# CSCI 476: Computer Security

Asymmetric Cryptography

Public Key Cryptography

Reese Pearsall

Spring 2023

# Announcements

Lab 8 Due TONIGHT

Research Project due on Sunday

Tomorrow @ 6:00 I'll be giving a talk about malware detection and analysis in NAH 149 (HackerCats)

Guest Speaker: Reese Pearsall on Malware Detection/Analysis

**Date and Time**
Thursday, April 20 2023 at 6:00 PM MDT to
Thursday, April 20 2023 at 7:00 PM MDT
Add To Google Calendar | iCal/Outlook

**Location**
NAH 149

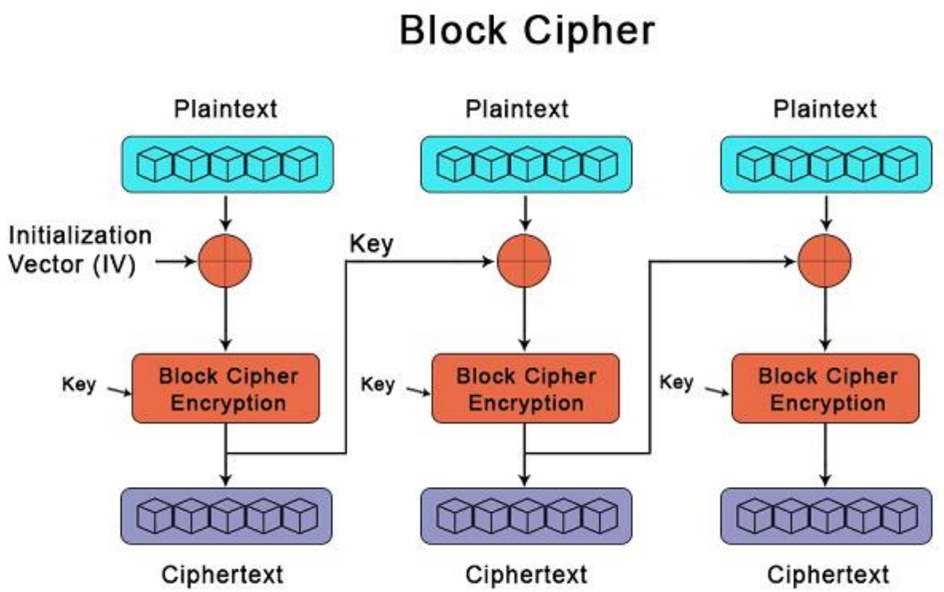
Graduating Seniors: RSVP for the pizza party by tomorrow

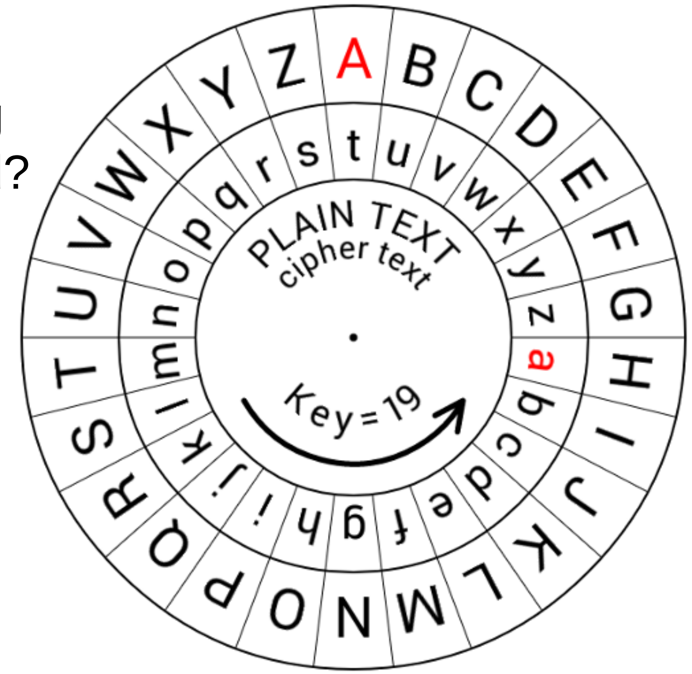
You can earn extra credit by attending lecture next week

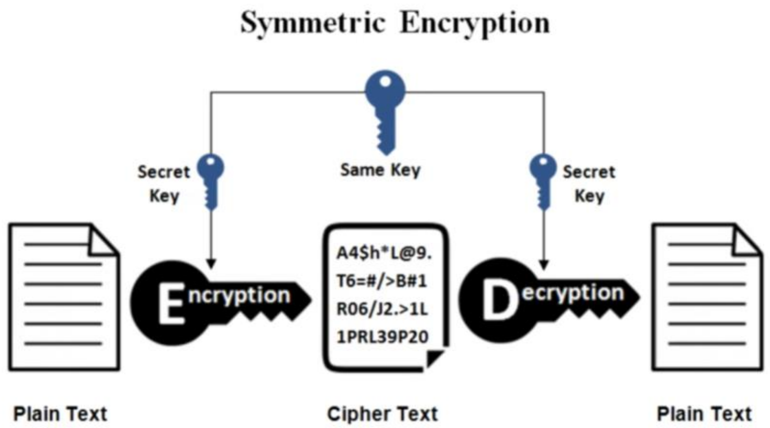
# Symmetric key encryption uses the same, **shared**, key for encrypting and decrypting

What is the one major hurdle we have not discussed yet?

How do the keys get sent without being intercepted? Do the keys get encrypted?



# Asymmetric Cryptography

AKA **Public key Cryptography**

The keys used for encrypting and decrypting data are *different*

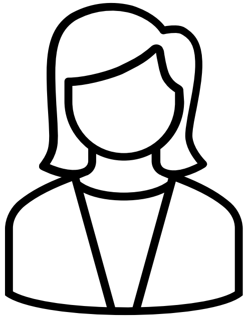
Additionally, each user now gets two-keys. A **public key**, and a **private key**

This involves some complicated math, and I won't go super deep into it. YouTube videos can explain it much better than I can

**RSA (Rivest–Shamir–Adleman)** is the most popular public key cryptosystem. We rely on it whenever we do communicate securely on the internet

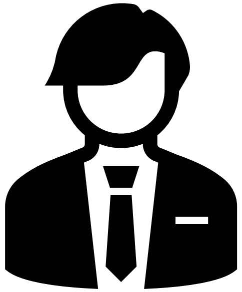
# Asymmetric Cryptography (RSA)

Alice



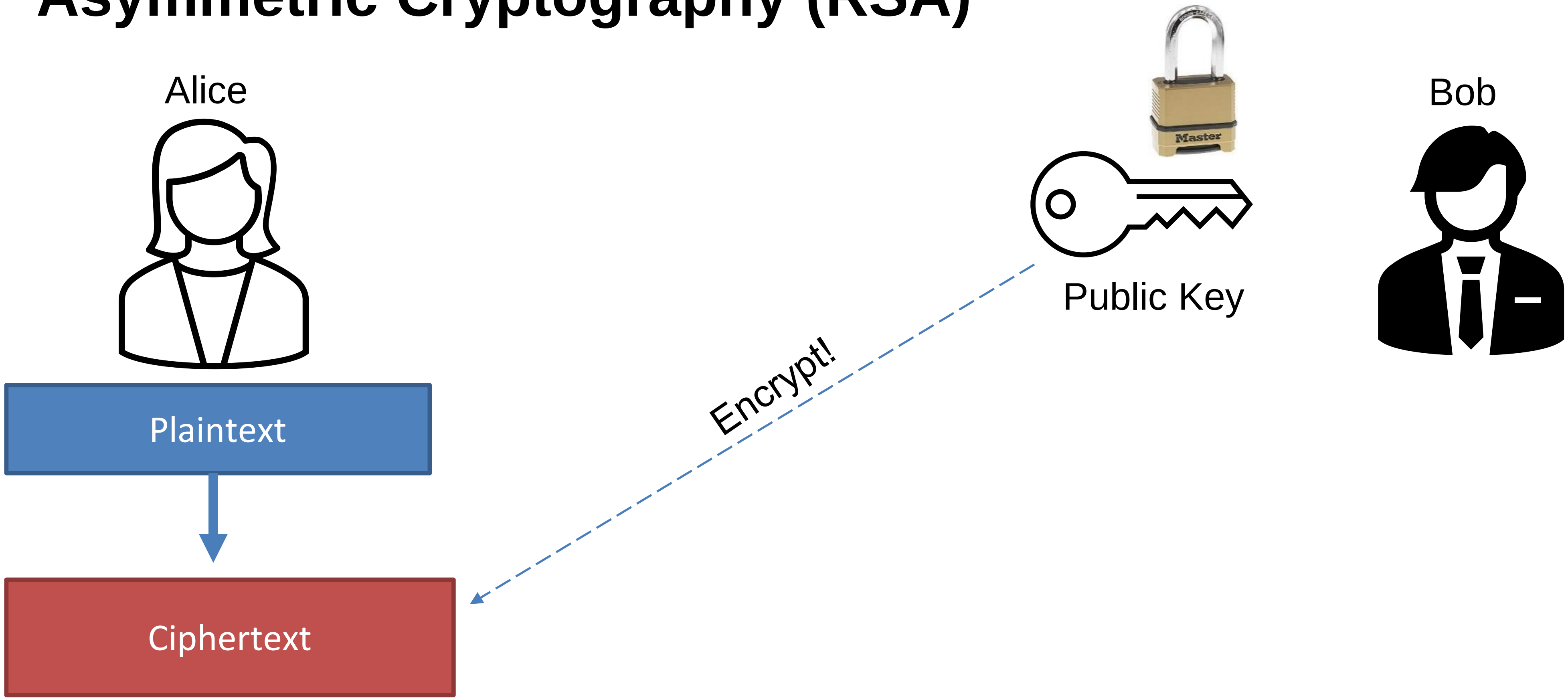
Plaintext

Bob



Alice has a plaintext that she wants to send to bob

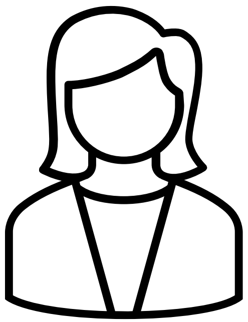
# Asymmetric Cryptography (RSA)




She uses Bob’s **public key** to encrypt her message

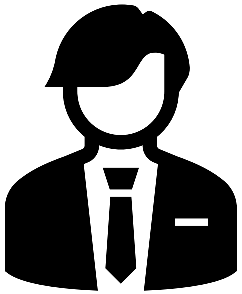
# Asymmetric Cryptography (RSA)

