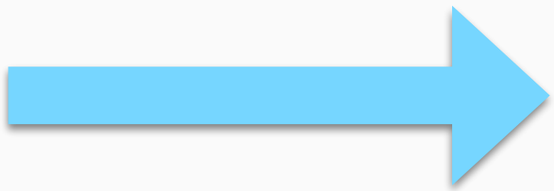


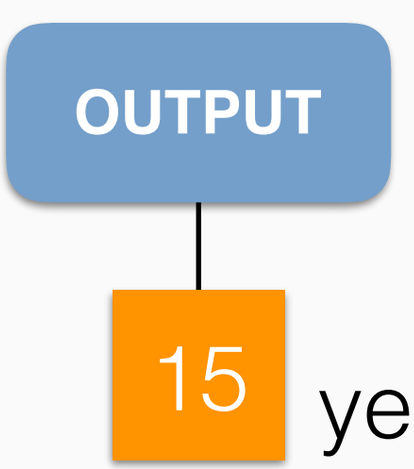
Direction & Functions

Addition Function:



Q: Given an input, and the function, can we **easily** compute the output?

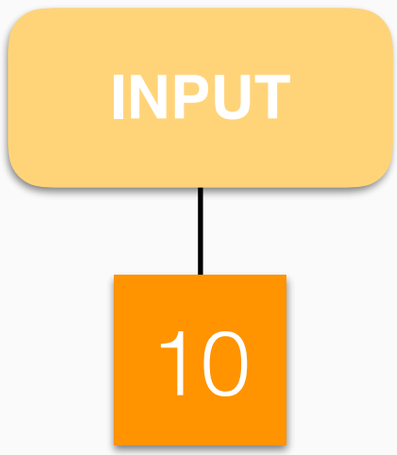




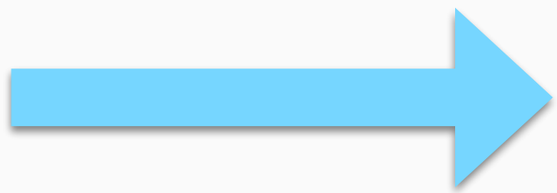
yes!

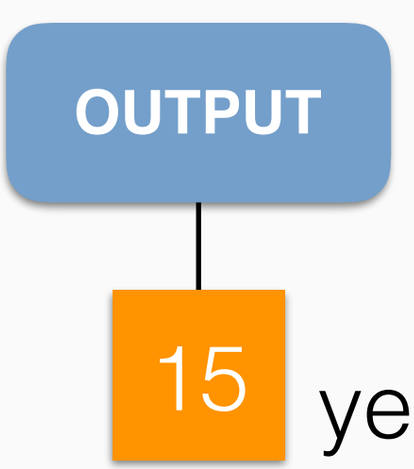
Direction & Functions

Addition Function:



Q: Given an input, and the function, can we **easily** compute the output?





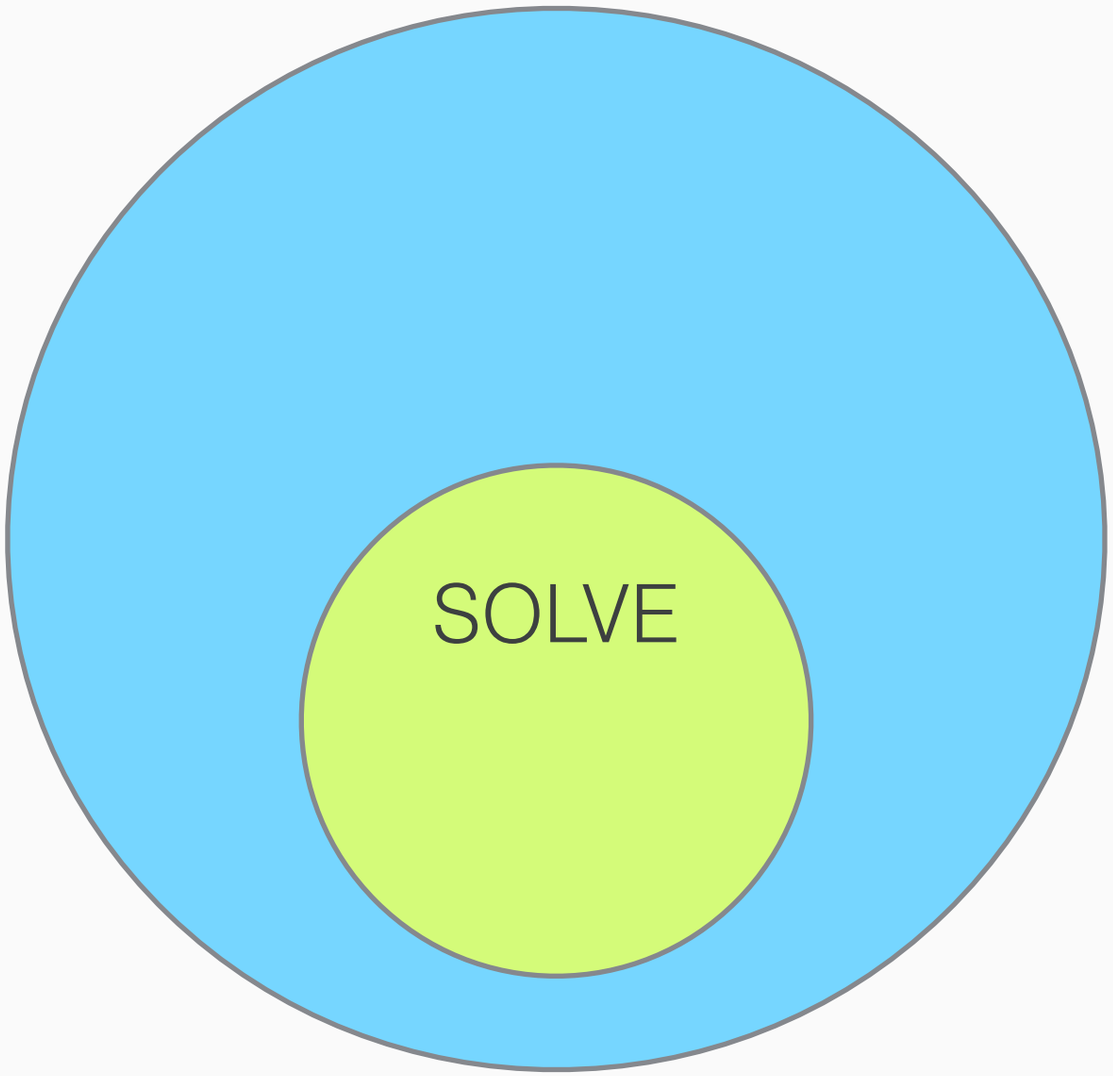
yes!

Easily: Is it a problem computers can solve efficiently?

QUESTION

Easily: Is it a problem computers can solve efficiently?

SOLVE

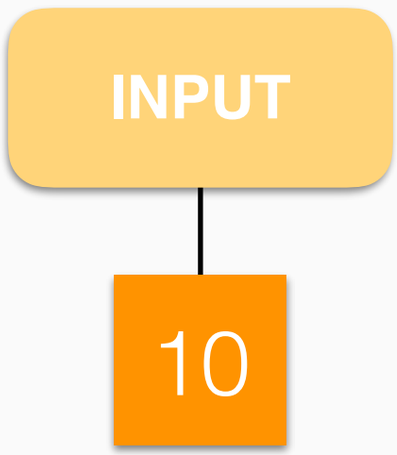


Q: Can we solve a problem efficiently?

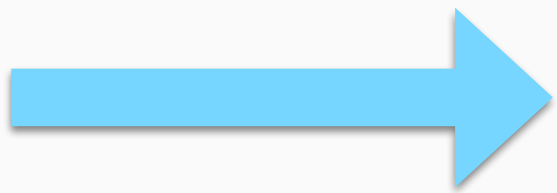
A: Is it in SOLVE?

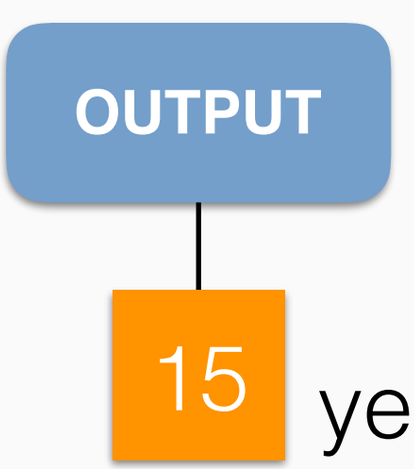
Direction & Functions

Addition Function:



Q: Given an input, and the function, can we **easily** compute the output?



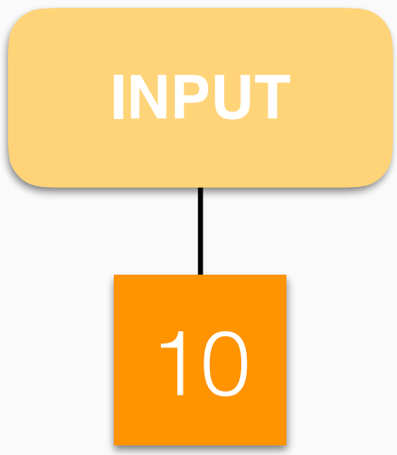


yes!

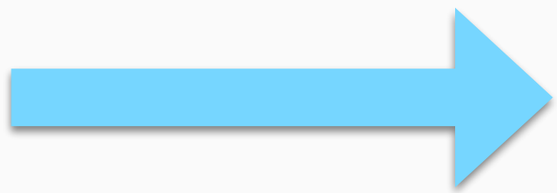
Easily: Is computing this in SOLVE?

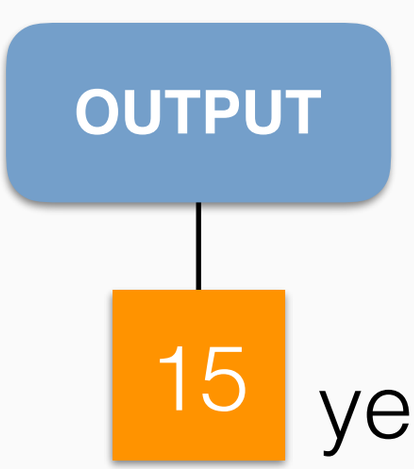
Direction & Functions

Addition Function:



Q: Given an input, and the function, can we **easily** compute the output?





yes!

Easily: Is addition in SOLVE?

Direction & Functions

Addition Function:

Direction One: Input to Output (*in SOLVE)*

Direction & Functions

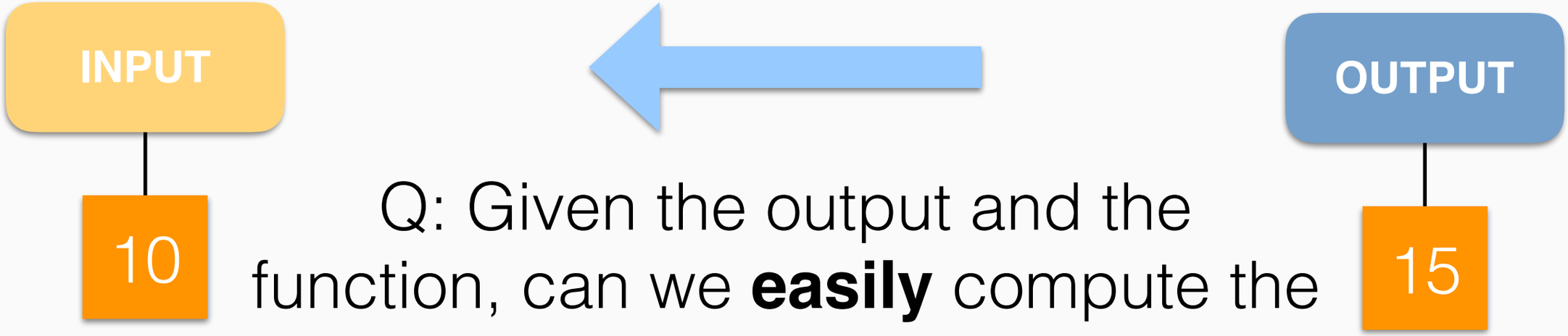
Addition Function:

Direction One: Input to Output (*in SOLVE)*

Direction Two: Output to Input (???)

Direction & Functions

Addition Function:



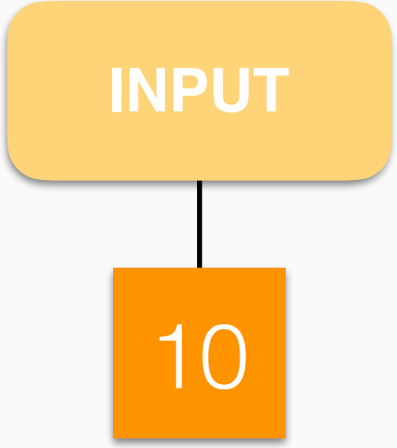
Q: Given the output and the function, can we **easily** compute the input?

Direction One: Input to Output (*in SOLVE)*

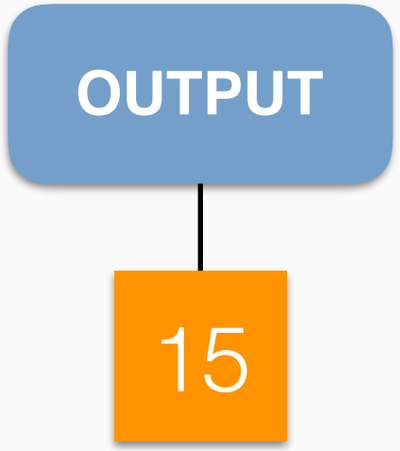
Direction Two: Output to Input (???)

Clicker Question!

Addition Function:



Q: Given the output and the function, can we **easily** compute the input?

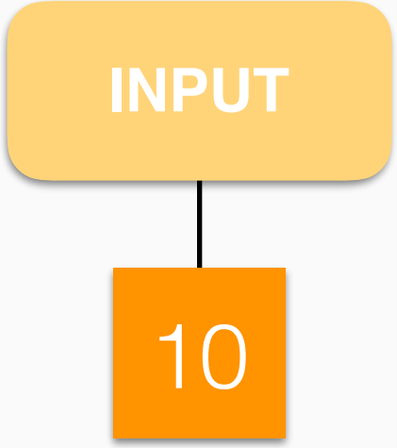


[A] Yes! And I'll explain why :) [B] No! And I'll explain why :)

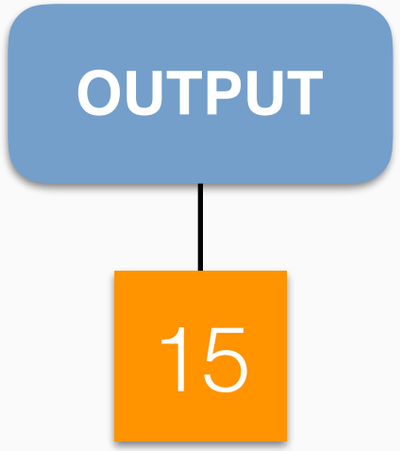
[C] I'm Confused

Clicker Answer!

Addition Function:



Q: Given the output and the function, can we **easily** compute the input?

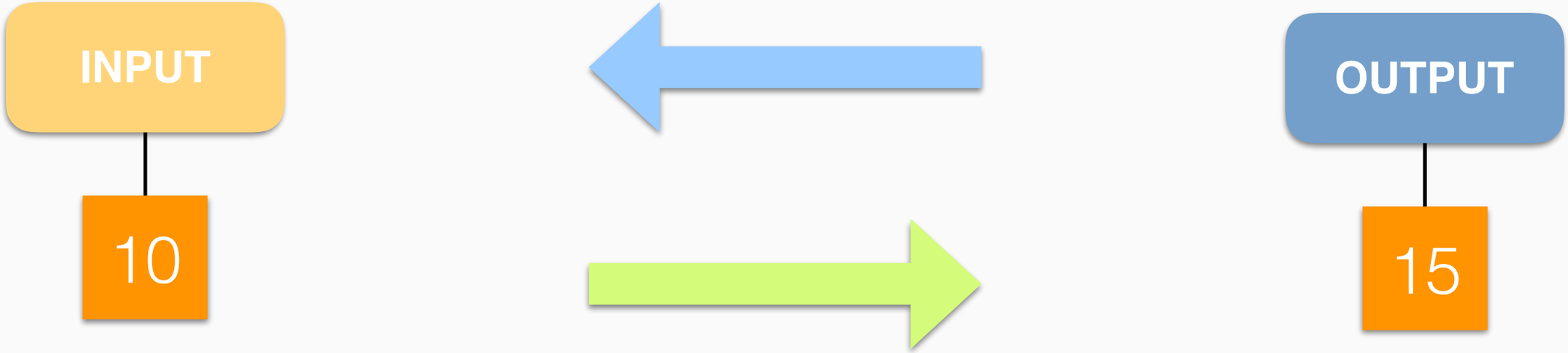


[A] Yes!

15, x + 5 —> Just subtract 5.

