Integrity and Pseudorandom Functions

Logistics

Exam was yesterday

- Grading is done, tabulation and upload will happen this weekend

MP3 is out!

- CP1 due on Monday, October 14th, 6PM
- CP2 due on Wednesday, October 23rd, 6PM

Cryptography

What is cryptography?

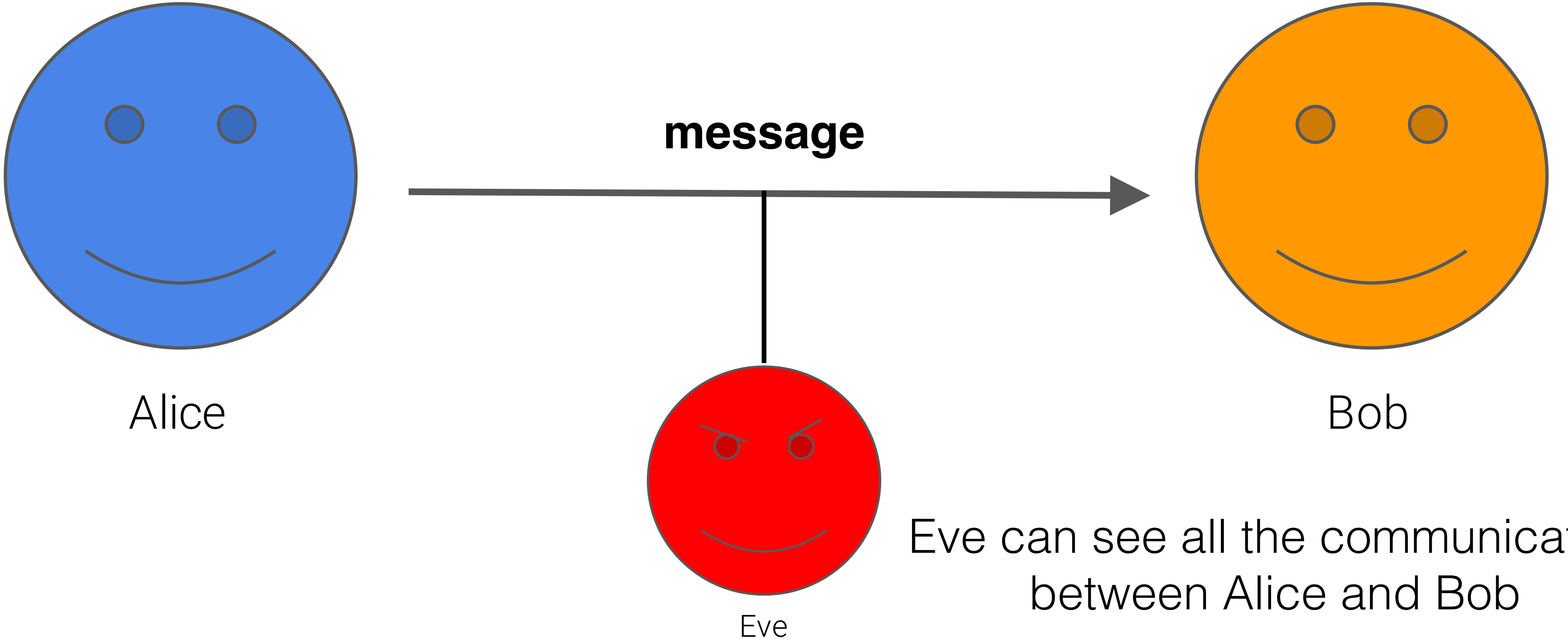
What can we do with cryptography?

How does one implement cryptography?

Where can it fail?

