

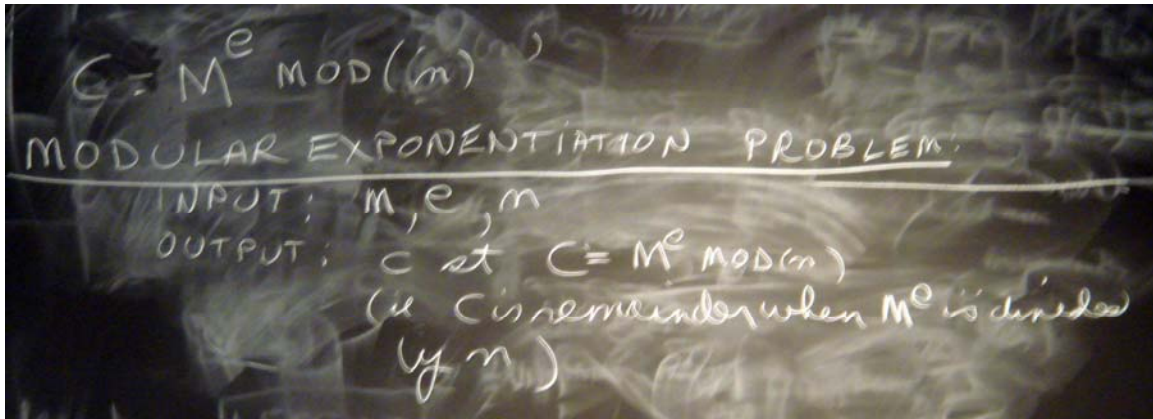
CSCI 303 Introduction to Algorithms
Spring 2007
March 28th, 2007 class notes

Quiz 2 will be on Monday, April 9th, in class. It will cover homeworks 6, 7, and 8.

Modular Exponentiation Problem:

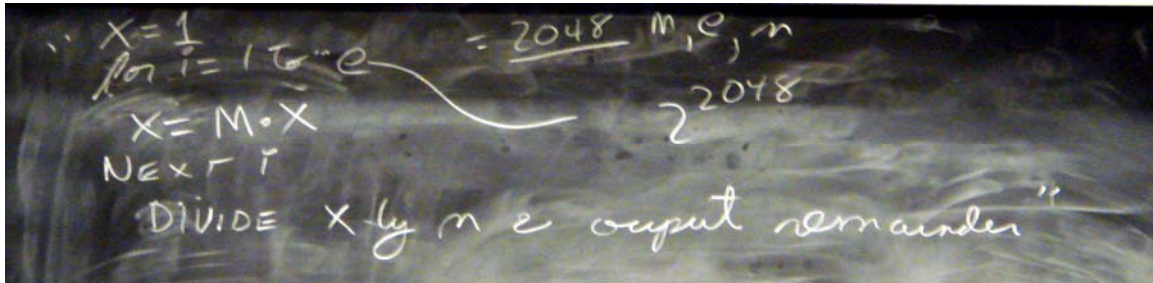
Input: m, e, n

Output: c such that $c = m^e \bmod n$ (i.e. c is the remainder of $\frac{m^e}{n}$).



Algorithm 1:

```
x = 1
for i = 1 to e
    x *= m
divide x by n and output the remainder
```