Direction & Functions

Addition Function:

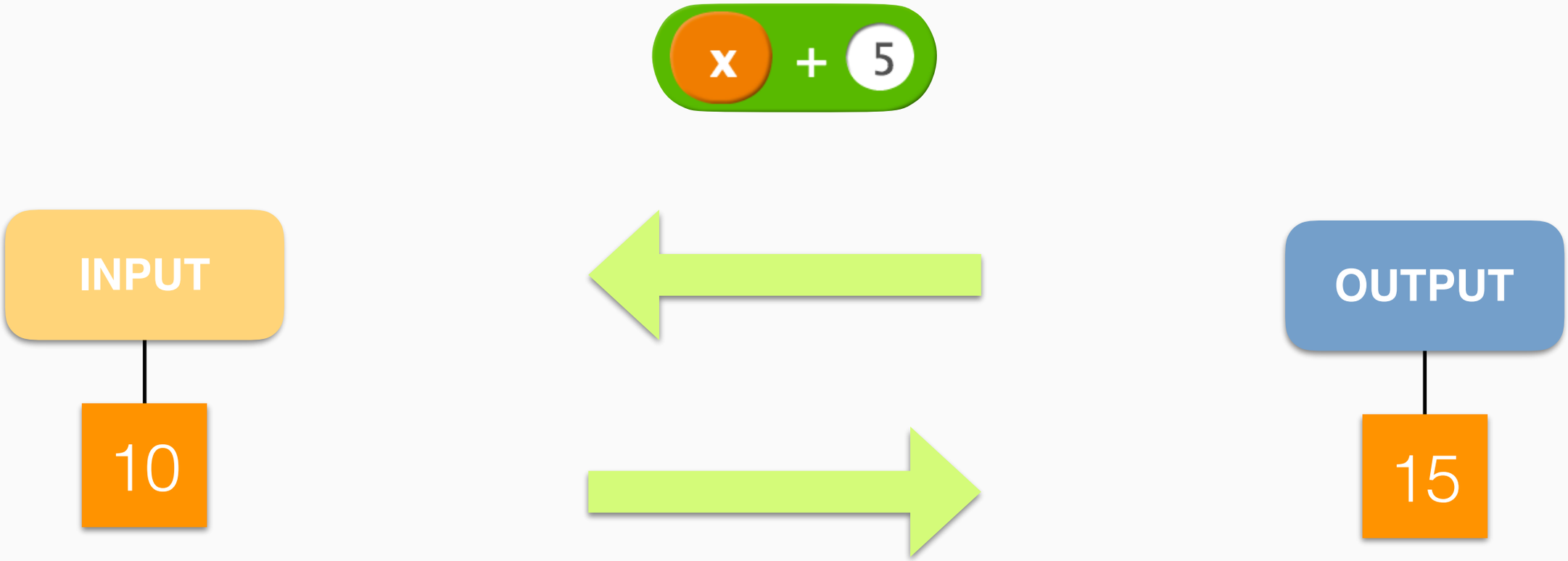


Direction One: Input to Output (*in SOLVE)*

Direction Two: Output to Input (???)

Direction & Functions

Addition Function:

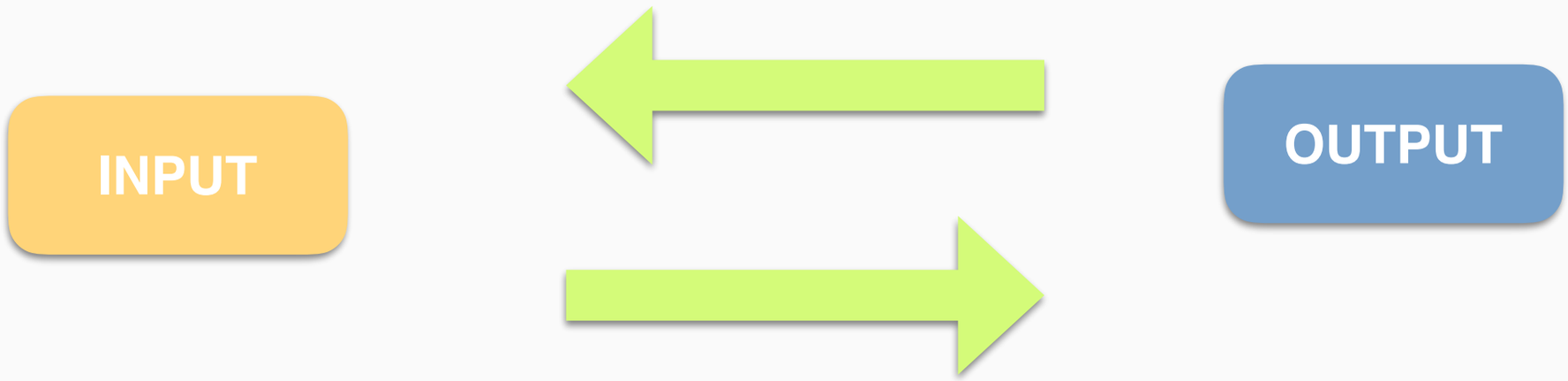


Direction One: Input to Output (*in SOLVE)*

Direction Two: Output to Input (*in SOLVE*)

Direction & Functions

Definition: A function is *one way* if going from Input to Output is in SOLVE, but going from Output to Input is **not in SOLVE**

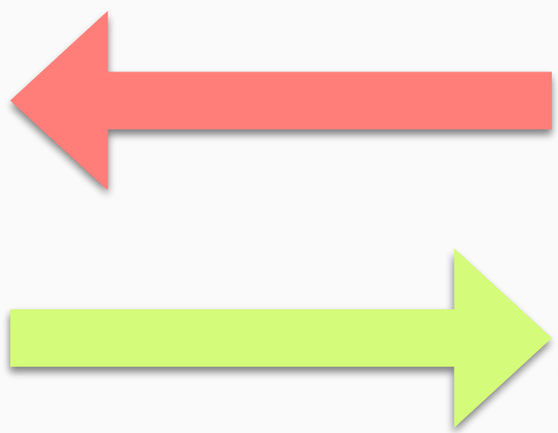


Direction One: Input to Output

Direction Two: Output to Input

Direction & Functions

Definition: A function is *one way* if going from **Input to Output is in SOLVE**, but going from **Output to Input is not in SOLVE**

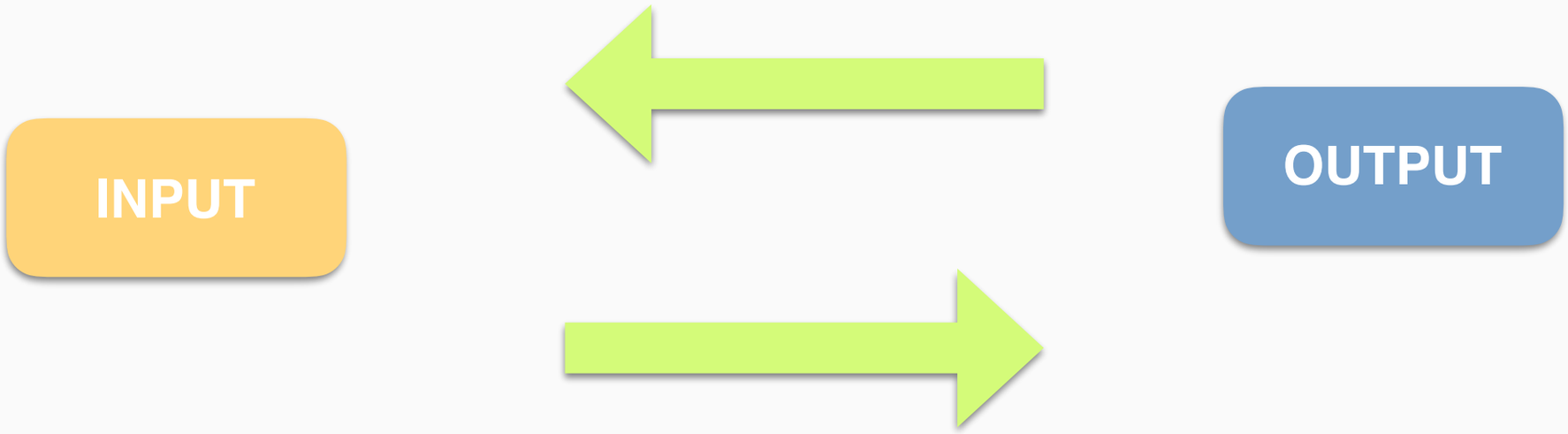


Direction One: Input to Output

Direction Two: Output to Input

Direction & Functions

So, the addition function is *not* a One Way Function, since going from output to input is in SOLVE.



Direction One: Input to Output

Direction Two: Output to Input

Prime ~~Rib~~ Numbers

- **Definition:** a *prime* number is one that is only divisible by itself and 1.
- Examples: 2, 3, 5, 7, 11, 13, 17, …
- Every natural number (1,2,3,4…) has what is called a *prime factorization*.
- Every number can be broken into some primes multiplied by each other.
- Example: 24 = 2*2*2*3, since 24 = 8*3, and 8=2*2*2

Clicker Question

Q: What is the prime factorization of 18?

[A] 2*3*3

[B] 3*6

[C] 2*9*1

[D] 1*18

[E] I’m confused

Clicker Answer

Q: What is the prime factorization of 18?

[A] 2*3*3

[B] 3*6

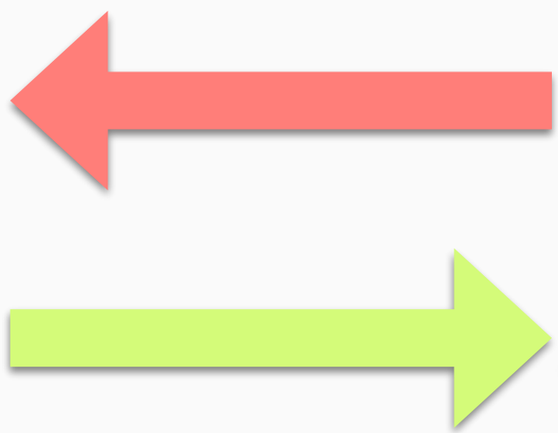
[C] 2*9*1

[D] 1*18

[E] I’m confused

One Way Functions

Definition: A function is *one way* if going from **Input to Output is in SOLVE**, but going from **Output to Input is not in SOLVE**

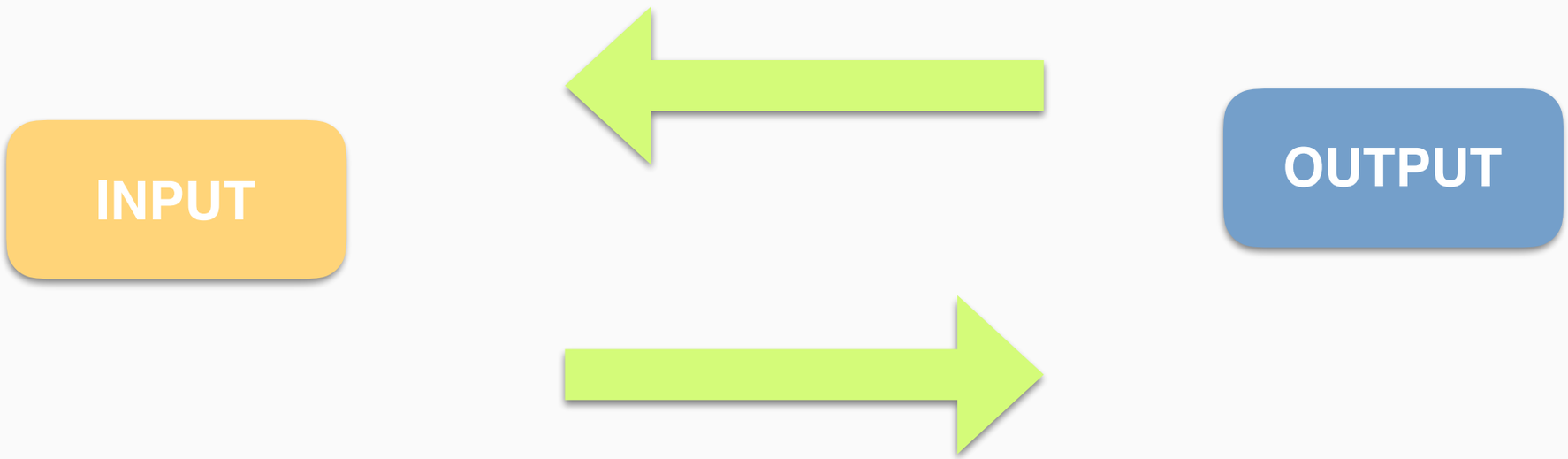


Direction One: Input to Output

Direction Two: Output to Input

One Way Functions

Example: multiplying some prime numbers.

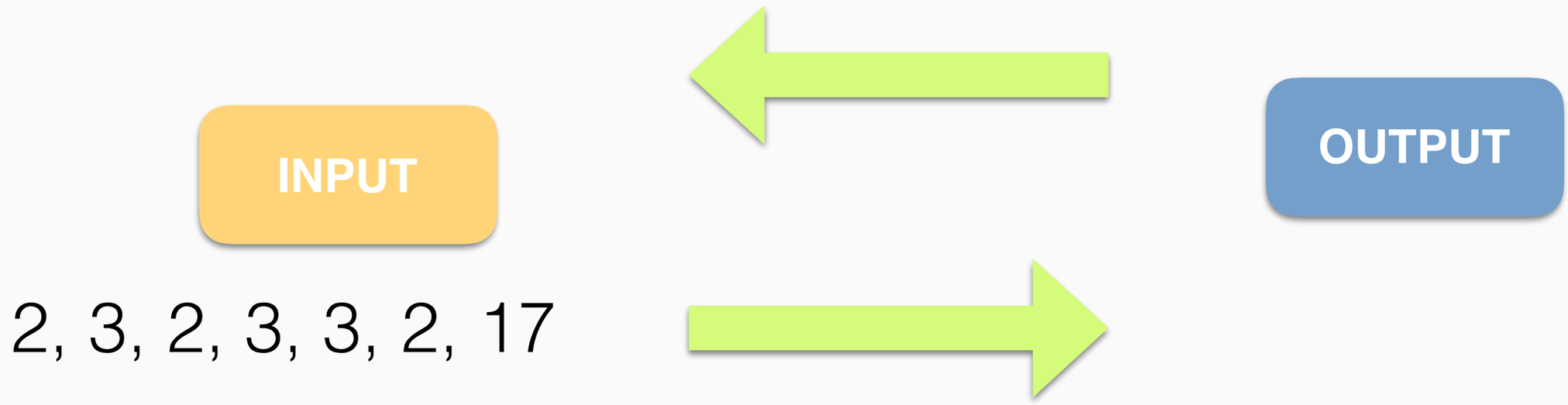


Direction One: Input to Output

Direction Two: Output to Input

Your First (byte of) One Way Functions

Example: multiplying some prime numbers.

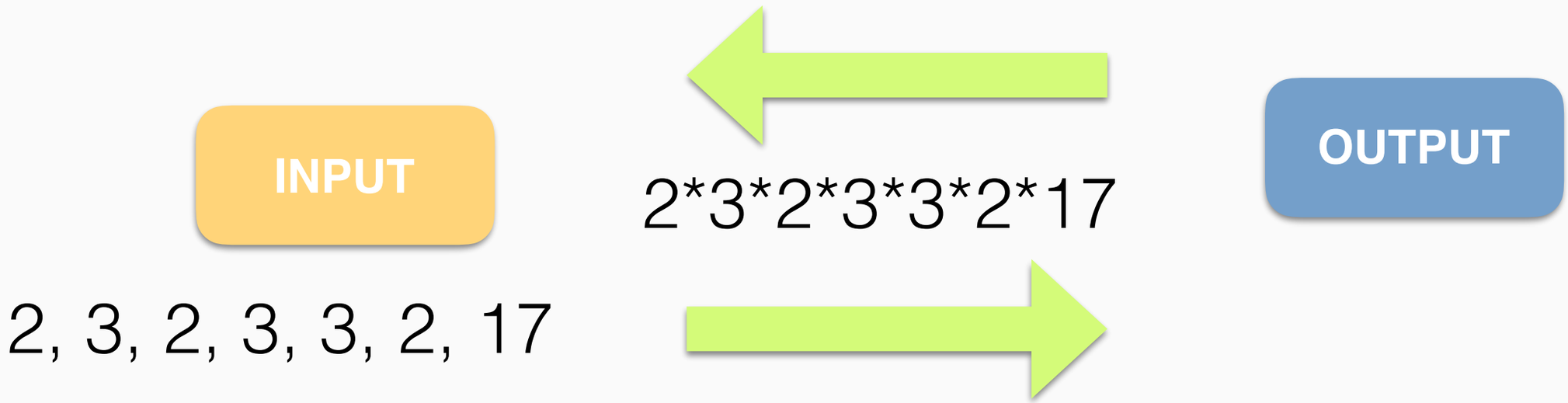


Direction One: Input to Output

Direction Two: Output to Input

Your First (byte of) One Way Functions

Example: multiplying some prime numbers.

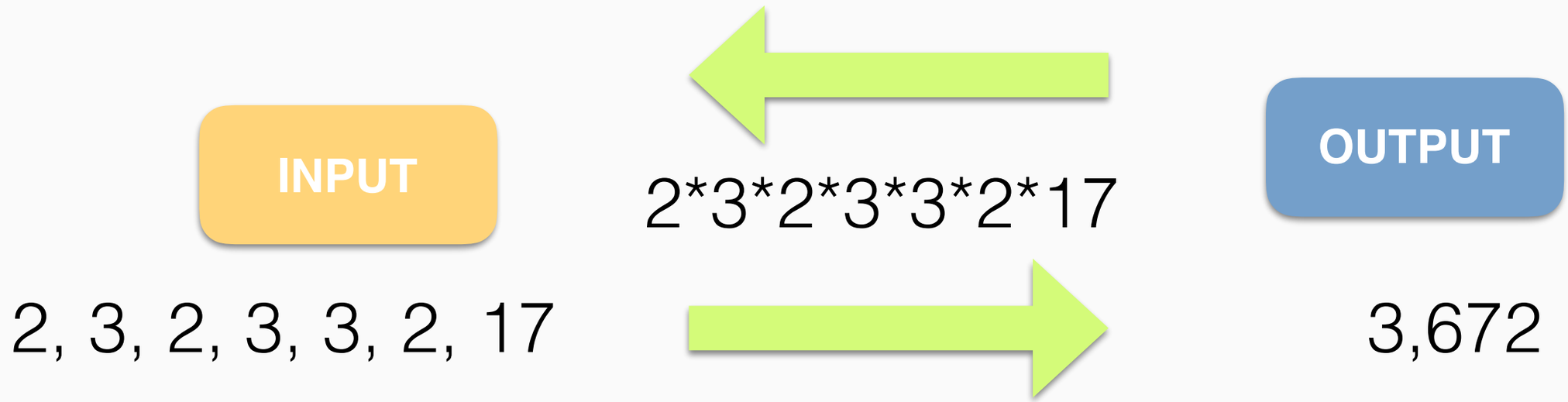


Direction One: Input to Output

Direction Two: Output to Input

Your First (byte of) One Way Functions

Example: multiplying some prime numbers.

