Primality testing

- Let \( \pi(N) \) be the number of prime numbers that are less than or equal to \( N \).
- Prime number theorem: \( \pi(N) \approx \frac{N}{\ln(N)} \)
  - If \( n \geq 17 \), \( \frac{N}{\ln(N)} < \pi(N) < 1.26 \frac{N}{\ln(N)} \)
- e.g. \( N = p \) or \( q \) in \( n=pq \). \( p \approx q \approx 2^{512} \)
  - \( \Pr(N \text{ is a prime} | \text{odd } N \approx 2^{512} ) \approx \frac{1}{\ln(2^{512})} = \frac{1}{355} \)
- Easier to prove “\( N \)” is composite than to prove “\( N \)” is a prime. (why ?)
Definition 5.1

- **yes-biased Monte Carlo algorithm**: a randomized algorithm in which “yes” is always correct, “no” may be incorrect.
  - “yes” or “no” is a decision by an algorithm for a problem (e.g. Is N composite ?)
  - \( \text{pr(incorrect answer| “yes”) = 0.} \)
- If \( \text{pr(incorrect answer| “no”) \leq \varepsilon} \), then the algorithm has error probability of \( \varepsilon \).
Problem 5.1 (page 179)

- Q: Is N a composite?
- If we answer “yes” because a factor found (or other method), then it is definitely composite.
  - for yes-biased Monte Carlo algorithm
    \[ \text{pr(composite|“yes”) = 1}. \]
- error probability \( \text{pr(composite|“no”) \leq \varepsilon} \).
  - \( \text{pr(prime|“no”) > 1-\varepsilon} \).
Primality test for a large integer

- **probabilistic test:**
  - can be highly efficient
  - tiny probability of making an error

- **deterministic test:**
  - no general polynomial-time algorithm available until Agrawal, Kayal, and Saxena (2002) [AKS algorithm]
  - AKS algorithm is still not yet practical for a large prime number.
Probabilistic composite tests

- Solovay-Strassen Algorithm (Algorithm 5.6, page 182) is a yes-biased algorithm with error probability at most $1/2$.
- Miller-Rabin Algorithm (Algorithm 5.7, page 188) is a yes-biased algorithm with error probability at most $1/4$.
- We can perform several independent random tests to drastically reduce the error probability.
Miller-Rabin Algorithm

- Let \( n-1 = 2^k \cdot m \), \( m \) is odd.
- Choose "\( a \)" randomly between 1 to \( n-1 \).
- \( b \leftarrow a^m \mod n \).
- if \( b = 1 \), then return ("prime")
- for \( i=0 \) to \( k-1 \) do
  - if \( b=-1 \mod n \), then return ("prime")
  - else \( b \leftarrow b^2 \mod n \).
- return ("composite")
Miller-Rabin Algorithm

- \( n-1 = 2^k \, m, \, m \text{ is odd.} \)
- Choose “\( a \)” randomly between 1 to \( n-1 \).
- \( b \leftarrow a^m \mod n. \)
- if \( b = 1 \), then return (“prime”)
- for \( i=0 \) to \( k-1 \) do
  - if \( b=-1 \), then return (“prime”)
  - else \( b \leftarrow b^2 \mod n. \)
- return (“composite”)

- Recall that for a prime \( n \),
  - \( a^{n-1}=1 \mod n. \)
- If \( a^{n-1}=1 \mod n \), then \( n \) can be a possible prime.
- If many such “\( a \)” are found, it is very likely \( n \) is a prime.
Factoring Algorithms

- Factorization is much harder than primality testing.
  - no known polynomial time algorithm.
- Trial division: try all primes $p \leq n^{0.5}$.
  - simple but not very efficient for a large $n$.
  - suitable when $n$ has many small prime factors.
  - not effective for RSA (why not?).
Pollard p-1 Algorithm (n,B)

- Choose a “suitable” bound B
- \( a \leftarrow 2 \)
- for j from 2 to B do
  - \( a \leftarrow a^j \mod n \)
- \( d = \gcd(a-1, n) \)
- if(1 < d < n) return (d) else return (“fail”)
  - useful when all prime factor of \((p-1) < B\), where \( p | n \).
Pollard rho Algorithm \((n, x_1)\)

- Choose a initial sees \(x_1\) and \(f(x) = x^2 + 1\)
- \(x \leftarrow x_1, x' \leftarrow f(x) \mod n,\)
- \(p \leftarrow \gcd(x-x', n).\)
- while (\(p=1\)) do
  - \(x \leftarrow f(x) \mod n,\)
  - \(x' \leftarrow f(x') \mod n, x' \leftarrow f(x') \mod n\)
  - \(p \leftarrow \gcd(x-x', n)\)
- if(\(p<n\)) return (\(p\)) else return ("fail")
## Factoring Algorithm in Practice

<table>
<thead>
<tr>
<th>Factoring Algorithms</th>
<th>Asymptotic Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>quadratic sieve</td>
<td>$O(\exp[(1+o(1)) \left[(\ln n) \ln \ln(n)\right]^{0.5}])$</td>
</tr>
<tr>
<td>elliptic curve (p is smallest prime</td>
<td>$n)</td>
</tr>
<tr>
<td>number field sieve</td>
<td>$O(\exp[(1.92+o(1)) \left((\ln n)^{1/3} \left[\ln \ln(n)\right]^{2/3}\right)])$</td>
</tr>
</tbody>
</table>
RSA Challenges

- Early 1990s. RSA-d, \( d = \# \text{of digits in } n=\text{pq} \). RSA-100, RSA-110, ..., RSA-500.
  - RSA-160 found in 2003.

