Cryptography

What is it?

I know that when I first heard the word cryptography, I saw it as this scary concept that would be impossible to grasp at even a basic level; but it doesn't have to be. The Cambridge Dictionary defines **cryptography** as *"the practice of creating and understanding codes that keep information secret".* We've probably all used some sort of cryptography in our lives; I remember when I was younger, my sister and I tried to create our own wildly complicated and ineffective language.

Examples

Caesar Cipher:

One of the earliest and most known examples of cryptography was the Caesar Cipher. The infamous Roman Emperor Julius Caesar used this method of encryption when sending messages. It involves shifting all letters by a set number of letters. For example, if I was to encode the word DOG using a shift of +2, the corresponding encrypted message would be FQI, as D->F, O->Q, G->I.

The issue with this is that in a time where cryptography is no longer an unknown field – I'm pretty sure I remember seeing questions involving ciphers in the general ability section of the Selective High School Placement Test over a decade ago – it is not too hard to crack the code with little effort.

Public Key Cryptography:

What had become apparent was that all previous encryption methods relied on the sharing of secrets, and this made them vulnerable – as seen with the Germans' brilliant but not 'unbreakable' Enigma Machine during World War II. This question was also on the mind of Ralph Merkle when he was an undergraduate computing student at UC Berkley in 1974. He actually came up with the idea as part of a project for one of his classes. "If the opponent knows everything, how do you establish secure communications over an open communications line when the enemy is listening in?" he said as part of a an interview with the Computer History Museum in 2011 (Ralph Merkle Interview; 2011-03-11

(computerhistory.org) – seriously it's pretty interesting give it a read). This is the basis of public-key cryptography.

In the end, this project was the invention of a concept now known as Merkle's puzzles, an early form of public-key cryptography. It is a secure method of sending messages without having any shared secrets beforehand, Merkle himself explains it in the above interview but I know you don't have all day, so I'll give you a summary:

> I send you 20 000 puzzles – a message encrypted with a weak cipher, with each one containing an identifying number and a key. You pick one of those puzzles and solve it, and so you now have the identifying number and key. You send me a message back saying that you'd like to use the key from puzzle number #X, and since I made the puzzles, I of course know what that key is. That will be the encryption key used between us in our following conversations.
>
> Sure, an attacker can also solve the puzzle to obtain the key, but they must solve the *exact* puzzle you did, and on average they'll do that after checking half of the puzzles, i.e., 10 000. So, if it takes you a day to solve the puzzle, it will most likely take the attacker 10 000 days (or 27 years) until he obtains the correct key.

Unfortunately, Merkle's Puzzles are not practical in modern applications as it is extremely difficult to find a balance when creating a large number of puzzles (or puzzles of a hard difficulty) such that the attacker has to do an extreme amount of work while it also not being too inconvenient for the participants of the conversation.

Following this, Merkle collaborated with Whitfield Diffie and Martin Hellman, students at Stanford, who published a paper on the Diffie-Hellman Key Exchange in 1976, which is seen as the earliest publicly known work on public-key cryptography.

This caught the attention of MIT colleagues Ron Rivest, Adi Shamir, and Leonard Adleman, who came up with and published the RSA algorithm (based on the first letter of their surnames). It involves the use of multiplying two large prime numbers, which is a relatively inexpensive cost in comparison to deducing the two original numbers. I'll outline an example of RSA below.

Example:

I recently saw the movie "The Creator" and really enjoyed it. So, I converted that into a number (by changing each letter to its corresponding position in the alphabet, but you can encode this however you like) to get the message:

$$\mathbf{x} = 20\,080\,503\,190\,501\,201\,518.$$

Now it's time for me to choose my primes p and q to start encrypting it. These primes need to be large enough such that p * q = n has more digits than the encoded message x above. Since x is 20 digits, I eventually chose the 11-digit primes:

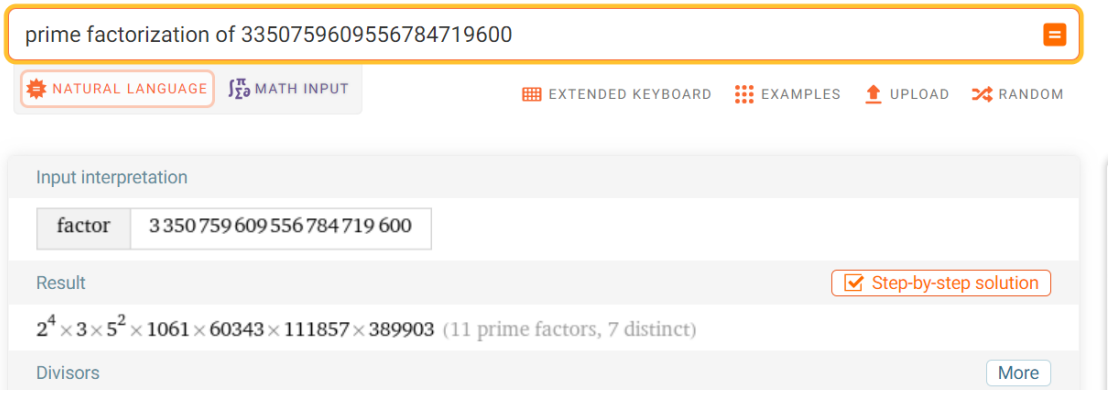
p = 47 055 833 459

q = 71 208 166 201.

n = pq = 3 350 759 609 675 048 719 259

m = (p-1)(q-1) = 3 350 759 609 556 784 719 600

Now I need to pick my public encryption key e. The value of this key doesn't matter as long as it is coprime with m. Therefore, I used WolframAlpha to show me the prime factorization of m.



And from this I chose e = 343.

Then I finally used the encryption algorithm to encode the message:

x^e mod n

= (20 080 503 190 501 201 518)^(343) mod (3 350 759 609 675 048 719 259)

= 1 003 160 170 847 806 648 231

This is my encrypted message. The only way to decrypt this is by using the private key d, which is calculated as follows:

d = inverse of e mod m

= 343 mod (3 350 759 609 556 784 719 600)

= 3 321 452 674 196 229 751 207

Now similarly to before, we apply the algorithm to decode the encrypted message y:

$$y^d \bmod n$$

$$= (1\,003\,160\,170\,847\,806\,648\,231)^{(3\,321\,452\,674\,196\,229\,751\,207)} \bmod (3\,350\,759\,609\,675\,048\,719\,259)$$

$$= 20\,080\,503\,190\,501\,201\,518$$

Which is the original message! To summarise, let's say an organization wants users to be able to send data to them securely. The organization will make the value of n and their public key e available to anyone, allowing them to encrypt a message. Once that message is sent to the organization, they can use the private key d (which is known only by them) to decrypt the message. Obviously to ensure security, the primes used are extremely large, typically 2048 bits, which is over **600** digits!