

In[12]:= 1 + 1

Out[12]= 2

Bob's m = 11 100 058 808 629 299 314 389, and his value of e = 3 are public.

In[13]:= mb = 11 100 058 808 629 299 314 389

Out[13]= 11 100 058 808 629 299 314 389

In[14]:= eb = 3

Out[14]= 3

Bob has a decryption key, which satisfies $e * d = 1 \pmod{\phi(m)}$. He computes as follows :

Bob's m is the product of two primes, which only he knows. They are $p = 97\,460\,021\,213$ and $q = 113\,893\,457\,753$

Hence he can easily compute $\phi(m) = \phi(p * q) = (p - 1) * (q - 1)$.

phim = 97 460 021 212 * 113 893 457 752

11 100 058 808 417 945 835 424

He now uses the ExtendedGCD command to compute his d. ExtendedGCD[e, phi(m)] gives {g, {r, s}} where g is the gcd and $g = er + \phi(m)s$. Thus a solution to $e * r$ congruent to $g = 1 \pmod{\phi(m)}$ is the value of $r \pmod{\phi(m)}$. Add a multiple of $\phi(m)$ if necessary to get $1 < r < \phi(m)$.

ExtendedGCD[3, phim]

{1, {-3 700 019 602 805 981 945 141, 1}}

This says $3(-3\,700\,019\,602\,805\,981\,945\,141) + 11\,100\,058\,808\,417\,945\,835\,424 * 1 = 1$, i.e. $3 * (-3\,700\,019\,602\,805\,981\,945\,141) = 1 \pmod{\phi(m)}$.

Since he wants d positive he takes $d = -3\,700\,019\,602\,805\,981\,945\,141 + \phi(m)$, which is also calculated by :

Mod[-3 700 019 602 805 981 945 141, phim]

7 400 039 205 611 963 890 283

So this is his value of d, which we will call db.

db = 7 400 039 205 611 963 890 283

7 400 039 205 611 963 890 283

`Mod[eb * db, phim]`

1

Note that only Bob knows db.

Alice wishes to send a message to Bob. The message she decides to send is : LETS HAVE SOME FUN . She looks up Bob ' s value of m and e (encryption code) , which are public. Namely (mb, eb) = (11 100 058 808 629 299 314 389, 3) .

She enters Bob ' s m into her computer ' s memory.

`mb = 11 100 058 808 629 299 314 389`

11 100 058 808 629 299 314 389

She first wants to break her message up into blocks of letters. Each block of letters will be represented by a number. To decide how many letters for each block, she wants k so that $10^{(2k)} < m$ but $(10^{(2k-1)})^e > m$ (so that before encrypting the letters will be $< m$, but after raising to the e - th power there is encryption from the wrap - around effect of (mod m) .

For instance she could start by trying k = 4.

`mb - 10^8`

11 100 058 808 629 199 314 389

She sees from this that $10^8 < mb$.

However

`In[15]= PowerMod[10^7, 3, mb]`

`Out[15]= 1 000 000 000 000 000 000 000`

The above calculation shows there would NOT be encryption for all numbers with eight digits. So she tries k = 5.

`mb - 10^10`

11 100 058 808 619 299 314 389

```
In[16]:= PowerMod[10^9, 3, mb]
Out[16]= 6 801 989 395 054 066 009 379
```

The above two calculations shows that $k = 5$ would be a good choice, i.e. she should break her message into blocks of five letters.

Her message is LETS HAVE SOME FUN. Each blank space will count as a letter, so in blocks we have LETS (), HAVE (), SOME (), FUN () (), where I used () to represent blanks.

Looking up the numbers for LETS (consulting our chart) she gets 22 153 029. We ' ll use 99 to represent blanks . So she adds that to get 2 215 302 999. For HAVE () she gets 1 811 321 599. For SOME () she gets 2 925 231 599. For FUN () () she gets 1 631 249 999.

She is now going to encrypt the message by raising each of the numbers to Bob ' s e - th power modulo his m.