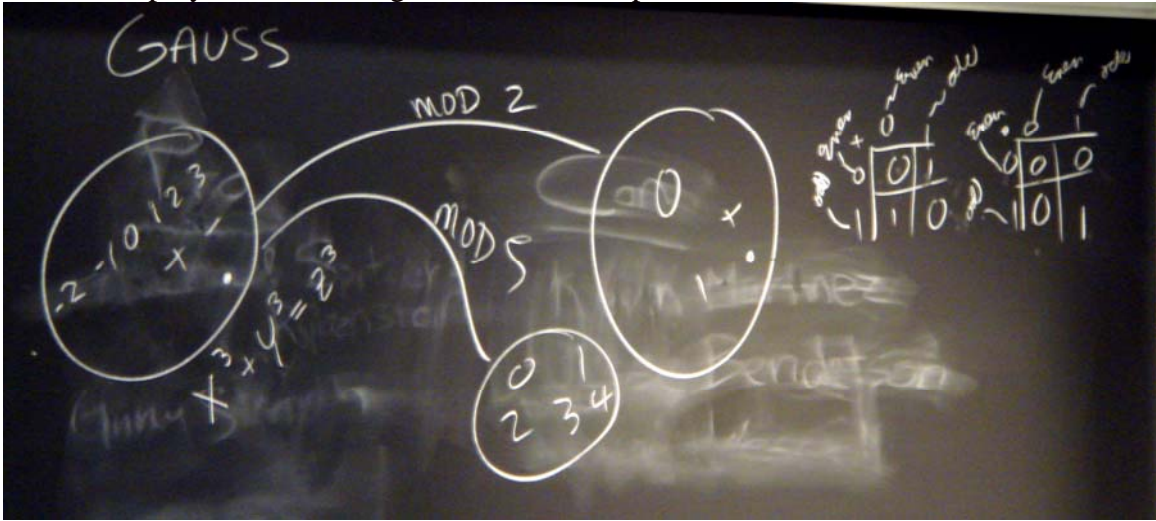
Algorithm 1 is not polynomial!

Algorithm 2:

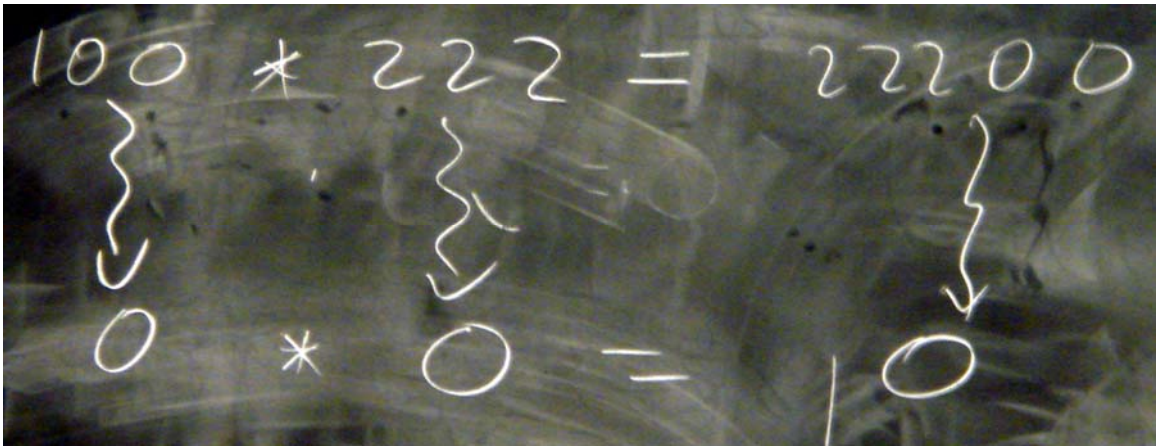
We can use repeated squaring. For $e \approx 2^k$, find m^{2^i} for $i = 1, 2, \dots, k$ and multiply the m^{2^i} for all i such that 2^i is in the binary representation of e .

