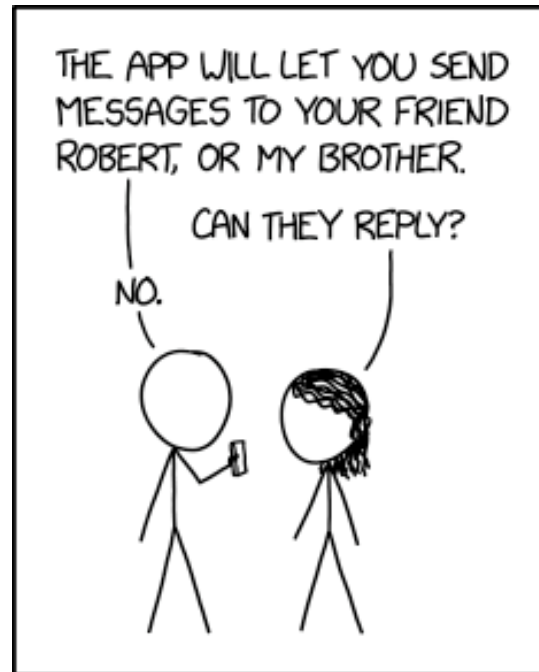


RSA:

Public-Key Encryption and Signatures



MY NEW SECURE TEXTING APP
ONLY ALLOWS PEOPLE NAMED
ALICE TO SEND MESSAGES
TO PEOPLE NAMED BOB.

What is Public-Key Encryption?

Last time we showed that Alice and Bob, who have never met, can establish a shared secret over a public channel.

(That was Diffie-Hellman *key exchange*, not public-key *encryption*)

Public-key encryption: Alice can publicly publish a key that allows anyone to send her secret messages.

Alice doesn't need to communicate individually to Bob for him to be able to send her secret messages.

A very powerful idea!

Public-Key Encryption Analogy

- Central Tower Emperor leaves a box of open locks out in the street.
- North Tower Emperor uses one of them to lock the gift and send it.



North Tower



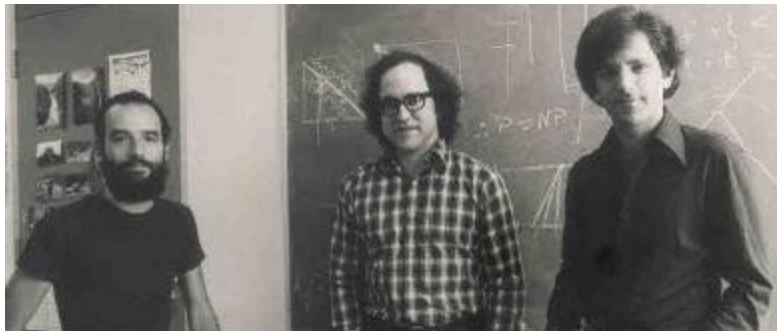
Central Tower

Every time you use a url with “https” you are using RSA!



Today: RSA

First* public-key encryption
First digital signature scheme



Adi Shamir Ron Rivest Len Adleman (1977)



Also discovered by Clifford Cocks at British Intelligence in 1973; classified until 1997.

*Diffie-Hellman can also be made into PKE but people didn't realize it at the time (see HW)

RSA



Here's an idea



That doesn't work, try again



Here's another idea



That still doesn't work, try again



⋮

Apparently this happened 42 times



Another “Hard Problem”: Factoring

Factoring Assumption: Given a number $n = pq$ where p and q are randomly chosen secret primes, there is no *efficient* algorithm for finding p, q .

The best known attack for RSA is to solve **factoring**.

(Recall that the best known attack for Diffie-Hellman is to solve **discrete log**.)

Factoring (like discrete log):

- is conjectured to be NP-intermediate.
- has a polynomial-time *quantum* algorithm [Shor, 1994]

RSA Factoring Challenge

In 2005, J. Franke et al. **won \$20,000 for showing:**

$n=310741824049004372135075003588856793003734602284272754572016148823206$
 $440580815045563468296717232867824379162728380334154710731085019195485$
 $29007337724822783525742386454014691736602477652346609$

is the product of

$p=1634733645809253848443133883865090859841783670033092312181110852389$
 $333100104508151212118167511579$

and

$q=1900871281664822113126851573935413975471896789968515493666638539088$
 $027103802104498957191261465571$

RSA Factoring Challenge

RSA \$100,000 challenge (defunct): factor n into two large primes:

$n=135066410865995223349603216278805969938881475605667027524$
4851438515265106048595338339402871505719094417982072821644715
513736804197039641917430464965892742562393410208643832021103
729587257623585096431105640735015081875106765946292055636855
294752135008528794163773285339061097505443349998111500569772
36890927563

MIT time capsule



1999



2019

$2^{79685186856218}$

(mod product of two 1024-bit primes)



Goal for Public-Key Cryptography

Create a poly-time encryption algorithm **E** (using **public key**).

Create a poly-time decryption algorithm **D** (using **secret key**).

Requirement: For any message **m**, $D(E(m)) = m$.

i.e. **D** is the (left) inverse of **E**.