"Don't roll your own Crypto!" - Most Cryptographers

The Crypto Gang



Eve can see all the communication between Alice and Bob

The Crypto Gang

Alice

Bob

The Crypto Gang

Alice

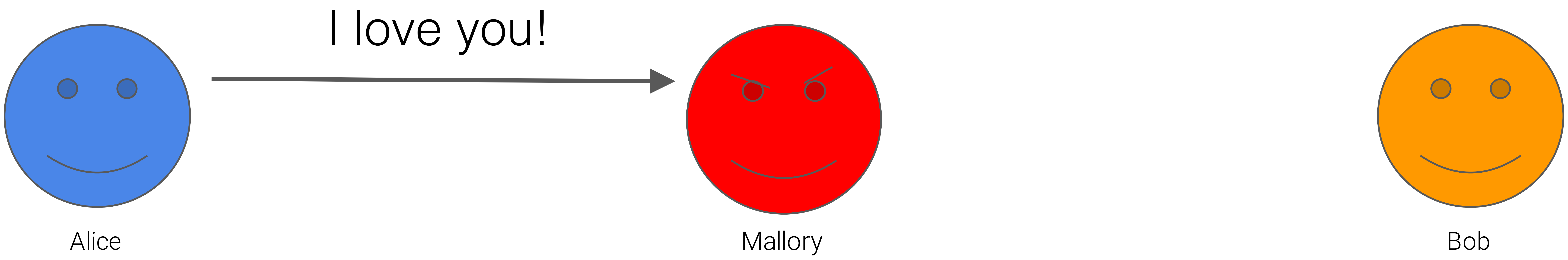
Mallory

Bob

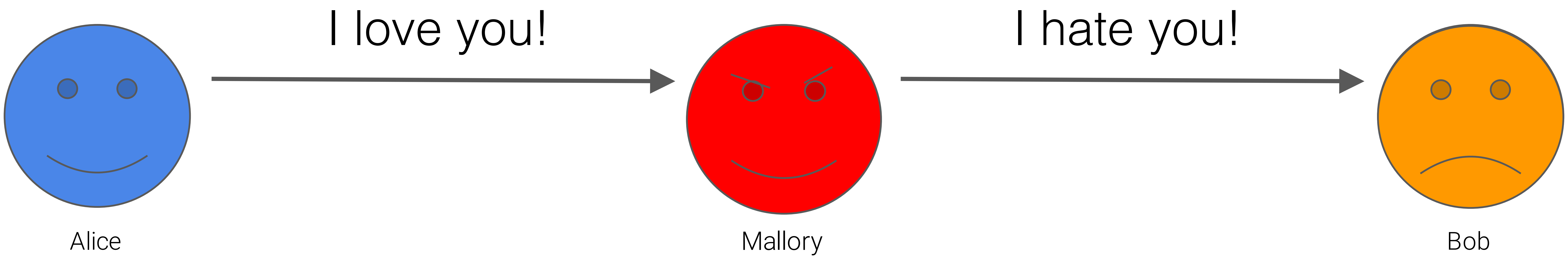
The Crypto Gang

Mallory can insert, delete, or modify communication!