Alice



Public Key

Bob

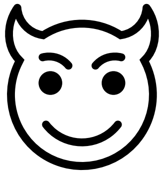


Plaintext

↓

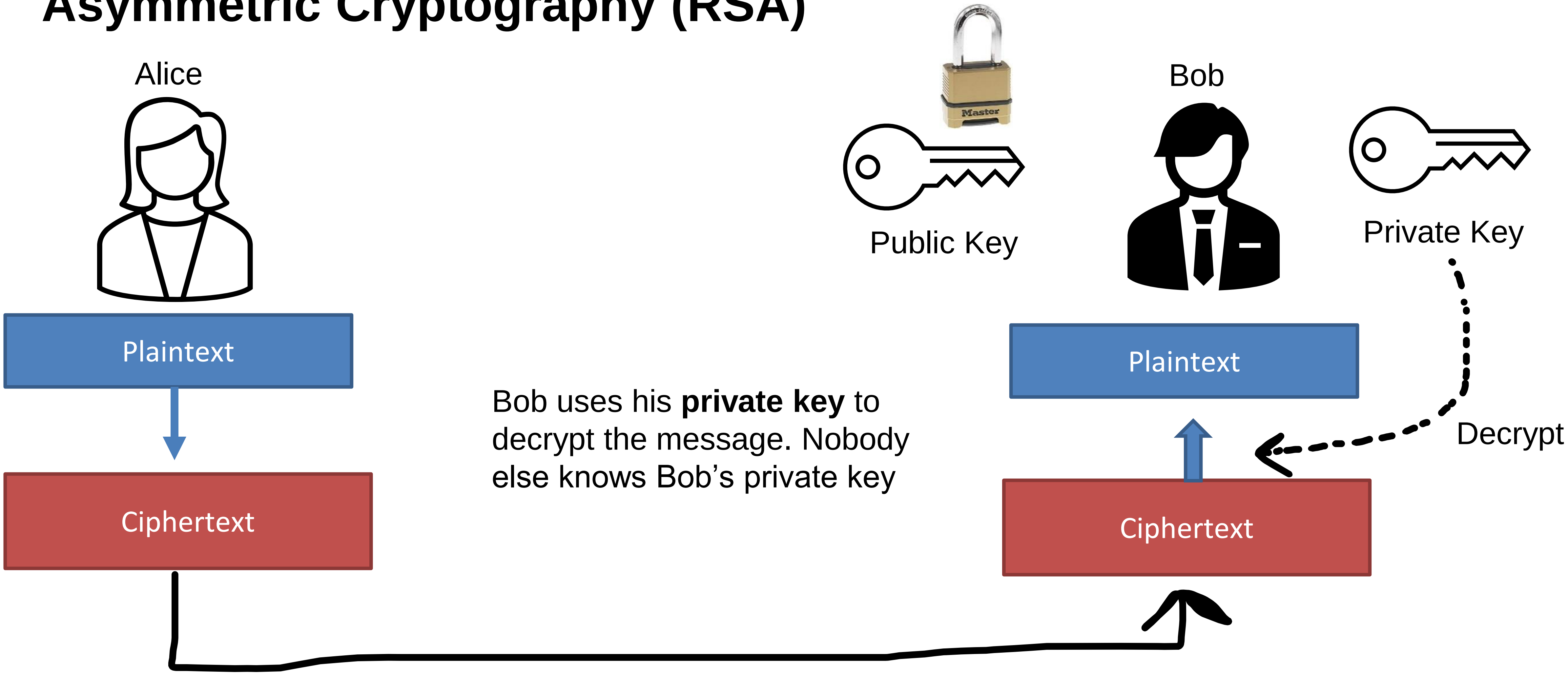
Ciphertext

Ciphertext is sent over some medium



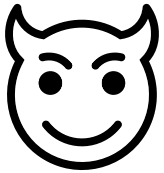
Eve can intercept this message,
But **can't** decrypt it (public key is not used for decrypting!)

# Asymmetric Cryptography (RSA)




Bob uses his **private key** to decrypt the message. Nobody else knows Bob's private key

Ciphertext is sent over some medium



Eve can intercept this message,
But **can't** decrypt it (public key is not used for decrypting!)

# Asymmetric Cryptography (RSA)

If you multiply two prime numbers (**p** and **q**) together, the product can only be divisible by those two number

5183

??? * ??? = 5183

This is very difficult to figure out for the people that don't know p or q

In fact, there is not an *efficient* program that can calculate the factors of integers

This problem is in NP

# Asymmetric Cryptography (RSA)

If you multiply two prime numbers (**p** and **q**) together, the product can only be divisible by those two number

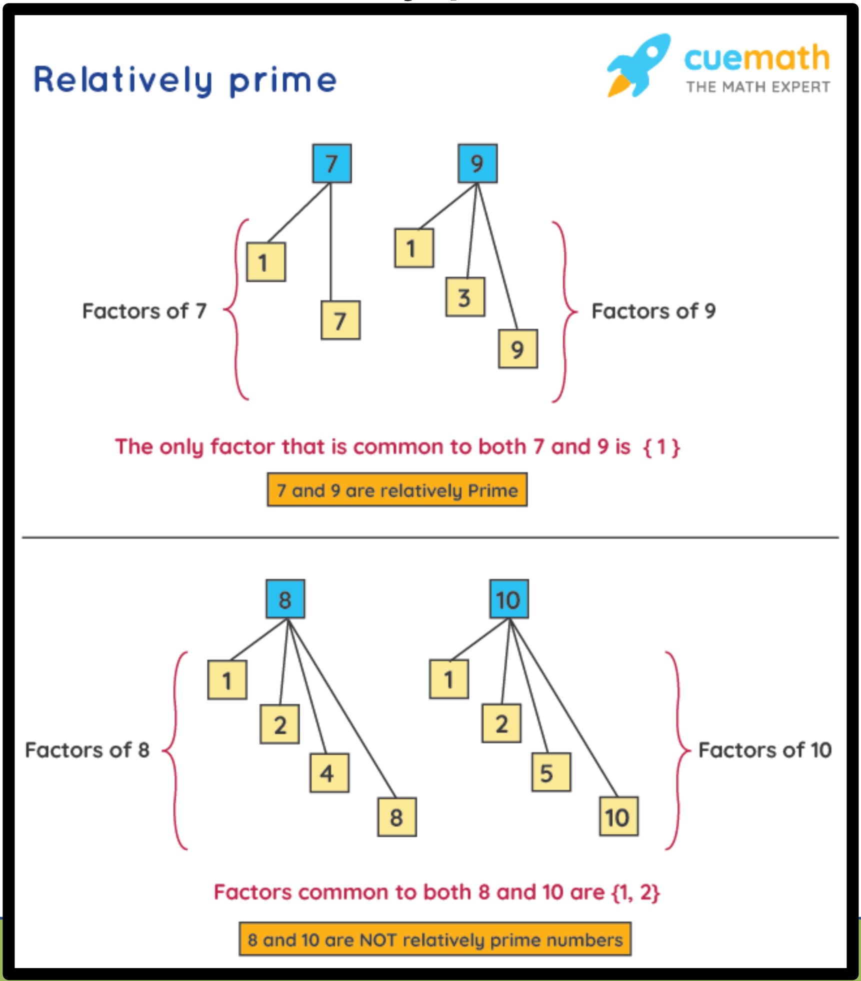
RSA is based on large numbers that are difficult to factorize
The public and private keys are derived from these prime numbers

How long should RSA keys be? 1024 or 2048 bits long!

The longer the key = the more difficult to crack (exponentially)

# Euler's Totient Function

Φ(n) = number of values less than n which are *relatively prime* to n




Φ(3127)

1
2
3
...
3125
3126

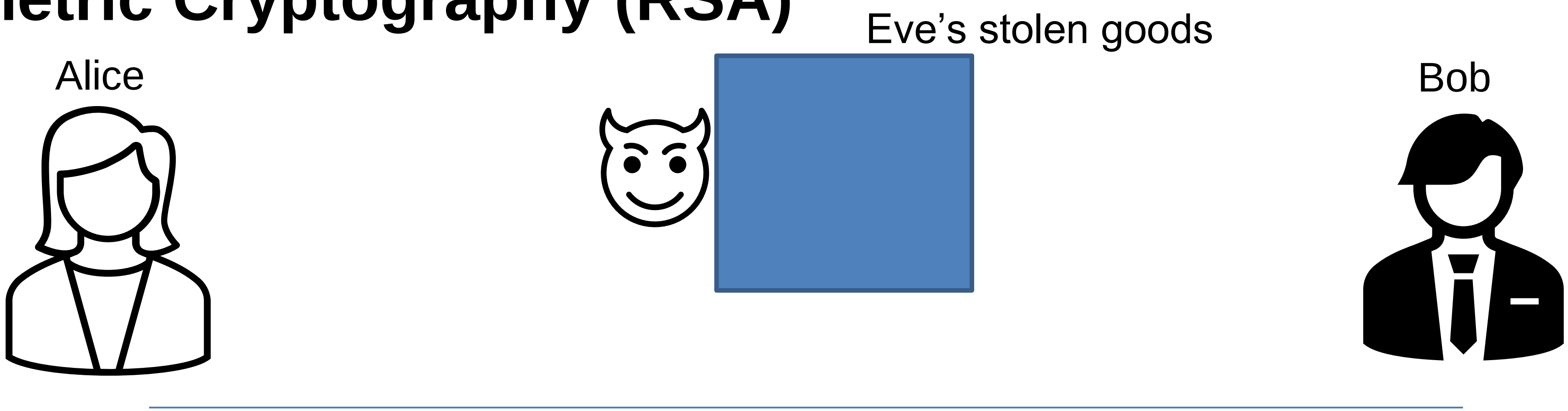
How many of these numbers are relatively prime w/ 3127?

Difficult.. But very easy for the product of two prime #s!