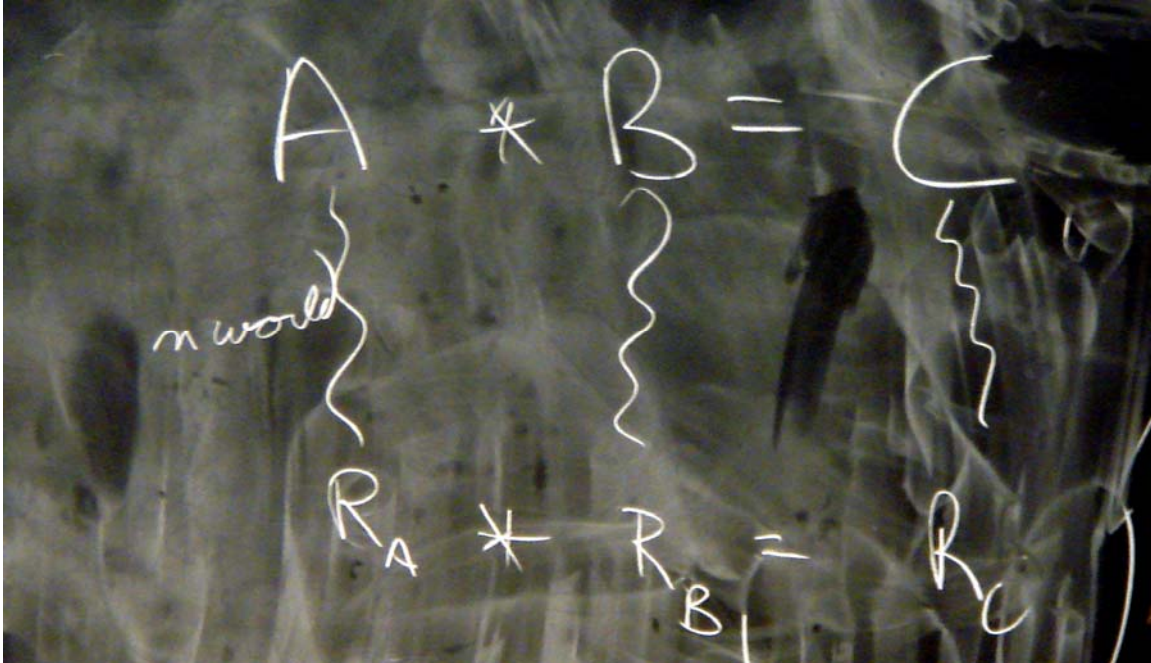
Algorithm 2 is still not polynomial! Note that m^e , for $m, e \approx 2^k$ is HUGE. $m^e \approx (2^k)^{2^k}$ (we can't even write that many digits down!)

We want a polynomial time algorithm. Gauss explored modular arithmetic:

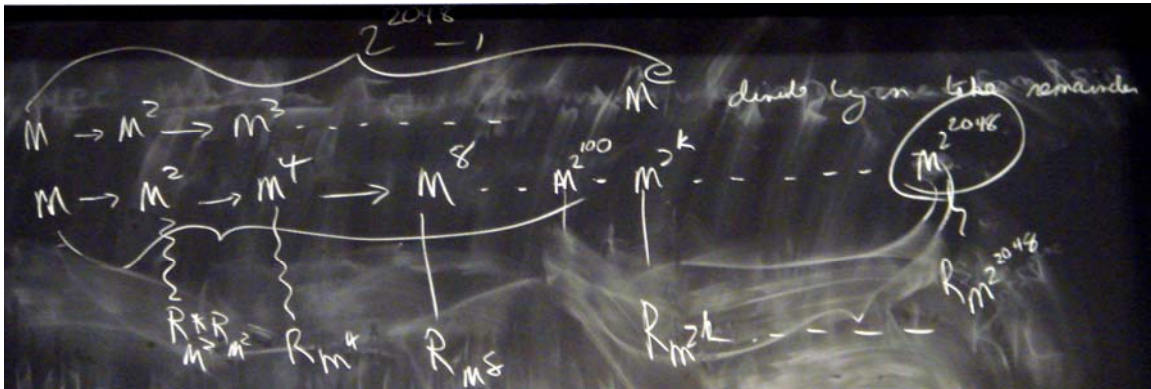


Note that $100 \bmod 2 * 222 \bmod 2 = (100 * 222) \bmod 2$. In general:
 $A * B = C \Rightarrow A \bmod n * B \bmod n = C \bmod n$.





Algorithm 3: use repeated squaring. For $e \approx 2^k$, find m^{2^i} for $i = 1, 2, \dots, k$ and multiply the m^{2^i} for all i such that 2^i is in the binary representation of e ; however, perform all the multiplications mod n , so that you never have to write down more digits than n has.



So Modular Exponentiation Problem is solvable in polynomial time. Therefore, we can encrypt and decrypt in polynomial time.

We have covered all of RSA, except how to find large primes quickly.

