# Crypto

Deepak Kumar

# whoami

# Educational Goals

- Basics of cryptography
- Get familiar with hex, binary, decimal conversions needed for CP1
- Think adversarially about cryptography
- Learn the definition and properties of hashes
- Implement hashes and investigate their properties using PyCrypto
- Learn the definitions of symmetric and asymmetric cryptography
- Implement AES encryption and decryption with PyCrypto

# Alice and Bob want to tell each other a secret.

Alice

Bob

# Alice and Bob want to tell each other a secret. But they have to do it in public.

Alice

Bob

Alice and Bob want to tell each other a secret.
But they have to do it in public.
How might they do this?

Alice

Bob

# How does this change when someone else *really* wants to learn the secret?

Alice
Bob
Eve

# Enter Cryptography

- From Greek
  - *Kryptos - secret*
  - *Grafein - to write*

# Enter Cryptography

- From Greek
  - *Kryptos - secret*
  - *Grafein - to write*

*Cryptography is the study of secure communication techniques that enable only the sender and intended recipient of a message to learn its true contents*

# Cryptography is all about hex, decimal, bytes

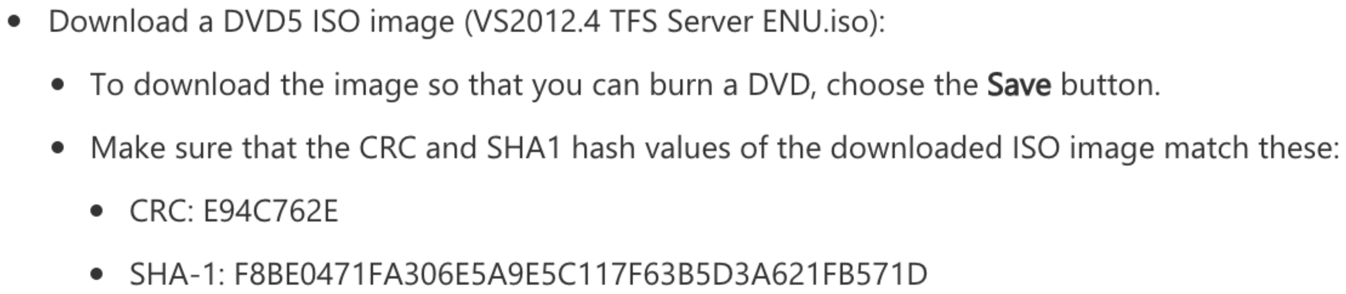
Demo

# Confidentiality and Integrity

Confidentiality – Keep the contents of a message secret from an eavesdropper (Eve)

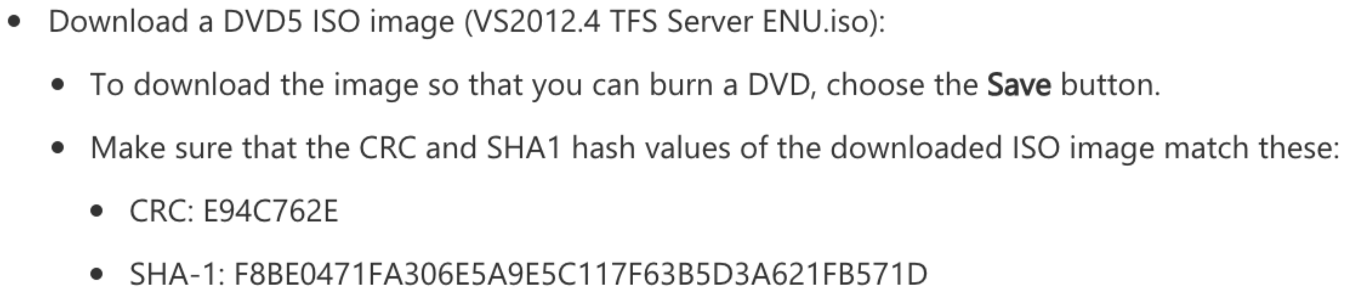
Integrity – Ensure a message has not been tampered with or altered

# Confidentiality or Integrity?



- Download a DVD5 ISO image (VS2012.4 TFS Server ENU.iso):
  - To download the image so that you can burn a DVD, choose the **Save** button.
  - Make sure that the CRC and SHA1 hash values of the downloaded ISO image match these:
    - CRC: E94C762E
    - SHA-1: F8BE0471FA306E5A9E5C117F63B5D3A621FB571D