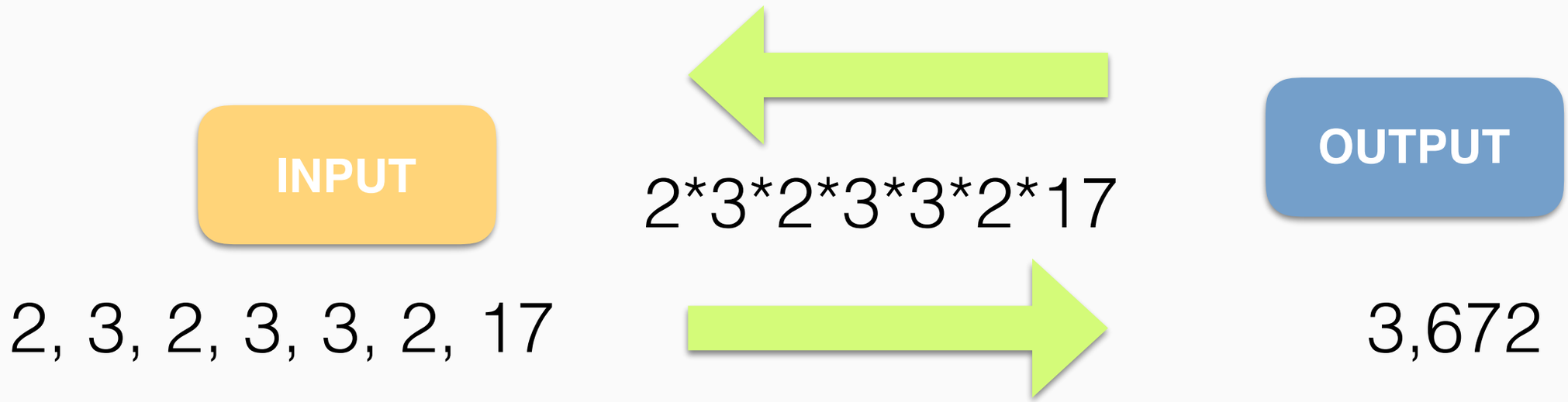


Direction One: Input to Output

Direction Two: Output to Input

Your First (byte of) One Way Functions

Example: multiplying some prime numbers.

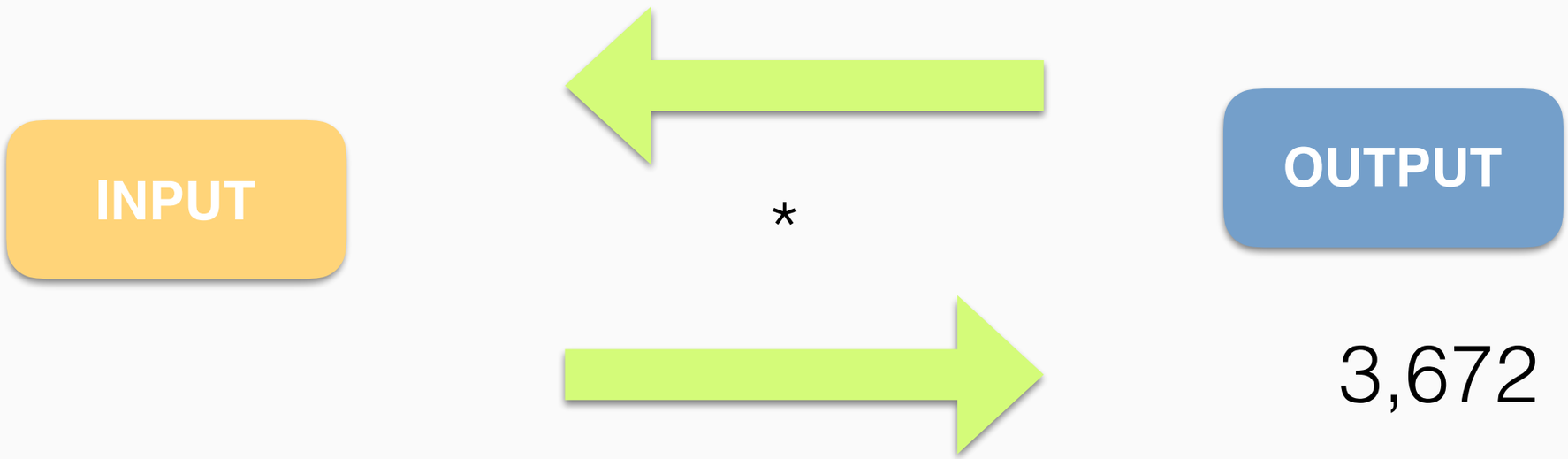


Direction One: Input to Output (*in SOLVE)*

Direction Two: Output to Input

Your First (byte of) One Way Functions

Example: multiplying some prime numbers.



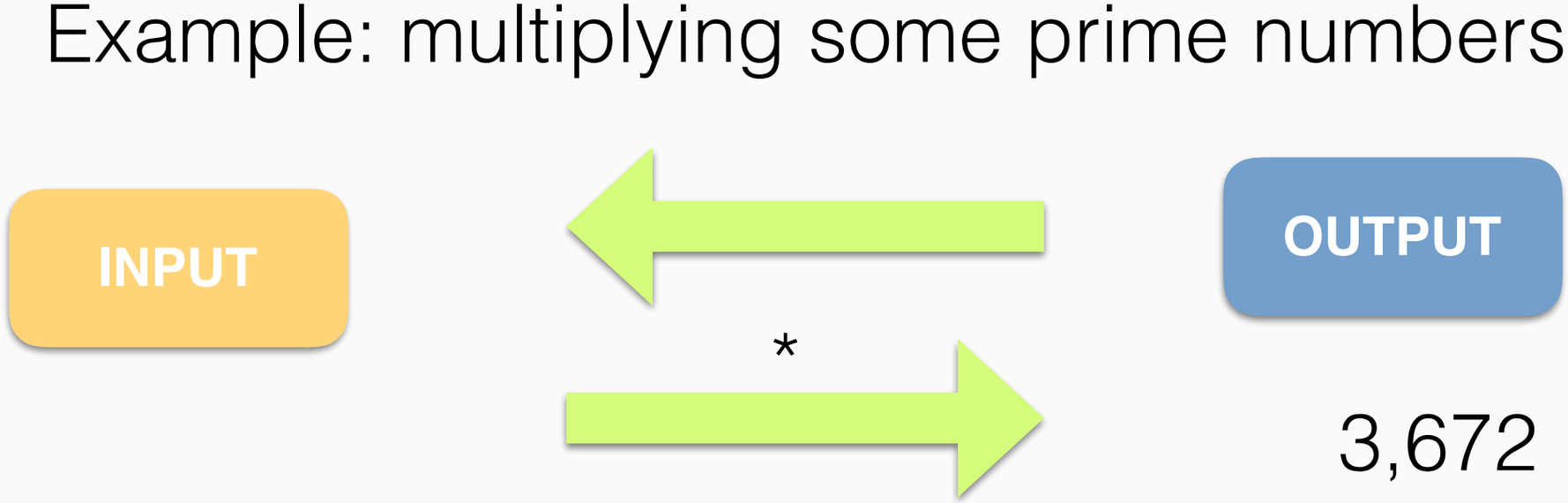
Direction One: Input to Output (*in SOLVE)*

Direction Two: Output to Input (*???*)

Your First (byte of) One Way Functions

Given a number, computing its prime factorization is **not in SOLVE**.

Example: multiplying some prime numbers.



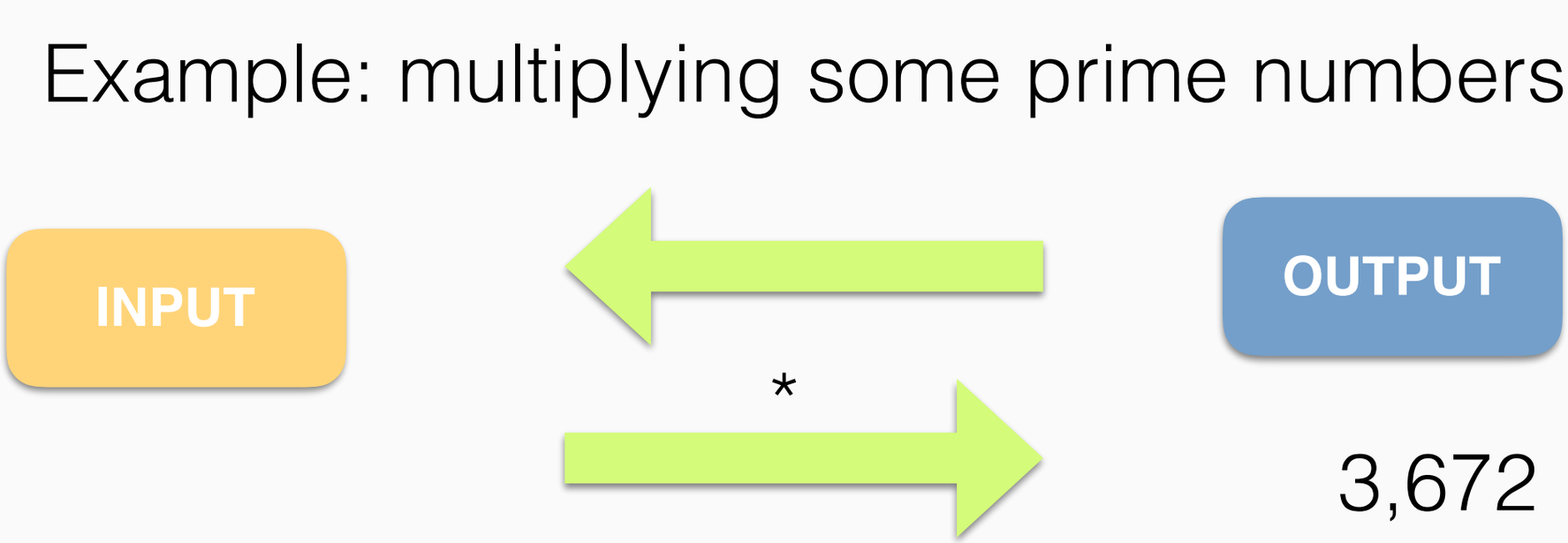
Direction One: Input to Output (*in SOLVE)*

Direction Two: Output to Input (*???*)

Your First (byte of) One Way Functions

Given a number, computing its prime factorization is **not in SOLVE**.

Example: multiplying some prime numbers.

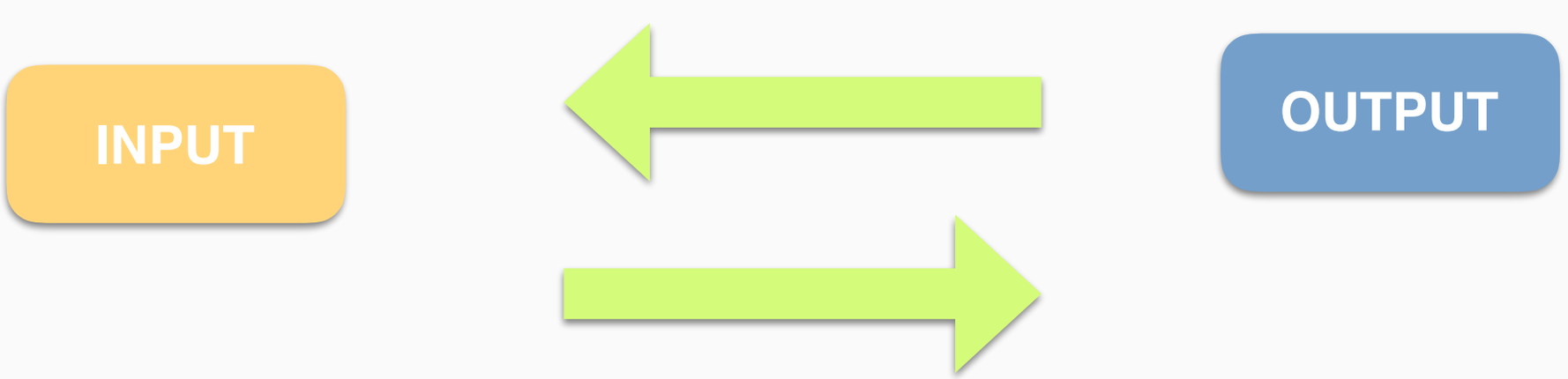


Direction One: Input to Output (*in SOLVE)*

Direction Two: Output to Input (*not in SOLVE*)

Multiplying Primes

Takeaway: Multiplying Primes is a One Way Function!



Direction One: Input to Output (*in SOLVE)*

Direction Two: Output to Input (*not in SOLVE*)

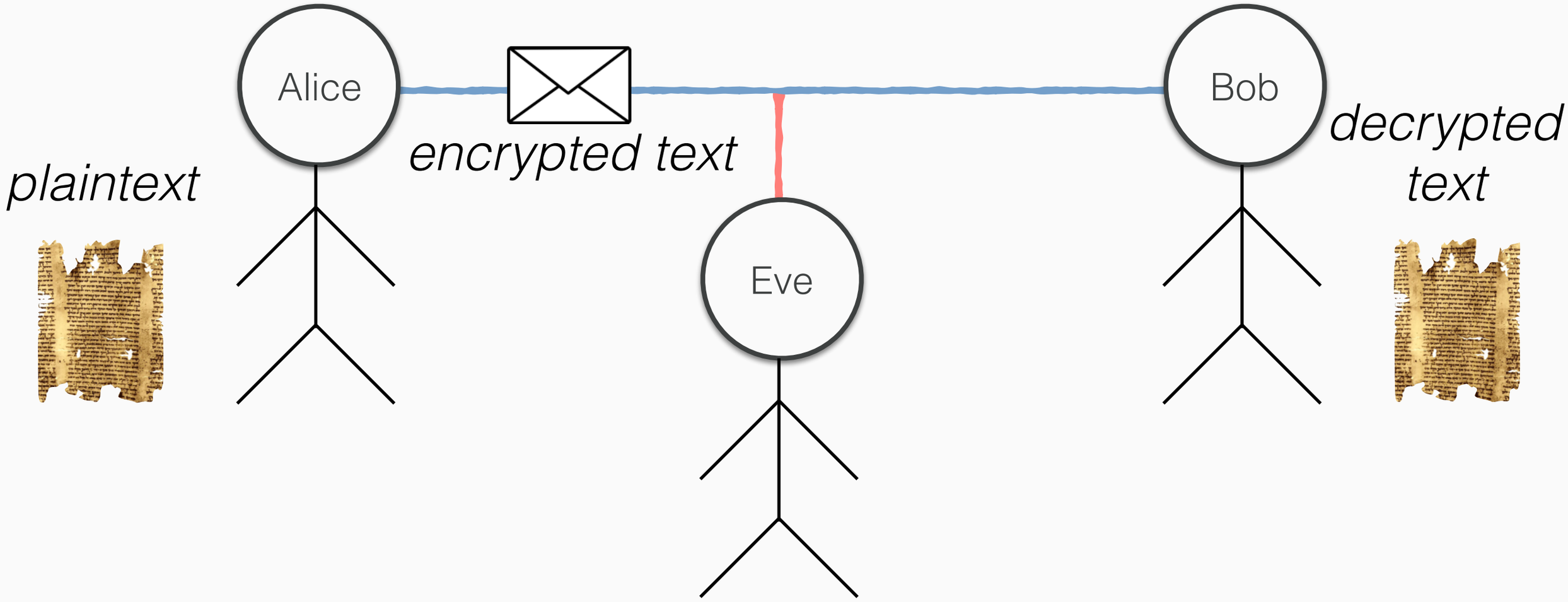
Why Should I Care?