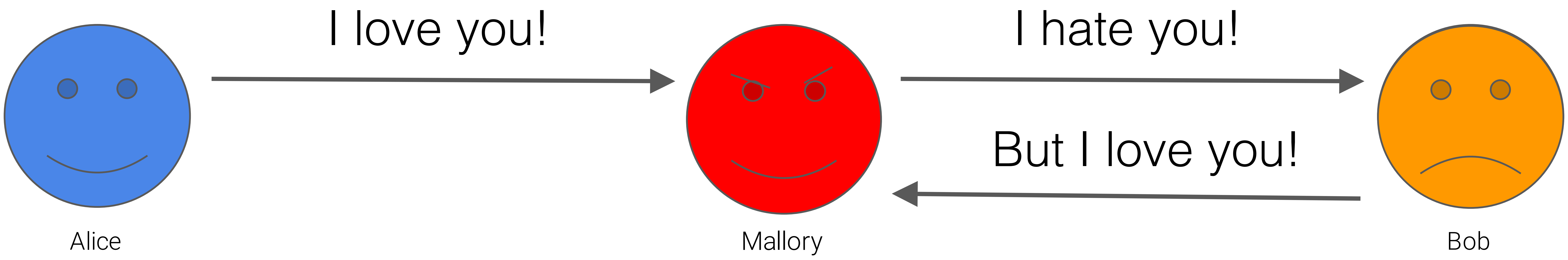
The Crypto Gang



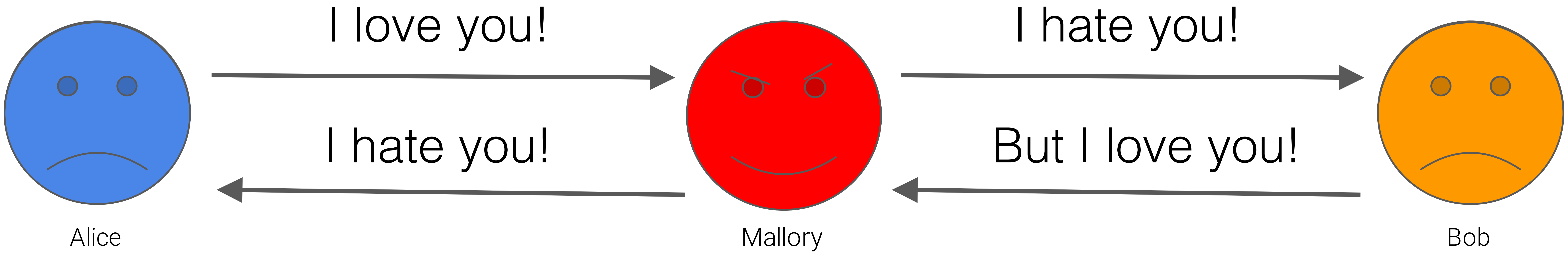
The Crypto Gang



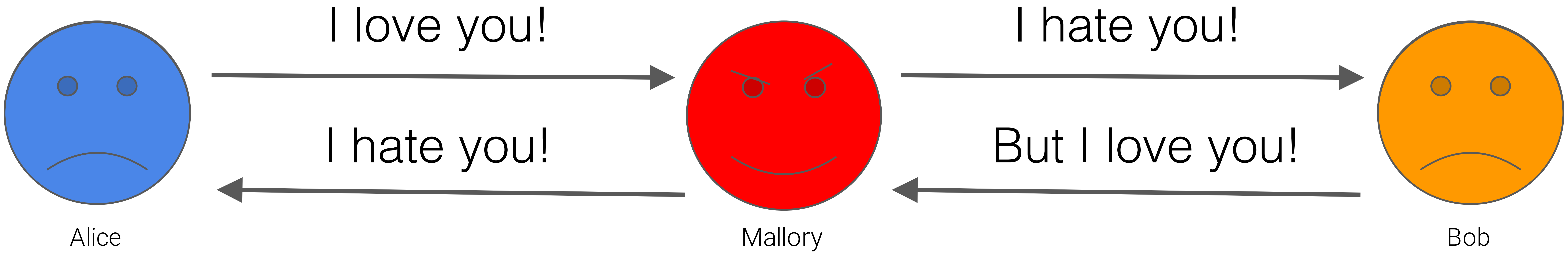
The Crypto Gang



The Crypto Gang



The Crypto Gang



How can we make the communication channel shared by Alice and Bob *secure?*

Secure Channels

Confidentiality

Keep the message contents ***secret*** from a passive eavesdropper

Integrity

Ensure the message has not been tampered with or altered without being detected

Authentication

Verify the identity of who you’re talking to

Secure Channels

Confidentiality

Keep the message contents *secret* from a passive eavesdropper

Integrity

Ensure the message has not been tampered with or altered without being detected