The Φ(n) of a product of two prime numbers will always be (p-1)(q-1)

A number is relatively prime to n if they share no common factors

# Asymmetric Cryptography (RSA)



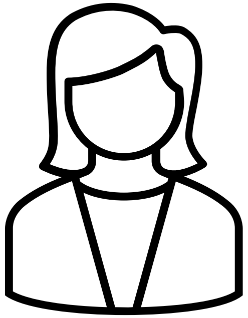

p = 53
q = 59

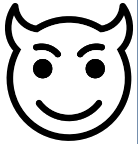
Step 1: Choose two large primer numbers, p and q

# Asymmetric Cryptography (RSA)

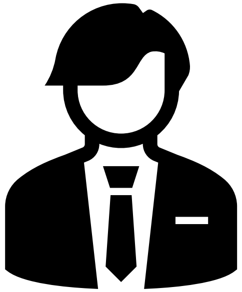
Eve’s stolen goods

Alice





Bob



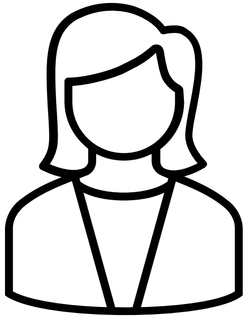
p = 53
q = 59
n = 3127

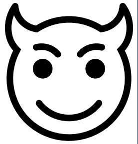
Step 1: Choose two large primer numbers, p and q
Step 2: Calculate the product n

# Asymmetric Cryptography (RSA)

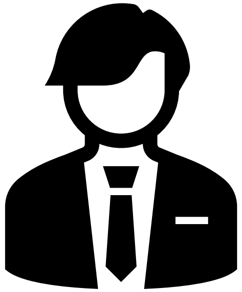
Eve's stolen goods

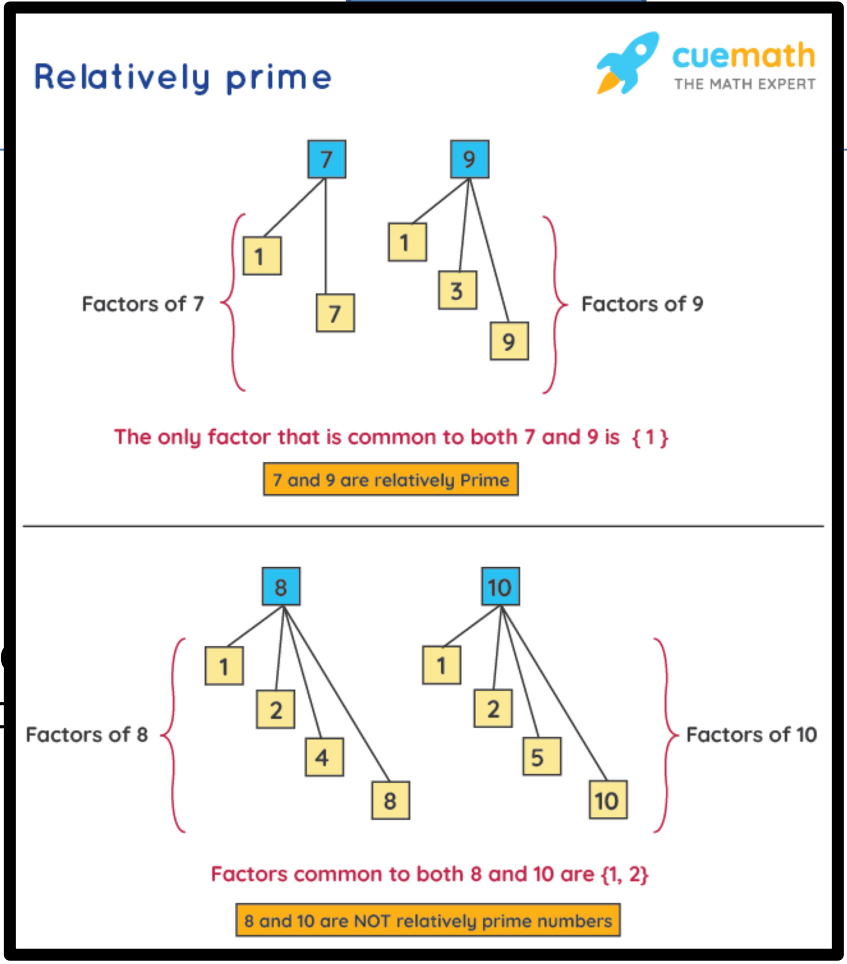
Alice





Bob






p = 53
q = 59
n = 3127

Step 1: Choose two larg
Step 2: Calculate the pro
Step 3: Calculate **Φ(n)**

Φ(n) = number of values less than n which are *relatively prime* to n

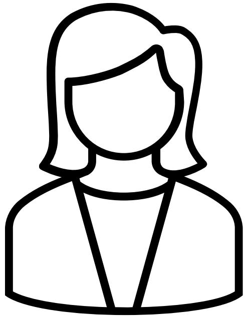
1
2
3
…
3125
3126

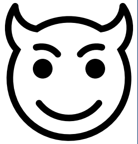
How many of these numbers are relatively prime w/ 3127?

# Asymmetric Cryptography (RSA)

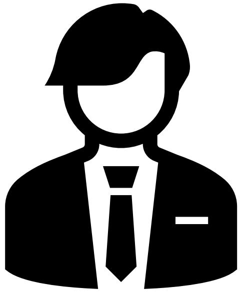
Eve's stolen goods

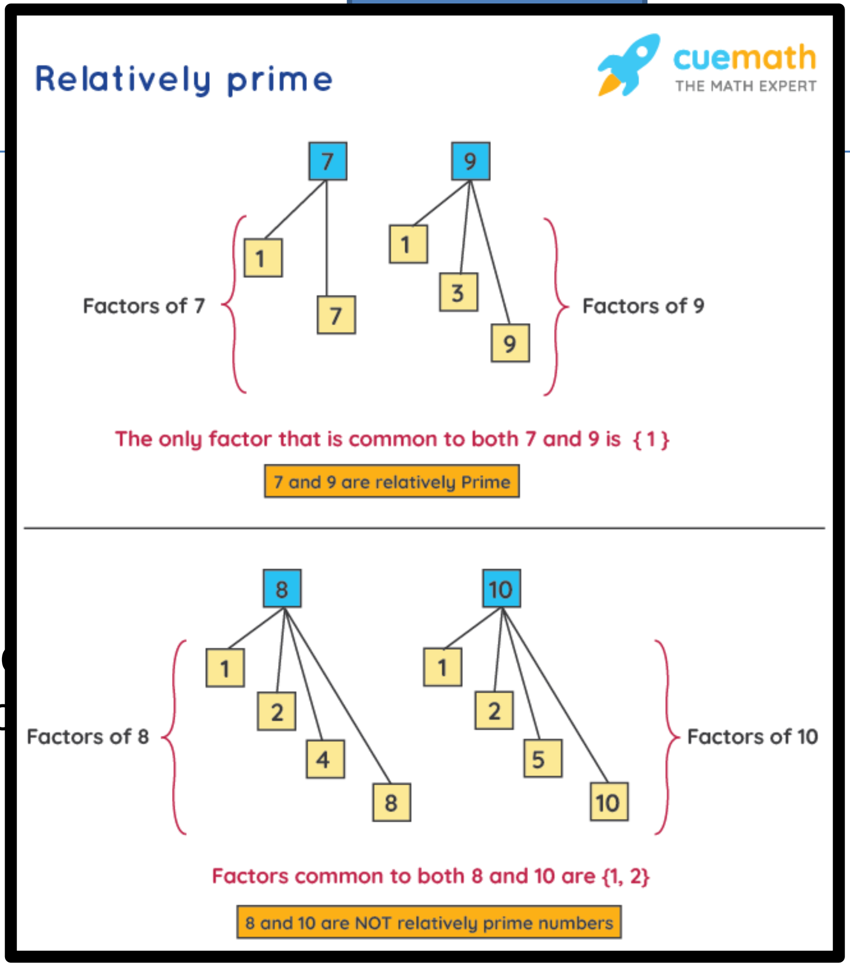
Alice





Bob






p = 53
q = 59
n = 3127

Step 1: Choose two larg
Step 2: Calculate the pro
Step 3: Calculate **Φ(n)**

Φ(n) = number of values less than n which are *relatively prime* to n

1
2
3
…
3125
3126

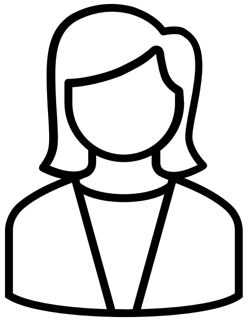
How many of these numbers are relatively prime w/ 3127?

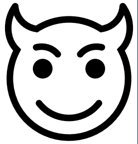
Difficult.. But very easy for the product of two prime #s!

# Asymmetric Cryptography (RSA)

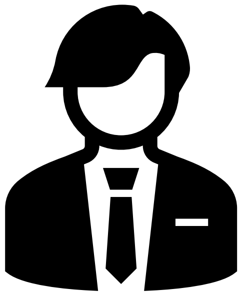
Eve's stolen goods

Alice





Bob



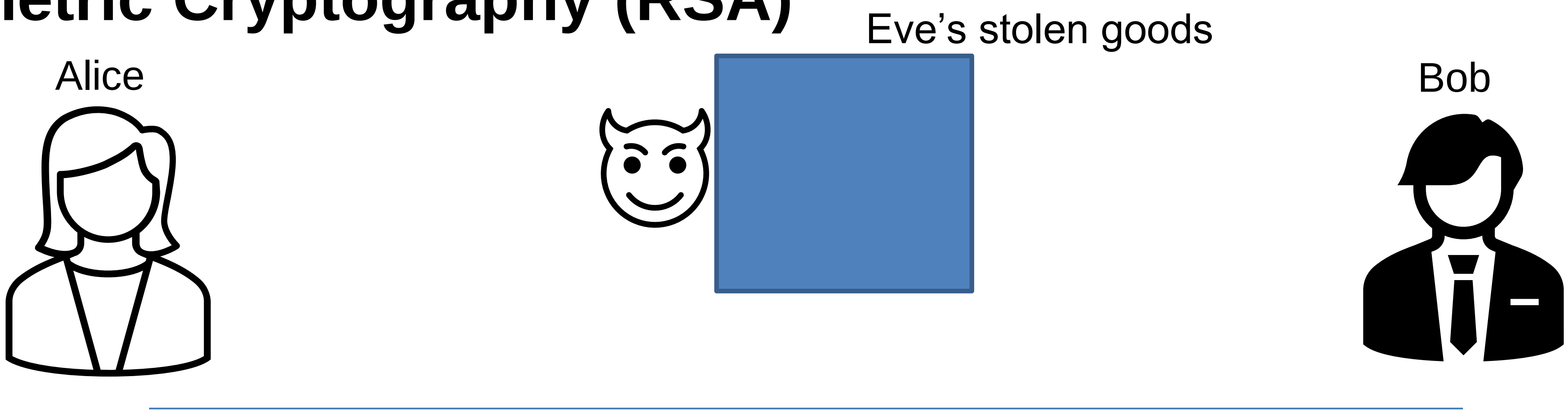
$p = 53$
$q = 59$
$n = 3127$

$\Phi(n)$ = number of values less than n which are *relatively prime* to n

The $\Phi(n)$ of a product of two prime numbers will always be $(p-1)(q-1)$

Step 1: Choose two large primer numbers, p and q
Step 2: Calculate the product n
Step 3: Calculate **$\Phi(n)$**