- A OWF should sound *like* cryptography
  - If Eve sees the output, can she guess the input?
- All of modern cryptography relies on One Way Functions! (As we'll see).
- Limits of human understanding: we're actually *not sure* if they exist. We believe that prime factorization is not in SOLVE, but we're not sure yet.
- If OWFs exist, SOLVE ≠ VERIFY. So if we can prove OWFs exist…
- The real world seems to have One Way Functions…

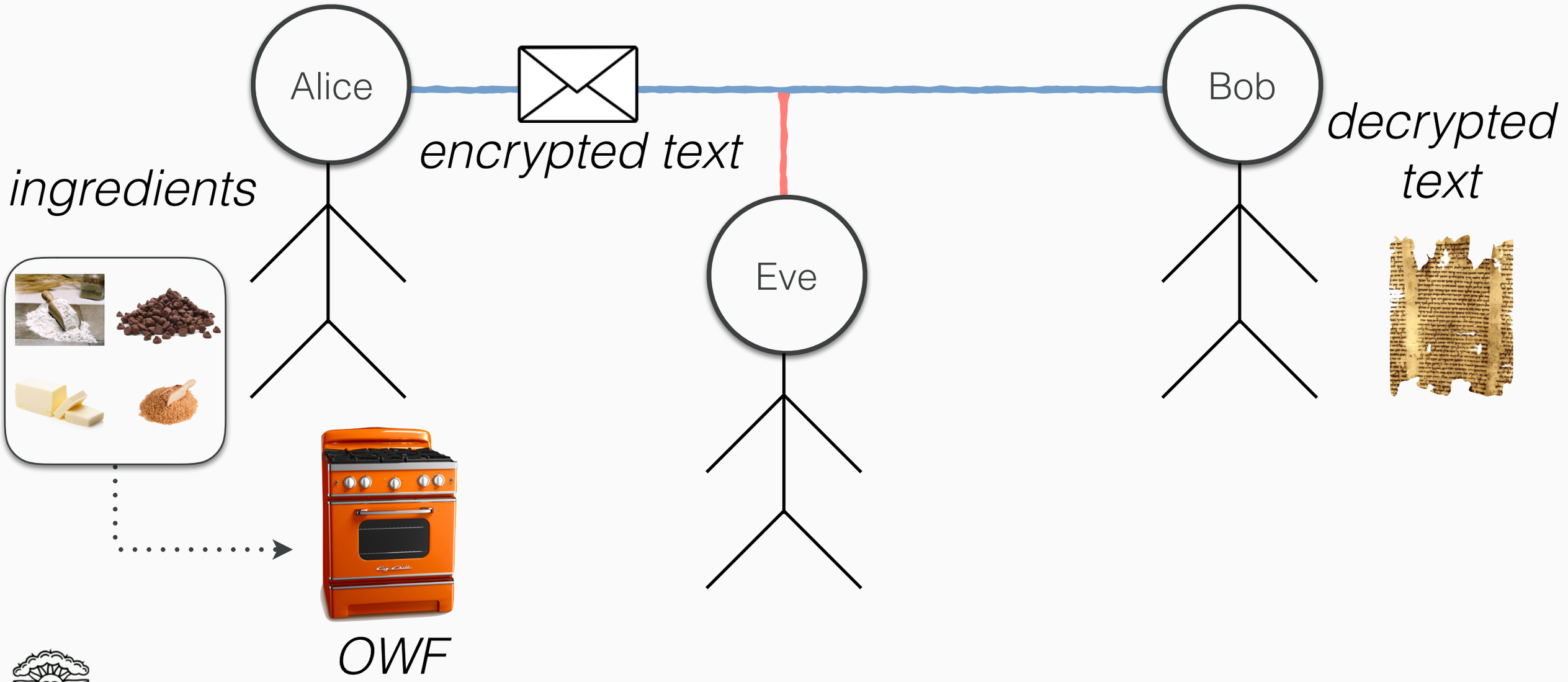
Why Should I Care?

- If SOLVE = VERIFY, then factoring primes *is* easy for computers to solve (since it's currently in VERIFY).
- That means that any cryptographic system that relies on prime factorization being a OWF could be broken!

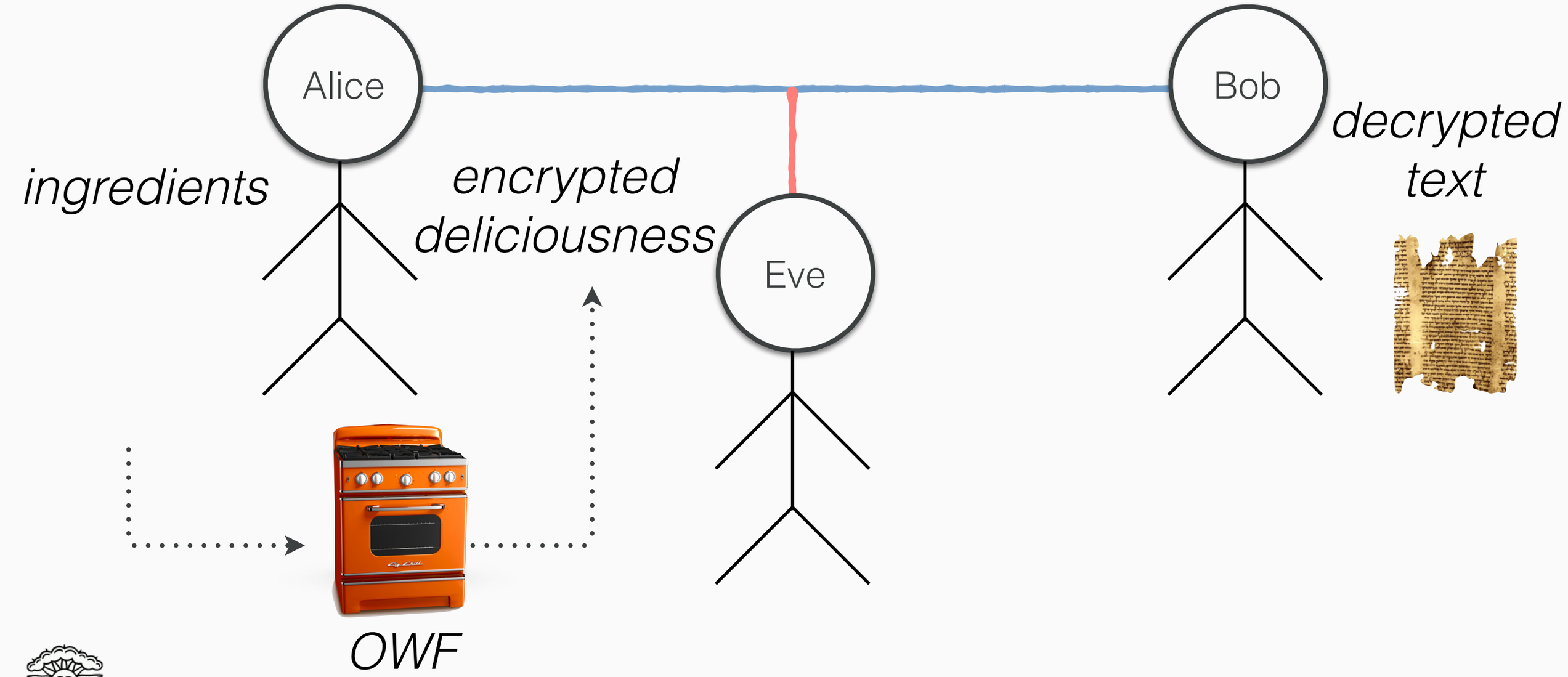
OWFs and Cryptography



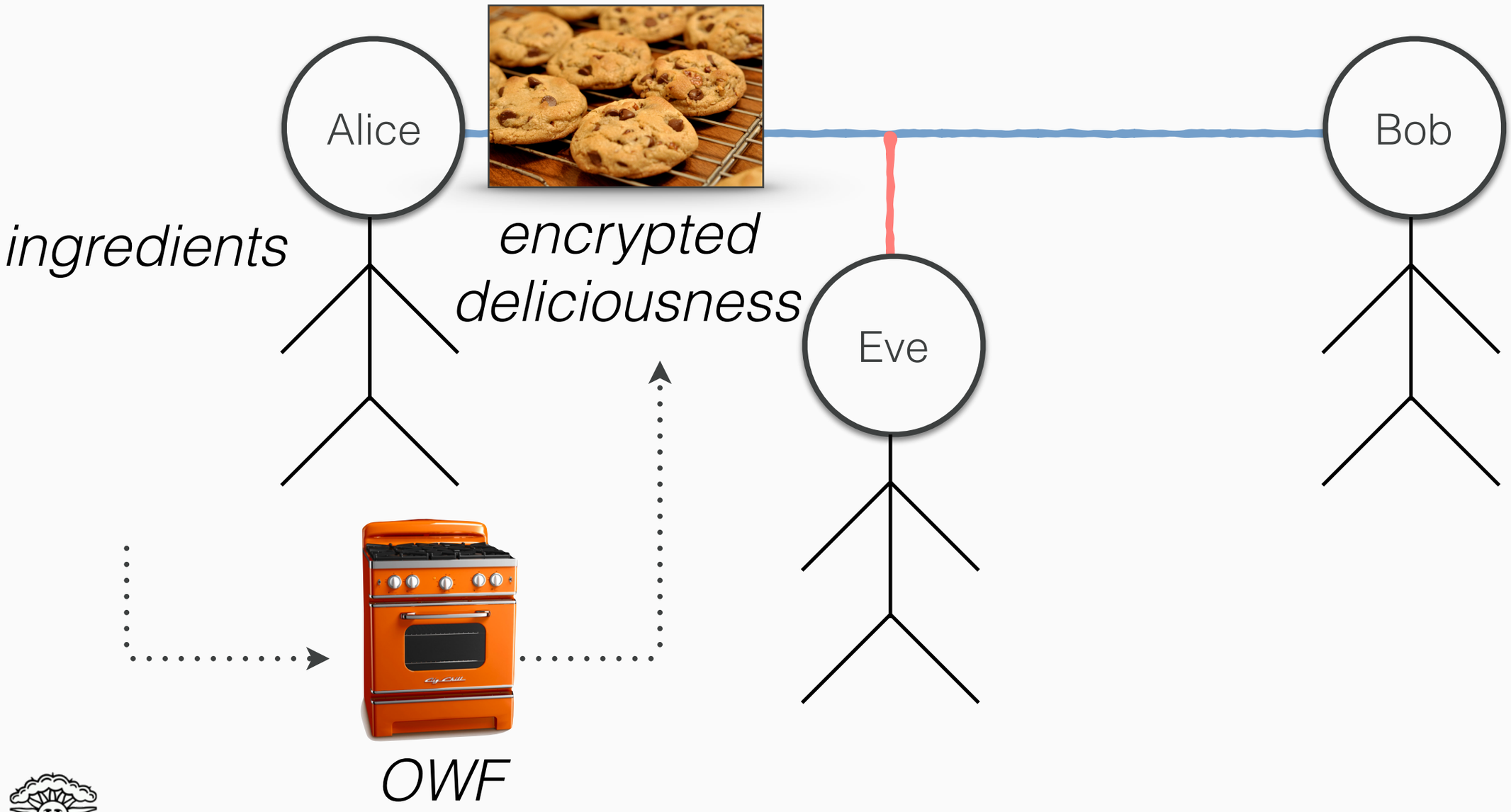
OWFs and Cryptography



OWFs and Cryptography



OWFs and Cryptography



OWFs and Cryptography

- Now we just need a way for Bob to decrypt it!
- Similar strategy as before: can Alice and Bob agree on some information in *advance* that will let Bob break it, but not Eee?
- For Caesar, this was the shift value.
- For Substitution, this was the shift alphabet.
- For a One Way Function, the details are a bit messy (like baking cookies).

OWFs and Cryptography

- For a One Way Function, the details are a bit messy (like baking cookies).

- In short, Alice and Bob *share a private key* that "breaks" the One Way Function.

OWFs and Cryptography

- For a One Way Function, the details are a bit messy (like baking cookies).

- In short, Alice and Bob *share a private key* that "breaks" the One Way Function.

The One Way Function is like a safe

OWFs and Cryptography

- For a One Way Function, the details are a bit messy (like baking cookies).

- In short, Alice and Bob *share a private key* that "breaks" the One Way Function.

(easy to put stuff in the safe, hard to take out)

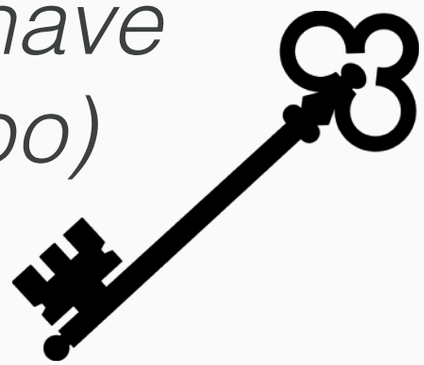
The One Way Function is like a safe

OWFs and Cryptography

- For a One Way Function, the details are a bit messy (like baking cookies).

- In short, Alice and Bob *share a private key* that "breaks" the One Way Function.

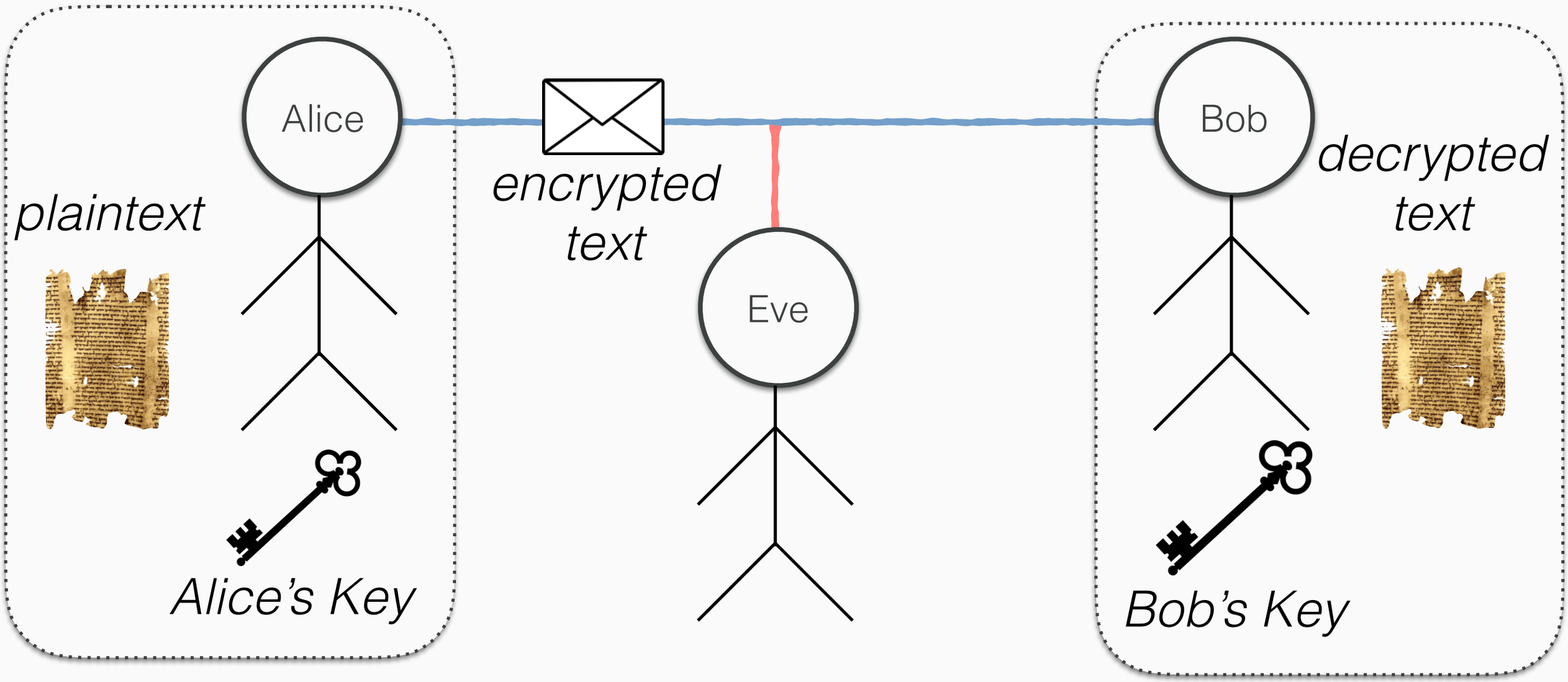
(unless you have a key/combo)



(easy to put stuff in the safe, hard to take out)

The One Way Function is like a safe

OWFs and Cryptography



Reflection: OWFs

- Functions have two directions:
  - Direction One: Input to Output
  - Direction Two: Output to Input

- **Definition:** A function is *one way* if going from Input to Output is in SOLVE, but going from Output to Input is not in SOLVE
- With certain information, called a *private key*, going from Output to Input is easy (i.e. in SOLVE).
- Therefore, decryption *requires* a private key!
- So Eve is out of luck (as long as OWFs exist…).

OWF Problem

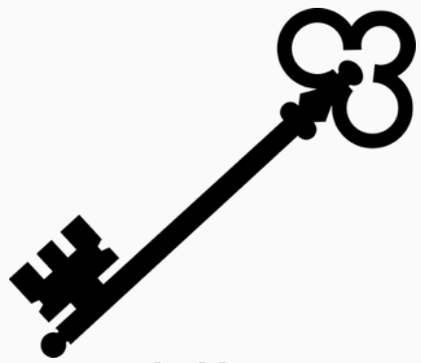
Any thoughts on what might be problematic here?

There's still something going on that makes this impractical.

OWF Problem

Any thoughts on what might be problematic here?

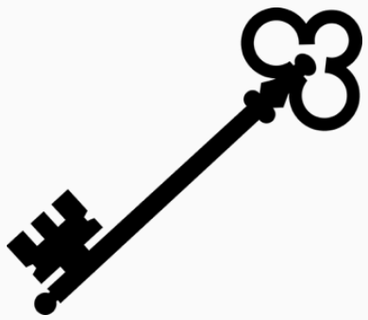
There's still something going on that makes this impractical.



Hint

OWF Problem

Any thoughts on what might be problematic here? There's still something going on that makes this impractical.



Hint

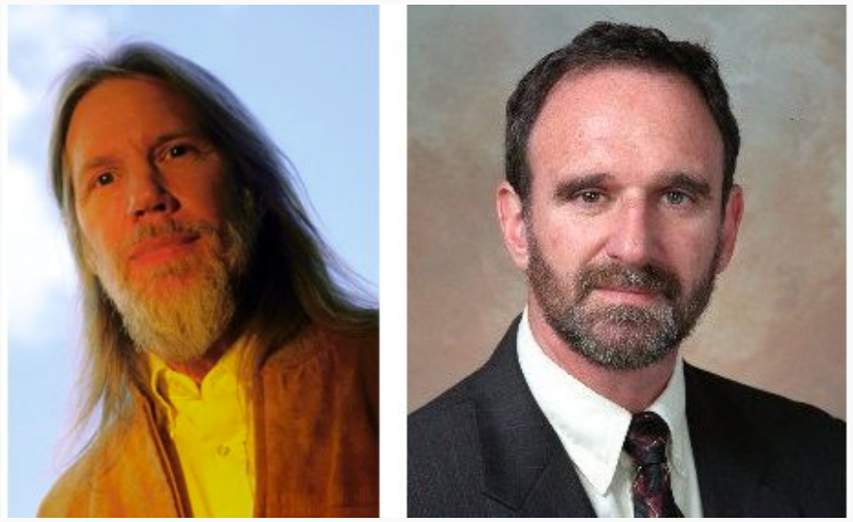
Alice and Bob still need to meet in advance to exchange secret keys.