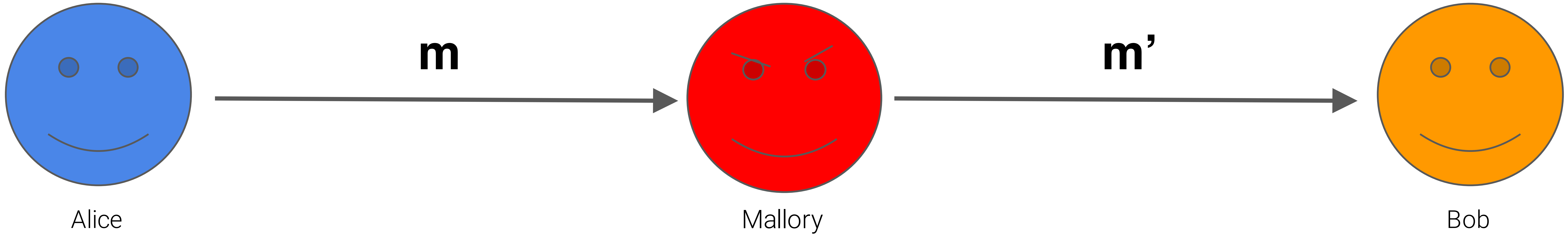
Authentication

Verify the identity of who you're talking to

Goal: Message Integrity

Alice wants to send a message **m** to Bob

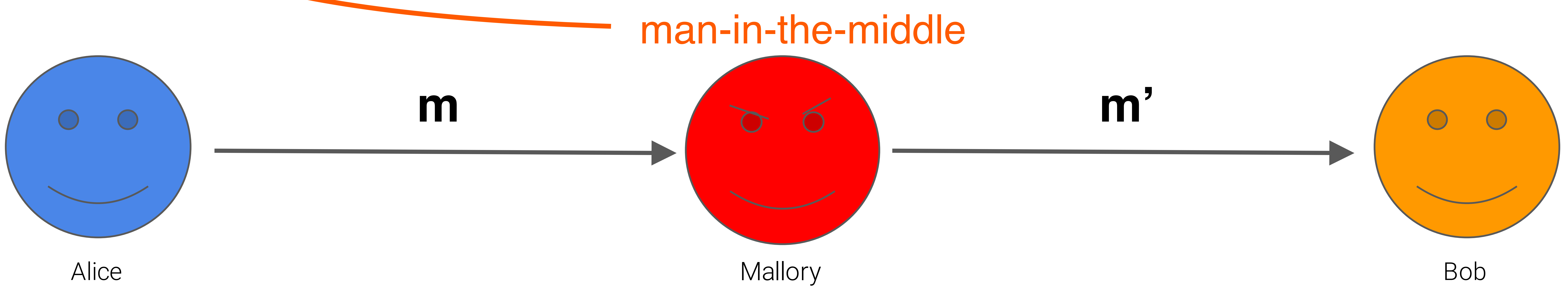
Mallory wants to trick Bob into accepting a message that Alice didn’t send



Goal: Message Integrity

Alice wants to send a message **m** to Bob

Mallory wants to trick Bob into accepting a message that Alice didn’t send

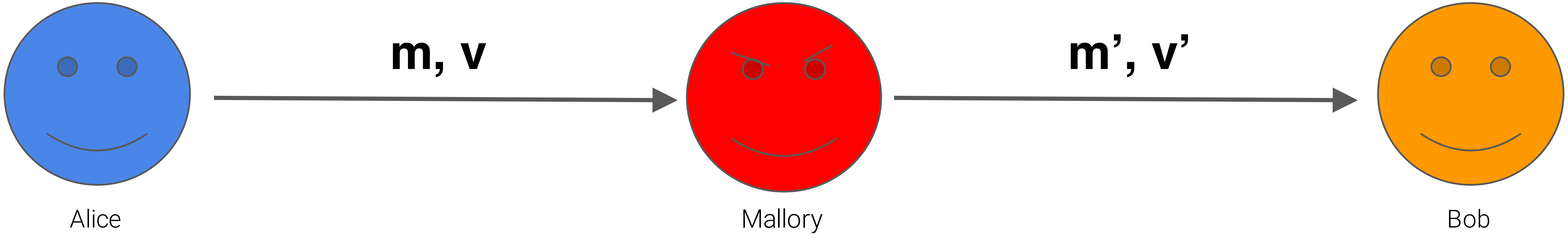


One Idea!

Alice computes v := f(m), sends along with message m

m = “I love you!”, f(m) = 24937859230785097341 0….

Bob verifies that v’ = f(m’), and accepts message if and only if this is true

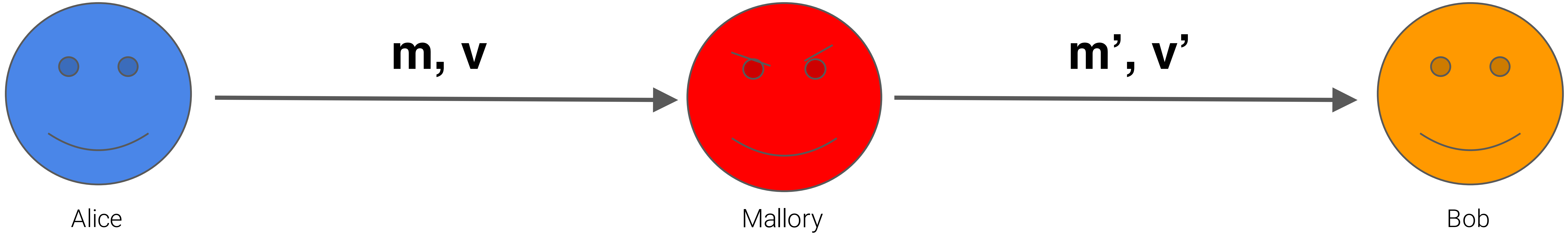


What should f be?