# Asymmetric Cryptography (RSA)




p = 53
q = 59
n = 3127
Φ(n) = 52*28 = 3016

Φ(n) = number of values less than n which are *relatively prime* to n

The Φ(n)  of a product of two prime numbers will always be (p-1)(q-1)

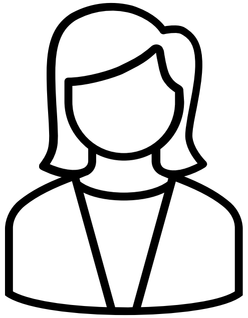
Step 1: Choose two large primer numbers, p and q
Step 2: Calculate the product n
Step 3: Calculate **Φ(n)**

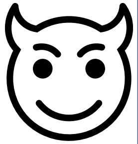
# Asymmetric Cryptography (RSA)

Φ(n) = number of values less than n which are *relatively prime* to n

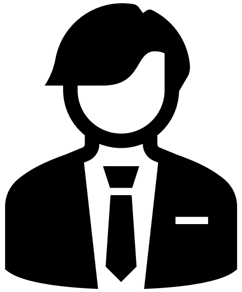
Eve's stolen goods

Alice





Bob



p = 53
q = 59
n = 3127
Φ(n) = 3016

e = 1 < e < **Φ(n)**
Not be a factor of n, but an integer

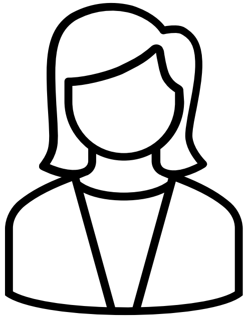
Step 1: Choose two large primer numbers, p and q
Step 2: Calculate the product n
Step 3: Calculate **Φ(n)**
Step 4: Choose public exponent **e**

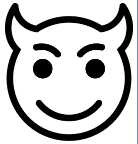
# Asymmetric Cryptography (RSA)

$\Phi(n)$ = number of values less than n which are *relatively prime* to n

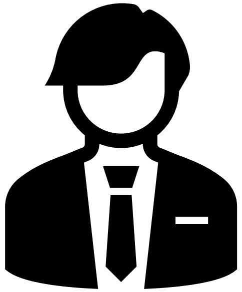
Eve's stolen goods

Alice





Bob



$p = 53$
$q = 59$
$n = 3127$
$\Phi(n) = 3016$
$e = 3$

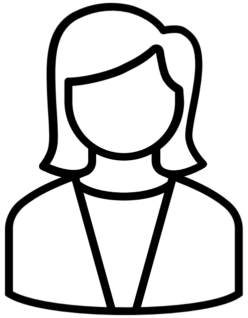
$e = 1 < e < \boldsymbol{\Phi(n)}$
Not be a factor of n, but an integer

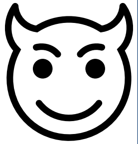
Step 1: Choose two large primer numbers, p and q
Step 2: Calculate the product n
Step 3: Calculate **Φ(n)**
Step 4: Choose public exponent **e**

# Asymmetric Cryptography (RSA)

Φ(n) = number of values less than n which are *relatively prime* to n

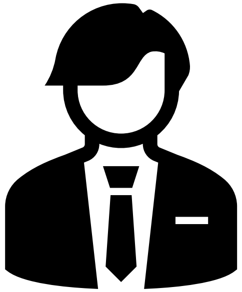
Alice





Eve's stolen goods

Bob



p = 53
q = 59
n = 3127
Φ(n) = 3016
e = 3

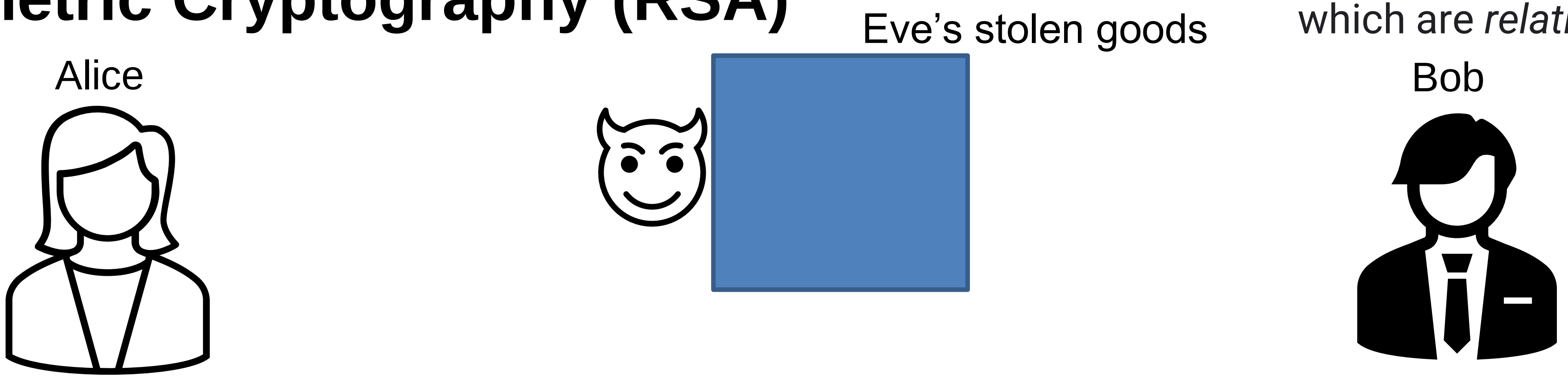
$$d = \frac{K * \Phi(n) + 1}{e}$$

K = some integer that will make the quotient an integer

Step 1: Choose two large primer numbers, p and q
Step 2: Calculate the product n
Step 3: Calculate **Φ(n)**
Step 4: Choose public exponent **e**
Step 5: Select private exponent d

# Asymmetric Cryptography (RSA)

Φ(n) = number of values less than n which are *relatively prime* to n




p = 53
q = 59
n = 3127
Φ(n) = 3016
e = 3

$$d = \frac{2 * 3016 + 1}{3}$$

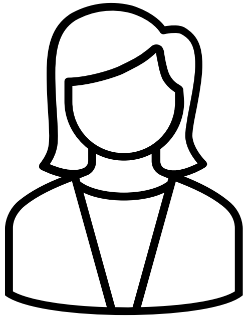
Step 1: Choose two large primer numbers, p and q
Step 2: Calculate the product n
Step 3: Calculate **Φ(n)**
Step 4: Choose public exponent **e**
Step 5: Select private exponent d

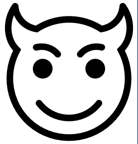
K = some integer that will make the quotient an integer

# Asymmetric Cryptography (RSA)

$\Phi(n)$ = number of values less than n which are *relatively prime* to n

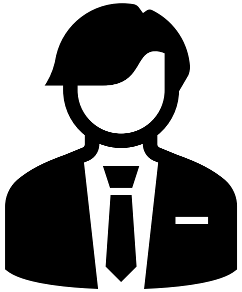
Alice





Eve's stolen goods

Bob



p = 53
q = 59
n = 3127
$\Phi(n)$ = 3016
e = 3
d = 2011

$$d = \frac{2 * 3016 + 1}{3}$$

Step 1: Choose two large primer numbers, p and q
Step 2: Calculate the product n
Step 3: Calculate **$\Phi$(n)**
Step 4: Choose public exponent **e**
Step 5: Select private exponent d

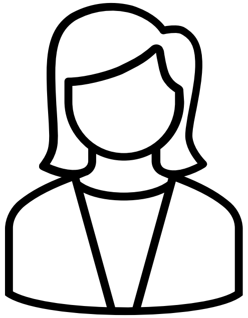
K = some integer that will make the quotient an integer

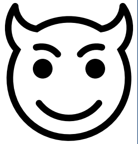
# Asymmetric Cryptography (RSA)

Φ(n) = number of values less than n which are *relatively prime* to n

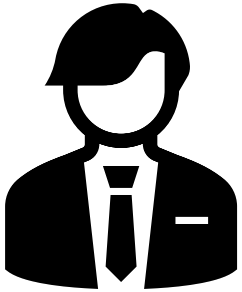
Eve's stolen goods

Alice





Bob



Alice's Public Key

n = 3127
e = 3

Secret Information

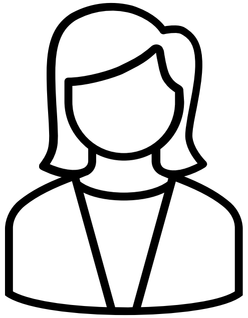
p = 53
q = 59
Φ(n) = 3016
d = 2011

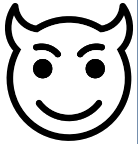
# Asymmetric Cryptography (RSA)

$\Phi(n)$ = number of values less than n which are *relatively prime* to n

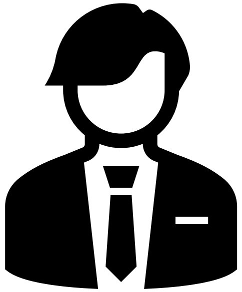
Eve's stolen goods

Alice





Bob



Alice's Public Key

n = 3127
e = 3

Bob has a message to send to Alice

**HI → 89**

Message must be converted into a number

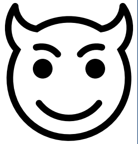
Secret Information

p = 53
q = 59
$\Phi(n)$ = 3016
d = 2011

# Asymmetric Cryptography (RSA)

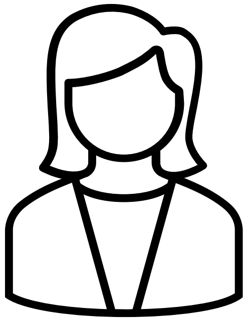
Φ(n) = number of values less than n which are *relatively prime* to n