Solution: Key Exchange

Alice and Bob still need to meet in advance to exchange secret keys.

Solution: Key Exchange

Alice and Bob still need to meet in advance to exchange secret keys.

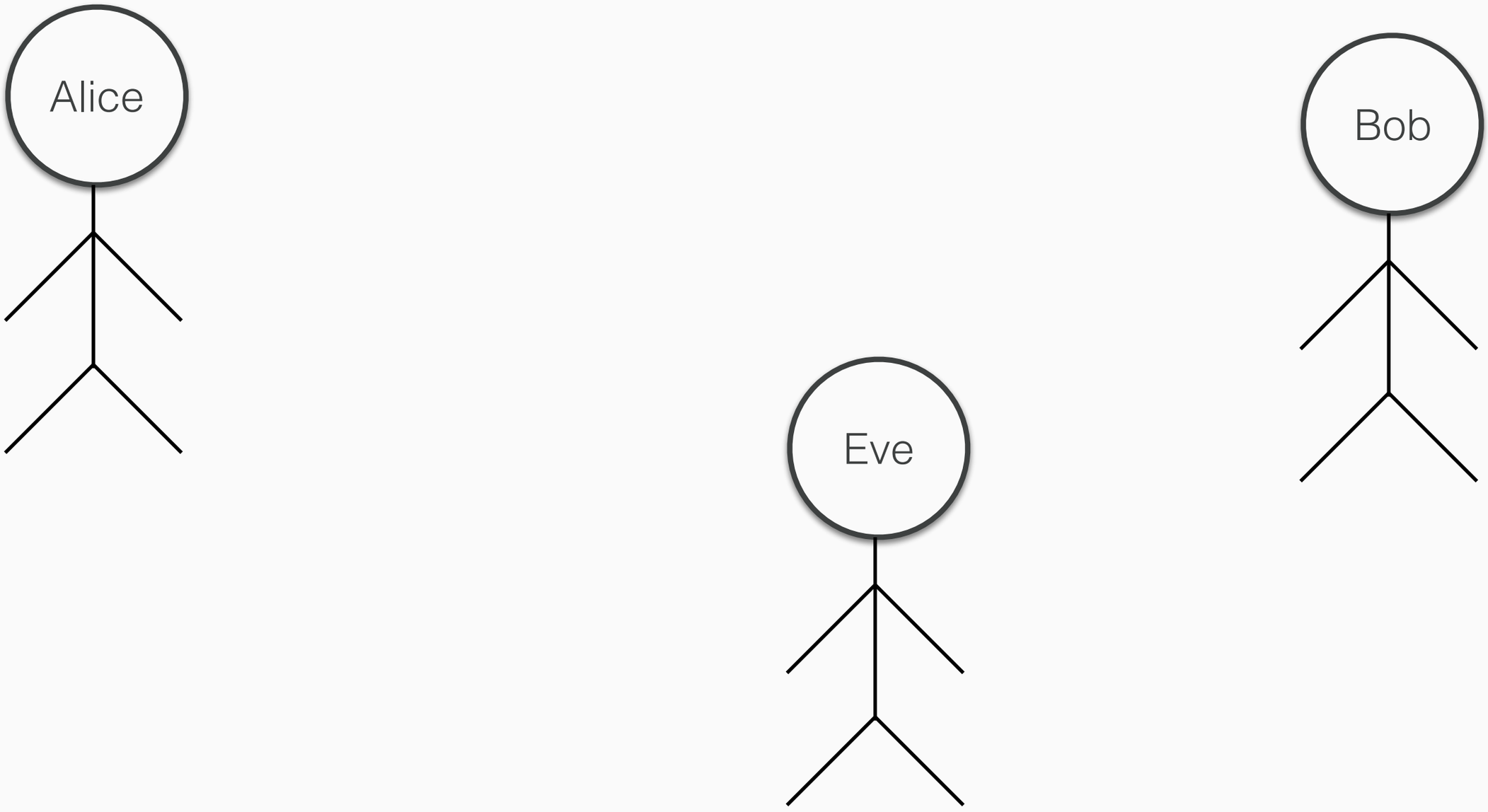
Diffie  Helman

https://upload.wikimedia.org/wikipedia/commons/8/88/Diffie_and_Hellman.jpg

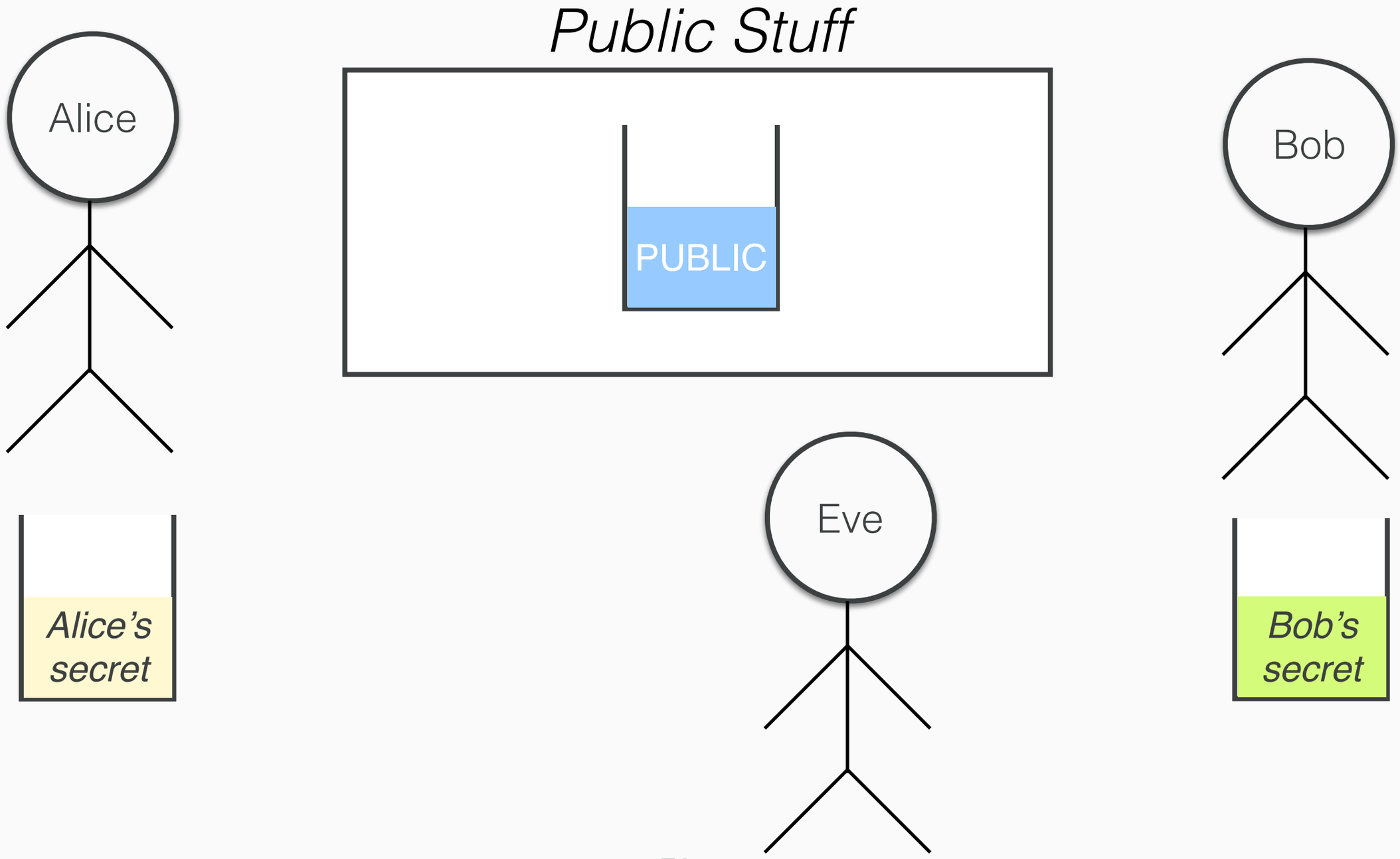
Diffie-Helman Key Exchange

Goal: We want Alice and Bob to both end up sharing a secret (key).

