

A NOTE ON DIGITAL SIGNATURES AND CHOOSING APPROPRIATE SIZE BLOCKS.

One sometimes has to be careful choosing block sizes for signatures. It is always important for each block to be a number less than m . Since digital signatures is a two - stepped process with raising to powers for two $(\text{mod } m)$, it is possible for the output of the first step to be too large for the second step. Here is an example of how block sizes should be handled.

For this example we will assume Alice ' s (m_a, e_a, d_a) are : $(10573258010425563407, 775523, 4081115647521015467)$

Alice = 1122191315

Here is $(\text{Alice})^d \pmod{m}$ (Her d and m).

```
In[23]:= PowerMod[1122191315, 4081115647521015467, 10573258010425563407]
Out[23]= 1076266189596082476
```

Now let ' s suppose Bob ' s $(m_b, e_b, d_b) = (30493567675321, 3, 20329028612363)$

Alice checks to see if the first step of her signature is bigger than his m .

```
In[24]:= 1076266189596082476 - 30493567675321
Out[24]= 1076235696028407155
```

It is, so she must break this message into blocks that are less than his m before encrypting it. 1076266189 , 596082476

She now encrypts each block using his e and m .

```
In[25]:= PowerMod[1076266189, 3, 30493567675321]
Out[25]= 20610599363873
```

```
In[26]:= PowerMod[596082476, 3, 30493567675321]
```

```
Out[26]= 5115985047749
```

So Alice ' s signature she sends to Bob is then

20610599363873, 5115985047749

Now Bob receives the above. He first raises the above to d (mod m) for his d and m. (First noting that each is less than his m) .

```
In[27]:= PowerMod[20610599363873, 20329028612363, 30493567675321]
```

```
Out[27]= 1076266189
```

```
In[28]:= PowerMod[5115985047749, 20329028612363, 30493567675321]
```

```
Out[28]= 596082476
```

So he gets : 1076266189, 596082476 . Since he expects a signature from Alice he raises each to the e power (mod m) for HER e AND m. He get :

```
In[29]:= PowerMod[1076266189, 775523, 10573258010425563407]
```

```
Out[29]= 9084275315466891653
```

```
In[30]:= PowerMod[596082476, 775523, 10573258010425563407]
```

```
Out[30]= 4865827453965816132
```

However these look like garbage ! I.e. do not correspond to letters.

But if he considers the blocks 1076266189, 596082476 he realizes that ignoring the comma gives the integer 1076266189596082476 which is still less than Alices m. I.e.

```
In[31]:= 10573258010425563407 - 1076266189596082476
```

```
Out[31]= 9496991820829480931
```

So he raises this combination of the two blocks to Alice ' s e power (mod m) for her m and gets :

```
In[32]:= PowerMod[1076266189596082476, 775523, 10573258010425563407]
```

```
Out[32]= 1122191315
```

Now he consults the table and indeed sees that this indeed corresponds to Alice !

MORAL : Since digital signatures involve two different m 's you have to always make sure your "blocks" are $(\text{mod } m)$. Since decrypting a signature also involves two m 's if you get nonsense after the raising to the power for the second m , see if putting two (or more) blocks you got together is an integer still less than m , in which case you should treat it as a single block.

Moral : You want $(b^e)^d = (b^d)^e = b \pmod{m}$, but this only holds if the block b is the appropriate size.