Eve's stolen goods

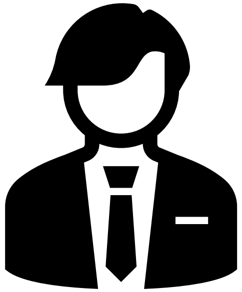


n = 3127
e = 3

Alice



Bob



Alice's Public Key

n = 3127
e = 3

Bob has a message to send to Alice

**89**

Use Alice's Public Key to encrypt

$$m^e \bmod 3127$$

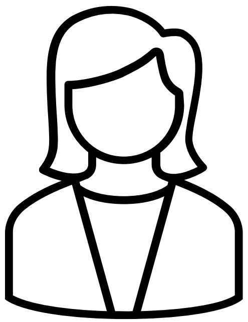
Secret Information

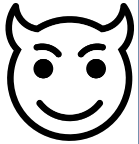
p = 53
q = 59
Φ(n) = 3016
d = 2011

# Asymmetric Cryptography (RSA)

$\Phi(n)$ = number of values less than n which are *relatively prime* to n

Alice

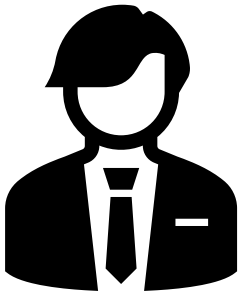




Eve's stolen goods

n = 3127
e = 3

Bob



Alice's Public Key

n = 3127
e = 3

Bob has a message to send to Alice

**89**

Use Alice's Public Key to encrypt

$89^3 \bmod 3127$

$C = 1394$

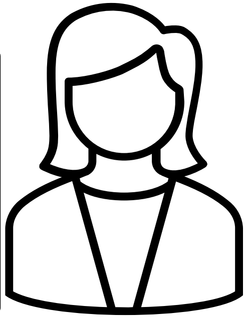
Secret Information

p = 53
q = 59
$\Phi(n)$ = 3016
d = 2011

# Asymmetric Cryptography (RSA)

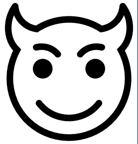
$\Phi(n)$ = number of values less than n which are *relatively prime* to n

Alice



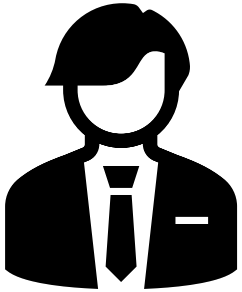
p = 53
q = 59
$\Phi(n) = 3016$
d = 2011

Eve's stolen goods



n = 3127
e = 3
c=1394

Bob



Alice's Public Key

n = 3127
e = 3

Bob has a message to send to Alice

**89**

Use Alice's Public Key to encrypt

$89^3 \bmod 3127$

$C = 1394$

$1394^{2011} \bmod 3127$

Alice decrypts message using her private key

# Asymmetric Cryptography (RSA)

$\Phi(n)$ = number of values less than n which are *relatively prime* to n

Alice

p = 53
q = 59
$\Phi(n)$ = 3016
d = 2011

Eve's stolen goods

n = 3127
e = 3
c=1394

Bob

Alice's Public Key

n = 3127
e = 3

Bob has a message to send to Alice

**89**

Use Alice's Public Key to encrypt

$89^3 \bmod 3127$

$C = 1394$

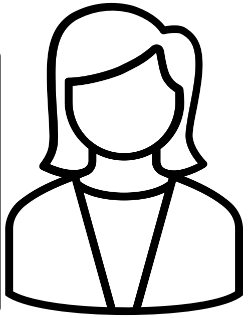
$1394^{2011} \bmod 3127 = \mathbf{89}$ ✓

Alice decrypts message using her private key

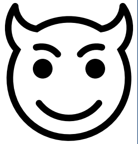
# Asymmetric Cryptography (RSA)

$\Phi(n)$ = number of values less than n which are *relatively prime* to n

Alice



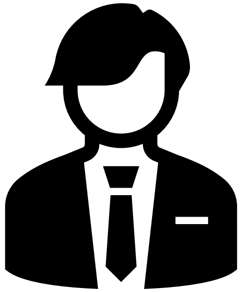
p = 53
q = 59
$\Phi(n)$ = 3016
d = 2011



Eve's stolen goods

n = 3127
e = 3
c=1394

Bob



Alice's Public Key

n = 3127
e = 3

Bob has a message to send to Alice

**89**

Alice's Private Key

n = 3127
d = 2011

What does eve know??

$???^3 \bmod 3127 = 1394$

These is very difficult to figure out, since she does not know the factorization of n

Use Alice's Public Key to encrypt

$89^3 \bmod 3127$

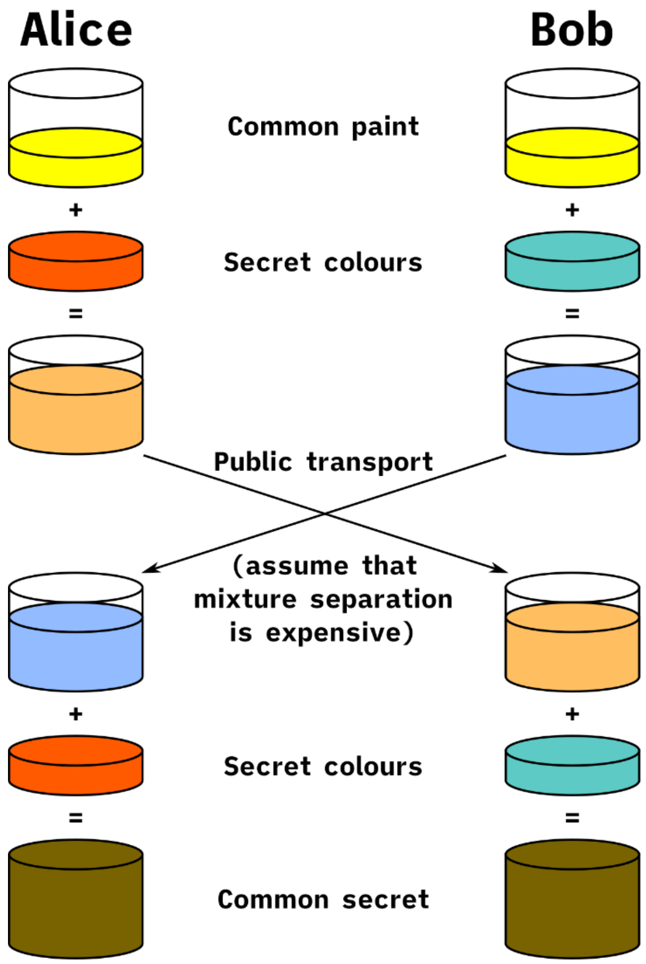
$C = 1394$

# Asymmetric Cryptography (RSA)

We now have a method for sending secure messages over a possibly unsecure channel!

This method is known as the **Diffie Helman Key Exchange**

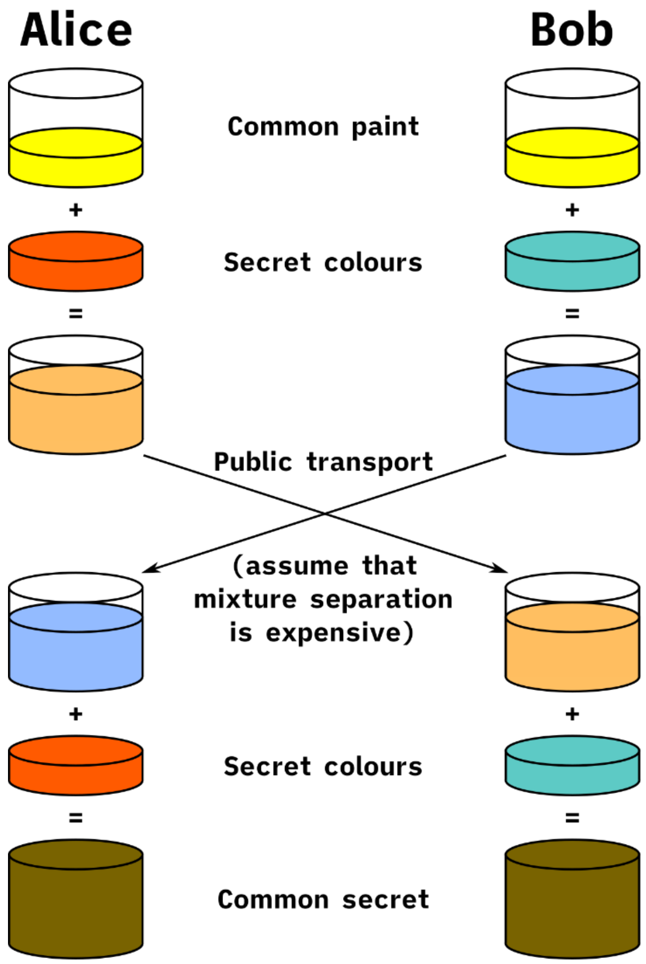
RSA is built on DHKE to create an encryption/decryption algorithm



# Asymmetric Cryptography (RSA)

We now have a method for sending secure messages over a possibly unsecure channel!

Limitation of RSA: Can only encrypted data that is smaller or equal to key length ( < 2048 bits )

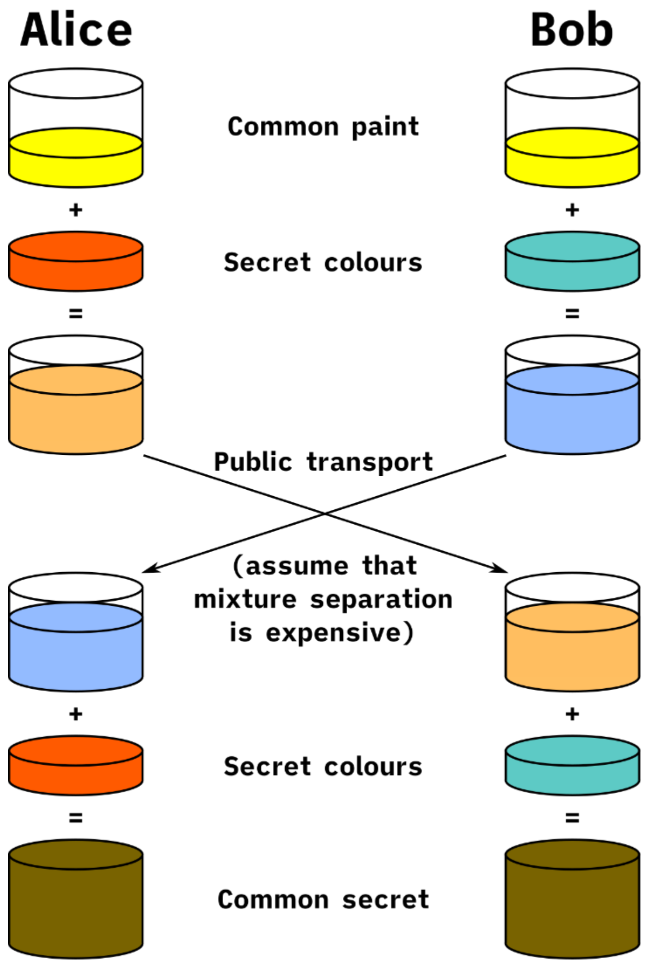


# Asymmetric Cryptography (RSA)

We now have a method for sending secure messages over a possibly unsecure channel!