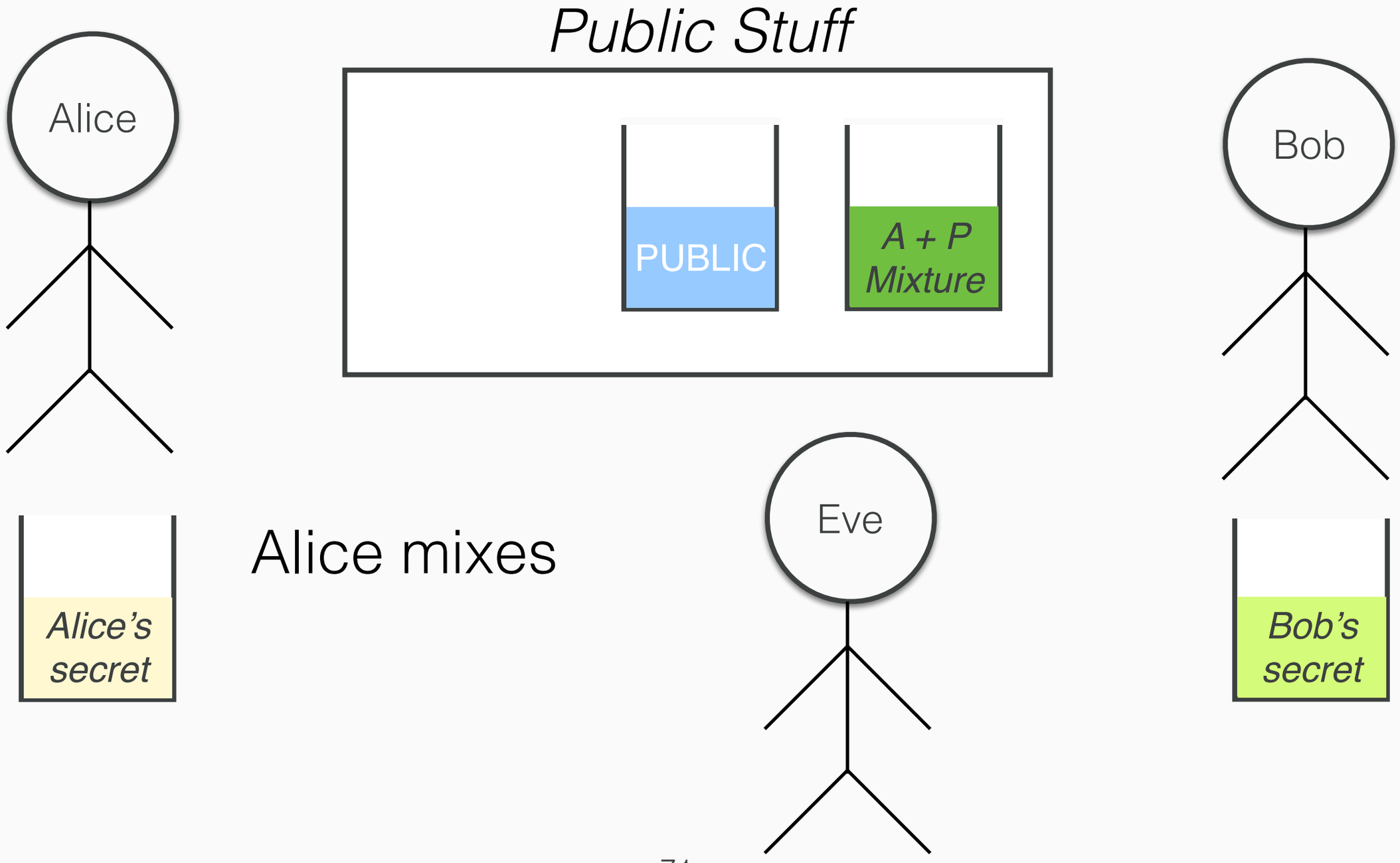
Diffie-Helman Key Exchange: Paint



Diffie-Helman Key Exchange: Paint

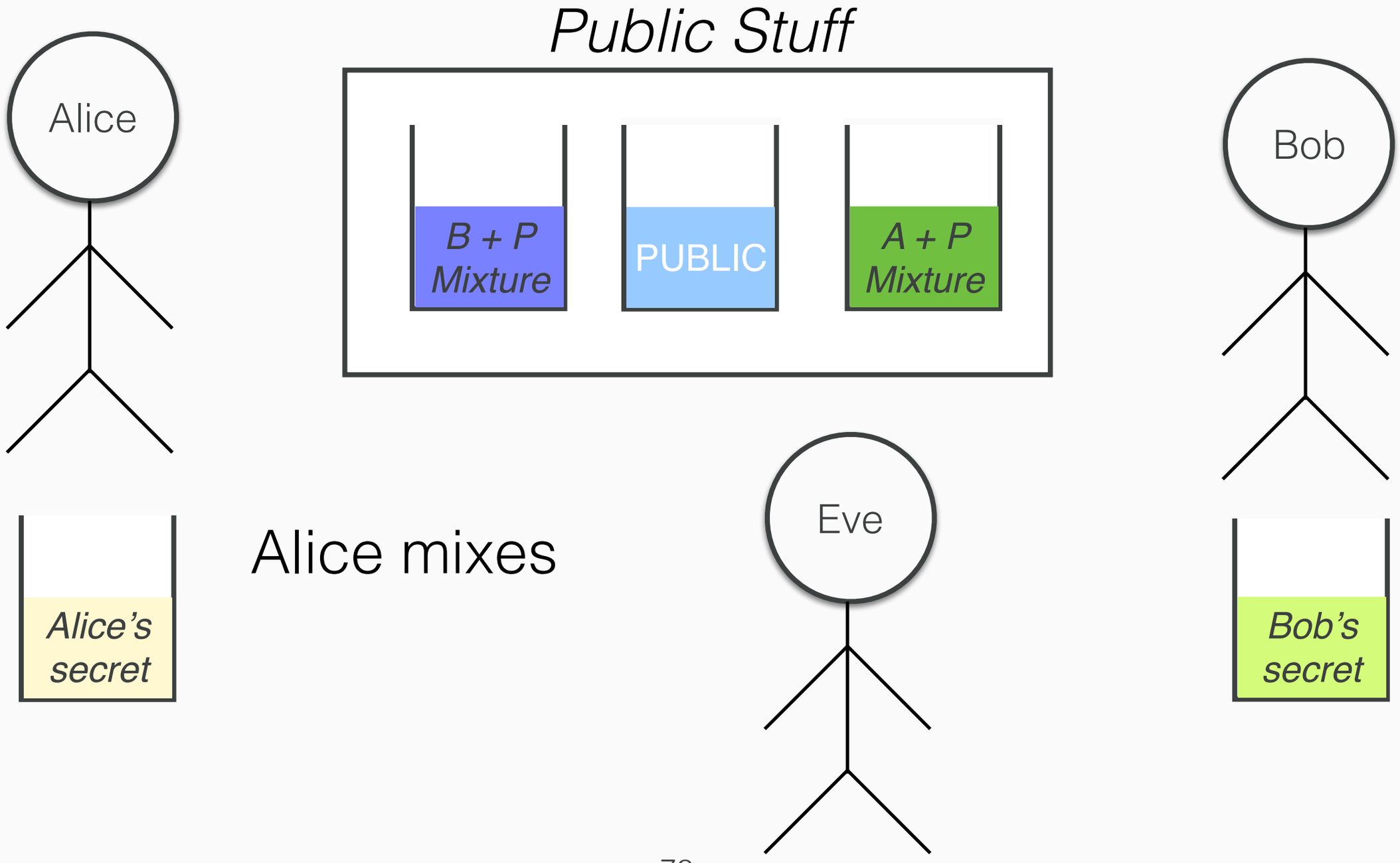


Diffie-Helman Key Exchange: Paint

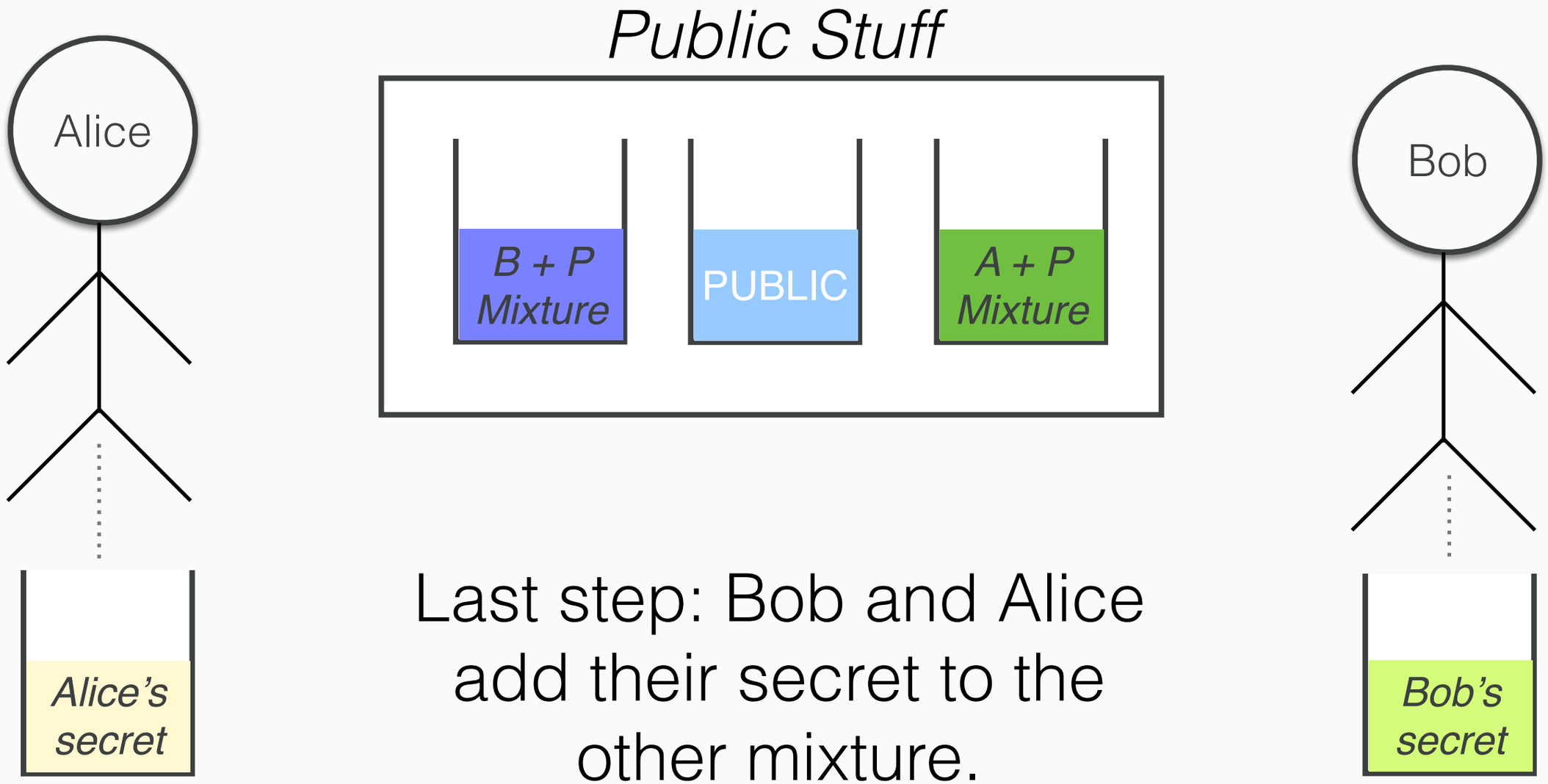


Alice mixes

Diffie-Helman Key Exchange: Paint

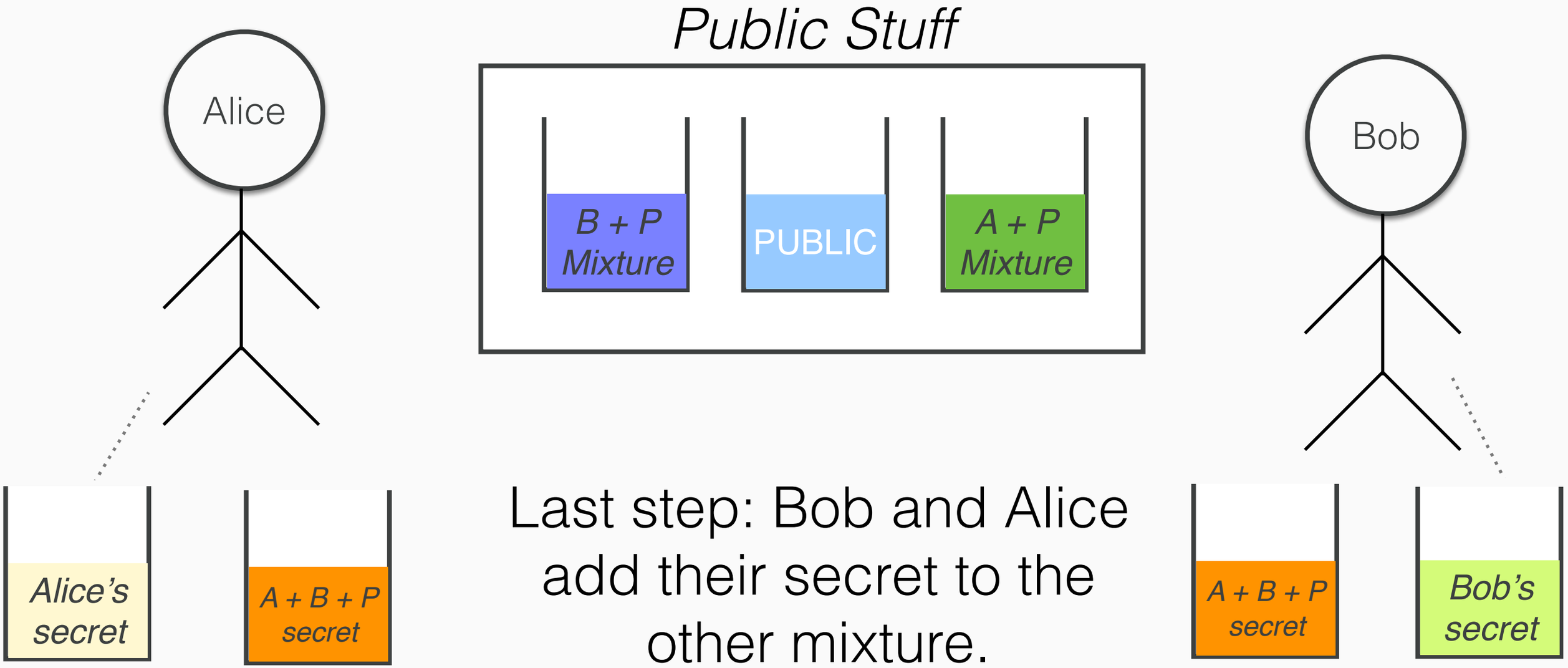


Diffie-Helman Key Exchange: Paint



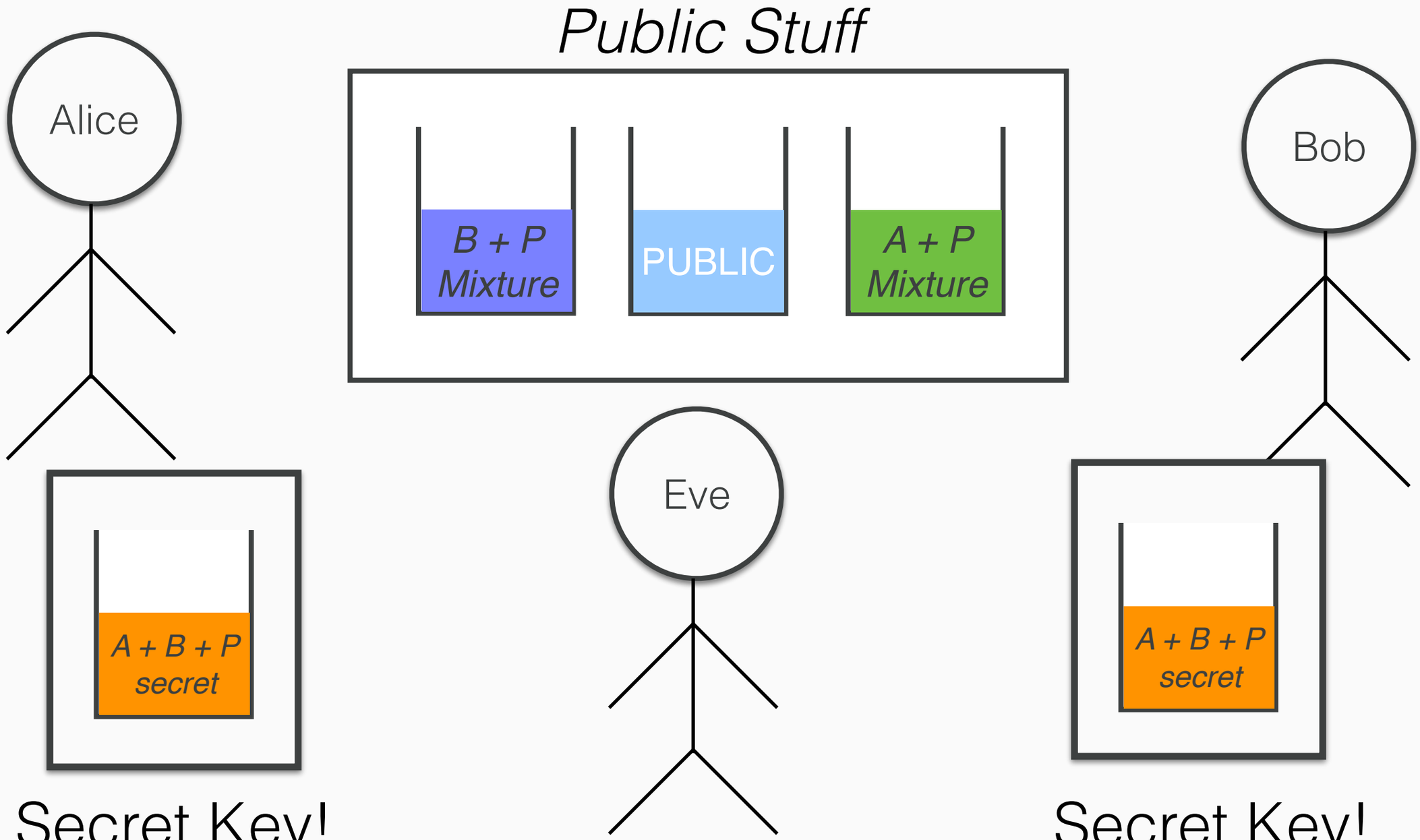
Last step: Bob and Alice add their secret to the other mixture.

Diffie-Helman Key Exchange: Paint



Last step: Bob and Alice add their secret to the other mixture.

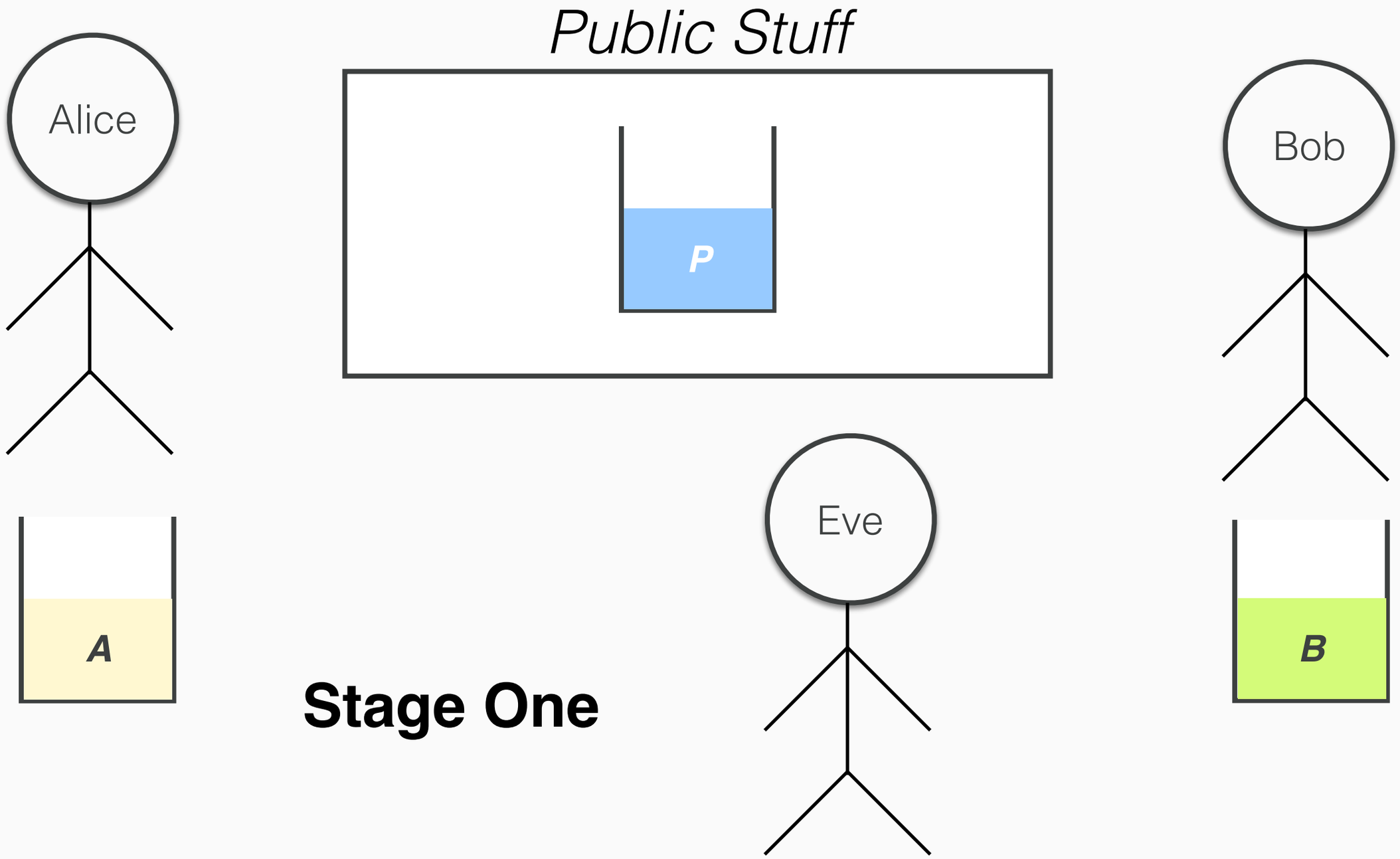
Diffie-Helman Key Exchange: Paint



Secret Key!

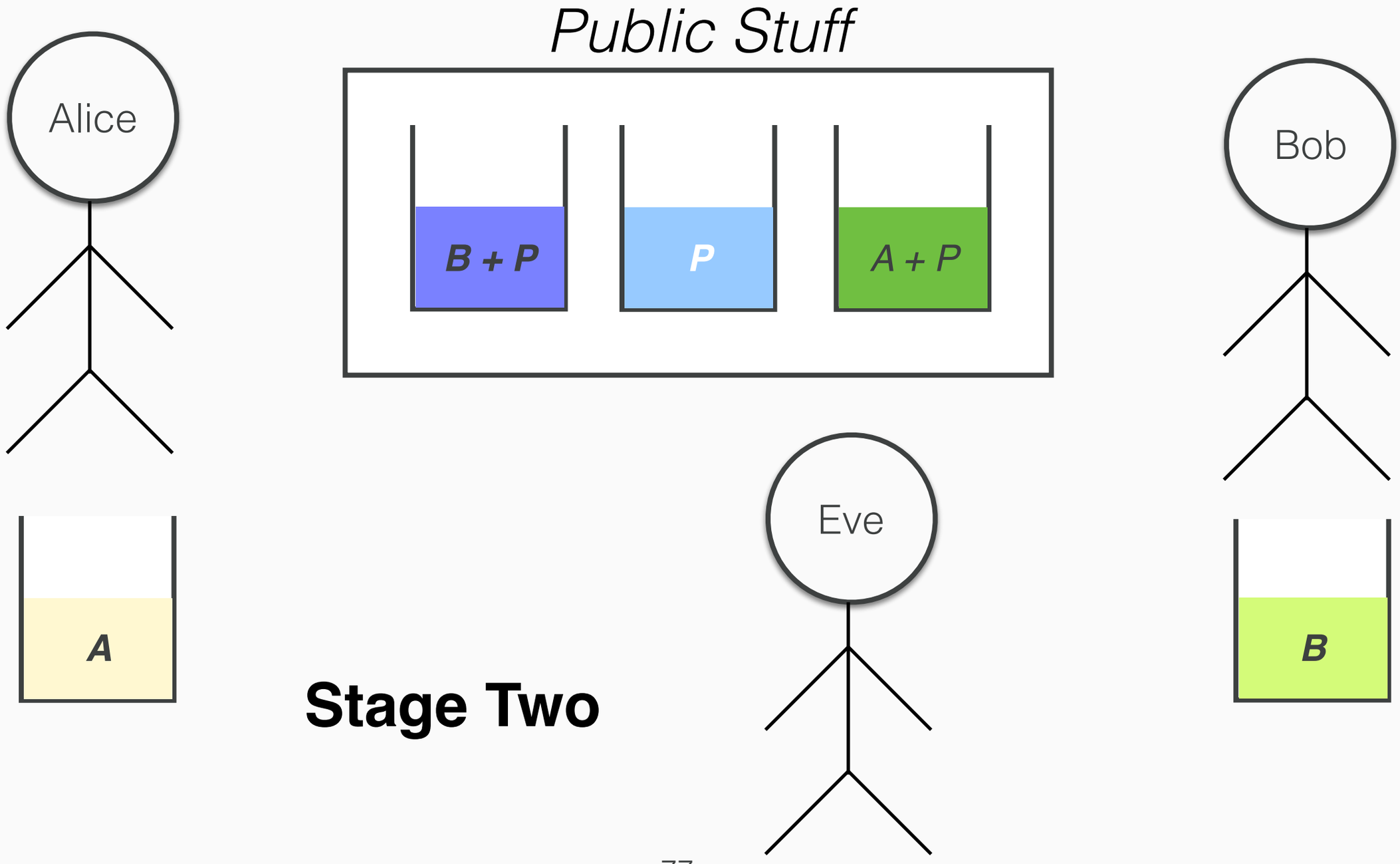
Secret Key!

