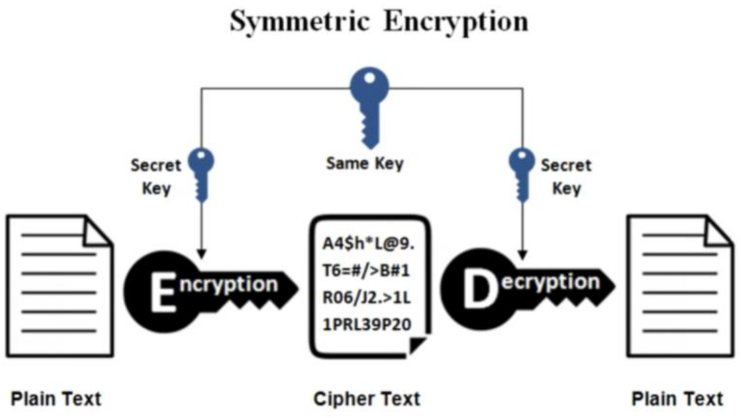
CSCI 466: Networks

Network Security (Hashing)

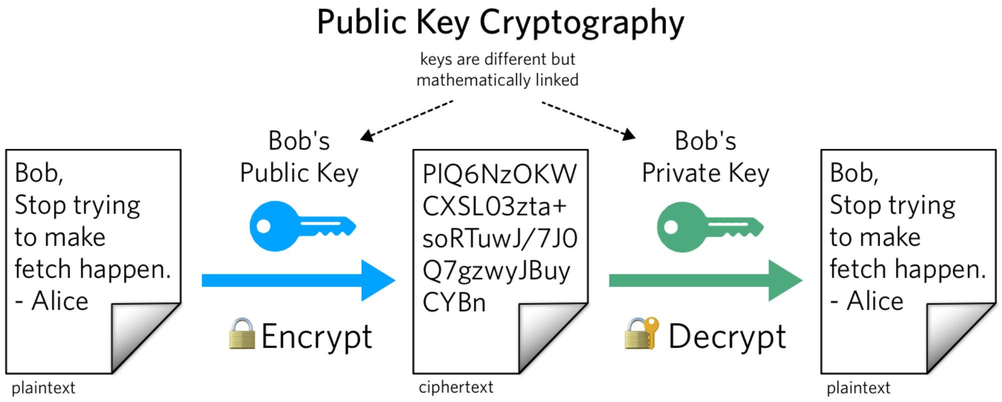
Reese Pearsall

Fall 2023

Review



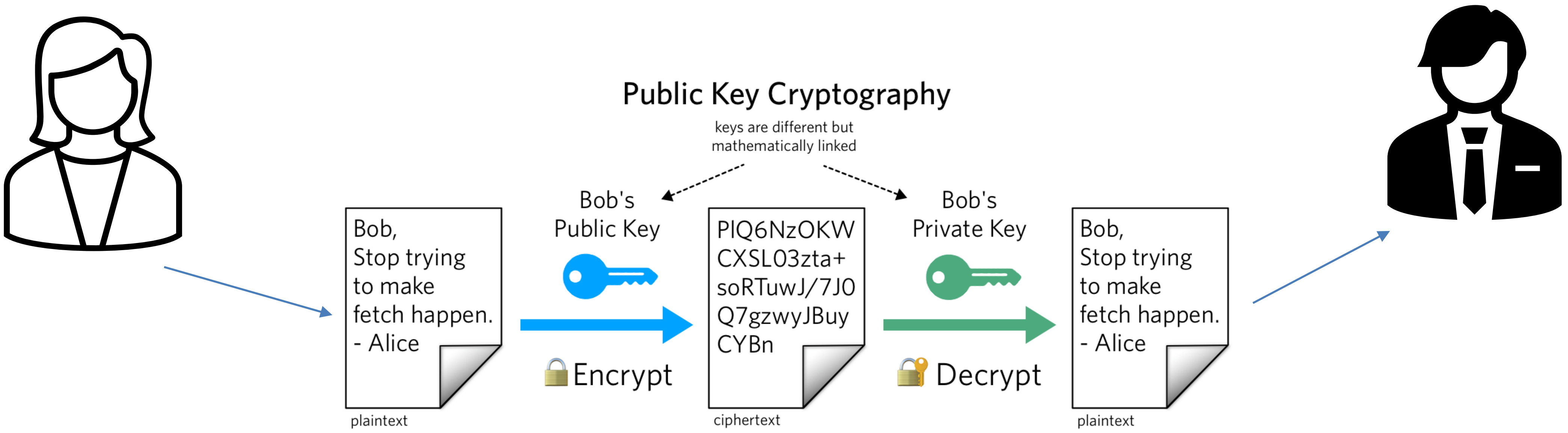
- Same key used for encrypting and decrypting
- Using block ciphers (AES), we can encrypt an arbitrary size of data
- Issue: How to securely share secret keys with each other?



- Two keys: Public Key (a lock), and a price key (the key)
- Public key is used to encrypt. Private key used to decrypt message
- Using math, we can securely send messages over an unsecure channel without sharing any sensitive information
- Issue: We can not encrypt stuff bigger than our key (2048 bits)

- Often times, symmetric and asymmetric cryptography are used **together**

(use RSA to send the key for symmetric crypto!)