Limitation of RSA: Can only encrypted data that is smaller or equal to key length ( < 2048 bits )

What could we encrypt instead??



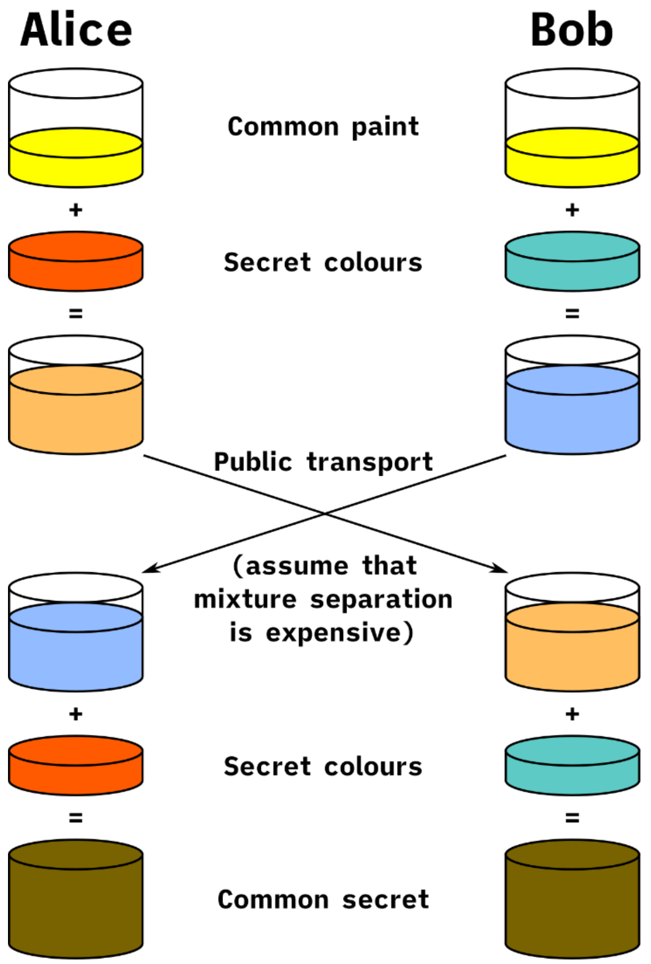
# Asymmetric Cryptography (RSA)

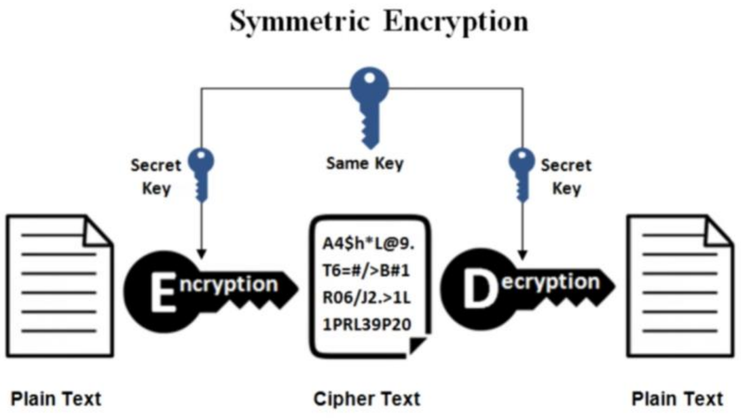
We now have a method for sending secure messages over a possibly unsecure channel!

Limitation of RSA: Can only encrypted data that is smaller or equal to key length ( < 2048 bits )

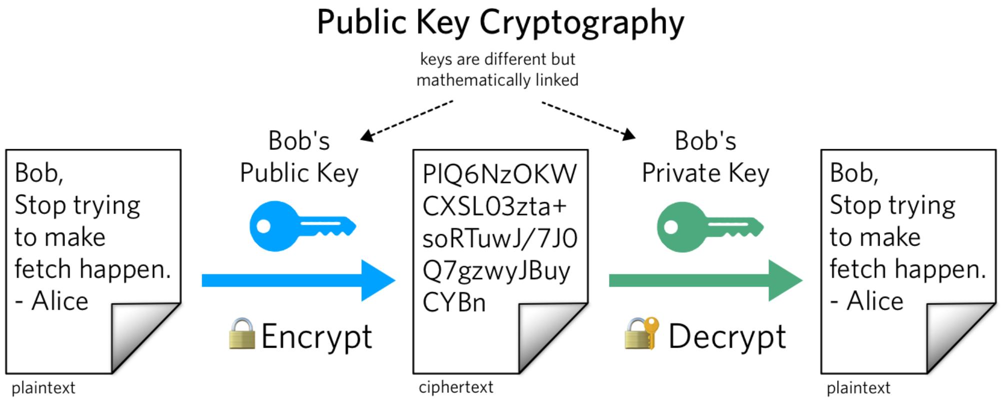
What could we encrypt instead??

The key for a symmetric cryptography algorithm! ( < 2048 bits)






- 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!)

We know that Public and Private keys are derived from big prime numbers

(We are talking hundreds of digits long…)

Our computer can't compute products and exponents for such large numbers

OpenSSL on our VMs has tools for generating public/private RSA keys

```
[11/29/22]seed@VM:~$ sudo openssl genrsa -aes128 -out private.pem 1024
```

**Example:** generate a 1024-bit public/private key pair

- Use **openssl genrsa** to generate a file, **private.pem**
- private.pem is a Base64 encoding of DER generated binary output

```
$ openssl genrsa -aes128 -out private.pem 1024 # passphrase csci476
$ more private.pem
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,C30BF6EB3FD6BA9A81CCB9202B95EC1A

sLIQ7Fs5j5zOexdWkZUoiv2W82g03gNERmfG+fwnVnbsIZAuW8E9wiB7tqz8rEL+
xfL+U20lyQNxpmOTUeKlN3qCcJROcGYSNd1BeNpgLWV1bN5FPYce9GRb4tFr4bhK
...
RPtJNKUryhVnAC4a3gp0gcXk1IQLeHeyKQCPQ1SckQRdrBzHjjCNN42NlCVEpcsF
WJ8ikqDd9FslGHc1PT6ktW5oV9cB8G2wfo7D85n91SQfSzuwAcyx7Ecir1o4PfKG
-----END RSA PRIVATE KEY——
```

# The *actual* content of **`private.pem`**:

```
$ openssl rsa -in private.pem -noout -text
Enter pass phrase for private.pem: csci476
Private-Key: (1024 bit)
modulus:
    00:b8:52:5c:25:cc:7c:f2:ef:a6:35:9d:de:3d:5d: ...
publicExponent: 65537 (0x10001)
privateExponent:
    4b:0d:ce:53:dd:e6:6b:0d:c6:82:42:9c:42:24:a7: ...
prime1:
    00:ef:14:46:57:9c:d0:4c:98:de:c3:0b:aa:d8:72: ...
prime2:
    00:c5:5d:f8:0b:f9:75:dc:88:ea:d4:d0:56:ee:f9: ...
exponent1:
    00:e6:49:9a:44:14:19:94:5e:7f:dc:52:65:bb:5d: ...
exponent2:
    7c:ad:77:dc:58:a2:13:c6:8a:52:15:aa:55:1c:22: ...
coefficient:
    3a:7c:b9:a0:12:e8:fa:88:b8:6f:38:4a:ed:bc:17: ...
```

# The *actual* content of `public.pem`:

```
$ openssl rsa -in private.pem -pubout > public.pem
Enter pass phrase for private.pem: csci476
writing RSA key
$ more public.pem
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC4UlwlzHzy76Y1nd49XakNUwqJ
Ud3ph0uBWWfnLnjIYgQL/spg9WE+1Q1YPp2t3FBFljhGHdWMA8abfNXG4jmpD+uq
Ix0WVyXg12WWilkY2/vs8xI1K+PumWTtq8R8ueAq7RzETc3873DO1vjMxXWqau7k
zIkUuJ/JCjzjYfbsDQIDAQAB
-----END PUBLIC KEY-----
```

```
$ openssl rsa -in public.pem -pubin -text -noout
Public-Key: (1024 bit)
Modulus:
    00:b8:52:5c:25:cc:7c:f2:ef:a6:35:9d:de:3d:5d: ...
Exponent: 65537 (0x10001)
```

n (beside "Modulus:")
e (beside "Exponent:")

→ (e,n) = public key!

# OpenSSL Tools: Encryption and Decryption

- Create a plaintext message:

```
$ echo "This is a secret." > msg.txt
```

- Encrypt the plaintext:

```
$ openssl rsautl -encrypt -inkey public.pem -pubin -in msg.txt -out msg.enc
```

# OpenSSL Tools: Encryption and Decryption

- Create a plaintext message:

```
$ echo "This is a secret." > msg.txt
```

- Encrypt the plaintext:

```
$ openssl rsautl -encrypt -inkey public.pem -pubin -in msg.txt -out msg.enc
```

- Decrypt the ciphertext:

```
$ openssl rsautl -decrypt -inkey private.pem -in msg.enc
Enter pass phrase for private.pem: csci476
This is a secret.
```

# OpenSSL Tools: Encryption and Decryption

- Create a plaintext message:

```
$ echo "This is a secret." > msg.txt
```

- Encrypt the plaintext:

```
$ openssl rsautl -encrypt -inkey public.pem -pubin -in msg.txt -out msg.enc
```

- Decrypt the ciphertext:

```
$ openssl rsautl -decrypt -inkey private.pem -in msg.enc
Enter pass phrase for private.pem: csci476
This is a secret.
```