# Confidentiality or Integrity?



- Download a DVD5 ISO image (VS2012.4 TFS Server ENU.iso):
  - To download the image so that you can burn a DVD, choose the **Save** button.
  - Make sure that the CRC and SHA1 hash values of the downloaded ISO image match these:
    - CRC: E94C762E
    - SHA-1: F8BE0471FA306E5A9E5C117F63B5D3A621FB571D

**Integrity:** want to ensure the ISO has not been modified in the download process

# Confidentiality or Integrity?

# Confidentiality or Integrity?

**Confidentiality:** Ensure that German military strategies and secrets were communicated secretly

# Integrity – Hashes

A hash is a cryptographic function H that takes an *arbitrary length input* and produces a *fixed size output*

A good hash function follows three properties:

1. First pre-image resistant
   a. If I know H(m), I can’t know m
2. Second pre-image resistant
   a. If I know m1, I can’t find m2 such that H(m2) == H(m1)
3. Collision resistant
   a. Can’t find *any* m1, m2 such that H(m1) == H(m2)

# Integrity – Hashes

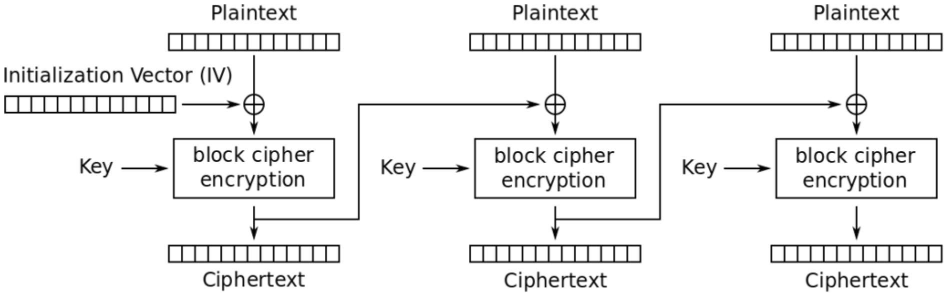
Demo

# Confidentiality – Symmetric Key Cryptography

Use a cipher where *secret key is shared between two parties in advance*

# Confidentiality – Block Ciphers

- A block cipher is a cipher that operates on blocks of input rather than bit-by-bit.
  - AES
  - 16-byte blocks
- Multiple modes (“ways”) of using a block cipher
  - ECB, CBC, OFB, CTR, CBF…
- In this MP, we’ll focus on **Cipher Block Chaining (CBC)**



Cipher Block Chaining (CBC) mode encryption

# Confidentiality – Block Ciphers

AES Demo

# Confidentiality – Asymmetric Cryptography

- We can use *two* keys, one **private key** and one **public key** to achieve confidentiality!
- These keys are the **inverse** of one another
  - Message x
  - Priv(Pub(x)) == x
  - Pub(Priv(x)) == x
- It is *computationally infeasible* to “guess” the private key given the public key

# Confidentiality – Textbook RSA

Choose two large primes, *p* and *q* at random

# Confidentiality – Textbook RSA

Choose two large primes, *p* and *q* at random

N = p * q

# Confidentiality – Textbook RSA

Choose two large primes, *p* and *q* at random

N = p * q

Choose *e* such that GCD(e, (p - 1) * (q - 1)) = 1, *coprime*

# Confidentiality – Textbook RSA

Choose two large primes, *p* and *q* at random

N = p * q

Choose *e* such that GCD(e, (p - 1) * (q - 1)) = 1, *coprime*

Find d such that e * d ≡ 1 mod ((p - 1) * (q - 1)), *d is modular multiplicative inverse of e for mod N*

# Confidentiality – Textbook RSA

Choose two large primes, *p* and *q* at random

N = p * q

Choose *e* such that GCD(e, (p - 1) * (q - 1)) = 1, *coprime*

Find d such that e * d ≡ 1 mod ((p - 1) * (q - 1)), *d is modular multiplicative inverse of e for mod N*

Public Key: (e, N)

Private Key: (d, N)

# Confidentiality – Encryption with RSA

Encrypt a message x to A, public key (e, N)

$c = x^e \bmod N$

Decrypt a message as A, private key (d, N)