Today's Goals:

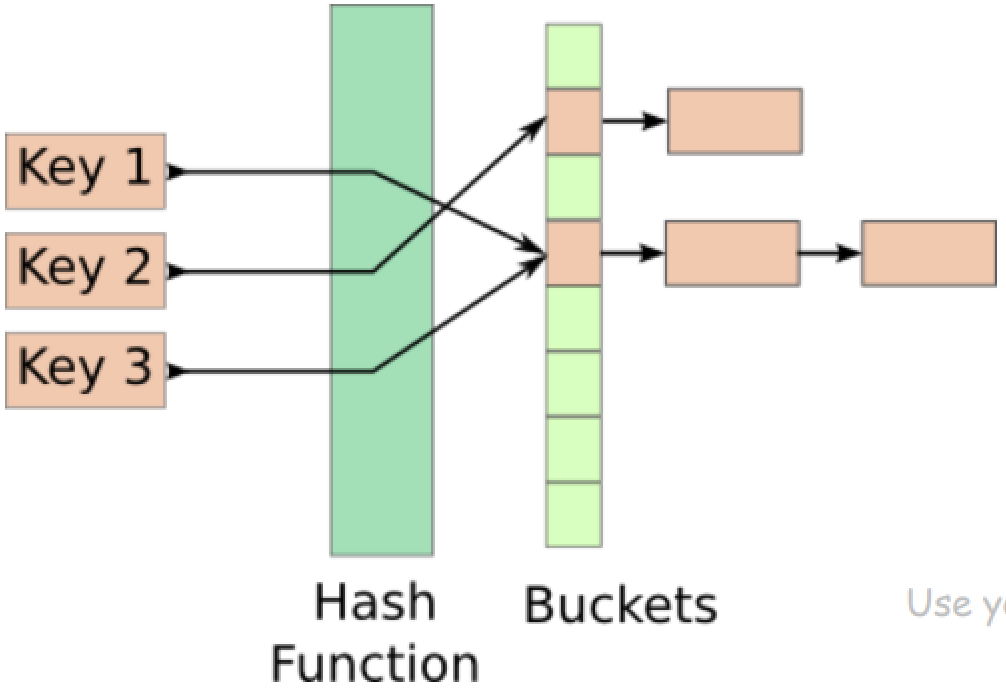
- We need a way to make sure our message does not get tampered with before arrival (**message integrity**)
- We need to find a way to make sure Bob knows these messages are truly coming from Alice (and not someone lying) → **Authentication**

Hash Functions

Hash Functions map arbitrary size data to data of fixed size

- An essential building block in cryptography, with desirable practical and security properties

Ex. **f(x) = x mod 100**



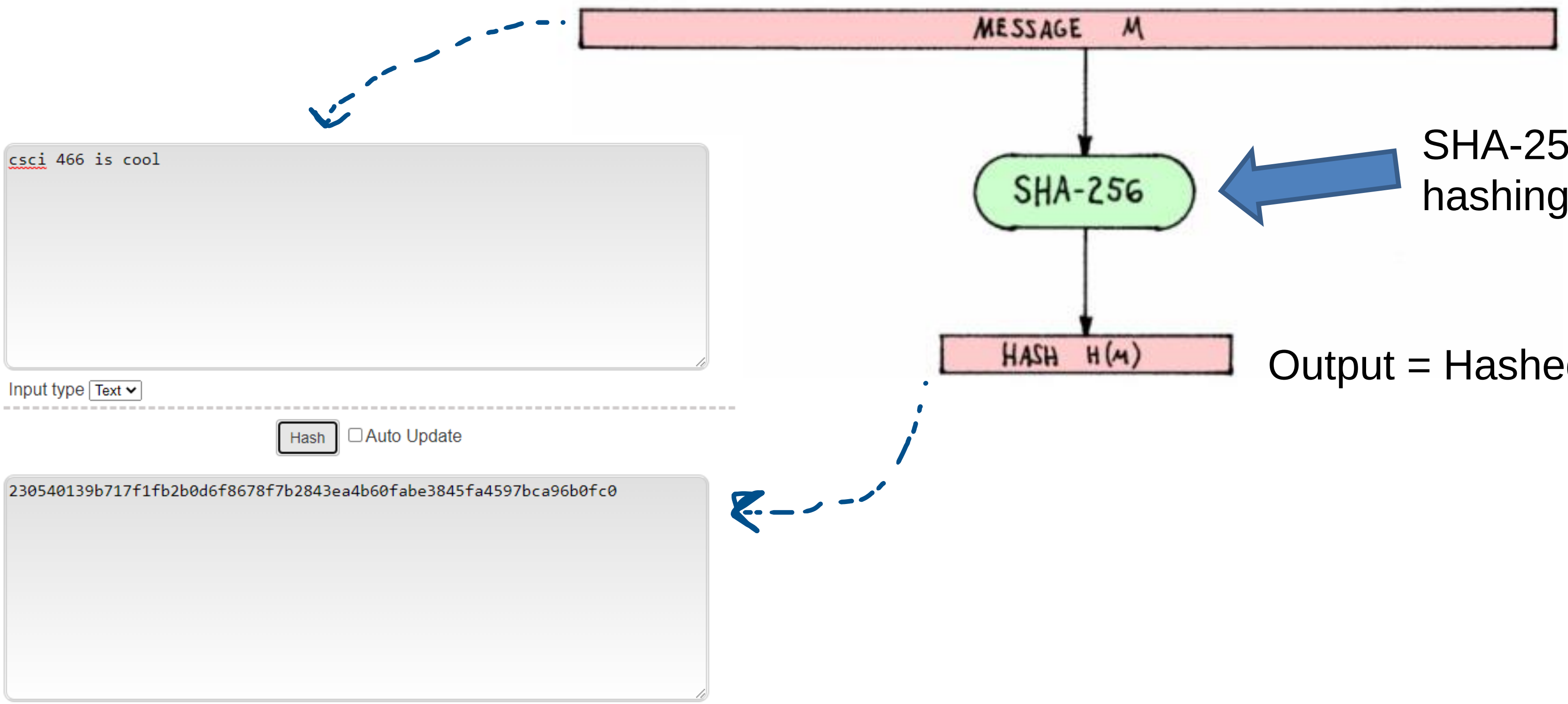
How many buckets?
What to do if two keys map to the same bucket?

Collisions happen…
Use your favorite collision resolution technique
(open addressing, chaining, etc.)

