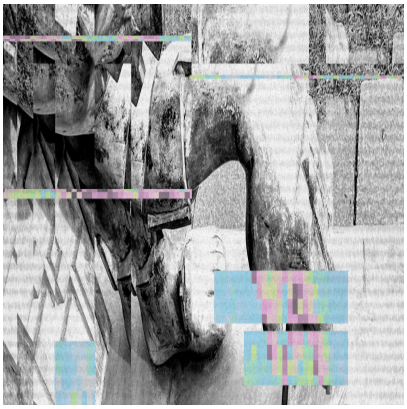


Asymmetric key encryption

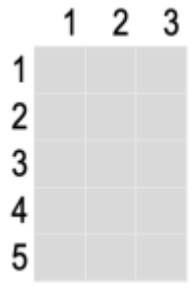
Description

An encryption key is a string of characters that you feed into an encryption algorithm to either encrypt or decrypt a message. An asymmetric key system has two keys. Thereâ??s a public key to encrypt a message. Itâ??s public because anyone can see it and use that key. But once the message is encrypted using the public key, the message can only be retrieved by someone with the private key. Only the sender and the receiver should know the private key.

The encryption algorithm uses the properties of prime numbers to encrypt a message. As I explored in the last post, two randomly selected really large prime numbers (***p*** and ***q***), when multiplied together produce an even larger number ***n*** that is virtually impossible to factorize, i.e. to discover the ***p*** and ***q*** primes that produced ***n***. We are here talking about an integer ***n*** that has over 100 digits. Hereâ??s how the encryption algorithm works. I wonâ??t attempt to explain why it does, just what it does, and with diagrams.

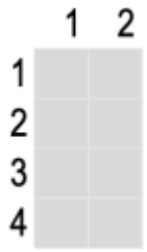
Multiplying primes

The encryption software chooses two random large prime numbers ***p*** and ***q***, and multiplies them together. Think of ***p*** and ***q*** as the length of the side of a rectangle. To keep this demonstration simple, Iâ??ll choose two very small primes, ***p***=3 and ***q***=5. So, ***n*** = ***p***.***q*** = 15. (Pretend that ***p***, ***q*** are so large that no one could guess them as factors of ***n***, i.e. ***n*** is impossible to factorize.) ***n*** = ***p***.***q*** = 3 x 5 = 15. These values are then fixed and hidden within this particular encryption algorithm. They donâ??t vary with the message or the occasion in which it is used.



The algorithm now calculates a smaller version of this product by subtracting 1 from each of the primes and multiplying them together.

z = (***p***-1).(***q***-1) = 8



The algorithm then has to create a new number ***e*** that is less than ***n*** and is not a factor of ***z***.

e < ***n***
z mod **e** â? 0

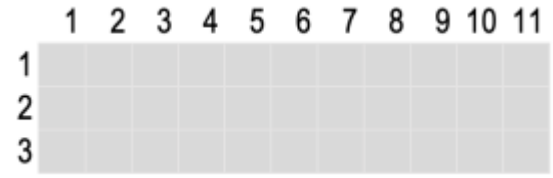
In this demonstration, with the values already decided, that means ***e*** must be less than 15 and doesnâ??t divide into 8 a whole number of times.

In this case the numbers that meet that requirement are 3, 5, 7, 9, 10, 11, 12, 13, 14. They are all less than 15 and none are factors of 8. The algorithm selects (randomly) ***e*** = 11.

The algorithm then has to create a number ***d*** such that when itâ??s multiplied by ***e*** and divided by ***z*** it produces a remainder of just 1. Some candidates for ***d*** are 3, 11, 19, 27, etc. The algorithm randomly chooses ***d*** = 3.

d x ***e*** = 3 x 11 = 33
d.***e*** mod ***z*** must equal 1
33 mod 8 = 1 (it is)

Hereâ??s a spatial representation of the ***d*** and ***e*** relationship.



The public and private keys are number pairs.