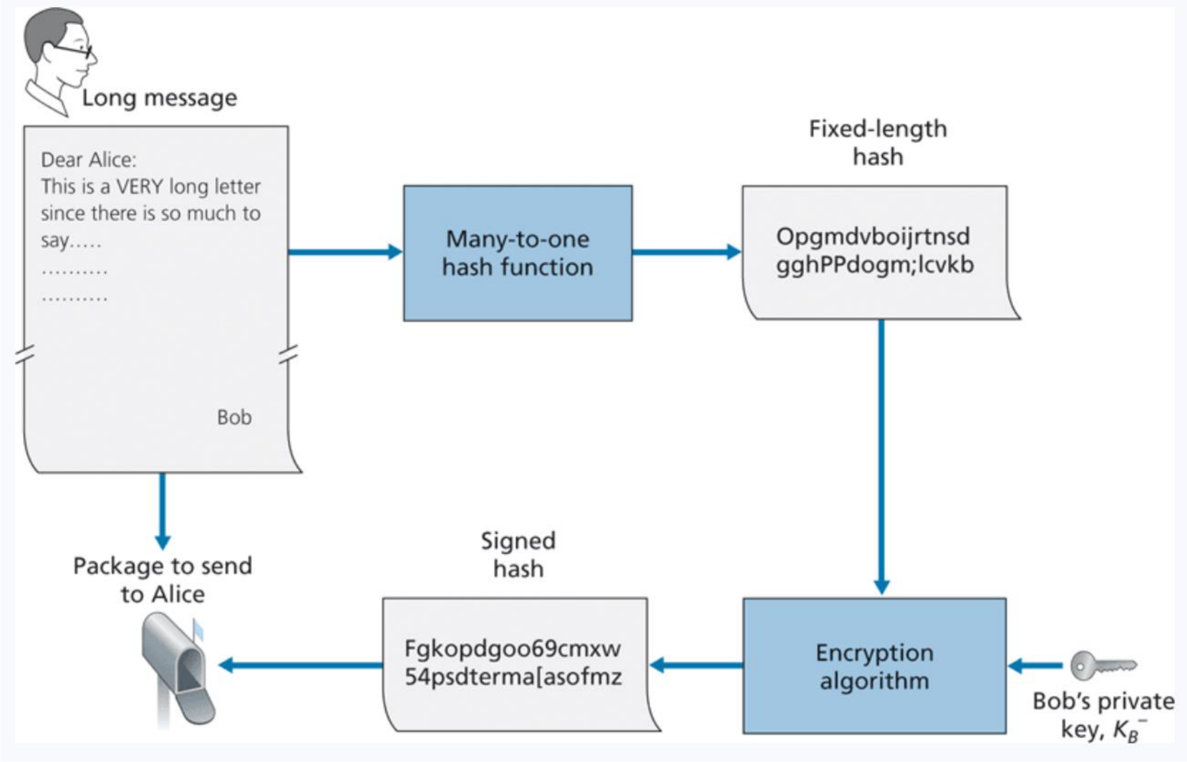
# BIG NUM API

```
int main ()
{
  BN_CTX *ctx = BN_CTX_new();

  BIGNUM *p, *q, *n, *phi, *e, *d, *m, *c, *res;
  BIGNUM *new_m, *p_minus_one, *q_minus_one;
  p = BN_new(); q = BN_new(); n = BN_new(); e = BN_new();
  d = BN_new(); m = BN_new(); c = BN_new();
  res = BN_new(); phi = BN_new(); new_m = BN_new();
  p_minus_one = BN_new(); q_minus_one = BN_new();

  // Set the public key exponent e
  BN_dec2bn(&e, "65537");

  // Generate random p and q.
  BN_generate_prime_ex(p, NBITS, 1, NULL, NULL, NULL);
  BN_generate_prime_ex(q, NBITS, 1, NULL, NULL, NULL);
  BN_sub(p_minus_one, p, BN_value_one());     // Compute p-1
```

# 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

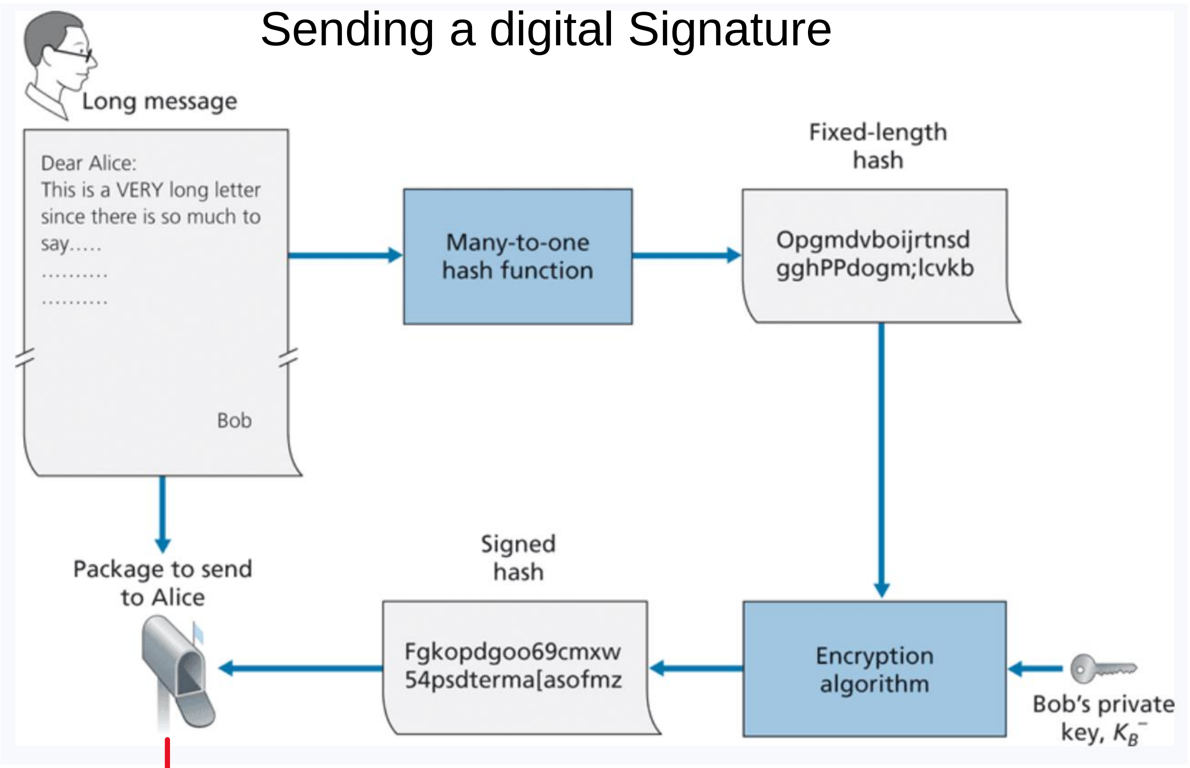
She will use Bob's **public key**

# Digital Signatures

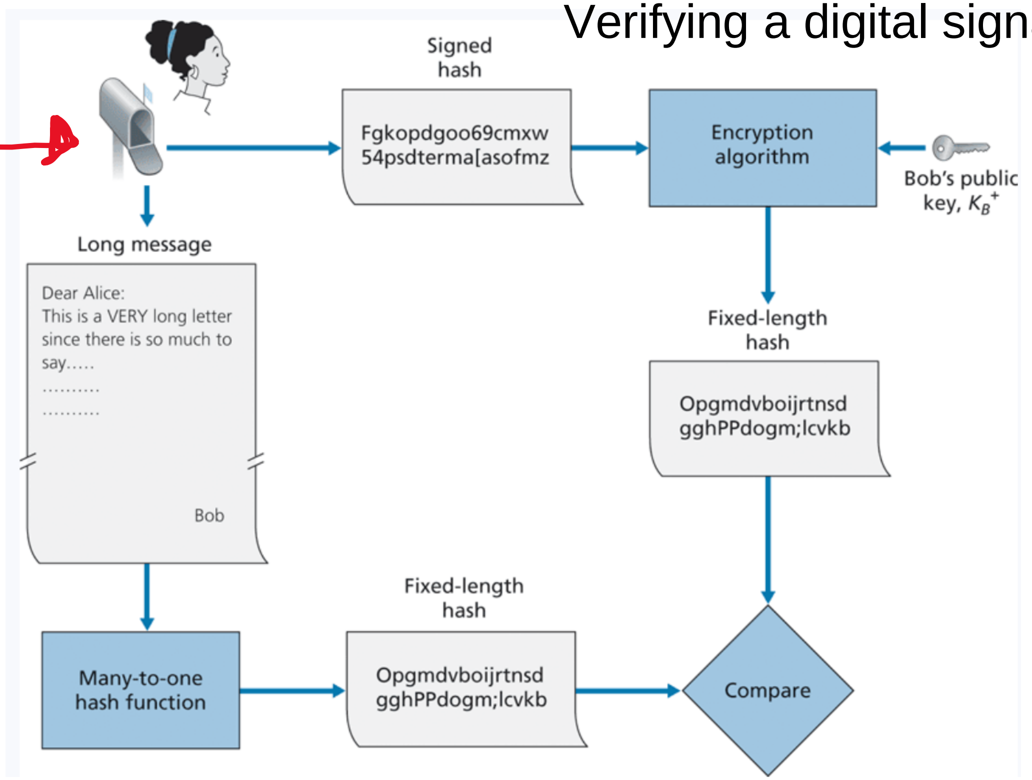
- What is a unique identifier for bob? What is something that only bob knows and nobody else?
- ➢ His **private key**