Goal: Find a function **E** that is easy to evaluate, hard to invert, but easy to invert with a “trapdoor” (secret key).



The public key is like an open lock from the box on the street, and the secret key is like the Emperor's key to that lock.



Preview of the Structure of RSA

One-time pad

key = k

E is “add k (mod 26)”

D is “subtract k (mod 26)”

E is easy to invert
(i.e. D is easy to find given E)

RSA

public key = (n, e) , secret key = d

E is “take e^{th} power (mod n)”

D is “take d^{th} power (mod n)”

We’ll carefully choose e, n, d so that E, D are inverses, i.e. $m^{ed} \equiv m \pmod{n}$

Moreover, E will be hard to invert without knowing d (i.e. D will be hard to find)

Why are we taking the mod anyway?

To find such e, n, d , Fermat’s Little Theorem will be useful...



Fermat's Little Theorem (1640)

Fermat's Little Theorem (FLT):

If p is prime, then for any $a, k \in \mathbb{Z}$,

$$a^{1+k(p-1)} \equiv a \pmod{p}.$$

any number $\equiv 1 \pmod{p-1}$

E.g. $3^{13} \equiv 3 \pmod{7}$

I never would have predicted this would be so useful one day!



*proof on last slide and in course notes

(Not to be confused with Fermat's Last Theorem)



"It is impossible... for any number which is a power greater than the second to be written as the sum of two like powers. I have a truly marvelous demonstration of this proposition which this margin is too narrow to contain."

Public-Key Encryption: Attempt #1

How?

I pick:

1. large random prime p .
2. e, d so that
 $e \cdot d \equiv 1 \pmod{p-1}$.

*pick random number
check if its prime*

(Supposedly) secret information is **bold**

I compute
 $c = m^e \pmod{p}$

(p, e)

$c = m^e \pmod{p}$



A Bob wants to send Alice a message $m < p$

d is supposed to be secret but I can compute it! hehehe

$$c^d \pmod{p} = \underline{m^{ed}} \pmod{p} = m \quad \text{FLT}$$

I decrypt by taking
 $c^d \pmod{p}$, which I claim is m .

Why?



EEA

$2^{16} + 1$

The Issue with Attempt #1

Alice calculated d from (p, e) by solving $e \cdot d \equiv 1 \pmod{p-1}$ where the modulus $p-1$ was **public**!

We would like this modulus to be private to Alice.

Then Alice could still compute d from e and the **private modulus**, but Eve couldn't!

An extension of Fermat's Little Theorem will help...

Euler's Theorem (1763)

(A special case of) Euler's Theorem:

If $n = p \cdot q$ is the product of two distinct primes, then for any $a, k \in \mathbb{Z}$:

$$a^{\underbrace{1+k(p-1)(q-1)}} \equiv a \pmod{n}.$$

any number $\equiv 1 \pmod{(p-1)(q-1)}$

E.g. setting $n = 2 \cdot 5 = 10$, $a = 4$, $k = 3$, we have $4^{13} \equiv 4 \pmod{10}$

I, too, never would
have predicted this
would be so useful
one day!



RSA: Public-Key Encryption

I pick:

1. large random primes p, q .
2. Set $n = p \cdot q$.
3. e coprime to $(p-1)(q-1)$, and pick d so that
 $e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$.

Secret information is **bold**

I compute
 $c = m^e \pmod n$

(n, e)

$c = m^e \pmod n$

A Bob wants to
send Alice a
message $m < n$

I decrypt by taking
 $c^d \pmod n$, which I
claim is m .

Why?

$$p=3, q=17, e=3$$

$$c^d \pmod n = m^{ed} \pmod n \stackrel{\text{Euler}}{=} m$$

$2^{16} + 1$
EEA



RSA Example

- * Set $n = p \cdot q = \underline{3} \cdot \underline{17} = \boxed{51}$ (the primes are secret)
- * Generate matching public/private key pair $(e, d) = (\boxed{3}, \underline{\underline{11}})$
 - * $e \cdot d \equiv 1 \pmod{32}$ $\overset{2}{(p-1)} \overset{16}{(q-1)}$
 - * E.g., pick e coprime to 32 and compute inverse d using EEA
- * Alice sends $(n, e) = \boxed{(51, 3)}$ to Bob
- * To send $m = \underline{4}$, Bob sends the ciphertext:
$$m^e \equiv 4^3 \equiv \boxed{13} \pmod{51}$$
- * After receiving $c = 13$, Alice computes:
$$c^d \equiv 13^{\underline{11}} \equiv \underline{4} \pmod{\boxed{51}}$$

RSA Security

Why can't Eve figure out m ?



Well... because we assume that.

RSA assumption: For randomly chosen m , there is no efficient algorithm that given $n, e, m^e \pmod{n}$, finds m .

what Eve knows

Best known attack: Factor n to find p, q , then compute d by solving $e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$ using Extended Euclid.

One-Time Pad Cons from last time

- **Con 1:** It's insecure to use the same key twice.  We didn't fix this one
- **Con 2:** Alice and Bob must privately agree on the secret key beforehand.  Diffie-Hellman and RSA fix this one

The RSA protocol you just saw suffers from **Con 1** because the encryption algorithm is **deterministic**.

