RSA

Last time, we continued our exploration of when ones appear in the power tables. We ended with a theorem that says more than Fermat's Little Theorem, namely:

Euler's Theorem

Let m be a positive integer and a be an integer with $\gcd(a,m)=1$. Then

$$a^{\varphi(m)} \equiv 1 \pmod{m}.$$

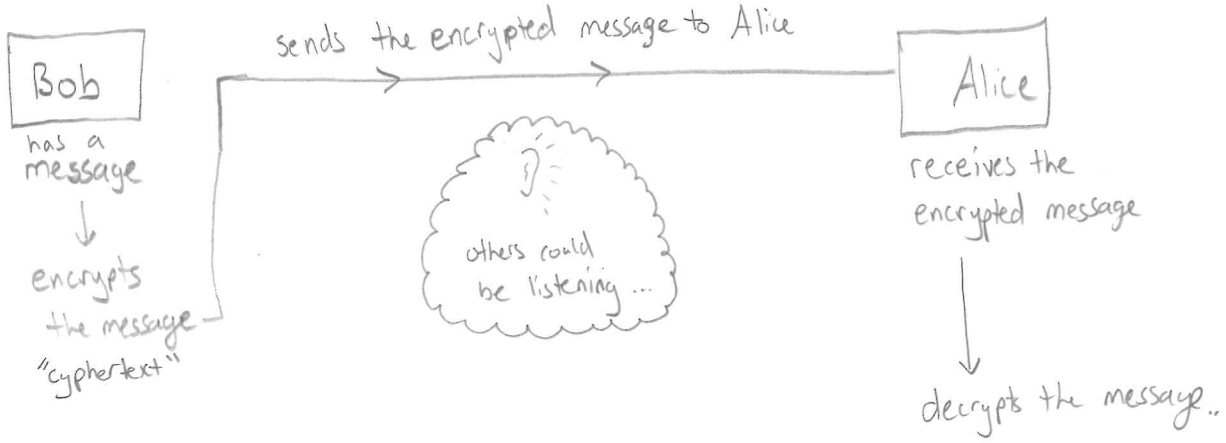
Did you figure out the Challenge problem? What are the last two digits of 3^{84}?

Let's try anothr (more straightforward):

Question: Suppose we want to find $6^{122} \pmod{77}$. How might we use Euler's Theorem to help?

Great! But... the point of learning these properties of a new kind of arithmetic (modular arithmetic) was to learn about a more advanced cryptosystem: the RSA (Rivest-Shamir-Adleman) cryptosystem.

Recall our classic cryptography set-up:

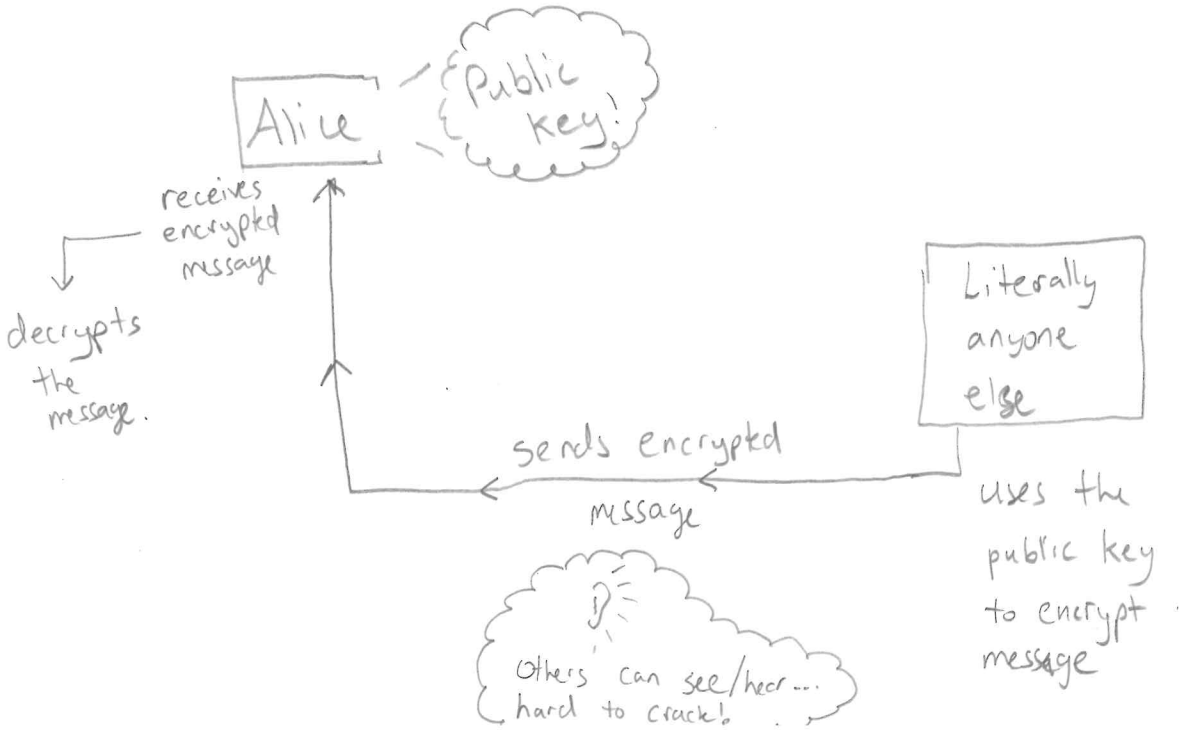


How does Alice decrypt the message?