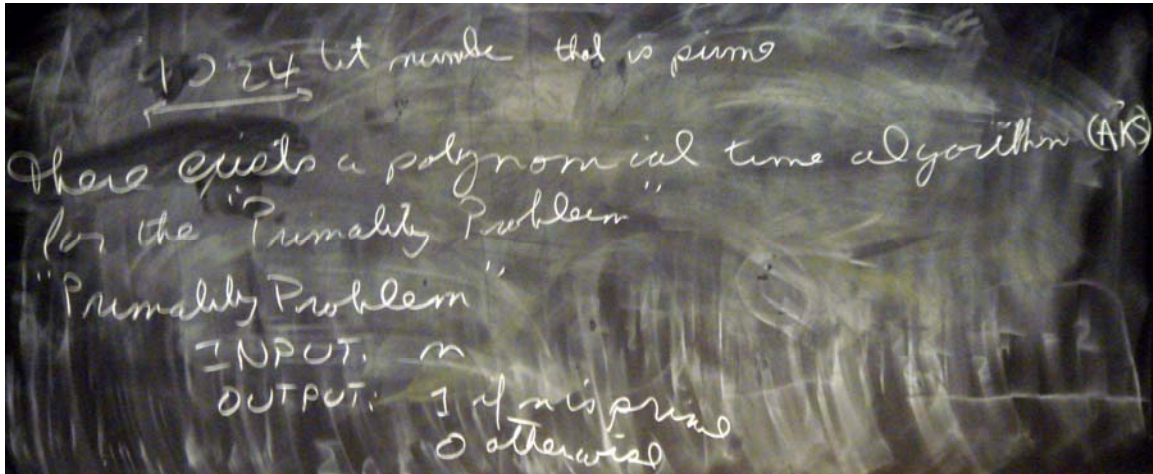
Primality Problem:

Input: n

Output: 1 if n is prime and 0 if n is composite

There exists a polynomial time algorithm for the Primality Problem: the AKS algorithm.

Gauss conjectured in 1801 and Hadamard & Valle Poisson proved in 1898 the prime number theorem: $\text{Prob}[\text{a number around } 2^n \text{ is prime}] \approx 1/n$.

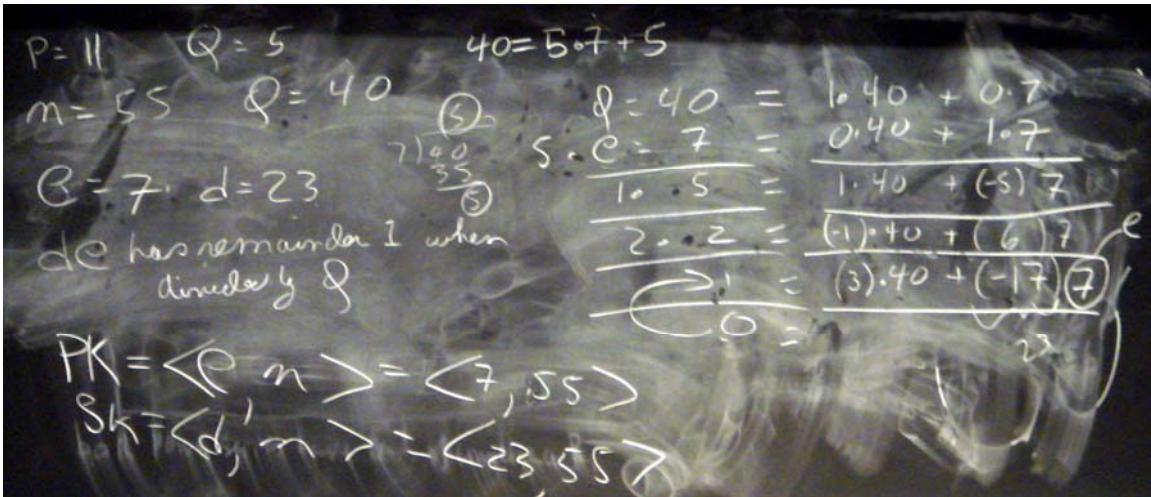


Thus we can guess a prime and check if it is in fact a prime. This approach will take us polynomial expected time to find a prime.