Hash Functions

Cryptographic Hash Functions map arbitrary size data to data of fixed size

- But with **three** additional important properties



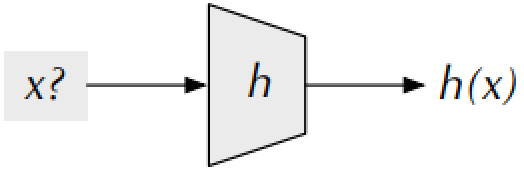
Input = Message M

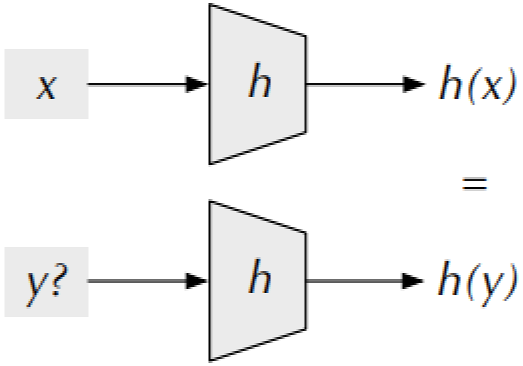
SHA-256 = A cryptographically secure hashing function

Output = Hashed Message M

Hash Functions

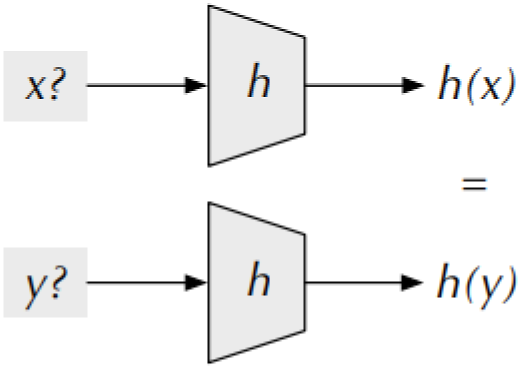
- **Preimage Resistance ("One-Way")**
  Given h(x) = z, hard to find x
  (or any input that hashes to z for that matter)





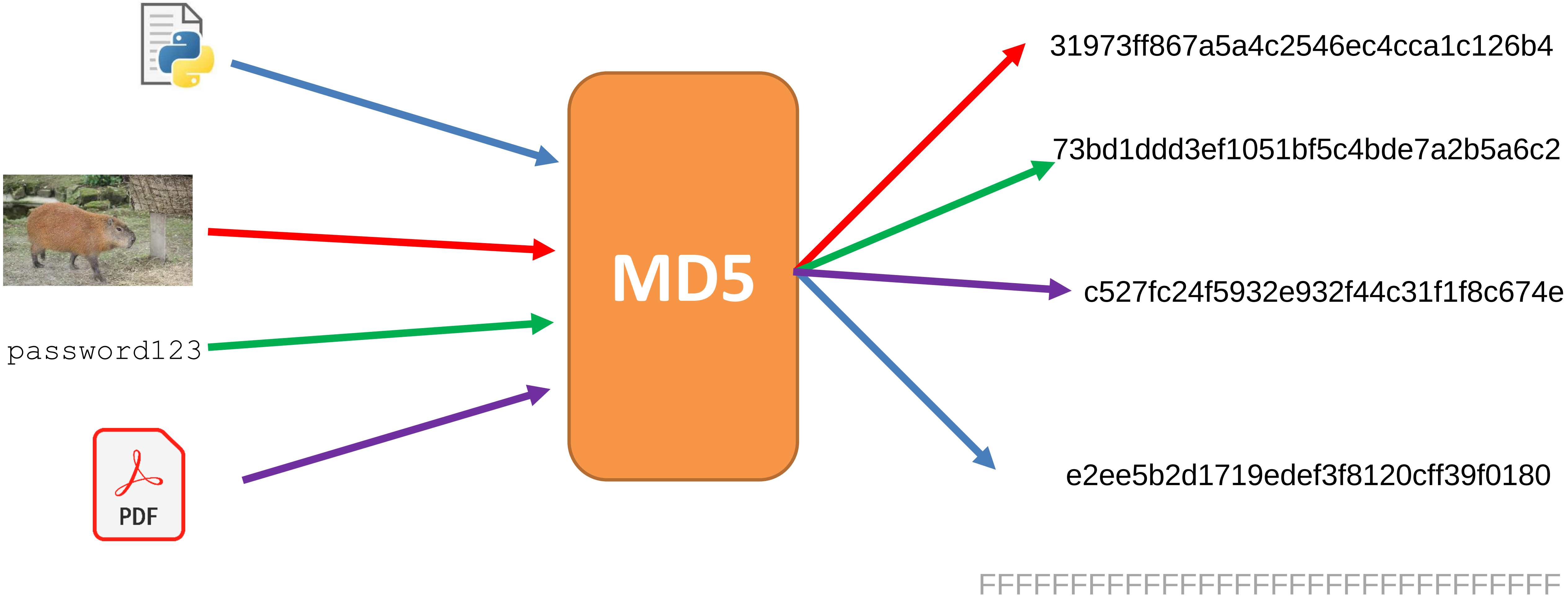
- **Second Preimage Resistance**
  Given x and h(x), hard to find y s.t. h(x) = h(y)

- Collision Resistance (or, ideally, "Collision Free")
  Difficult to find x and y s.t. hash(x) = hash(y)



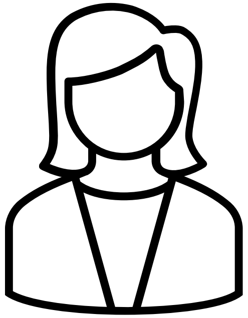
Applications of Hashing

Output space of MD5 (128 bits)



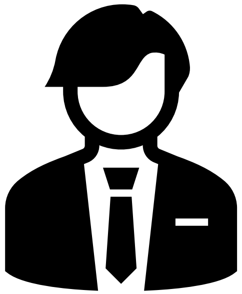
What are some uses for hashing?