Diffie-Helman Key Exchange



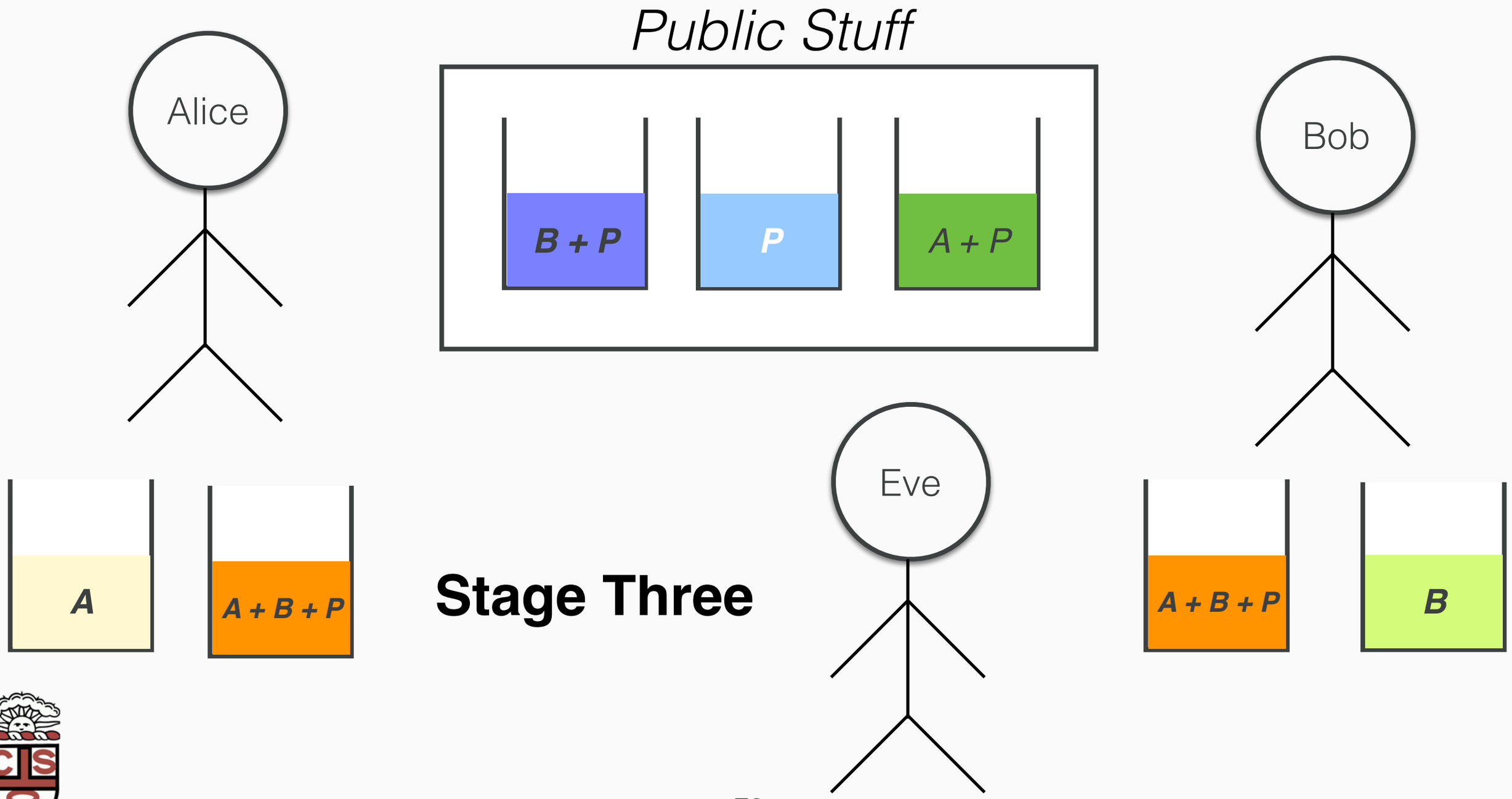
Stage One

Diffie-Helman Key Exchange



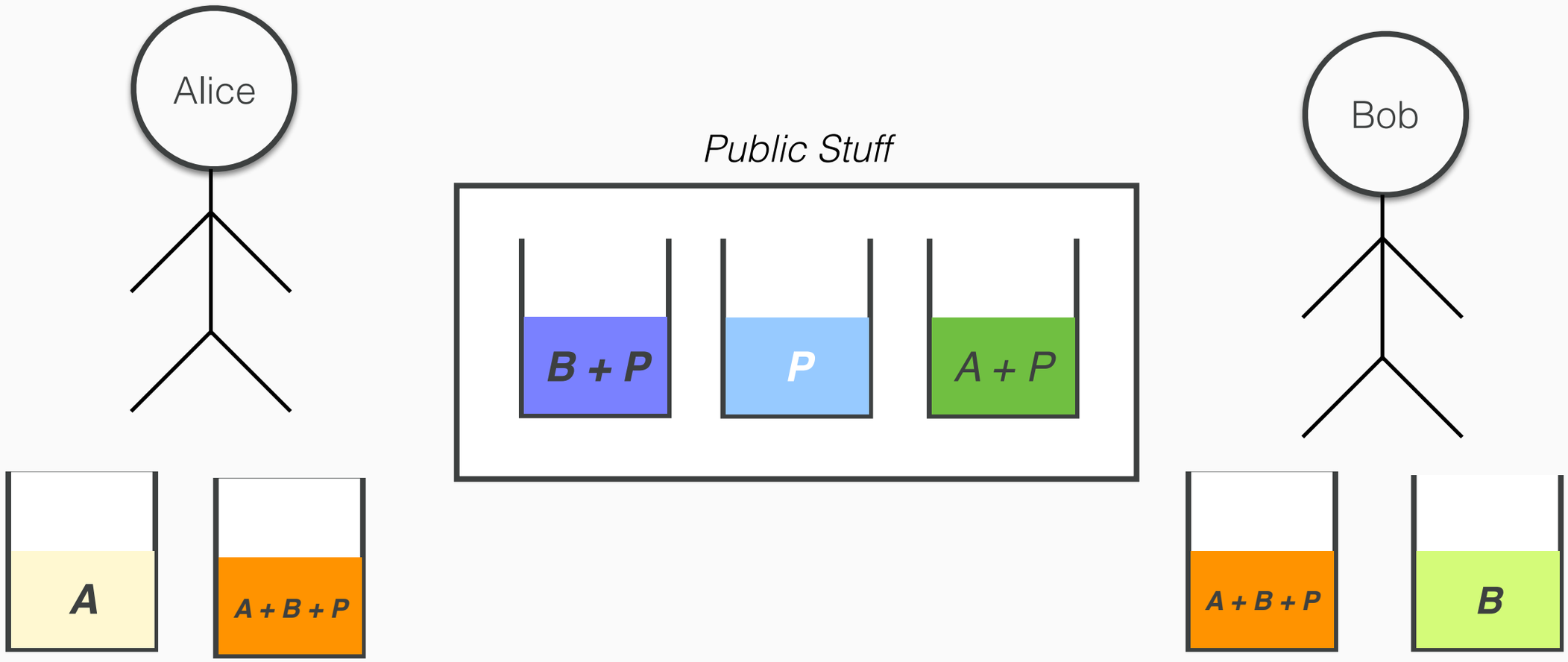
Stage Two

Diffie-Helman Key Exchange



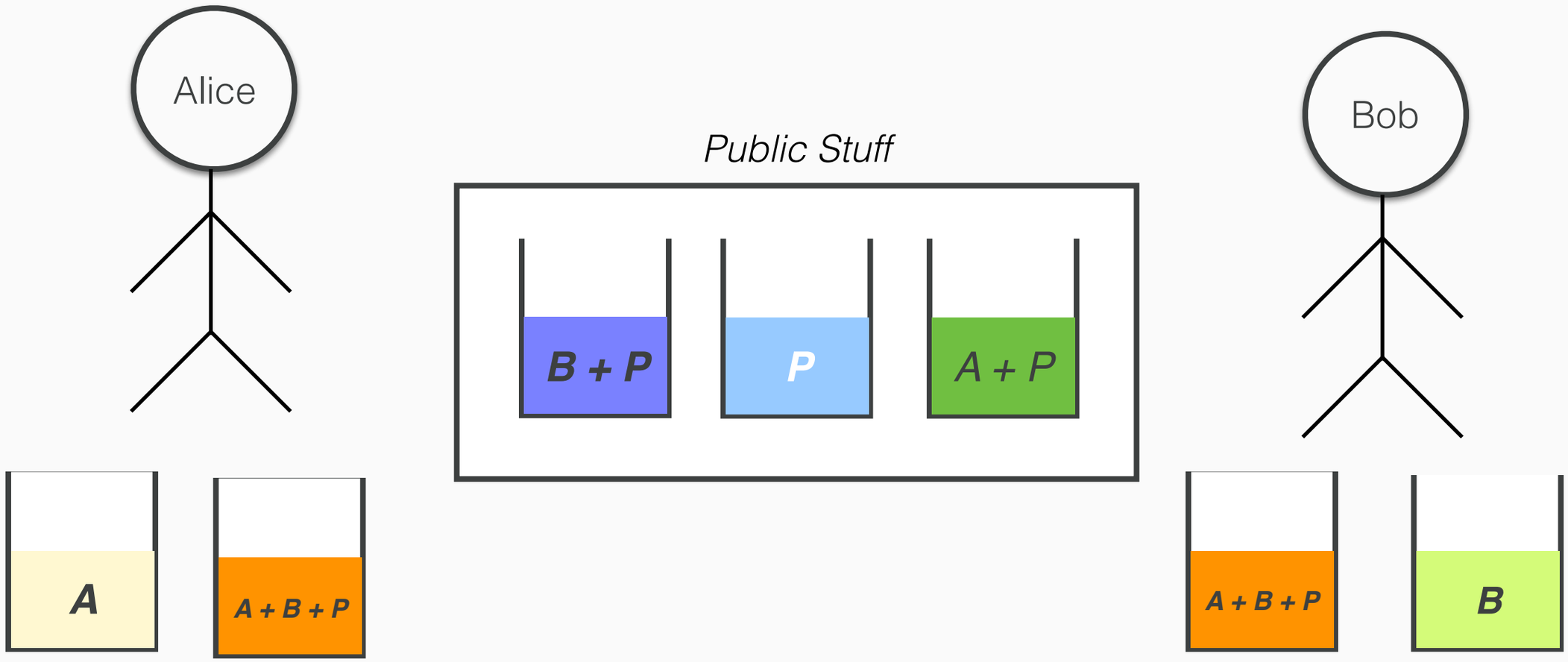
Stage Three

Question!



Q: Why do we use Diffie-Helman to exchange *keys* and not our message?

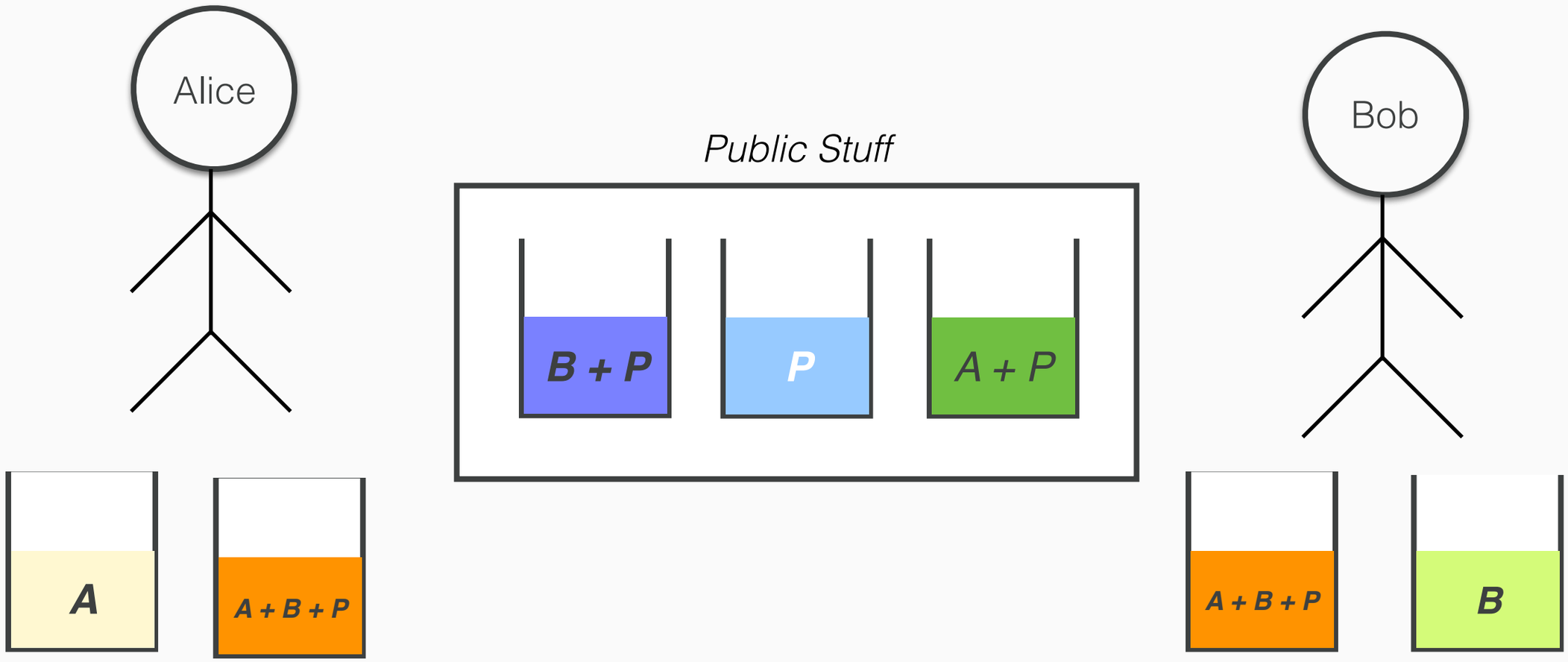
Question!



Q: Why do we use Diffie-Helman to exchange *keys* and not our message?

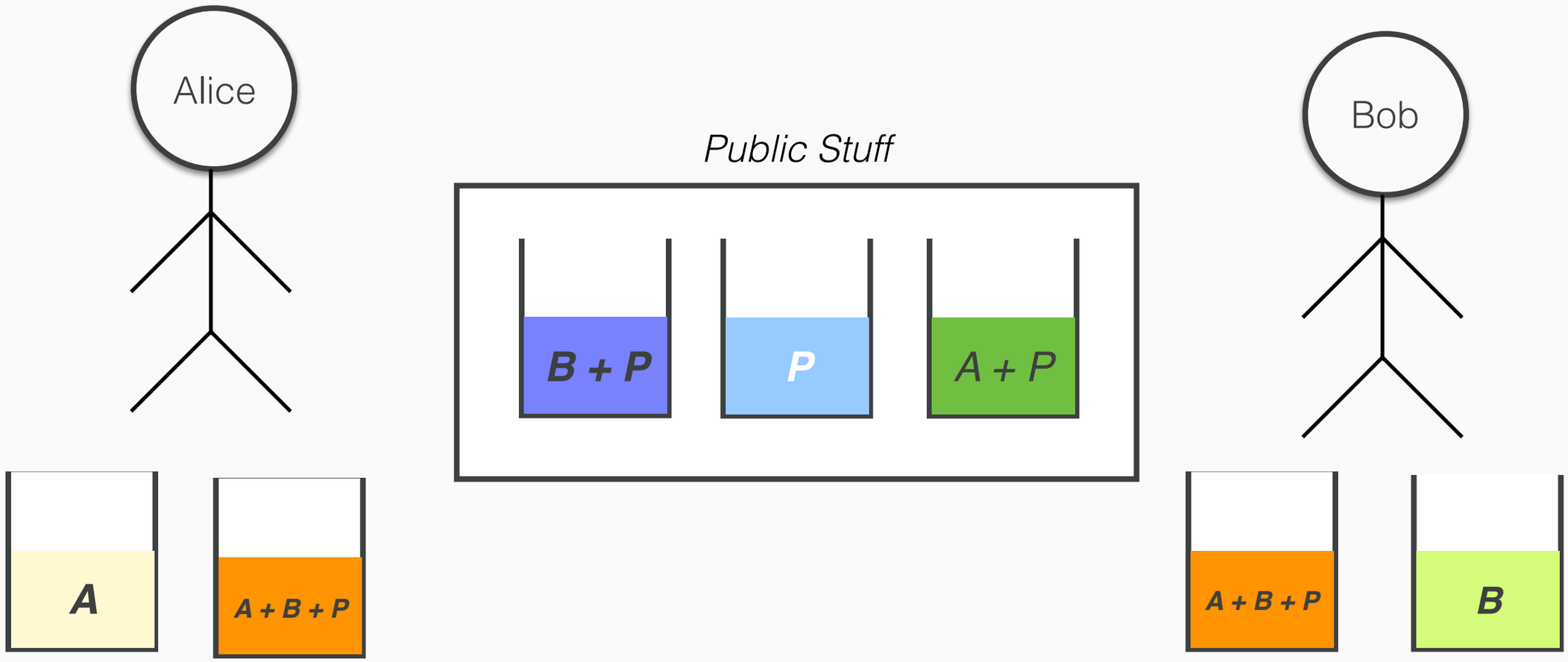
A: We don't get to control **the secret**!

Another Question!



Q: Why can’t Eve just make A + B + P herself?