Summary of RSA, with an example:

Alice creates public and secret keys:

1. Pick 2 primes by trial and error.
 $p = 11, q = 5$
2. $n = 55, \phi(n) = 40$
3. By trial and error, find e such that $\text{GCD}(e, 40) = 1$.
Use Euclid's algorithm to check the GCD.
 $e = 7$
4. Find d such that $ed \equiv 1 \pmod{55}$
Use extended Euclid's algorithm.
 $d = 23$
5. $\text{PK} = \langle 7, 55 \rangle$
6. $\text{SK} = \langle 23, 55 \rangle$



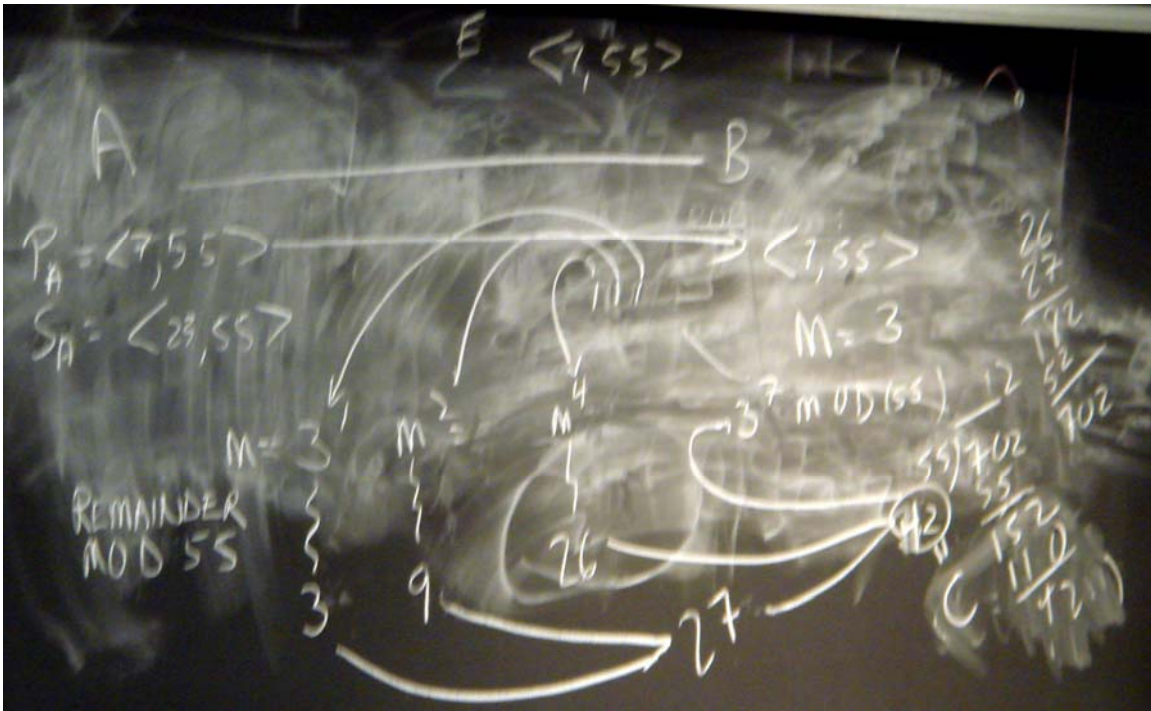
Bob wants to send Alice a message, so he asks for her public key. She sends $\langle 7, 55 \rangle$. Bob's message $m = 3$.

Bob computes $c = m^e \bmod n = 3^7 \bmod 55$. Note that $7 = 111_2$.

Normal world:	$m = 3$	3^2	3^4
Mod 55 world	3	9	$81 \bmod 55 = 26$

$$3^7 \bmod 55 = 3^1 3^2 3^4 \bmod 55 = 27 * 26 \bmod 55 = 702 \bmod 55 = 42 = c = P_A(m).$$

Bob sends $c = 42$ back to Alice.