What weaknesses does this system have?

In our classical set-up, the key Alice has is a private key, a key only Alice (and Bob) know. There is another (more secure) set-up in which the key is a public key, i.e. everyone knows the key. How in the world could this be more secure?

Well, it works a bit like this. Alice announces a public key to everyone (Bob included), and with this key, anyone can send Alice an encrypted message that only Alice can decrypt.



Sounds like wishful thinking! How could something like this actually work? Let's look at an example... but we will use "small numbers," making it "easy" for listeners to crack the encryption. Later, we will think about what happens when we use "large numbers." We start the example by assuming that Alice and Bob have instructions for the algorithm.

Alice

Step 1 !

Alice needs to pick a public key. To do this, she picks two primes

$$\begin{cases} p = \_\_\_\_ \\ q = \_\_\_\_ \end{cases}$$

but she doesn't tell anyone what two primes she picked. She sets $n = p \cdot q = \_\_\_\_$. (She will eventually share n.)

Now, she notices

$$\varphi(n) = \varphi(p \cdot q) = (p-1)(q-1) = \boxed{}.$$

She looks for some e with the property that $\gcd(e, \boxed{}) = 1$.

She picks $e =$ _____. (She will eventually share e.)

Now, in preparation for when she receives a message, she computes a number d which is important for decrypting a message.

She finds d such that $e \cdot d \equiv$ _____ $\equiv 1 \pmod{\varphi(n)}$

She computes $d =$ _____.

Now, she has five numbers.

p, q, d : She does not share

n, e : She shares.

That makes $n, e =$ ___, ___ the public key.

She announces them to everyone.

Bob ↙ Step 2! Encryption

Bob decides he wants to send Alice a message, so he looks up her public key: n, e. (___ and ___).

He decides to encrypt a single number: $m =$ ___.

(The number could represent many things: a special code, a letter... or, what is often the case, a "key" for a classical cryptosystem, ie. Bob is sharing a private key for a simpler + faster way to send messages.)

He encodes the number (mod n), so we have ___ (mod n).

Notice, if n is really big, then this doesn't change anything! We just want to think of the number "mod n" for what comes next.

Now, Bob uses e=___ to encrypt the message:

$$(\ m\ )^{e} = (\_\_\_)^{-} \equiv \_\_\_\_ \pmod{n}, \quad n = \_\_\_\_$$

(Bob takes his chosen number, the message, and raises it to the power of e.)

The result, _____, is the encrypted message. Bob sends the encrypted message to Alice.

Alice

step 3! Decryption

Alice receives the encrypted message _____ from Bob.

To decrypt, she raises the encrypted message to the power d = _____, where d is the number she computed (but did not share with anyone!).

$$(\text{encrypted message})^{d} \equiv \_\_\_\_ \pmod{n}, \quad n = \_\_\_\_$$

The claim is that

$m =$ ___ , intended message

$n =$ ___ .

$$(\text{encrypted message})^d \equiv m \pmod{n} = \underline{\qquad\qquad} \pmod{\quad}$$

Is this true? Check it on a calculator... why is it working??

Well...

rem: $ed \equiv$ ___ $\pmod{\varphi(n)}$

$$(\text{encrypted message})^d = (m^e)^d = m^{\overline{ed}} = m^{\text{___}}$$

↘ exponent rules!

$$\equiv (m^{\text{___}})^{\text{___}} \cdot m^{\text{___}}$$

$$\equiv \underline{\qquad\qquad} \pmod{n = \text{___}}.$$

Questions to think about:

- If n is small, it is easy to factor into two primes. If we can factor n easily, how might we "crack" Bob's code? (Can we figure out d?)

In practice, we pick p and q to be very, very large primes! That way it takes a computer a really long time (think a billion years!) to factor n into p and q.

- If n cannot be factored in any reasonable amount of time, can you "crack" Bob's message?

NOTE Whether or not there is a faster way to factor primes is unknown, but a very, very difficult problem related to one of the most famous unsolved math problems: {P vs. NP.} It goes something like this: NP is a set of problems that, if given a proposed solution, can be checked quickly. (eg. is $p=11, q=5$ the prime factorization of $n=55$? ... or even big numbers p, q, n.) P is a set of problems which can be "quickly" solved (probably not finding the factorization of a large number...). We don't know whether or not P and NP are the same set of problems!

the solution to this problem is worth $1 million! (Clay Mathematics Institute.)