Prime Numbers

Prime Numbers

- prime numbers only have divisors of 1 and self
  - they cannot be written as a product of other numbers
  - note: 1 is prime, but is generally not of interest
- eg. 2,3,5,7 are prime, 4,6,8,9,10 are not
- prime numbers are central to number theory
- list of prime number less than 200 is:

  2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199

Prime Factorisation

- to **factor** a number `n` is to write it as a product of other numbers: `n=a x b x c`
- note that factoring a number is relatively hard compared to multiplying the factors together to generate the number
- the **prime factorisation** of a number `n` is when its written as a product of primes
  - eg. $91=7\times13$; $3600=2^4\times3^2\times5^2$

$$a = \prod_{p \in \mathrm{P}} p^{a_p}.$$

Relatively Prime Numbers & GCD

- two numbers `a, b` are **relatively prime** if have **no common divisors** apart from 1
  - eg. 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor
- conversely can determine the greatest common divisor by comparing their prime factorizations and using least powers
  - **eg.** $300=2^1 x 3^1 x 5^2$ $18=2^1 x 3^2$ **hence** $GCD(18,300)=2^1 x 3^1 x 5^0=6$

Fermat's Theorem

- $a^{p-1} = 1 \pmod{p}$
  - where p is prime and $\gcd(a,p)=1$
- also known as Fermat's Little Theorem
- also have: $a^p = a \pmod{p}$
- useful in public key and primality testing

Euler Totient Function ø(n)

- when doing arithmetic modulo n
- **complete set of residues** is: `0..n-1`
- **reduced set of residues** is those numbers (residues) which are relatively prime to n
  - eg for n=10,
  - complete set of residues is {0,1,2,3,4,5,6,7,8,9}
  - reduced set of residues is {1,3,7,9}
- number of elements in reduced set of residues is called the **Euler Totient Function ø(n)**

Euler Totient Function $\varnothing(n)$

- to compute ø(n) need to count number of residues to be excluded
- in general need prime factorization, but
  - for p (p prime) $\varnothing(p)=p-1$
  - for p.q (p,q prime) $\varnothing(p.q)=(p-1)\times(q-1)$
- eg.

$\varnothing(37) = 36$

$\varnothing(21) = (3-1)\times(7-1) = 2\times6 = 12$

Euler's Theorem

- a generalisation of Fermat's Theorem
- $a^{\varnothing(n)} = 1 \pmod{n}$
  - for any a,n where gcd(a,n)=1
- eg.

  $a=3; n=10;\ \varnothing(10)=4;$

  hence $3^4 = 81 = 1 \bmod 10$

  $a=2; n=11;\ \varnothing(11)=10;$

  hence $2^{10} = 1024 = 1 \bmod 11$
- also have: $a^{\varnothing(n)+1} = a \pmod{n}$

Primality Testing

- often need to find large prime numbers
- traditionally **sieve** using **trial division**
  - ie. divide by all numbers (primes) in turn less than the square root of the number
  - only works for small numbers
- alternatively can use statistical primality tests based on properties of primes
  - for which all primes numbers satisfy property
  - but some composite numbers, called pseudo-primes, also satisfy the property
- can use a slower deterministic primality test