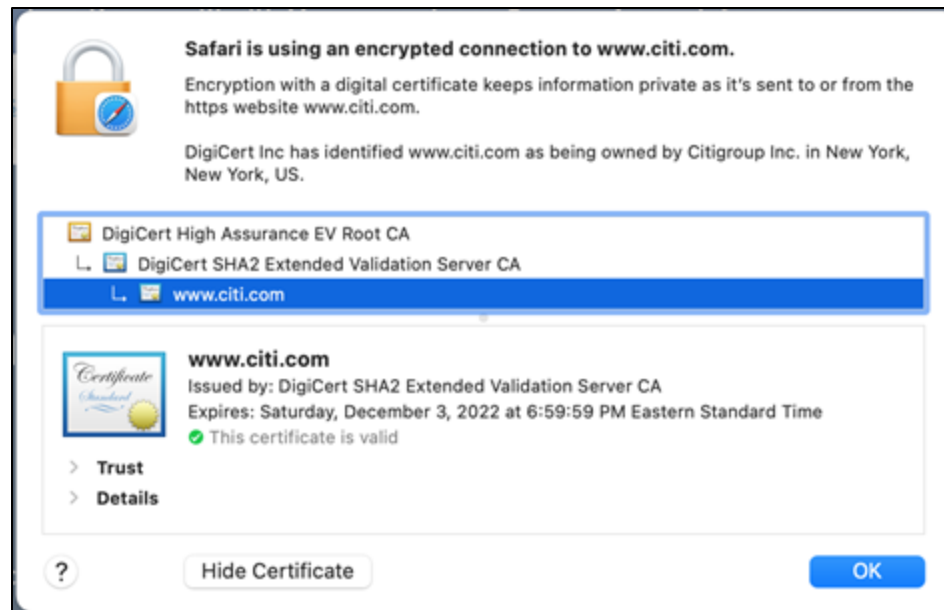
E.g. Eve can tell if the same message is sent twice.

This can be fixed by inserting some **randomization** into the encryption algorithm. (We won't show.)

RSA can also be used for “Signatures”

Goal of a signature:

1. Alice publishes a public key (n, e) .
2. Alice OR a malicious “forger” sends a public message m with a “signature” s .
3. Anyone can check whether the same entity did both 1. and 2.



This is my
signature!

λ



RSA Signature Goal

For any message m ,
I can compute a
valid signature s .



(n, e)

(m, s)

We will run a
verification algorithm
to check if the
signature is valid



I'm back! Did
you miss me?

I can't compute a
valid signature for
any other message :(

(m', s')



malicious adversary
trying to forge Alice's
signature

This is reminiscent of
verifying a certificate
for an instance of an
NP language



RSA Signatures: Run RSA “Backwards”

I pick:

1. large random primes p, q .
2. Set $n = p \cdot q$.
3. e coprime to $(p-1)(q-1)$, and pick d so that $e \cdot d \equiv 1 \pmod{(p-1)(q-1)}$.

I want to send message m .

I compute signature $s = m^d \pmod{n}$

Secret information is **bold**

Verification algorithm: check that

$$\underline{s^e} \equiv \underline{m} \pmod{n}$$

$$m^{de} \equiv m \pmod{n}$$



(n, e)

$(m, s = \underline{m^d} \pmod{n})$



(m', s')

I can't seem to compute a valid signature for a randomly chosen m' ...



RSA assumption: For randomly chosen m , there is no efficient algorithm that given $n, e, \underline{m^e} \pmod{n}$, finds \underline{m} .

m'

$m'^d \pmod{n}$

What if the forger gets to choose m' based on s '?

Then the forger can successfully forge Alice's signature by first picking s' and then letting $m' = s'^e \pmod n$!

This can be fixed by Alice applying some wild function H to m and sending $H(m), s = H(m)^d \pmod n$ (We won't show.)

Proof of Fermat's Little Theorem

- * **Lemma:** If p is prime and $a \not\equiv 0 \pmod{p}$, then the set of numbers $\{a, 2a, 3a, \dots, (p-1)a\} \pmod{p}$ is the same set as $\{1, \dots, p-1\}$.
 - 1) For every $i \in \{1, \dots, p-1\}$, ia is not a multiple of p since p does not divide either i or a (**Euclid's lemma**). Thus, each $ia \pmod{p} \in \{1, \dots, p-1\}$.
 - 2) For every $i, j \in \{1, \dots, p-1\}$, $i \neq j$, $(j-i)a$ is not a multiple of p . Thus, there are no collisions: $ia \not\equiv ja \pmod{p}$.
- * **Then:** Since the sets are the same, their products are too:
 - * $a \cdot 2a \cdots (p-1)a \equiv 1 \cdot 2 \cdots (p-1) \pmod{p}$
 - * Hence $a^{p-1} \equiv 1 \pmod{p}$. ($\{1, \dots, p-1\}$ all have inverses mod p , so multiply both sides by $1^{-1} \cdot 2^{-1} \cdots (p-1)^{-1} \pmod{p}$)