$x = c^d \bmod N$

*No one else can decrypt the message if it was encrypted with A’s public key*

# Confidentiality – Playing with RSA

Demo

# Some notes on this MP

- CP1 is a whirlwind tour of getting your feet with with crypto in Python
  - Hex, Binary, Bytearrays
  - AES
  - RSA
- CP2 is 5 different cryptographic attacks
  - Length-extension attack on Merkle-Damgard hashes
  - Collision attack on weak hash functions
  - Padding oracle attack
  - Weak RSA key generation attack
  - Colliding certificates

# Some notes on this MP

- CP1 is a whirlwind tour of getting your feet with with crypto in Python
  - Hex, Binary, Bytearrays
  - AES
  - RSA
- CP2 is 5 different cryptographic attacks
  - Length-extension attack on Merkle-Damgard hashes
  - Collision attack on weak hash functions
  - Padding oracle attack
  - Weak RSA key generation attack
  - Colliding certificates

**THIS MP IS VERY HARD AND YOU WILL FEELSBADMAN IF YOU START LATE**

# Educational Goals

- Be able to define cryptography
- Identify how confidentiality and integrity are implemented in cryptography
- Learn the difference between symmetric and asymmetric cryptography, and why we would use one over the other
- Learn the definitions and properties of hashes, HMACs
- Think adversarially about cryptography
- Be comfortable with the vocabulary to understand and complete CP1 of Crypto MP

# How do we implement confidentiality and integrity using cryptography?

# Confidentiality

haahjr ha khdu

# Confidentiality

attack at dawn

*Shift each letter in the alphabet by 7*

# Confidentiality

Is a Caesar Cipher good enough?

# Confidentiality

Is a Caesar Cipher good enough?

*No, everyone knows the Caesar cipher!*

# Confidentiality

- Cryptographers and mathematicians spent a considerable amount of time inventing more complicated ciphers, but *they kept getting broken*

*Kerckchoff's Principle: Use Secret Keys, **NOT** Secret Functions!*

# Confidentiality – Keyed Ciphers

- DES
    - Data Encryption Standard
- 3DES
    - Triple DES
- AES
    - Advanced Encryption Standard

All of these are examples of *symmetric* encryption, where two parties share a key in advance

# Confidentiality

What if you don’t have a shared secret key?

# Confidentiality – Asymmetric Cryptography

- We can use *two* keys, one **private key** and one **public key** to achieve confidentiality!
- These keys are the **inverse** of one another
  - Message x
  - Priv(Pub(x)) == x
  - Pub(Priv(x)) == x
- It is *computationally infeasible* to “guess” the private key given the public key

# Confidentiality – Textbook RSA

Choose two large primes, $p$ and $q$ at random

# Confidentiality – Textbook RSA

Choose two large primes, *p* and *q* at random

N = p * q

# Confidentiality – Textbook RSA

Choose two large primes, *p* and *q* at random

N = p * q

Choose *e* such that GCD(e, (p - 1) * (q - 1)) = 1, *coprime*

# Confidentiality – Textbook RSA

Choose two large primes, *p* and *q* at random

N = p * q

Choose *e* such that GCD(e, (p - 1) * (q - 1)) = 1, *coprime*

Find d such that e * d ≡ 1 mod ((p - 1) * (q - 1)), *d is modular multiplicative inverse of e for mod N*

# Confidentiality – Textbook RSA

Choose two large primes, $p$ and $q$ at random

$N = p * q$

Choose $e$ such that $\mathrm{GCD}(e, (p - 1) * (q - 1)) = 1$, *coprime*

Find d such that $e * d \equiv 1 \bmod ((p - 1) * (q - 1))$, *d is modular multiplicative inverse of e for mod N*

Public Key: $(e, N)$

Private Key: $(d, N)$

# Confidentiality – Encryption with RSA

Encrypt a message x to A, public key (e, N)

$c = x^e \bmod N$

Decrypt a message as A, private key (d, N)