Message Integrity



"I love you bob"

89defae676abd3e3a42b41df17c40096



Message Received:

"I love you bob"

↓

Hash

↓

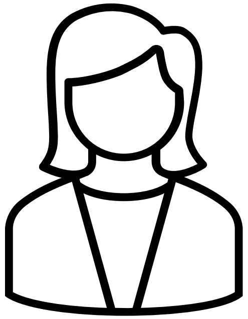
89defae676abd3e3a42b41df17c40096

1. Sarah computes the hash of message prior to sending

2. Bob receives the message, and computes the hash of the received message

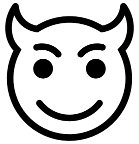
If the message was not tampered with, or modified, then the hashes should be the same

Message Integrity



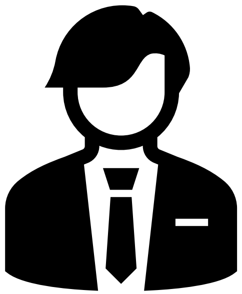
“I love you bob”

89defae676abd3e3a42b41df17c40096



Eve tampers with message

“I hate you bob”



Message Received:

“I hate you bob”

Hash

b0608c4e1775ad8f92e7b5c191774c5d

Different hashes = something fishy is going on

1. Sarah computes the hash of message prior to sending

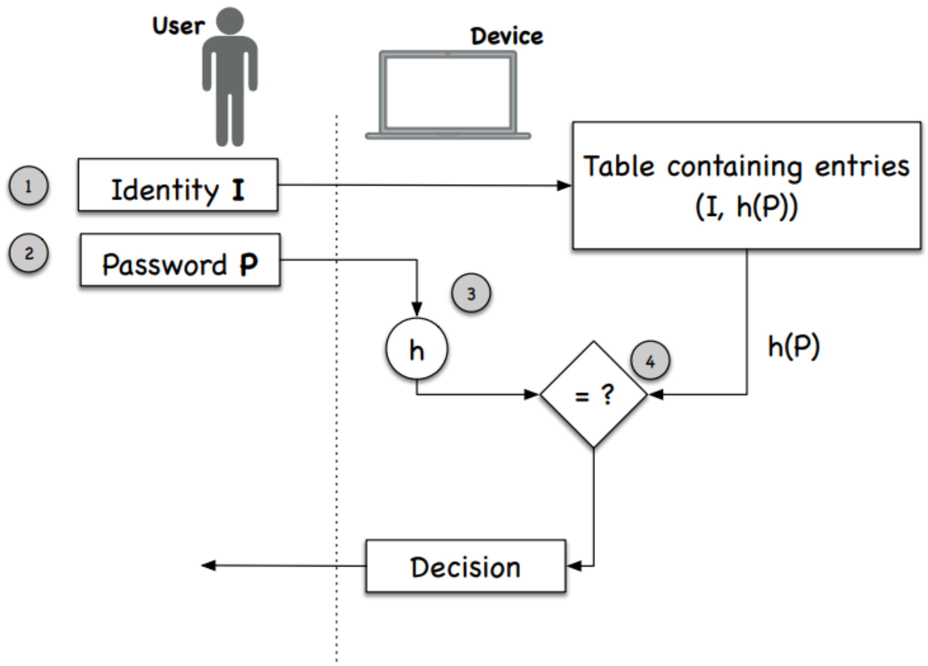
2. Bob receives the message, and computes the hash of the received message

If the message was not tampered with, or modified, then the hashes should be the same

Other Applications of Hashing

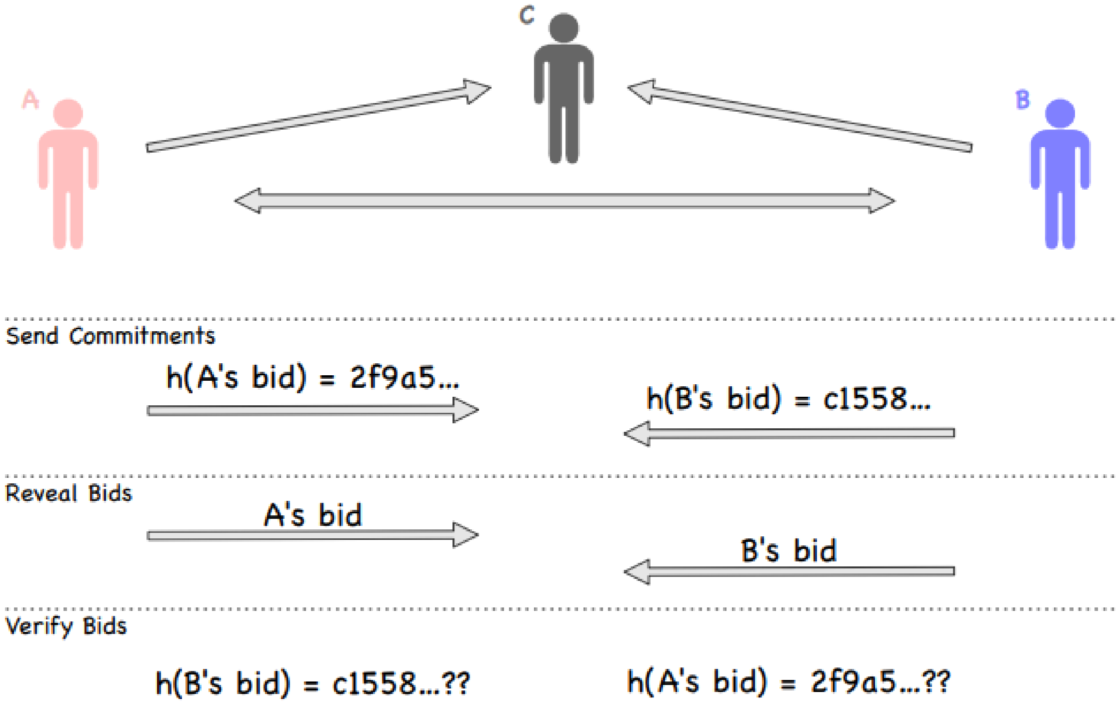
Password Storage

- Websites don’t store your password in plaintext, instead they store the **hash** of your password