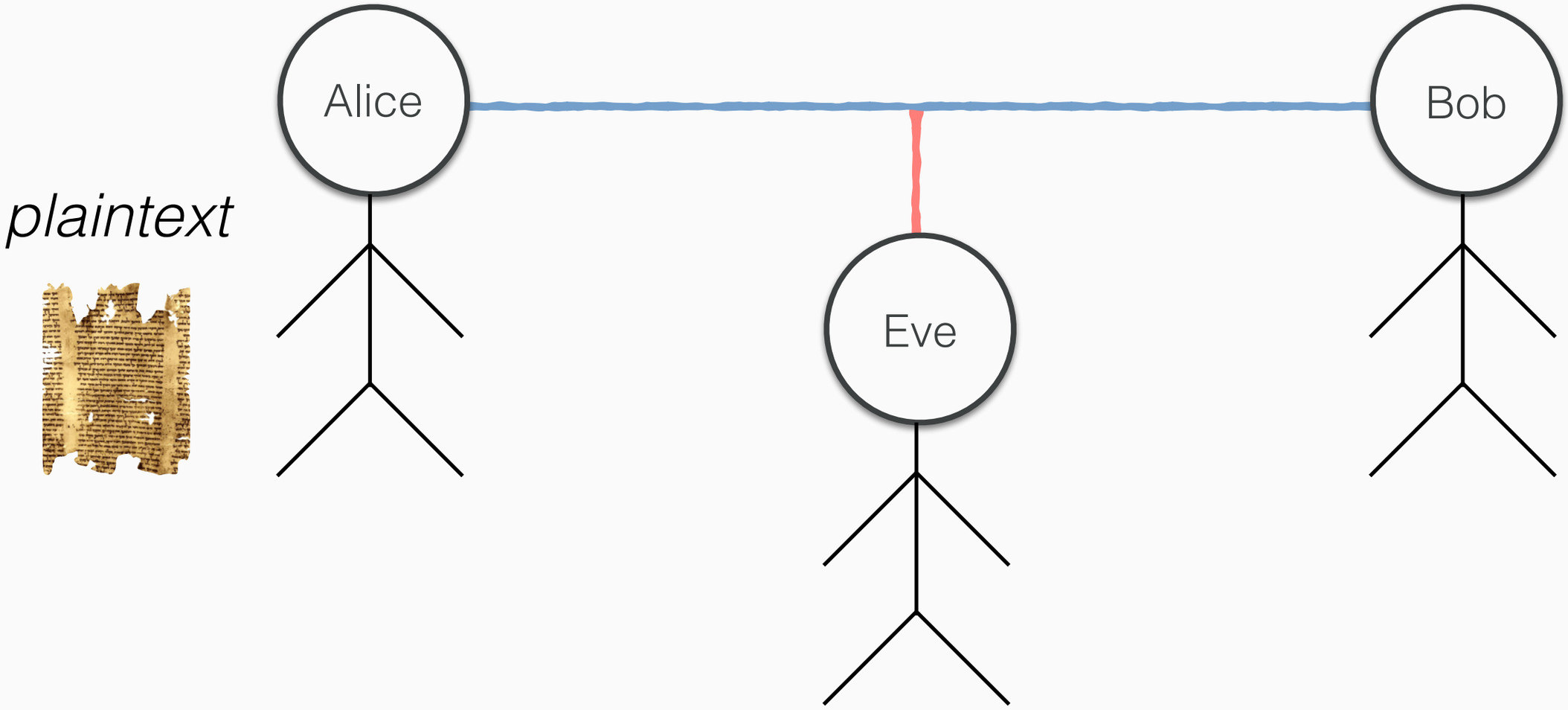
Another Question!



Q: Why can't Eve just make A + B + P herself?

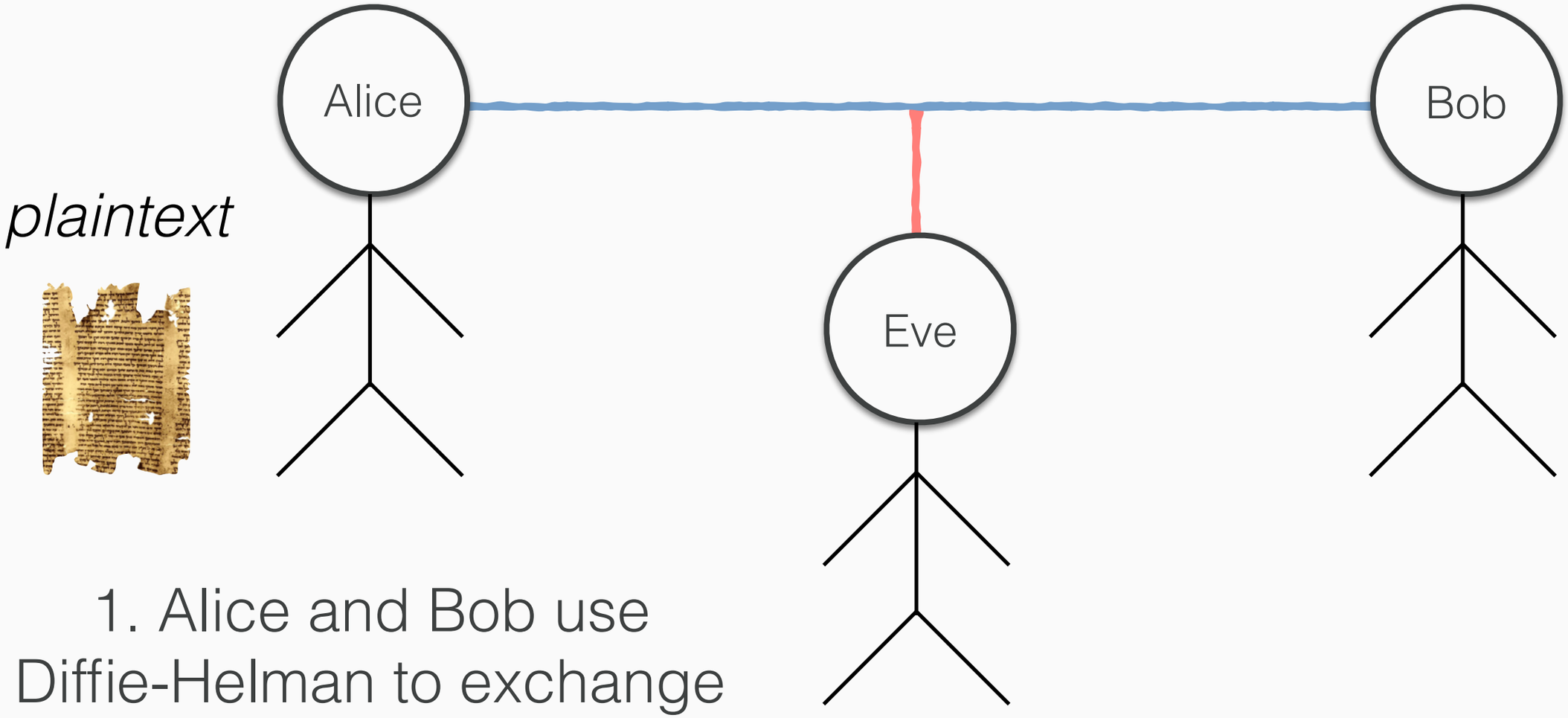
A: She can only make B + P + P, or A + P + P,
or A + B + P + P…

The Full Pipeline



plaintext

The Full Pipeline

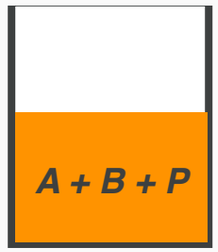


plaintext

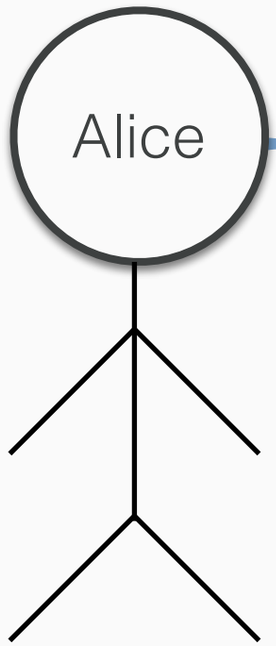
1. Alice and Bob use Diffie-Helman to exchange private keys

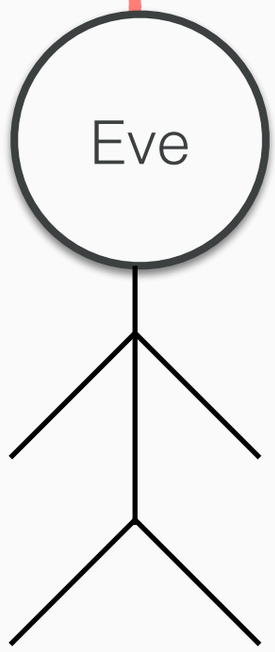
The Full Pipeline

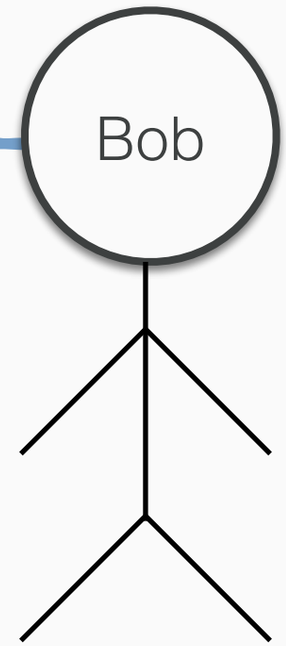
private key



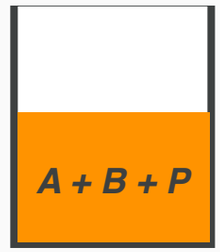
plaintext







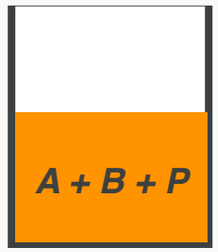
private key



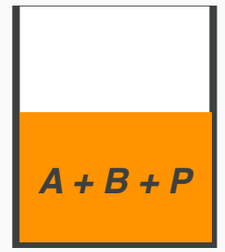
1. Alice and Bob use Diffie-Helman to exchange private keys

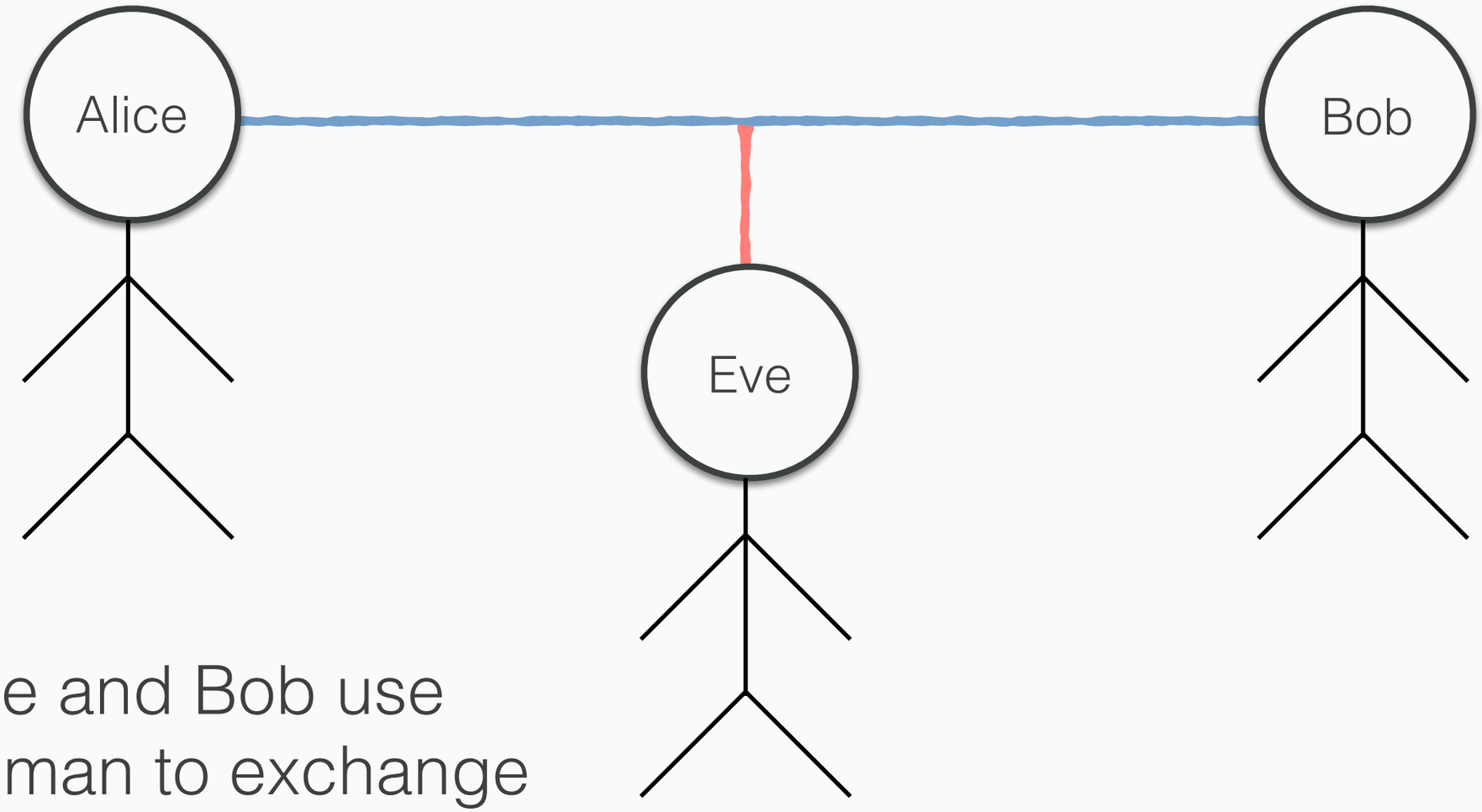
The Full Pipeline

private key



private key





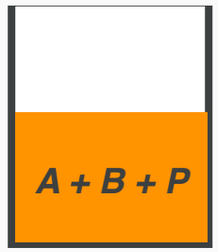
plaintext

1. Alice and Bob use Diffie-Helman to exchange private keys

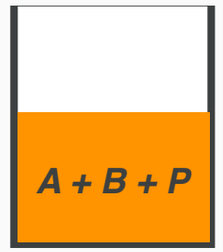
2. Alice *encrypts* the message

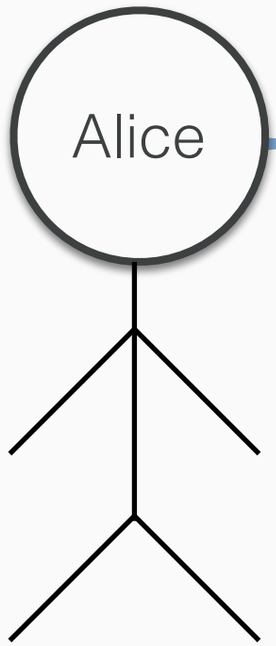
The Full Pipeline

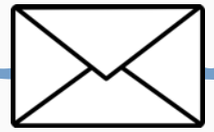
private key



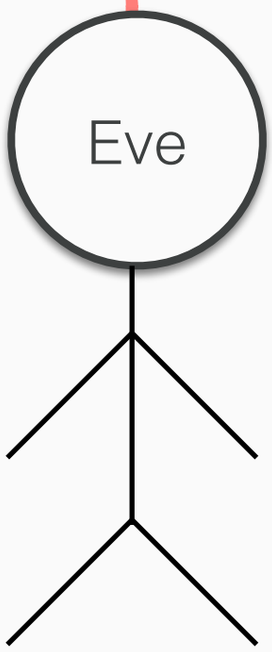
private key

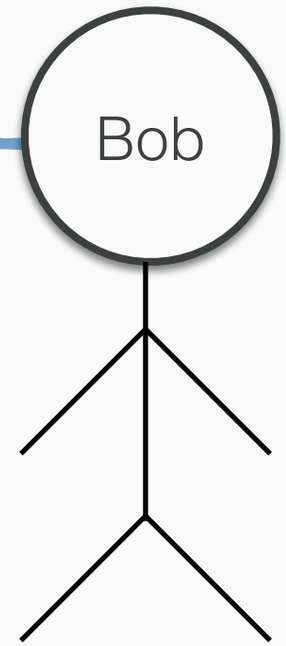






encrypted text





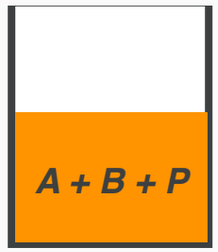
plaintext

1. Alice and Bob use Diffie-Helman to exchange private keys

2. Alice *encrypts* the message via OWF

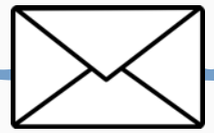
The Full Pipeline

private key



plaintext

Alice

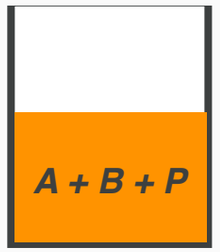


encrypted text

Eve

Bob

private key

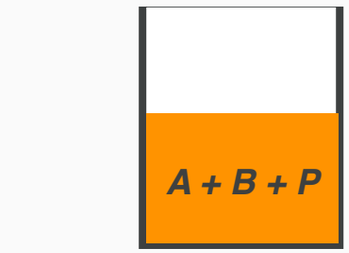


1. Alice and Bob use Diffie-Helman to exchange private keys

2. Alice *encrypts* the message via OWF

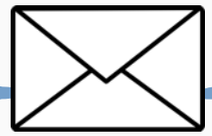
3. Can't decrypt without A+B+P, so Eve can't read, but Bob decrypts.

The Full Pipeline



Alice

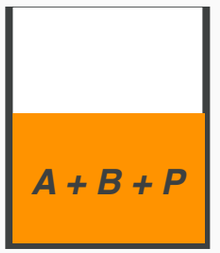
plaintext



encrypted text

Eve

Bob



plaintext

1. Alice and Bob use Diffie-Helman to exchange private keys

2. Alice *encrypts* the message via OWF

3. Can't decrypt without A+B+P, so Eve can't read, but Bob decrypts.