- Since 2001. RSA-b, \( b = \# \text{of bits in } n=\text{pq} \).
  - RSA-576, RSA-640, RSA-704, RSA-768, RSA-896, RSA-1024, RSA-1576.
  - Prizes: from $10,000 to $200,000.
Other Attacks on RSA

- computing \( \varphi(n) \)
- find decryption exponent (d) of RSA
- Wiener’s low decryption exponent attack.
Computing $\phi(n)$

- If both $n$ and $\phi(n)$ are known, then we can find the factorization of $n$. Why?
- $n=pq$, $\phi(n) = (p-1)(q-1)$.
- Plug $q=n/p$ in the equation for $\phi(n)$, $p^2 - (n - \phi(n)+1) p + n = 0$.
- We can then easily solve the quadratic equation of $p$. 
Example 5.13 (page 201)

- \( n = 84773093, \)
- \( \phi(n) = 84754668. \)
  - \( n - \phi(n) = 18425. \)

quadratic equation for \( p \) is

- \( p^2 - (n - \phi(n) + 1) p + n = 0. \)
- \( p^2 - 18426 p + 84773093 = 0. \)

Solutions (how ?) are: 9539 and 8887.
Find decryption exponent

- If the information about \( d \) (decryption exponent) is known, the \( n \) can be factored.
  - implication: we cannot not simply change other \( d \)'s with \( n \) unchanged.
  - Algorithm 5. 10: randomized algorithm that can factor \( n \) in a polynomial time (page 204)
Wiener’s low decryption exponent attack

- If $d$ (decryption exponent) is too small, the $n$ can be factored.
  - Algorithm 5. 11: based on extended Euclidean algorithm and continued fraction technique (page 204)
  - Implication: we cannot not use too small $d$’s as decryption exponent.
    - We choose simpler “$e$” as a public key then solve for “$d$” which is usually not too small.
Rabin Cryptosystem

- Another example of public key cryptosystem which is computationally secure against a chosen-plaintext attack (assuming \( n = pq \) is hard to be factored)

- **Drawback**: the encryption function is not 1-1. Decryption function may yield 4 possible plaintexts for a given ciphertext.
Algorithm 5.2 (Rabin)

- Let $n = p \cdot q$, $p = 3 \mod 4$, $q = 3 \mod 4$.
- $P = C = \mathbb{Z}_n^*$. $K = \{(n, p, q)\}$. $n$ is public key.
- Encryption:
  - $e_K(x) = x^2 \mod n$
- Decryption:
  - $d_K(y) = y^{1/2} \mod n$ (how?)
Find $y^{1/2} \mod n$?

- Require $n=pq$, $p=3 \mod 4$ and $q=3 \mod n$.
- Recall $y^{(p-1)} = 1 \mod p$ (and $y^{(q-1)} = 1 \mod q$).
- Since $y=x^2 \mod p$, we have $y^{(p-1)/2} = 1 \mod p$.
- We can show that $x= \pm y^{(p+1)/4} \mod p$ satisfies $x^2 = y \mod p$.
- Likewise, $x= \pm y^{(q+1)/4} \mod q$ satisfies $x^2 = y \mod q$.
- We can find four solutions using Chinese Remainder Theorem.
Example 5.18

- Choose small $n=77=7 \times 11$.
- $e_K(x)=x^2 \mod 77$, $d_K(y)=y^{1/2} \mod 77$.
- Suppose $y=23$ received. Find $x$?
  - $x = 23^{(7+1)/4} \mod 7 = 2^2 \mod 7 = 4$.
  - $x = 23^{(11+1)/4} \mod 11 = 1^3 \mod 11 = 1$.
- Use CRT, $x=\pm 4 \mod 7, x=\pm 1 \mod 11$, we find $x=\pm 10$ and $\pm 32 \mod 77$
  - $x=10, 32, 45, 67$. 
Security of Rabin Cryptosystem

- Rabin Cryptosystem can be proved using “Turing reduction”. (page 213)
- Factorization $\preceq_T$ Rabin decryption.
  - if we can solve Rabin decryption in polynomial time, then we can solve factorization problem in polynomial time.
  - factorization is hard, so is Rabin decryption.
Semantic Security of RSA

- total break: the private key is known to your adversary.
- partial break: some unseen ciphertexts can be decrypted by your adversary.
- distinguishability of ciphertexts: >1/2 chance to distinguish between encryption of two plaintexts.
Multi-precision arithmetic software packages

- Modular arithmetic operations are provided to compute residues, perform exponentiation, and compute multiplicative inverses:
  - **BigInteger** class in Java:
    - [http://java.sun.com/j2se/1.4.2/docs/api/java/math/BigInteger.html](http://java.sun.com/j2se/1.4.2/docs/api/java/math/BigInteger.html)
  - MAPLE or MATHEMATICA.
Summary

- Primality testing algorithms
- Factorization algorithms
- Attacks on RSA
- Rabin Cryptosystem
HW5

- (written assignment)
  - 5.3, 5.5, 5.6, 5.7, 5.33
- (program assignment)
  - 5.12.