CPSC 467b: Cryptography and Computer Security

Michael J. Fischer

Lecture 14
February 27, 2012

Quadratic Residues, Squares, and Square Roots

Square roots in $\mathbf{Z}_n^*$

Recall from lecture 13 that to find points on an elliptic curve requires solving the equation

$$y^2 = x^3 + ax + b$$

for $y \pmod{p}$, and that requires computing square roots in $\mathbf{Z}_p^*$.

Squares and square roots have several other cryptographic applications as well.

Today, we take a brief tour of the theory of *quadratic resides*.

Quadratic residues modulo n

An integer b is a *square root* of a modulo n if

$$b^2 \equiv a \pmod{n}.$$

An integer a is a *quadratic residue (or perfect square)* modulo n if it has a square root modulo n.

Quadratic residues in $\mathbf{Z}_n^*$

If $a, b \in \mathbf{Z}_n$ and $b^2 \equiv a \pmod{n}$, then

$$b \in \mathbf{Z}_n^* \text{ iff } a \in \mathbf{Z}_n^*.$$

Why? Because

$$\gcd(b, n) = 1 \text{ iff } \gcd(a, n) = 1$$

This follows from the fact that $b^2 = a + un$ for some u, so if p is a prime divisor of n, then

$$p \mid b \text{ iff } p \mid a.$$

Assume that all quadratic residues and square roots are in $\mathbf{Z}_n^*$ unless stated otherwise.

QR_n and QNR_n

We partition $\mathbf{Z}_n^*$ into two parts.

$$\mathrm{QR}_n = \{a \in \mathbf{Z}_n^* \mid a \text{ is a quadratic residue modulo } n\}.$$
$$\mathrm{QNR}_n = \mathbf{Z}_n^* - \mathrm{QR}_n.$$

QR_n is the *set of quadratic residues* modulo n.

QNR_n is the *set of quadratic non-residues* modulo n.

For $a \in \mathrm{QR}_n$, we sometimes write

$$\sqrt{a} = \{b \in \mathbf{Z}_n^* \mid b^2 \equiv a \pmod{n}\},$$

the *set of square roots* of a modulo n.

Quadratic residues in $\mathbf{Z}_{15}^*$

The following table shows all elements of $\mathbf{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$ and their squares.

b		$b^2 \bmod 15$
1		1
2		4
4		1
7		4
8	$= -7$	4
11	$= -4$	1
13	$= -2$	4
14	$= -1$	1

Thus, $\mathrm{QR}_{15} = \{1, 4\}$ and $\mathrm{QNR}_{15} = \{2, 7, 8, 11, 13, 14\}$.

Quadratic residues modulo an odd prime p

Fact

For an odd prime p,

- *Every $a \in QR_p$ has exactly two square roots in $\mathbf{Z}_p^*$;*
- *Exactly 1/2 of the elements of $\mathbf{Z}_p^*$ are quadratic residues.*

In other words, if $a \in \mathrm{QR}_p$,

$$|\sqrt{a}| = 2.$$

$$|\mathrm{QR}_n| = |\mathbf{Z}_p^*|/2 = \frac{p-1}{2}.$$

Quadratic residues in $\mathbf{Z}_{11}^*$

The following table shows all elements $b \in \mathbf{Z}_{11}^*$ and their squares.

b	$b^2 \bmod 11$
1	1
2	4
3	9
4	5
5	3

b	$-b$	$b^2 \bmod 11$
6	-5	3
7	-4	5
8	-3	9
9	-2	4
10	-1	1

Thus, $\mathrm{QR}_{11} = \{1, 3, 4, 5, 9\}$ and $\mathrm{QNR}_{11} = \{2, 6, 7, 8, 10\}$.

Proof that $|\sqrt{a}| = 2$ modulo an odd prime p

Let $a \in \mathrm{QR}_p$.