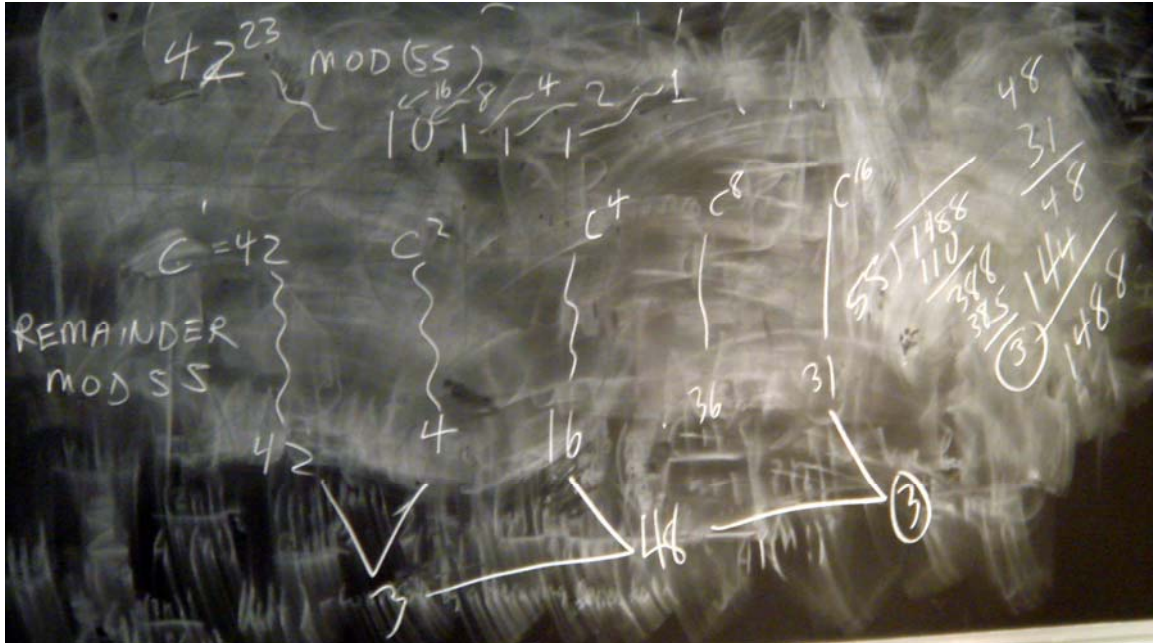


To decrypt, Alice has to compute $m = c^d \bmod n = 42^{23} \bmod 55$. Note that $23 = 10111_2$.

Normal world:	$m=42$	42^2	42^4	42^8	42^{16}
Mod 55 world:	42	4	16	36	31

$$42^{23} \bmod 55 = 42^1 42^2 42^4 42^{16} \bmod 55 = (42 * 4 * 16 * 31) \bmod 55 = (3 * 16 * 31) \bmod 55 = (48 * 31) \bmod 55 = 3 = m$$

Thus Alice has decoded m!



Eve has only $n = 55$, $e = 7$, $c = 42$.

Remember the five requirements for a PKC:

Requirements for a PKC (Public Key Cryptosystem):

- i) For all m , $P_A, S_A, S_A(P_A(m)) = m$.
- ii) The problem of generating a "mated pair" of keys $\langle P, S \rangle$ must be solvable in polynomial time.
- iii) The encryption problem:
 Input: P_A, m
 Output: $P_A(m)$
 must be solvable in polynomial time.
- iv) The decryption problem:
 Input: S_A, c
 Output: $S_A(c)$
 must be solvable in polynomial time.
- v) The code breaking problem:
 Input: P_A, c
 Output: m
 should not be solvable in polynomial time.

We have shown that RSA satisfied i, ii, iii, iv. As far as v goes, if one could factor in polynomial time, then Eve could see that $55 = 11 * 5$ and compute $\phi(n) = 40$, then find d by using extended Euclid algorithm on 55 and 7, and then decrypt! No one has proven factoring can or cannot be done in polynomial time though many have tried. It's also possible that there is some other way to break RSA. However, as far as we know, so far RSA has not been broken!