$x = c^d \bmod N$

*No one else can decrypt the message if it was encrypted with A’s public key*

# Integrity

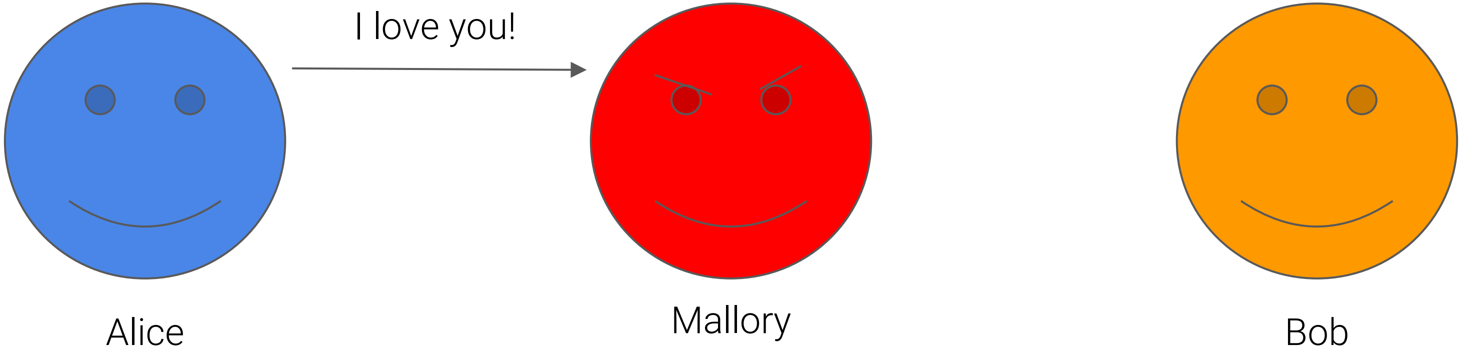
# Integrity

Alice

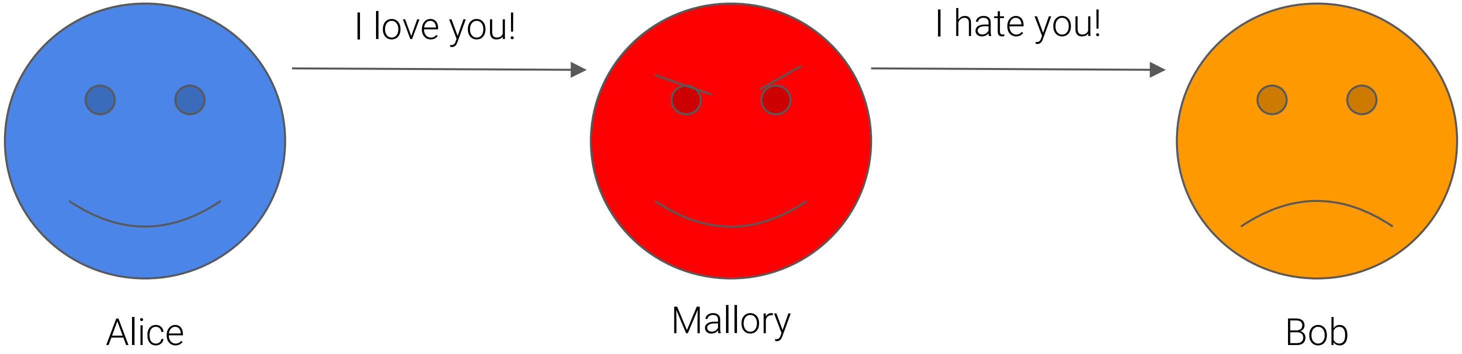
Mallory

Bob

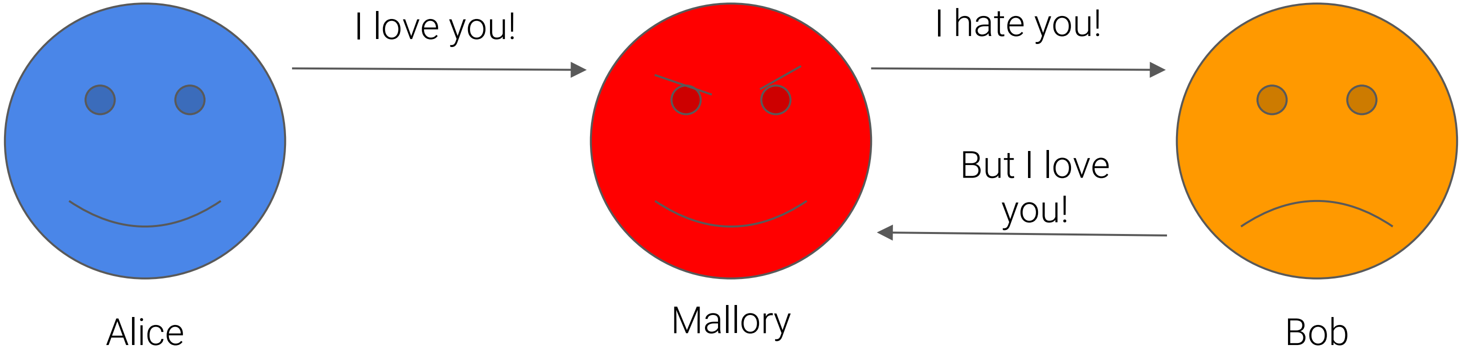
# Integrity



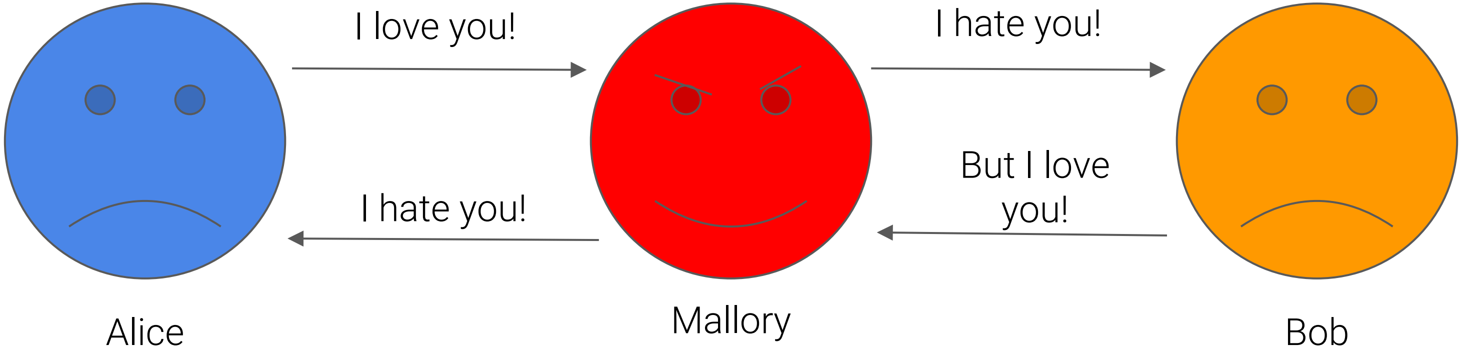
# Integrity



# Integrity



# Integrity

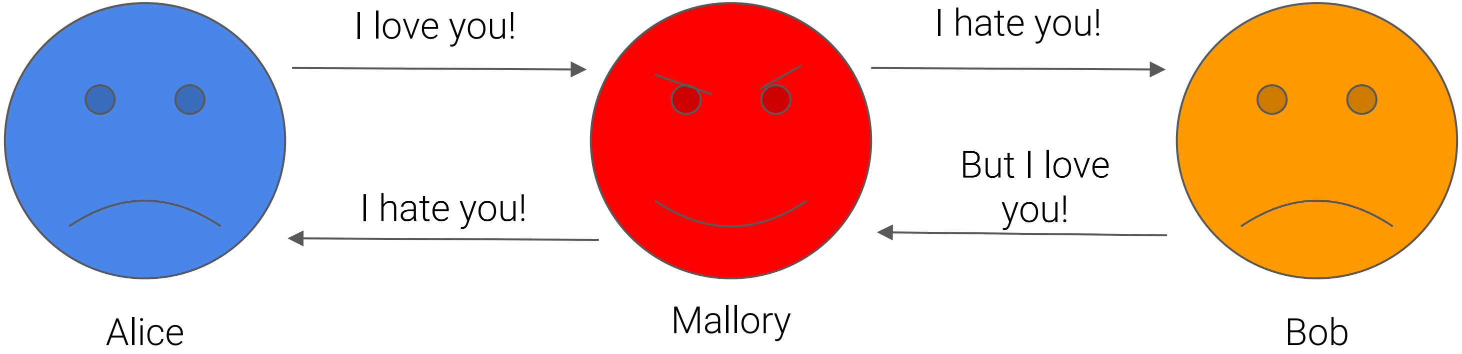

I love you!
I hate you!
I hate you!
But I love you!
Alice
Mallory
Bob

# Integrity – Hashes

A hash is a cryptographic function H that follows three properties

1. First pre-image resistant
   a. If I know H(m), I can’t know m
2. Second pre-image resistant
   a. If I know m1, I can’t find m2 such that H(m2) == H(m1)
3. Collision resistant
   a. Can’t find *any* m1, m2 such that H(m1) == H(m2)