Public key = [***n***, ***e***] = [15, 11]
Private key = [***n***, ***d***] = [15, 3]

Encrypting the message

The sender and receiver of the message have access to the same encryption algorithm with the same ***p*** and ***q*** values coded in. Person A wants to dispatch a secret message ***m***. To keep this demonstration

simple, Iâ??ll encrypt just a number, the message is â??7â?• i.e. ***m*** = 7.

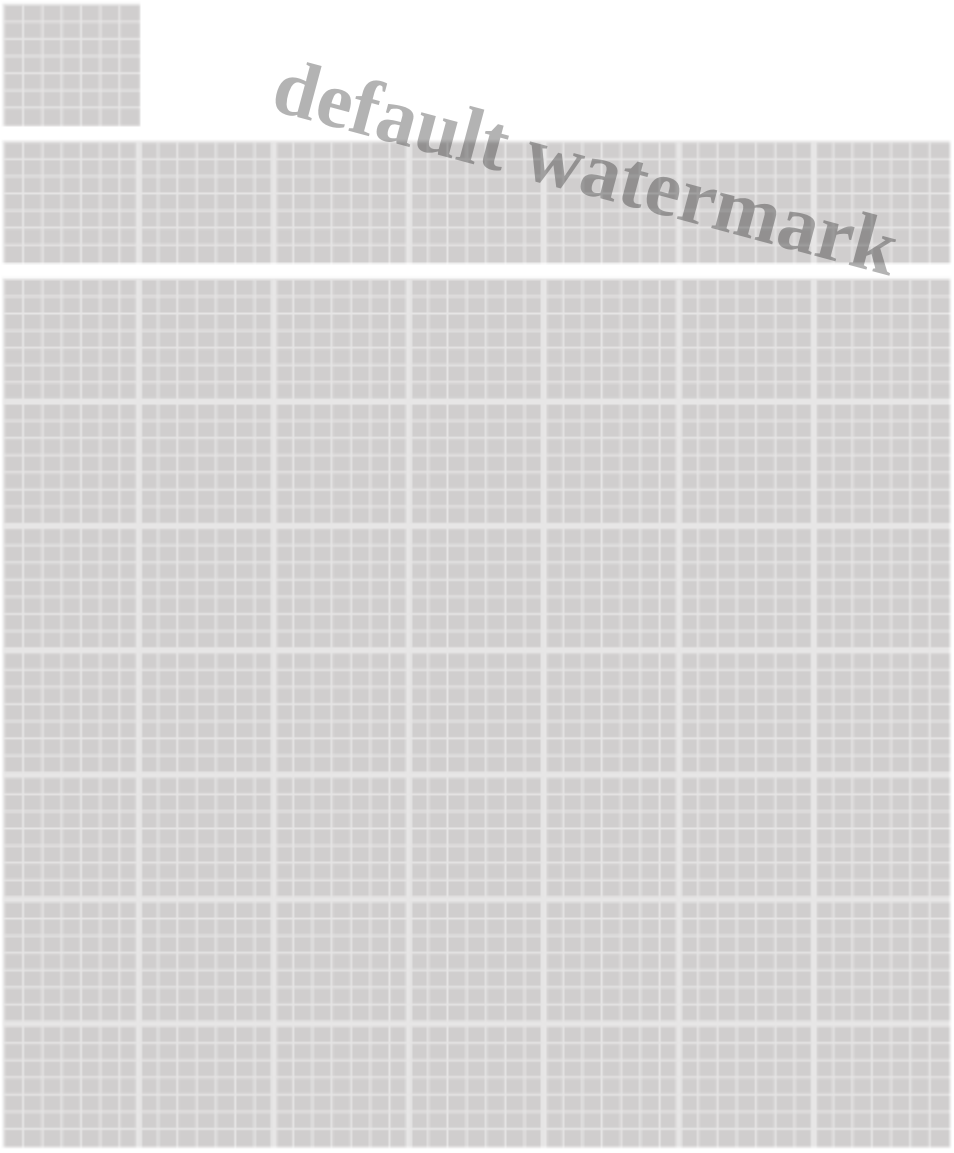
The formula to encrypt the message ***m*** into coded form ***c*** is

$c = m^e \bmod n$

n and ***e*** are parts of the public encryption key. The algorithm will multiply the number ***m*** by itself ***e*** times and divide the result by ***n***. The value of ***c*** is whatever is left over from that division (the mod operation).