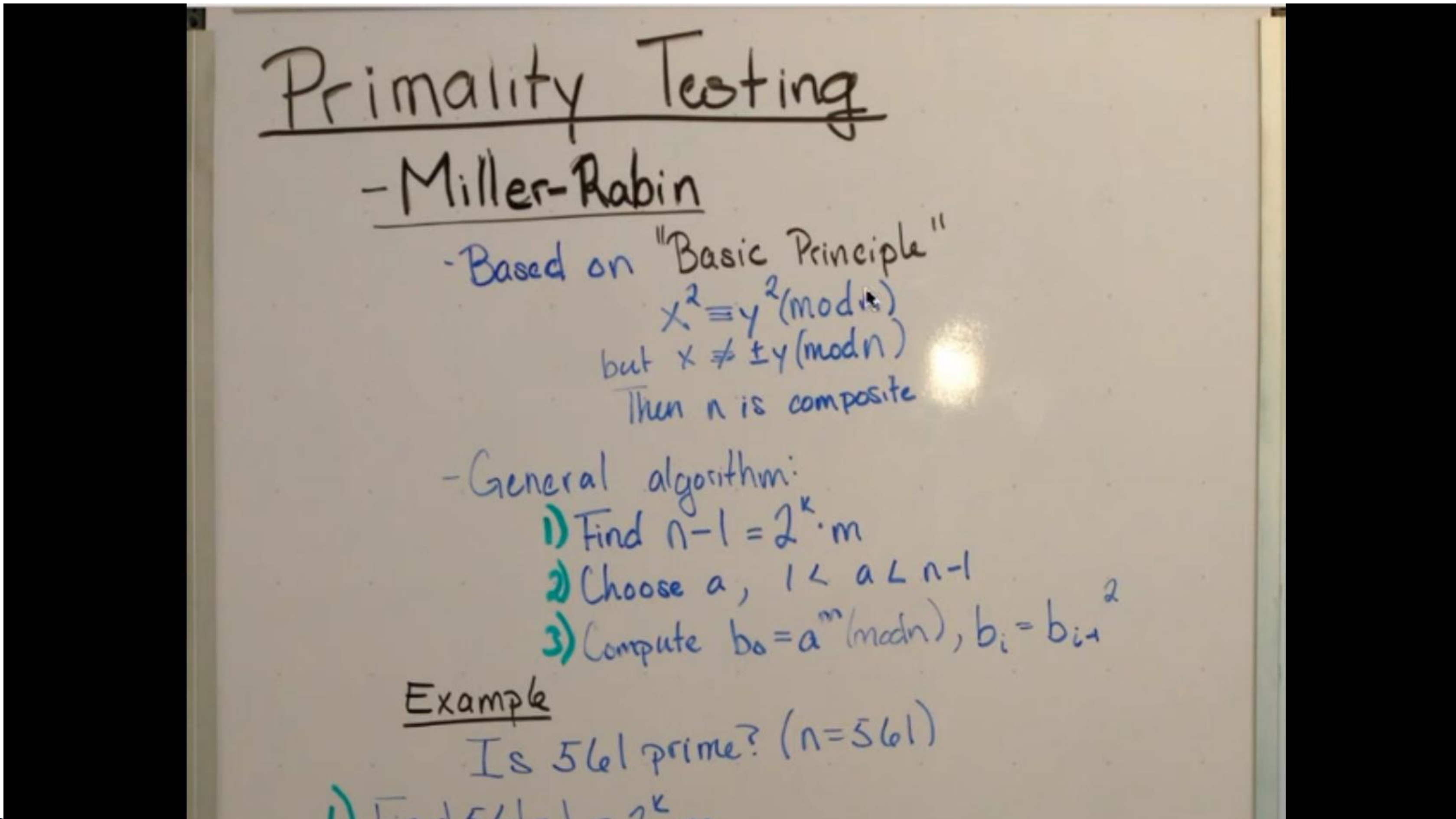
Miller Rabin Algorithm

- a test based on prime properties that result from Fermat's Theorem
- algorithm is:

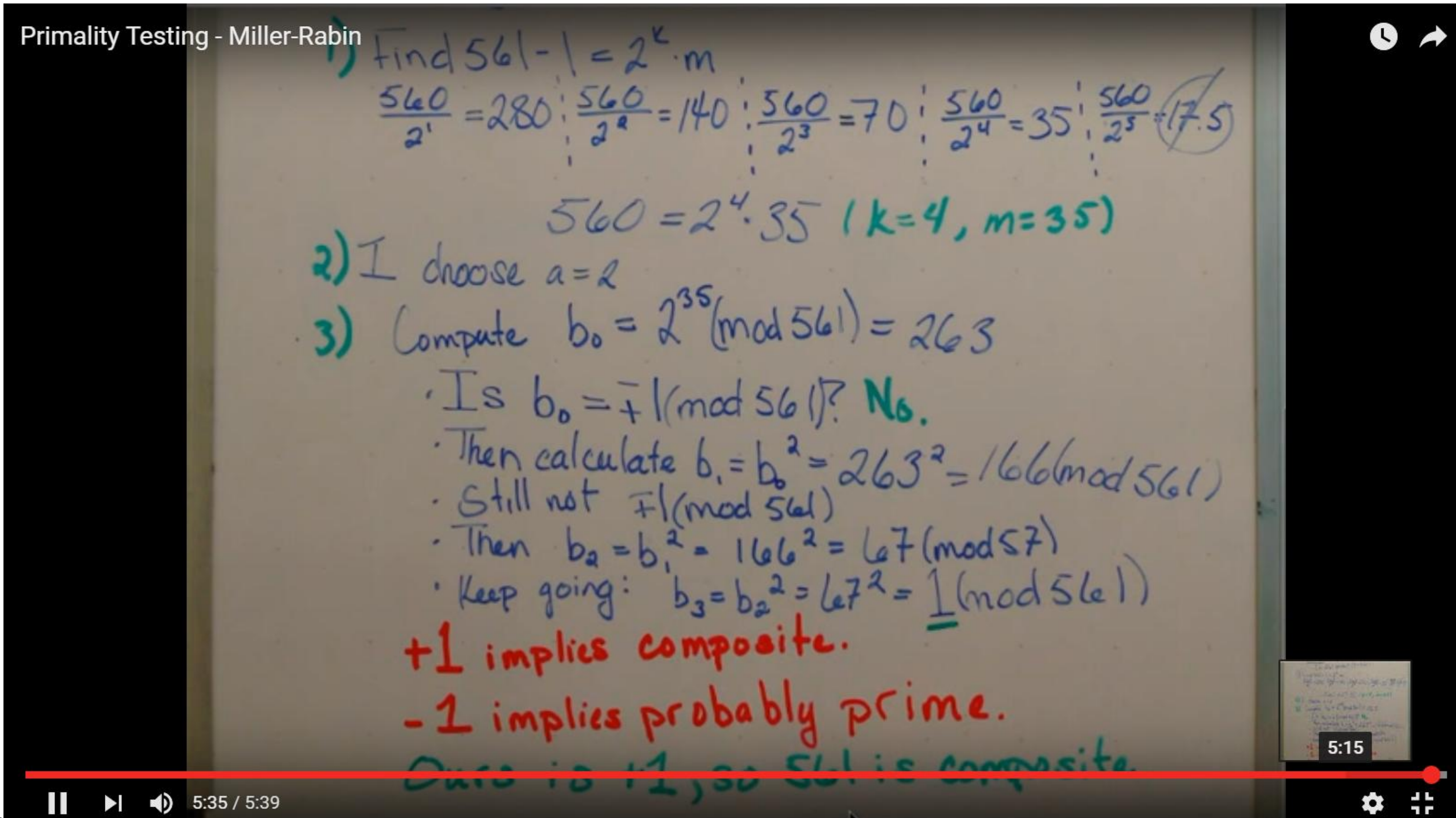
TEST (n) is:

1. Find integers k, q, $k > 0$, q odd, so that $(n-1)=2^k q$
2. Select a random integer a, $1<a<n-1$
3. **if** $a^q \bmod n = 1$ **then** return ("inconclusive");
4. **for** $j = 0$ **to** $k - 1$ **do**
   5. **if** ($a^{2^j q} \bmod n = n-1$)
      then return("inconclusive")
6. return ("composite")