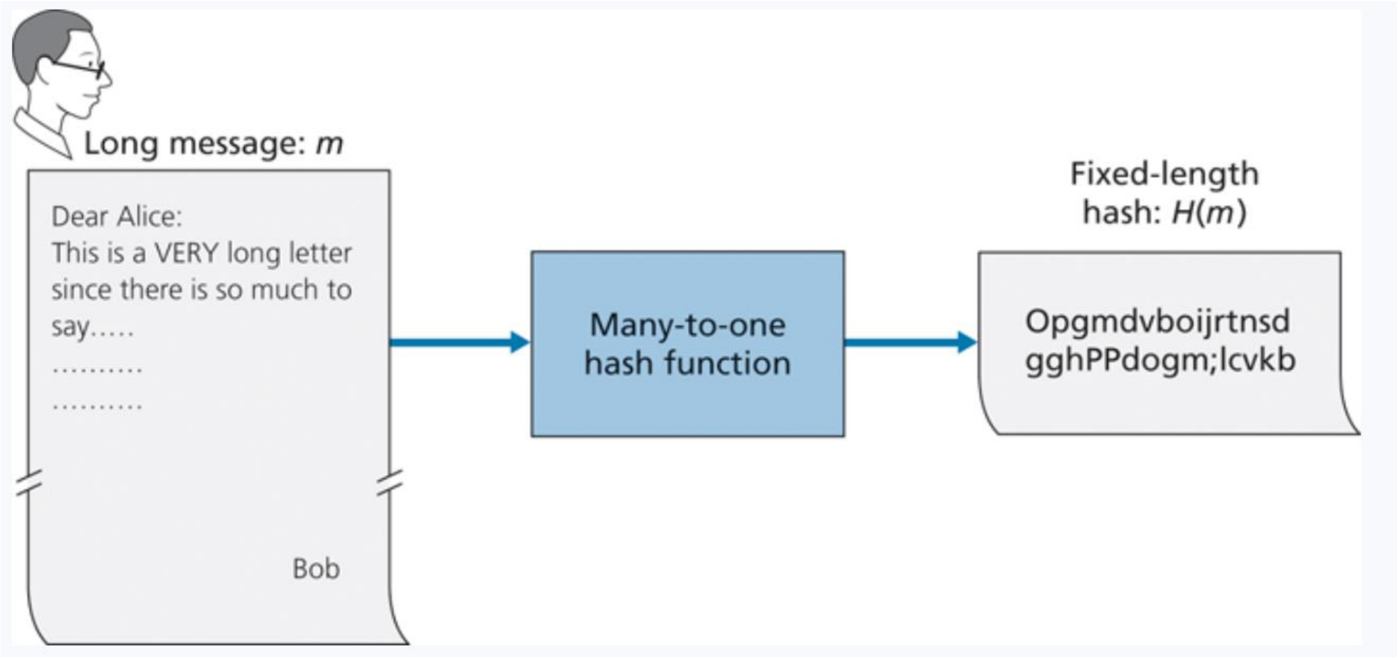


Fairness and Commitment

- Disclosing a hash does not disclose the original message
- Useful for committing a secret without disclosing the secret itself



Message Integrity

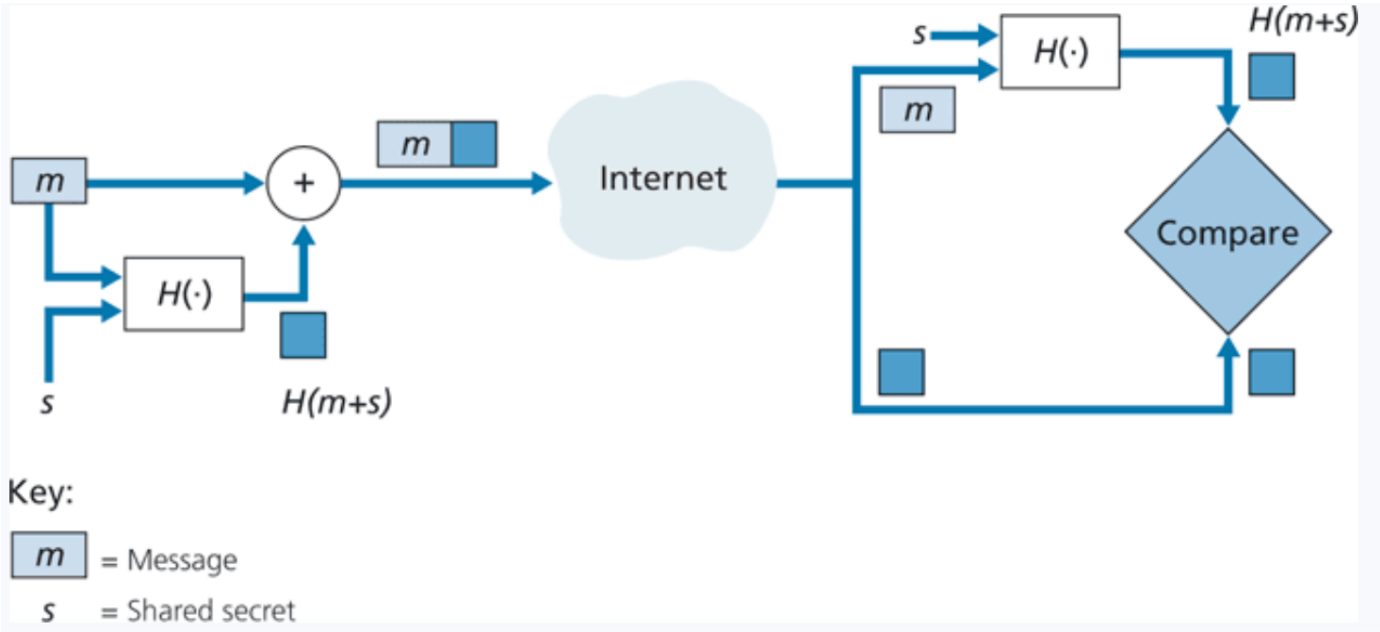


- Hashes provide an irreversible, unique* identifier for a message

Message Authentication Code (MAC)

1. Append a message with a shared secret (m + s)
2. Compute hash of (m+s) → H(m+s)
3. Send H(m+s) with message m
4. **Sender sends: (H(m+s), m)**

1. Receiver gets **(H(m+s), m)**
2. Append m with shared secret s (m + s)
3. Compute H(m+s)
4. The value receiver computed should match the H(m+s) he received