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

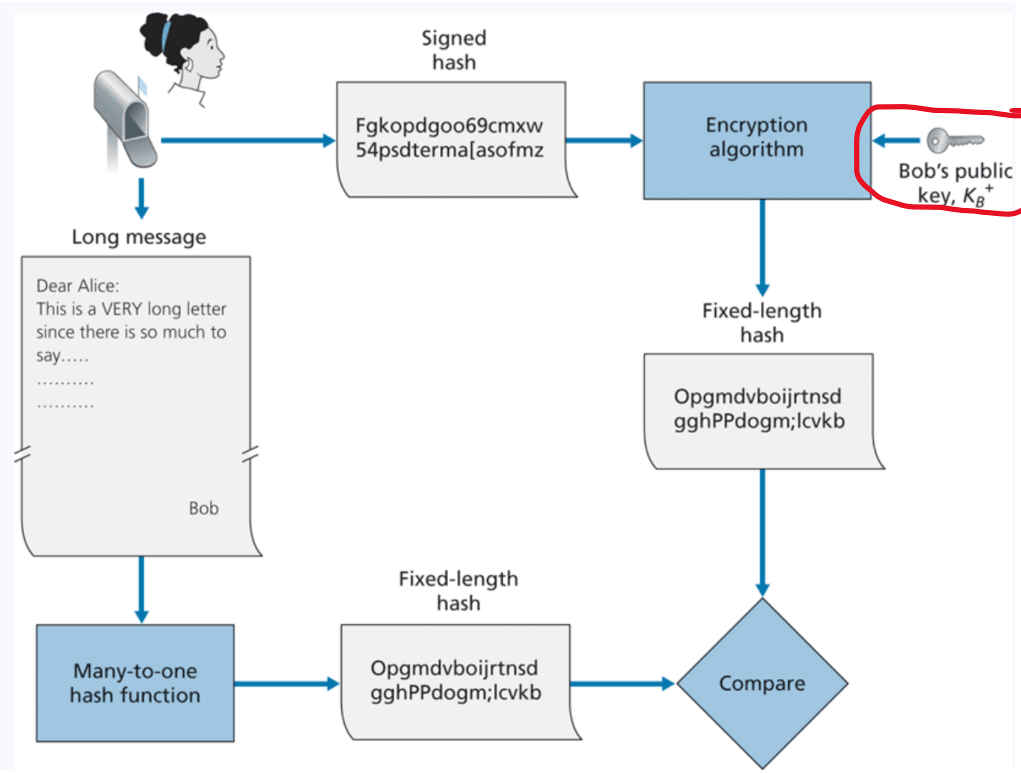
Sending a digital Signature




Verifying a digital signature

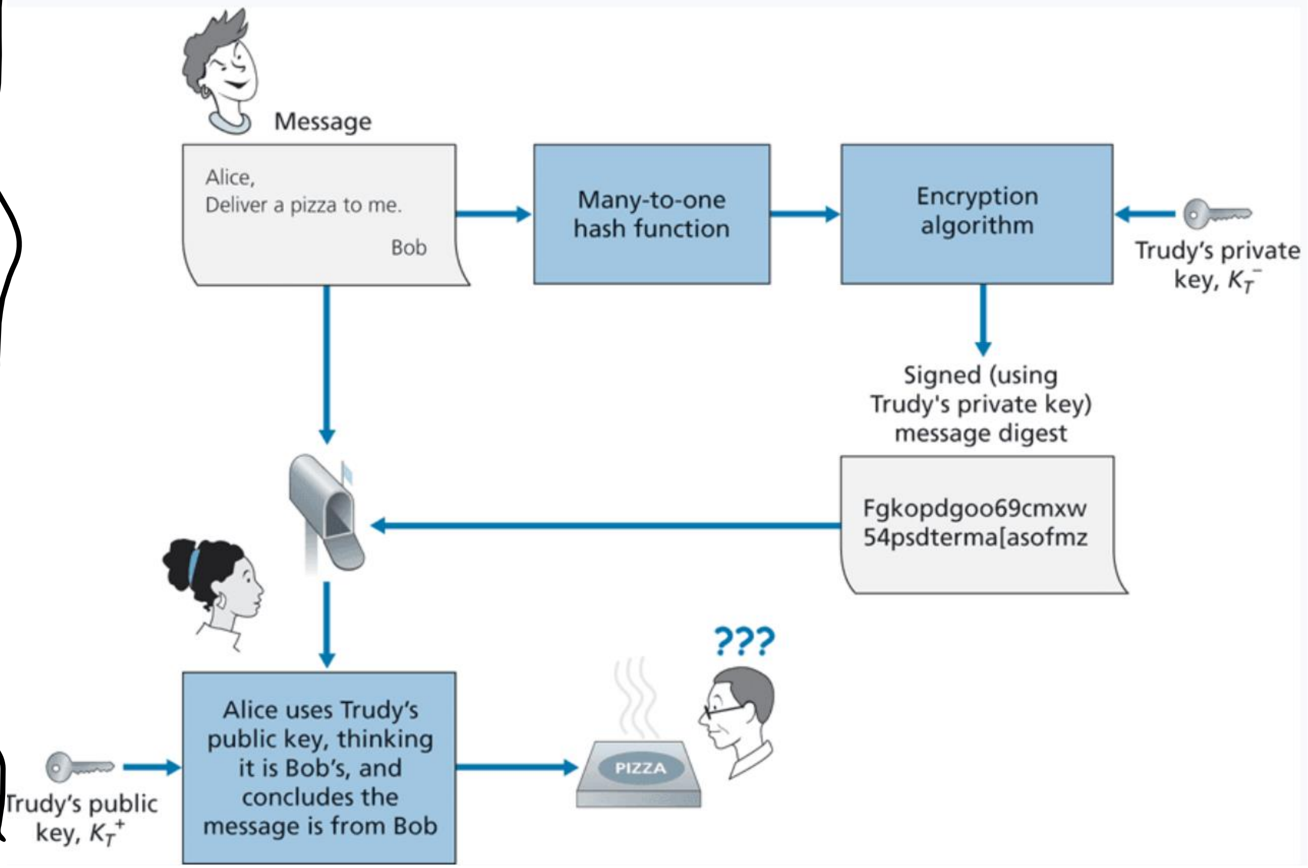


# Digital Signatures




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

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

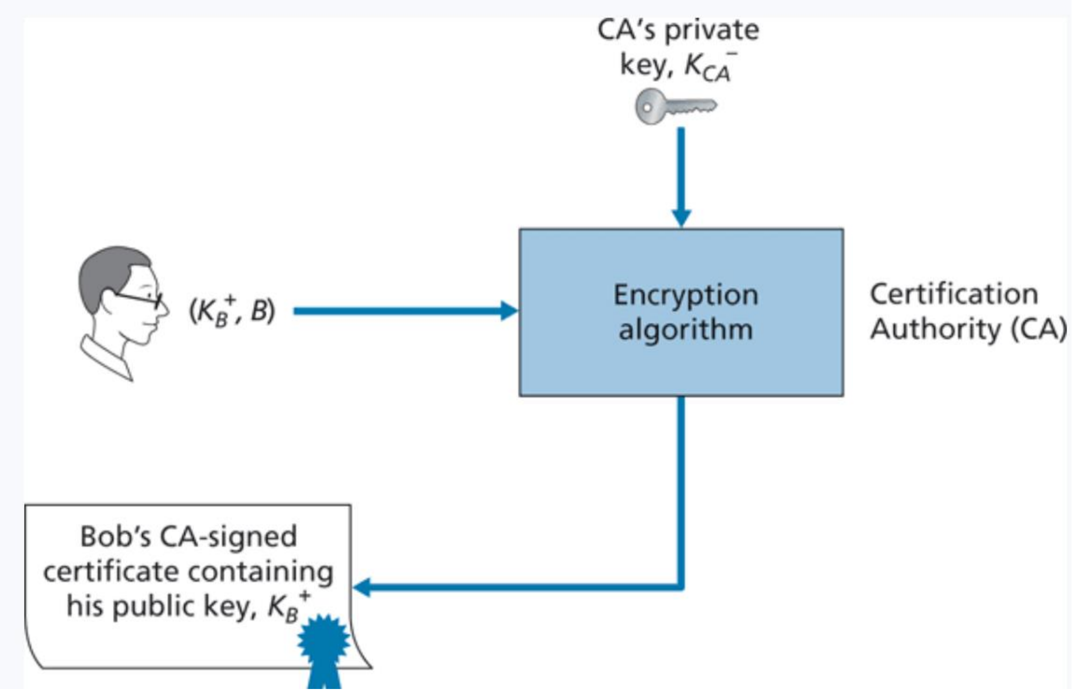


# Digital Certificates

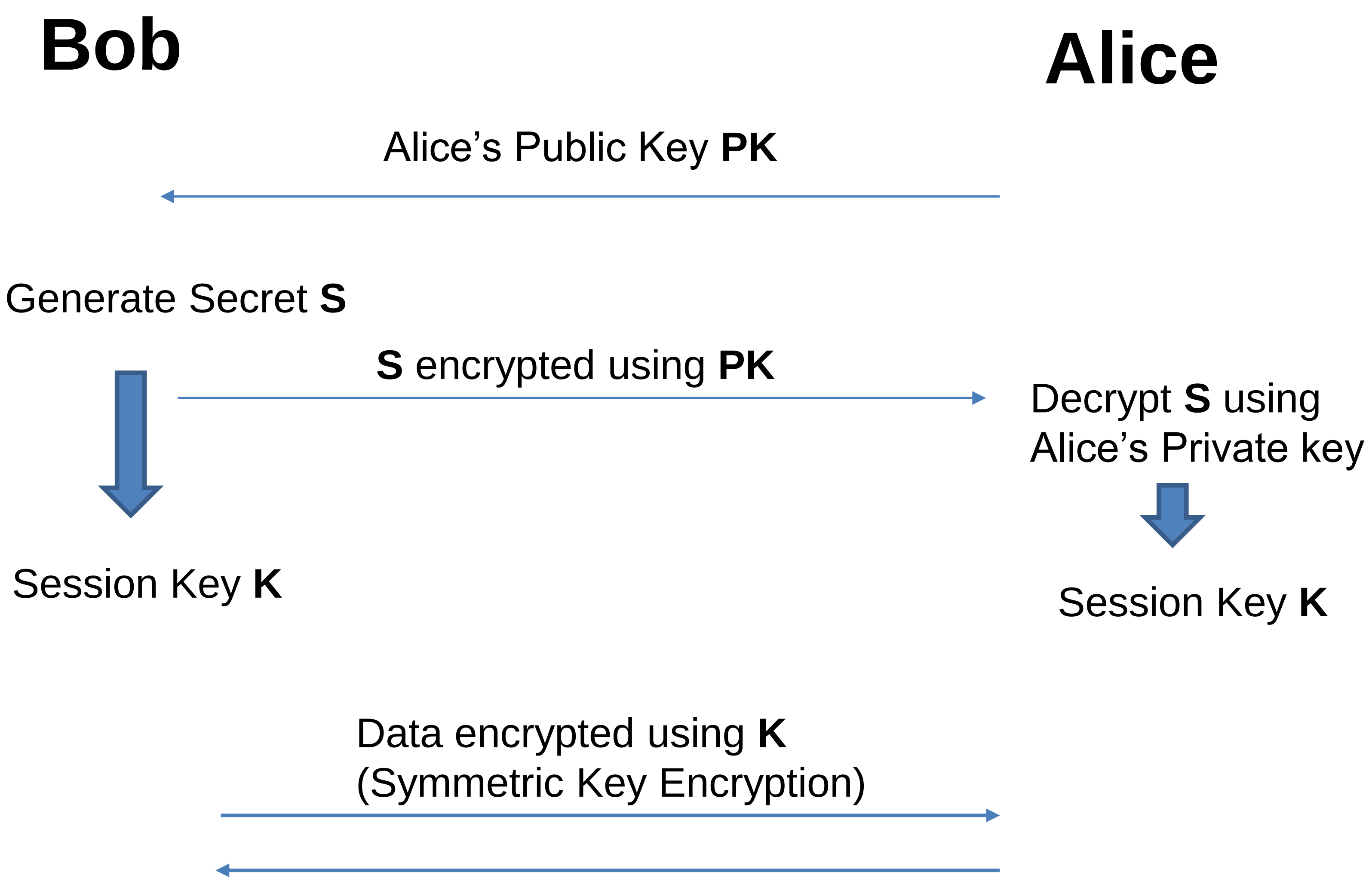
**Certificates** are an authoritative document that links entities (person, router, organization) to their public key

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




# Application: HTTPS and TLS/SSL

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