She does this using the PowerMod command

```
PowerMod[2 215 302 999, eb, mb]
7 029 906 317 731 427 576 340
```

```
PowerMod[1 811 321 599, eb, mb]
1 173 022 759 013 612 016 368
```

```
PowerMod[2 925 231 599, eb, mb]
4 802 105 580 056 640 357 905
```

```
PowerMod[1 631 249 999, eb, mb]
6 712 304 281 932 603 988 382
```

Breaking up each block by commas, the message Alice sends to Bob is :

```
7 029 906 317 731 427 576 340, 1 173 022 759 013 612 016 368,
4 802 105 580 056 640 357 905, 6 712 304 281 932 603 988 382
```

When Bob gets the message he decrypts it by computing $b^{(db)} \pmod{mb}$, where b is the message block.

Bob enters his db into his computer ' s memory .

```
db = 7 400 039 205 611 963 890 283
7 400 039 205 611 963 890 283
```

He decodes the first block $b = 7 029 906 317 731 427 576 340$

PowerMod[7 029 906 317 731 427 576 340, db, mb]
2 215 302 999

Bob sees this corresponds to LETS (). He now decodes the second block b = 1 173 022 759 013 612 016 368.

PowerMod[1 173 022 759 013 612 016 368, db, mb]
1 811 321 599

He sees this corresponds to the letters HAVE ().

Continuing he computes :

PowerMod[4 802 105 580 056 640 357 905, db, mb]
2 925 231 599

this corresponds to SOME (), and

PowerMod[6 712 304 281 932 603 988 382, db, mb]
1 631 249 999

which corresponds to FUN () ()

Now suppose Eve intercepts Alice ' s message to Bob.
Eve can look up Bob ' s value of m and e since these are public.

However the only way Eve can decrypt Alice ' s message is by knowing Bob ' s decryption code d. And the only way d can be computed is by knowing phi (m). The only way phi (m) can be computed is by factoring m. For the purposes of illustration we are using values of m which can be factored by the computer. However if m is very large there is no efficient way to factor m. Reasonably large m would take more than a lifetime to factor. This leads to unbreakable codes.

Now, Bob has decrypted Alice ' s message, and it makes sense,
but how can he be sure that she sent it and that i.e. Eve didn ' t send it?

Alice can also "sign" her message in a way that Bob can be
absolutely sure that the message came from her and from no one else.

To "sign" her message she needs to use her own (m, e, d). Thus we will now use
(ma, ea, da) for Alice ' s (m, e, d). Note that only (ma, ea) and (mb, db) are public.

Alice ' s (ma, ea) = (12 183 039 688 392 008 768 777, 7)

ma = 12 183 039 688 392 008 768 777
12 183 039 688 392 008 768 777

$ea = 7$

7

Alice has computed her own da . It is $da = 5\ 221\ 302\ 723\ 501\ 759\ 761\ 863$.

$da = 5\ 221\ 302\ 723\ 501\ 759\ 761\ 863$

5 221 302 723 501 759 761 863

To "sign" her message she puts her first name Alice into the corresponding numbers : 1 122 191 315.

Now she raises this to her OWN da - th power modulo her OWN ma .

$\text{PowerMod}[1\ 122\ 191\ 315, da, ma]$

7 788 990 147 451 777 481 373

Now when Bob gets the above block, which could be the last block of the message, he tries to decrypt it using HIS db and his mb as before.

$\text{PowerMod}[7\ 788\ 990\ 147\ 451\ 777\ 481\ 373, db, mb]$

10 092 914 785 471 903 555 415

Since this doesn't correspond to letters, and it is the last block, he suspects it is a signature from Alice. To check it is REALLY from Alice, he instead raises it to HER ea - th power mod HER ma =

$\text{PowerMod}[7\ 788\ 990\ 147\ 451\ 777\ 481\ 373, ea, ma]$

1 122 191 315

This does indeed correspond to Alice. The point is that the only way (encrypted signature) $^ (ea) =$ Alice is if encrypted signature = (Alice) $^ (da)$,

For instance Eve wouldn't have been able to produce this signature since she cannot calculate (Alice) $^ (da)$ since Eve doesn't know da !

Since Alice is the only one knowing da the message must really be from Alice!.

Alice can also first sign her name (or indeed the entire message!) by raising it to her own da - th power (mod ma) and THEN encrypt it as usual by raising it to the eb - th power (mod mb). This provides extra security. Using this approach Alice doesn't have to sign just her name, and she can sign the entire message!