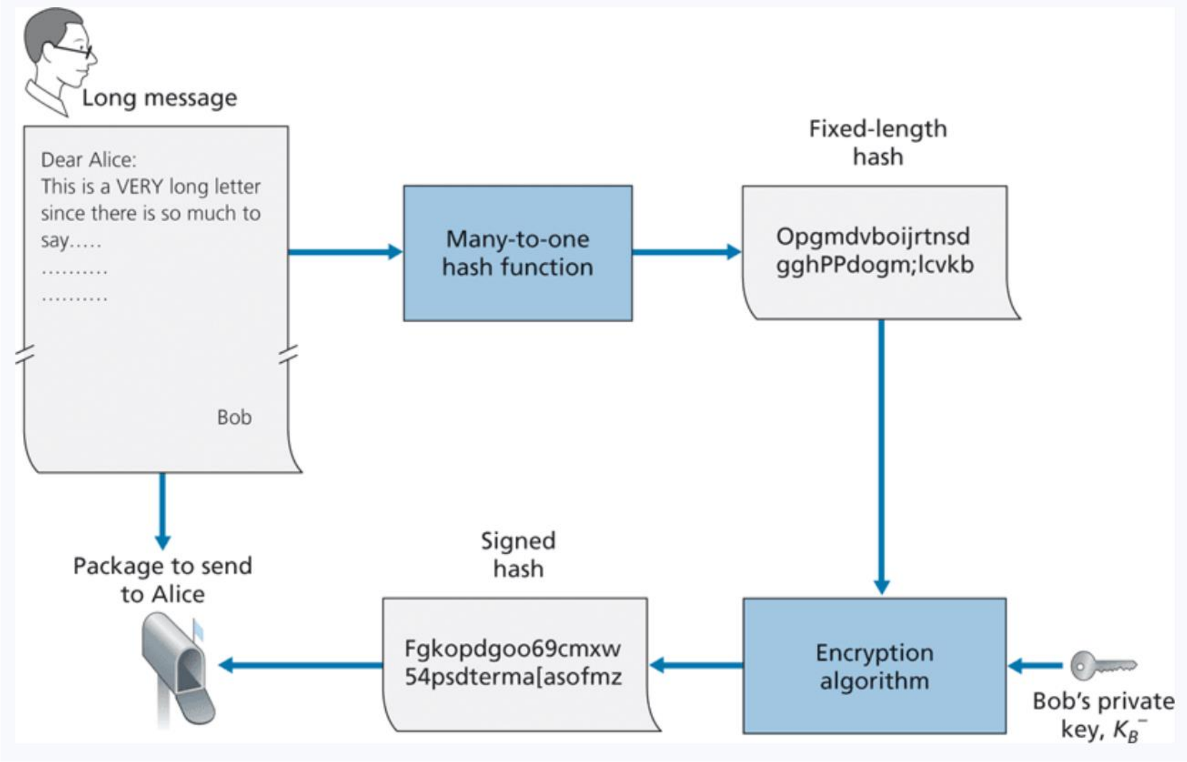


No encryption required!

Digital Signatures

- What is a unique identifier for bob? What is something that only bob knows and nobody else?
- ➢ His **private key**

Bob encrypts his hashed message using his **private key**, and sends the signed hash, along with message to Alice