Easily computable by Alice and Bob;

<u>Not</u> computable by Mallory

We're **sunk** if Mallory can learn **f(x)** for any <u>**x ≠ m**</u>



Candidate for f: Random Function

Input: Any size up to a huge maximum

Output: Fixed size (e.g., 256 bits)

Defined by a giant lookup table that's filled in by flipping coins

0	→	0011111001010001…
1	→	1110011010010100…
2	→	0101010001010000…
⋮		⋮

Candidate for f: Random Function

Input: Any size up to a huge maximum

Output: Fixed size (e.g., 256 bits)

Defined by a giant lookup table that's filled in by flipping coins

0	→	0011111001010001...
1	→	1110011010010100...
2	→	0101010001010000...
⋮		⋮

Provably <u>secure</u>

Candidate for f: Random Function

Input: Any size up to a huge maximum

Output: Fixed size (e.g., 256 bits)

Defined by a giant lookup table that's filled in by flipping coins

0	→	0011111001010001…
1	→	1110011010010100…
2	→	0101010001010000…
⋮		⋮

Provably <u>secure</u>

Completely <u>impractical</u>

Candidate for f: Pseudorandom Function

Want a function that is practical but “looks random”…

Candidate for f: Pseudorandom Function

Let's build one:

Start with a big family of functions $\mathbf{f_0()}$, $\mathbf{f_1()}$, $\mathbf{f_2()}$, … all known to Mallory

Use $\mathbf{f_k}$ where **k** is a secret value (or "key") known only to Alice/Bob

k is (for example) 256-bits, chosen randomly and shared

Candidate for f: Pseudorandom Function

Let's build one:

Start with a big family of functions $\mathbf{f_0()}$, $\mathbf{f_1()}$, $\mathbf{f_2()}$, … all known to Mallory

Use $\mathbf{f_k}$ where **k** is a secret value (or "key") known only to Alice/Bob

k is (for example) 256-bits, chosen randomly and shared

Kerckchoffs's Principle: A cryptosystem should be secure even if everything about the system, <u>except the key</u>, is public knowledge

Formal Definition of a PRF

1. Flip a coin secretly to get bit **b**

2. If **b = 0**, let g be a ***random*** function

   If **b = 1**, let g = f_k where k is a randomly chosen secret

3. Repeat until Mallory says “stop!”

   Mallory chooses an **x;** we announce **g(x)**

4. Mallory guesses **b**

Formal Definition of a PRF

1. Flip a coin secretly to get bit **b**

2. If **b = 0**, let **g** be a *random* function

   If **b = 1**, let **g** = $\mathbf{f_k}$ where k is a randomly chosen secret

3. Repeat until Mallory says "stop!"

   Mallory chooses an **x;** we announce **g(x)**

4. Mallory guesses **b**

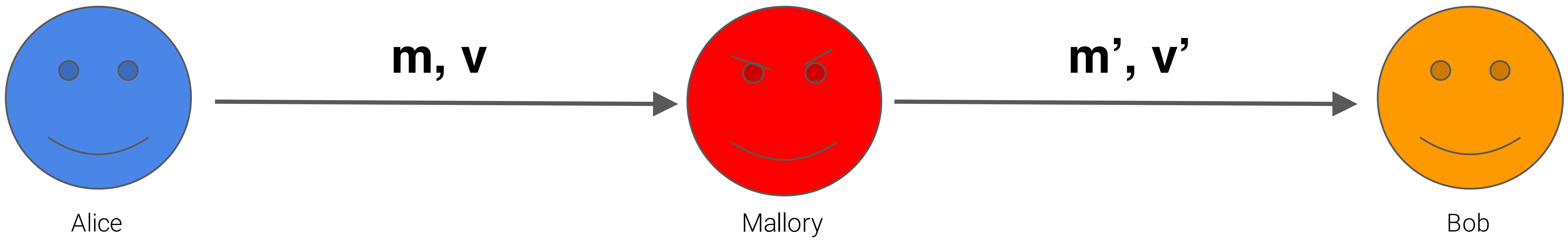
We say g() is a secure PRF if Mallory can't select b any better than random guessing!

A solution for Alice and Bob

Let **g() = $\mathbf{f_k}$()** be a secure PRF, where **k** is a random key known only to Alice and Bob

Alice computes **v := $\mathbf{f_k}$(m)**, Bob verifies **v' = $\mathbf{f_k}$(m')**

Mallory doesn't know key **k**, so she can't forge a message from Alice!



Message Authentication Codes

Message Authentication Code (MAC): *Effectively the same thing as a PRF*

Current, widely adopted approach – Hash-based MAC!

$\mathrm{HMAC}_k(m)$ =

$$\mathrm{SHA256}\left(k \oplus c_1 \parallel \mathrm{SHA256}(k \oplus c_2 \parallel m)\right)$$

So Far

Message Integrity, PRFs, MACs, HMACs

Next time...

Hashes for integrity

Hashes vs. HMACs

Confidentiality