- It must have a square root $b \in \mathbf{Z}_p^*$.
- $(-b)^2 \equiv b^2 \equiv a \pmod{p}$, so $-b \in \sqrt{a}$.
- Moreover, $b \not\equiv -b \pmod{p}$ since $p \nmid 2b$, so $|\sqrt{a}| \geq 2$.
- Now suppose $c \in \sqrt{a}$. Then $c^2 \equiv a \equiv b^2 \pmod{p}$.
- Hence, $p \mid c^2 - b^2 = (c-b)(c+b)$.
- Since p is prime, then either $p \mid (c-b)$ or $p \mid (c+b)$ (or both).
- If $p \mid (c-b)$, then $c \equiv b \pmod{p}$.
- If $p \mid (c+b)$, then $c \equiv -b \pmod{p}$.
- Hence, $c = \pm b$, so $\sqrt{a} = \{b, -b\}$, and $|\sqrt{a}| = 2$.

Proof that half the elements of $\mathbf{Z}_p^*$ are in QR_p

- Each $b \in \mathbf{Z}_p^*$ is the square root of exactly one element of QR_p.
- The mapping $b \mapsto b^2 \bmod p$ is a 2-to-1 mapping from $\mathbf{Z}_p^*$ to QR_p.
- Therefore, $|\mathrm{QR}_p| = \frac{1}{2}|\mathbf{Z}_p^*|$ as desired.

Quadratic residues modulo pq

We now turn to the case where $n = pq$ is the product of two distinct odd primes.

Fact

Let $n = pq$ for p, q distinct odd primes.

- *Every $a \in QR_n$ has exactly four square roots in $\mathbf{Z}_n^*$;*
- *Exactly 1/4 of the elements of $\mathbf{Z}_n^*$ are quadratic residues.*

In other words, if $a \in \mathrm{QR}_n$,

$$|\sqrt{a}| = 4.$$

$$|\mathrm{QR}_n| = |\mathbf{Z}_n^*|/4 = \frac{(p-1)(q-1)}{4}.$$

Proof sketch

- Let $a \in \mathrm{QR}_n$. Then $a \in \mathrm{QR}_p$ and $a \in \mathrm{QR}_q$.
- There are numbers $b_p \in \mathrm{QR}_p$ and $b_q \in \mathrm{QR}_q$ such that
  - $\sqrt{a} \pmod{p} = \{\pm b_p\}$, and
  - $\sqrt{a} \pmod{q} = \{\pm b_q\}$.
- Each pair (x, y) with $x \in \{\pm b_p\}$ and $y \in \{\pm b_q\}$ can be combined to yield a distinct element $b_{x,y}$ in $\sqrt{a} \pmod{n}$.[1]
- Hence, $|\sqrt{a} \pmod{n}| = 4$, and $|\mathrm{QR}_n| = |\mathbf{Z}_n^*|/4$.

[1] To find $b_{x,y}$ from x and y requires use of the Chinese Remainder theorem.

Testing for membership in QR_p

Theorem (Euler Criterion)

An integer a is a non-trivial[2] *quadratic residue modulo an odd prime p iff*

$$a^{(p-1)/2} \equiv 1 \pmod{p}.$$

Proof in forward direction.

Let $a \equiv b^2 \pmod{p}$ for some $b \not\equiv 0 \pmod{p}$. Then

$$a^{(p-1)/2} \equiv (b^2)^{(p-1)/2} \equiv b^{p-1} \equiv 1 \pmod{p}$$

by Euler's theorem, as desired. □

[2] A non-trivial quadratic residue is one that is not equivalent to $0 \pmod{p}$.

Proof of Euler Criterion

Proof in reverse direction.

Suppose $a^{(p-1)/2} \equiv 1 \pmod{p}$. Clearly $a \not\equiv 0 \pmod{p}$. We find a square root b of a modulo p.

Let g be a primitive root of p. Choose k so that $a \equiv g^k \pmod{p}$, and let $\ell = (p-1)k/2$. Then

$$g^{\ell} \equiv g^{(p-1)k/2} \equiv (g^k)^{(p-1)/2} \equiv a^{(p-1)/2} \equiv 1 \pmod{p}.$$

Since g is a primitive root, $(p-1) \mid \ell$. Hence, $2|k$ and $k/2$ is an integer.

Let $b = g^{k/2}$. Then $b^2 \equiv g^k \equiv a \pmod{p}$, so b is a non-trivial square root of a modulo p, as desired. □

Finding Square Roots

Finding square roots modulo prime $p \equiv 3 \pmod 4$

The Euler criterion lets us test membership in QR_p for prime p, but it doesn't tell us how to quickly find square roots. They are easily found in the special case when $p \equiv 3 \pmod 4$.