When Alice receives this message, she must find a way to decrypt the signed hash

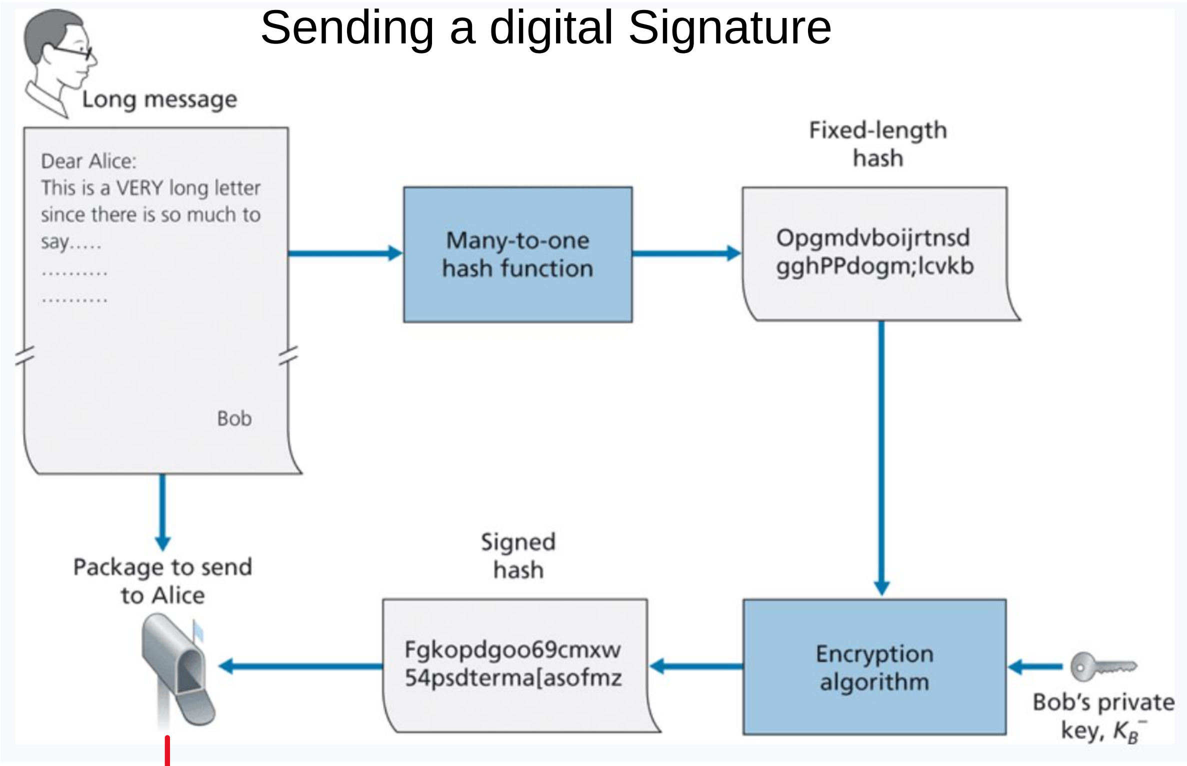
She will use Bob's **public key**

Digital Signatures

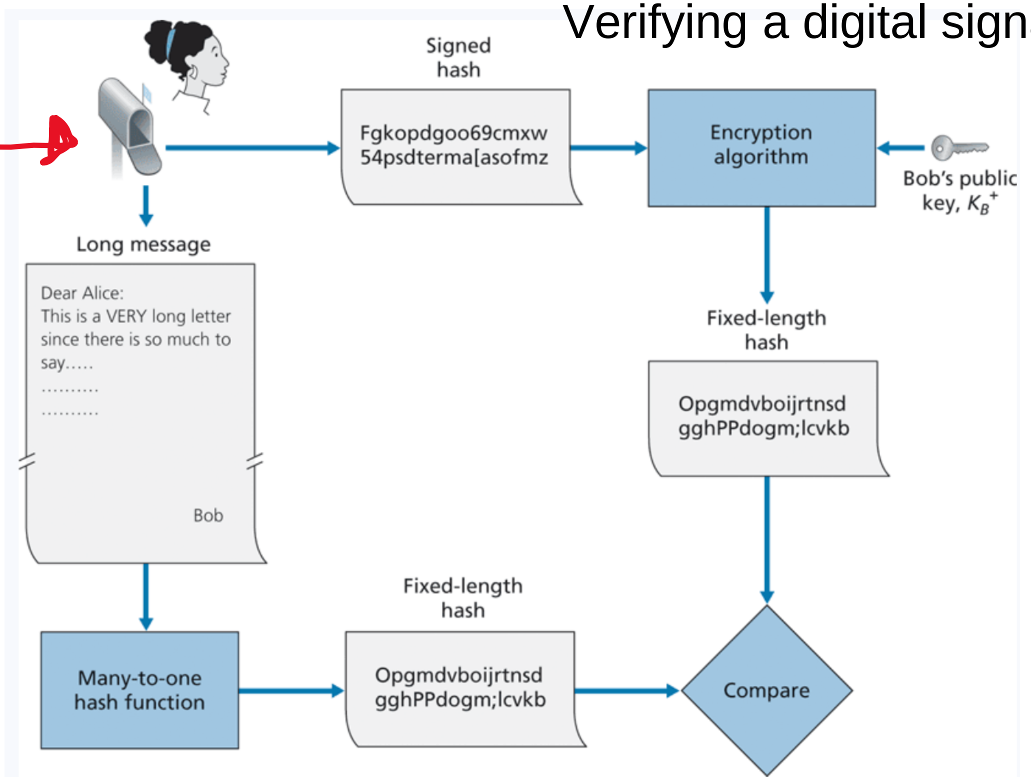
- What is a unique identifier for bob? What is something that only bob knows and nobody else?
- ➢ His **private key**

Bob encrypts his hashed message using his **private key**, and sends the signed hash, along with message to Alice. Alice decrypts using his **public key** and verifies that the hashes match