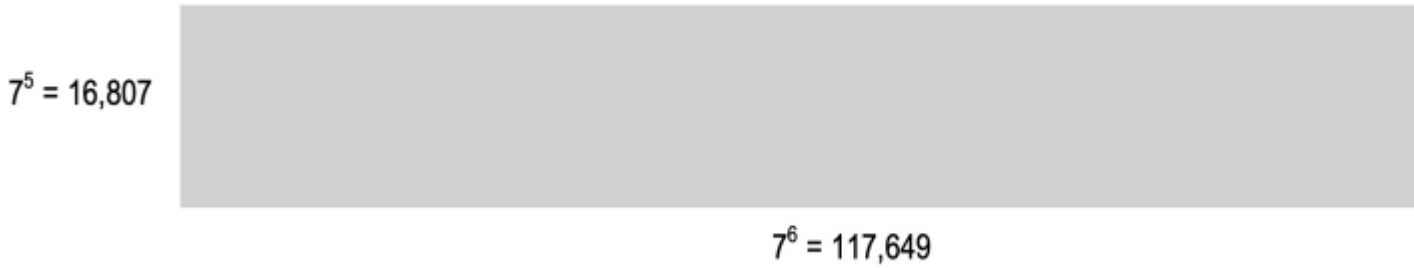
$c = 7^{11} \bmod 15$

m^e is going to be very large. For visual confirmation, here are three gridded rectangles. The first is 7^2, the second is 7^3 , the third is 7^4. I donâ??t have the space to show what happens when you keep multiplying ***m*** by ***m*** eleven times.



$m^e = 7^{11} = 1{,}977{,}326{,}743$

That number of grid units could be arranged in a huge rectangle with integer dimensions as follows. There are variants, but they are more stretched out than this one.



Divide that by ***n*** = 15 to give 131,821,782 whole numbers with 13 left over. That would be a very long thin rectangle of dimensions 15 x 131,821,782 with a tiny sliver of 13 cells left over at one end.

Recovering the message

So all those big number calculations result in ***c*** = 13 which is what gets transmitted as the encrypted message. The recipient B then uses the private encryption key pair ***d*** and ***n*** to decrypt the message.

$\boldsymbol{m} = \boldsymbol{c}^{\boldsymbol{d}} \bmod \boldsymbol{n}$
$\boldsymbol{m} = 13^3 \bmod 15 = 7$

Hereâ??s the 13^3 = 13 x 13 x 13 gridded rectangle made up of 146 grid units

Here are the same number of grid units rearranged to a 146 x 15 rectangle with 7 grid units left over, which is the original message ***m***.

Surprisingly, that tiny remaindered bit at the end is the original message. Even with very small primes (***p*** and ***q***) and a miniscule message ***m*** of just one number, the calculations involve extremely large integers and require iteration to find numbers that match various criteria.

So whatever the speed of the computer, this kind of encryption is computationally expensive. Itâ??s therefore used for handshaking protocols, setting up a secure connection that enables the exchange of keys used for faster, simpler encryption.

References

- Mann, Kathryn. 2017. The science of encryption: prime numbers and mod n arithmetic. Available online: https://math.berkeley.edu/~kpmann/encryption.pdf (accessed 3 May 2021).
- Seetharam, Anand. 2019. RSA (Rivest, Shamir, Adleman) Algorithm explained with example. *CSEdu4All*, 29 January. Available online: https://www.youtube.com/watch?v=KPkm2yvyGi8 (accessed 4 May 2021).

Note

- The references above explain the process. The diagrammatic approach and any errors that entails are my own.

Category

1. Architecture

Tags

1. cryptography
2. prime factors

Date Created
May 15, 2021
Author
rcoyne99