Theorem

Let $p \equiv 3 \pmod 4$, $a \in \mathrm{QR}_p$. Then $b = a^{(p+1)/4} \in \sqrt{a} \pmod p$.

Proof.

$p + 1$ is divisible by 4, so $(p + 1)/4$ is an integer. Then

$$b^2 \equiv (a^{(p+1)/4})^2 \equiv a^{(p+1)/2} \equiv a^{1+(p-1)/2} \equiv a \cdot 1 \equiv a \pmod p$$

by the Euler Criterion. □

Finding square roots for general primes

We now present an algorithm due to D. Shanks[3] that finds square roots of quadratic residues modulo any odd prime p.

It bears a strong resemblance to the algorithm presented in lecture 9 for factoring the RSA modulus given both the encryption and decryption exponents.

[3] Shanks's algorithm appeared in his paper, "Five number-theoretic algorithms", in Proceedings of the Second Manitoba Conference on Numerical Mathematics, Congressus Numerantium, No. VII, 1973, 51–70. Our treatment is taken from the paper by Jan-Christoph Schlage-Puchta", "On Shank's Algorithm for Modular Square Roots", *Applied Mathematics E-Notes, 5* (2005), 84–88.

Shank's algorithm

Let p be an odd prime. Write $\phi(p) = p - 1 = 2^s t$, where t is odd. (Recall: s is # trailing 0's in the binary expansion of $p - 1$.)

Because p is odd, $p - 1$ is even, so $s \geq 1$.

A special case

In the special case when $s = 1$, then $p - 1 = 2t$, so $p = 2t + 1$.

Writing the odd number t as $2\ell + 1$ for some integer ℓ, we have

$$p = 2(2\ell + 1) + 1 = 4\ell + 3,$$

so $p \equiv 3 \pmod 4$.

This is exactly the case that we handled above.

Overall structure of Shank's algorithm

Let $p - 1 = 2^s t$ be as above, where p is an odd prime.

Assume $a \in \mathrm{QR}_p$ is a quadratic residue and $u \in \mathrm{QNR}_p$ is a quadratic non-residue.

We can easily find u by choosing random elements of $\mathbf{Z}_p^*$ and applying the Euler Criterion.

The goal is to find x such that $x^2 \equiv a \pmod{p}$.

Shanks's algorithm

Let s, t satisfy $p - 1 = 2^s t$ and t odd.
Let $u \in \mathrm{QNR}_p$.
$k = s$
$z = u^t \bmod p$
$x = a^{(t+1)/2} \bmod p$
$b = a^t \bmod p$
while $(b \not\equiv 1 \pmod{p})$ {
    let m be the least integer with $b^{2^m} \equiv 1 \pmod{p}$
    $y = z^{2^{k-m-1}} \bmod p$
    $z = y^2 \bmod p$
    $b = bz \bmod p$
    $x = xy \bmod p$
    $k = m$
}
return x

Loop invariant

The congruence

$$x^2 \equiv ab \pmod{p}$$

is easily shown to be a loop invariant.

It's clearly true initially since $x^2 \equiv a^{t+1}$ and $b \equiv a^t \pmod{p}$.

Each time through the loop, a is unchanged, b gets multiplied by y^2 (lines 10 and 11), and x gets multiplied by y (line 12); hence the invariant remains true regardless of the value of y.

If the program terminates, we have $b \equiv 1 \pmod{p}$, so $x^2 \equiv a$, and x is a square root of $a \pmod{p}$.

Termination proof (sketch)

The algorithm terminates after at most $s - 1$ iterations of the loop.

To see why, we look at the orders[4] of b and $z \pmod p$ and show the following loop invariant:

> *At the start of each loop iteration (before line 8),* $\mathrm{ord}(b)$ *is a power of 2 and* $\mathrm{ord}(b) < \mathrm{ord}(z) = 2^k$.

After line 8, $m < k$ since $2^m = \mathrm{ord}(b) < 2^k$. Line 13 sets $k = m$ for the next iteration, so k decreases on each iteration.