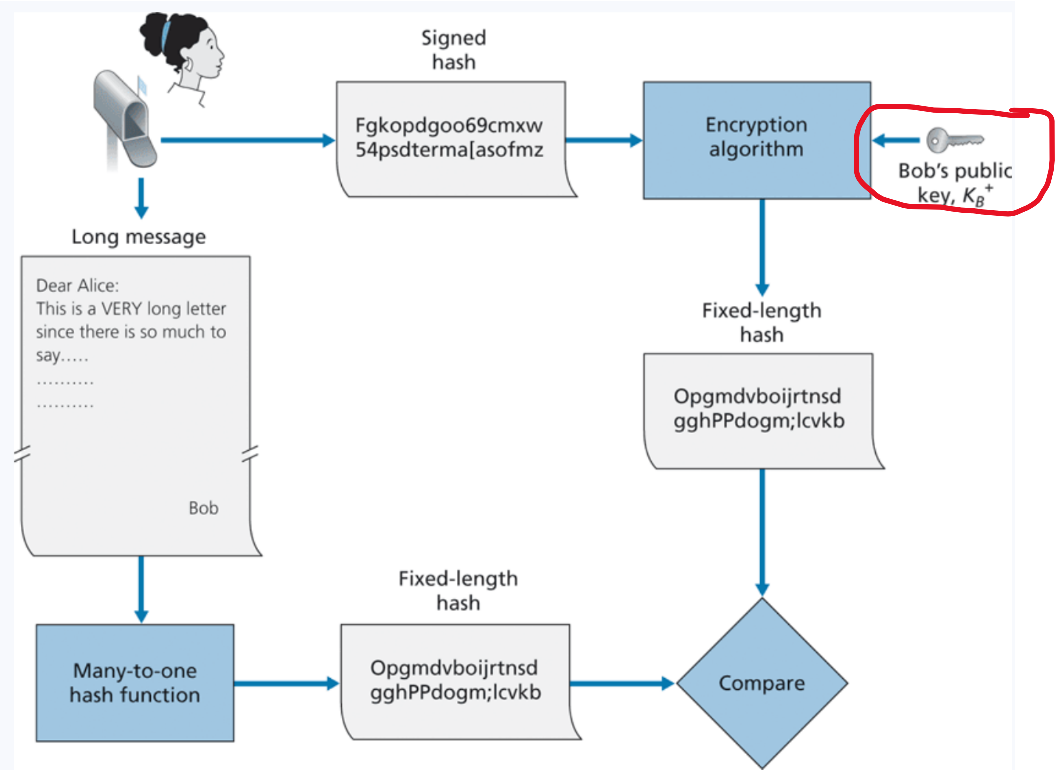
Sending a digital Signature



Verifying a digital signature

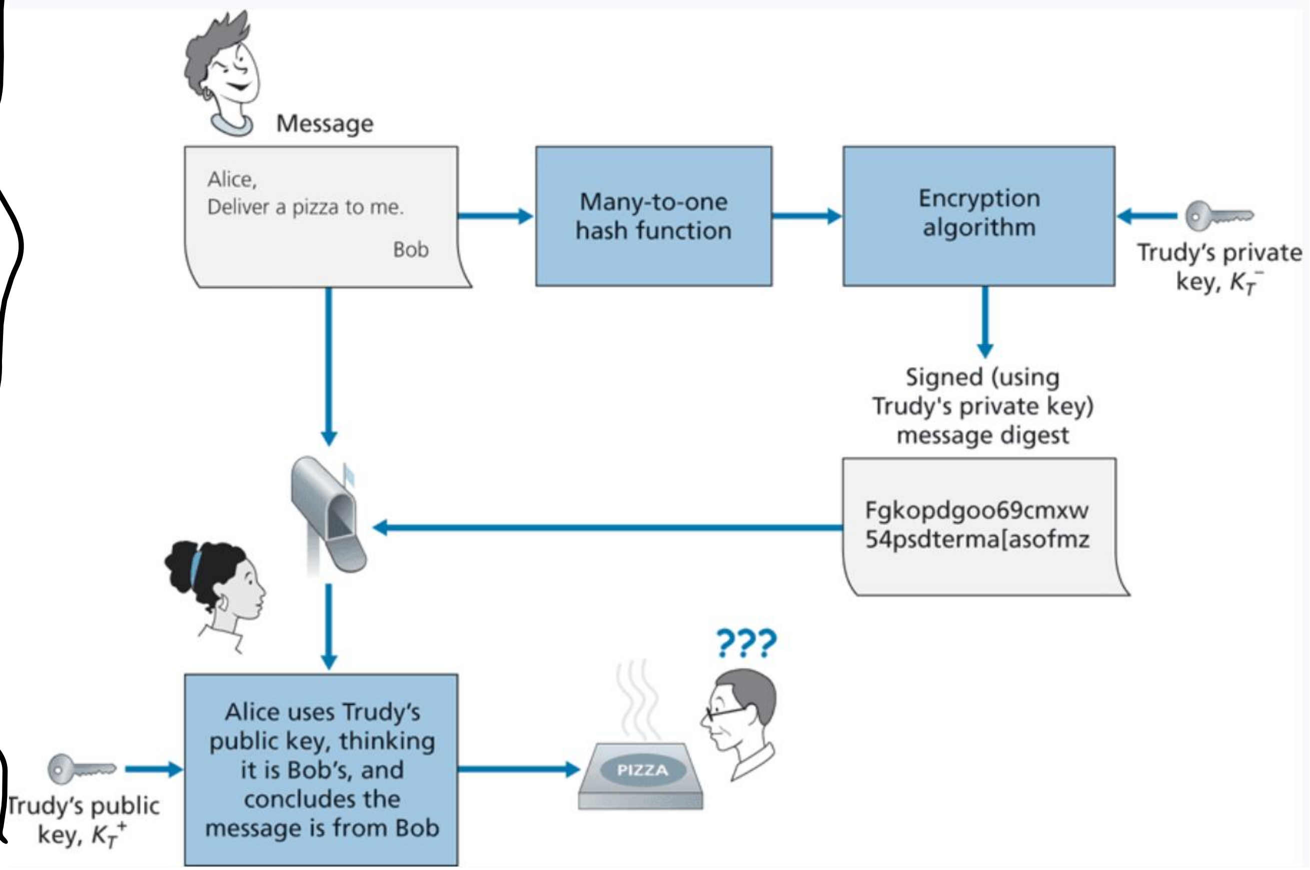


Digital Signatures



How do we know that this is **Bob's** public key ?

We don't have a way to link entities to their public keys

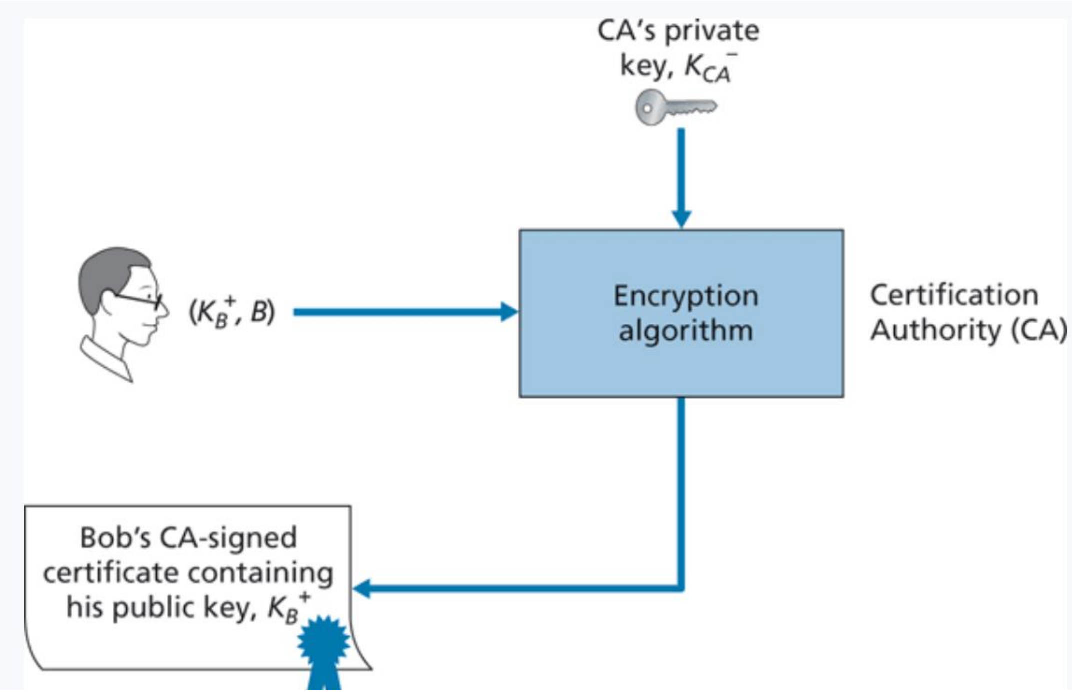


Digital Certificates

Certificates are an authoritative document that links entities (person, router, organization) to their public key

Creating certificates are done by a **Certification Authority** (digicert, lets encrypt, comodo)

Some are more trustworthy than others…



On your web browser, you exchange certificate information with the websites you are visiting

Securing Email

Symmetric Crypto, Asymmetric Crypto, and Hashing all work together to send secure, authentic messages