Miller Rabin Algorithm



Miller Rabin Algorithm



Probabilistic Considerations

- if Miller–Rabin returns “composite” the number is definitely not prime
- otherwise is a prime or a pseudo-prime
- chance it detects a pseudo-prime is $< {}^1/_4$
- hence if repeat test with different random a then chance n is prime after t tests is:
  - Pr(n prime after t tests) = $1-4^{-t}$
  - eg. for t=10 this probability is > 0.99999
- could then use the deterministic AKS test

Chinese Remainder Theorem

- used to speed up modulo computations
- if working modulo a product of numbers
  - eg. $\mathrm{mod}\ M = m_1 m_2 .. m_k$
- Chinese Remainder theorem lets us work in each moduli m_i separately
- since computational cost is proportional to size, this is faster than working in the full modulus M

Chinese Remainder Theorem

- can implement CRT in several ways
- to compute $A \pmod{M}$
  - first compute all $a_i = A \bmod m_i$ separately
  - determine constants c_i below, where $M_i = M/m_i$
  - then combine results to get answer using:

$$A \equiv \left(\sum_{i=1}^{k} a_i c_i\right) \pmod{M}$$

$$c_i = M_i \times \left(M_i^{-1} \bmod m_i\right) \quad \text{for } 1 \le i \le k$$

What's x such that:

$x \equiv 2 \pmod{3}$
$x \equiv 3 \pmod{5}$
$x \equiv 2 \pmod{7}$?

(So, $a_1 = 2$, etc. and $m_{1} = 3$ etc.)

$m = m_1 \cdot \ldots \cdot m_n$.

$M_i = m / m_i$

$y_i M_i \equiv 1 \pmod{m_i}$.

$x = \Sigma_i \, a_i y_i M_i$

Using the Chinese Remainder theorem:

- $m = 3 \times 5 \times 7 = 105$
- $M_1 = m/3 = 105/3 = 35$
- 2 is an inverse of $M_1 = 35 \pmod{3}$ (since $35 \times 2 \equiv 1 \pmod{3}$)
- $M_2 = m/5 = 105/5 = 21$
- 1 is an inverse of $M_2 = 21 \pmod{5}$ (since $21 \times 1 \equiv 1 \pmod{5}$)
- $M_3 = m/7 = 15$
- 1 is an inverse of $M_3 = 15 \pmod{7}$ (since $15 \times 1 \equiv 1 \pmod{7}$)
- So , $x \equiv 2 \times 2 \times 35 + 3 \times 1 \times 21 + 2 \times 1 \times 15 = 233 \equiv 23 \pmod{105}$
- So answer: $x \equiv 23 \pmod{105}$

We're solving equations in modular arithmetic!!

Primitive Roots

- from Euler's theorem have $a^{\varnothing(n)} \bmod n = 1$
- consider $a^m = 1 \pmod n$, $GCD(a,n)=1$
  - must exist for $m = \varnothing(n)$ but may be smaller
  - once powers reach m, cycle will repeat
- if smallest is $m = \varnothing(n)$ then a is called a **primitive root**
- if p is prime, then successive powers of a "generate" the group $\bmod p$
- these are useful but relatively hard to find

Powers mod 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

Discrete Logarithms

- the inverse problem to exponentiation is to find the **discrete logarithm** of a number modulo p
- that is to find $\mathtt{i}$ such that $\mathtt{b} = \mathtt{a}^{\mathtt{i}} \pmod{\mathtt{p}}$
- this is written as $\mathtt{i} = \mathtt{dlog}_{\mathtt{a}}\ \mathtt{b} \pmod{\mathtt{p}}$
- if $\mathtt{a}$ is a primitive root then it always exists, otherwise it may not, eg.

  $x = \log_3 4 \bmod 13$ has no answer

  $x = \log_2 3 \bmod 13 = 4$ by trying successive powers
- whilst exponentiation is relatively easy, finding discrete logarithms is generally a **hard** problem

Summary

- have considered:
  - prime numbers
  - Fermat's and Euler's Theorems & ø(n)
  - Primality Testing
  - Chinese Remainder Theorem
  - Primitive Roots & Discrete Logarithms