# Integrity



*What if Alice also sent along a SHA256 hash of her message to Bob?*

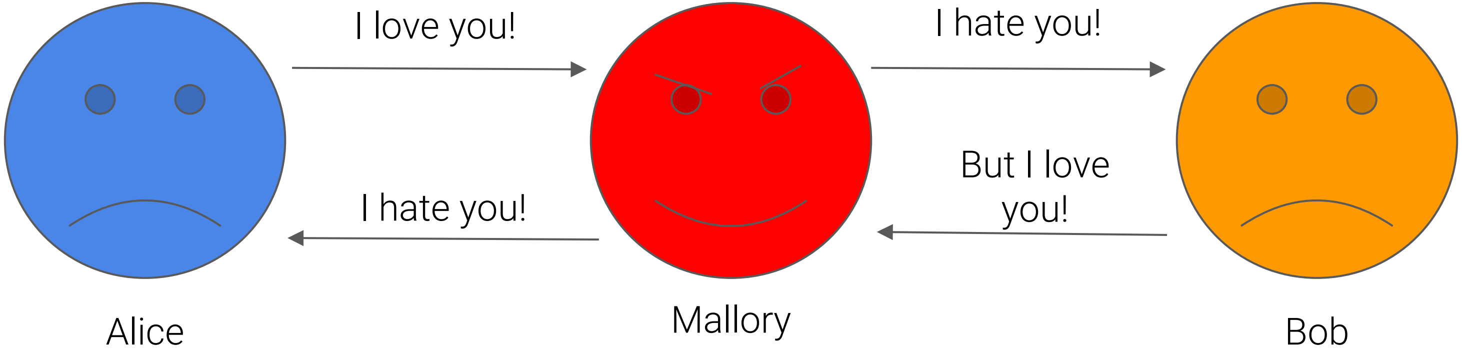
# Integrity – Keyed Hashes (HMAC)

Keyed Hash-based Message Authentication Code

$$\mathrm{HMAC}(K, m) = \mathrm{H}\Big(\big(K' \oplus opad\big) \parallel \mathrm{H}\big(\big(K' \oplus ipad\big) \parallel m\big)\Big)$$

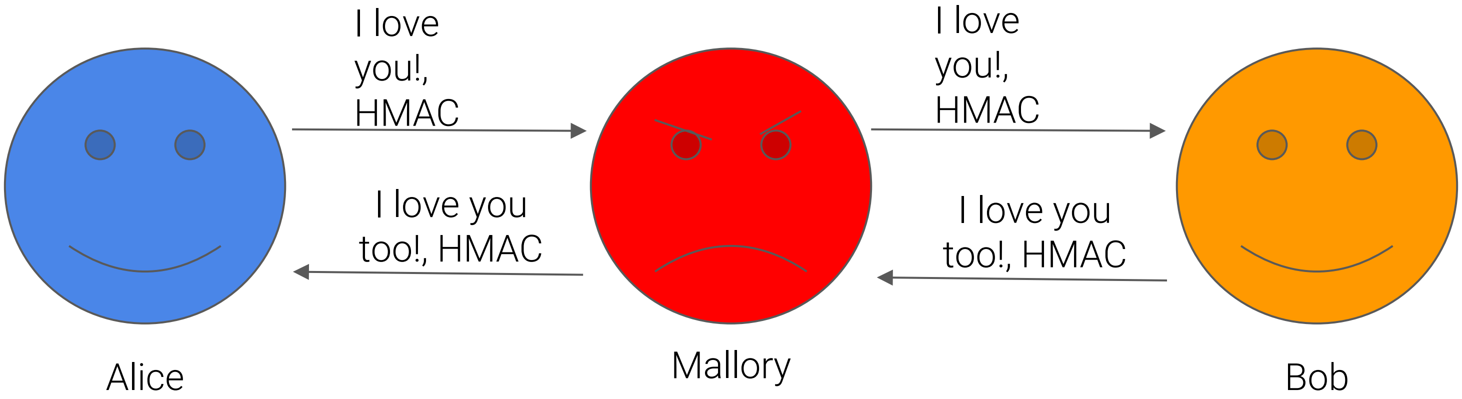
$$K' = \begin{cases} \mathrm{H}(K) & K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

# Integrity



*What if Alice also sent along a HMAC of her message to Bob?*

# Integrity



*What if Alice also sent along a HMAC of her message to Bob?*

# Integrity – Asymmetric Signatures

Sign a message x as A, private key (d, N)

$c = x^d \bmod N$

Verify a message from A, public key (e, N)

$x = c^e \bmod N$

*If the message is verified, only A could have signed it!*