The loop terminates when $b \equiv 1 \pmod p$. Then $\mathrm{ord}(b) = 1 < 2^k$, so $k \geq 1$. Hence, the loop is executed at most $s - 1$ times.

[4] Recall that the order of an element g modulo p is the least positive integer k such that $g^k \equiv 1 \pmod p$.

QR Probabilistic Cryptosystem

A hard problem associated with quadratic residues

Let $n = pq$, where p and q are distinct odd primes.

Recall that each $a \in \mathrm{QR}_n$ has 4 square roots, and 1/4 of the elements in $\mathbf{Z}_n^*$ are quadratic residues.

Some elements of $\mathbf{Z}_n^*$ are easily recognized as non-residues, but there is a subset of non-residues (which we denote as Q_n^{00}) that are *hard to distinguish* from quadratic residues without knowing p and q.

This allows for public key encryption of single bits: A random element of QR_n encrypts 1; a random element of Q_n^{00} encrypts 0.

Quadratic residues modulo $n = pq$

Let $n = pq$, p, q distinct odd primes.

We divide the numbers in $\mathbf{Z}_n^*$ into four classes depending on their membership in QR_p and QR_q.[5]

- Let $Q_n^{11} = \{a \in \mathbf{Z}_n^* \mid a \in \mathrm{QR}_p \cap \mathrm{QR}_q\}$.
- Let $Q_n^{10} = \{a \in \mathbf{Z}_n^* \mid a \in \mathrm{QR}_p \cap \mathrm{QNR}_q\}$.
- Let $Q_n^{01} = \{a \in \mathbf{Z}_n^* \mid a \in \mathrm{QNR}_p \cap \mathrm{QR}_q\}$.
- Let $Q_n^{00} = \{a \in \mathbf{Z}_n^* \mid a \in \mathrm{QNR}_p \cap \mathrm{QNR}_q\}$.

Under these definitions, $$\mathrm{QR}_n = Q_n^{11}$$

$$\mathrm{QNR}_n = Q_n^{00} \cup Q_n^{01} \cup Q_n^{10}$$

[5] To be strictly formal, we classify $a \in \mathbf{Z}_n^*$ according to whether or not $(a \bmod p) \in \mathrm{QR}_p$ and whether or not $(a \bmod q) \in \mathrm{QR}_q$.

Quadratic residuosity problem

Definition (Quadratic residuosity problem)

The *quadratic residuosity problem* is to decide, given $a \in Q_n^{00} \cup Q_n^{11}$, whether or not $a \in Q_n^{11}$.

Fact

There is no known feasible algorithm for solving the quadratic residuosity problem that gives the correct answer significantly more than 1/2 the time for uniformly distributed random $a \in Q_n^{00} \cup Q_n^{11}$*, unless the factorization of* n *is known.*

Goldwasser-Micali probabilistic cryptosystem

The Goldwasser-Micali cryptosystem is based on the assumed hardness of the quadratic residuosity problem.

The public key consist of a pair $e = (n, y)$, where $n = pq$ for distinct odd primes p, q, and y is any member of Q_n^{00}.
The private key consists of p.
The message space is $\mathcal{M} = \{0, 1\}$. (Single bits!)

To encrypt $m \in \mathcal{M}$, Alice chooses a random $a \in \mathrm{QR}_n$.
She does this by choosing a random member of $\mathbf{Z}_n^*$ and squaring it.

If $m = 0$, then $c = a \bmod n \in Q_n^{11}$.
If $m = 1$, then $c = ay \bmod n \in Q_n^{00}$.

The problem of finding m given c is equivalent to the problem of testing if $c \in \mathrm{QR}_n (= Q_n^{11})$, given that $c \in Q_n^{00} \cup Q_n^{11}$.

Decryption in Goldwasser-Micali encryption

Bob, knowing the private key p, can use the Euler Criterion to quickly determine whether or not $c \in \mathrm{QR}_p$ and hence whether $c \in Q_n^{11}$ or $c \in Q_n^{00}$, thereby determining m.

Eve's problem of determining whether c encrypts 0 or 1 is the same as the problem of distinguishing between membership in Q_n^{00} and Q_n^{11}, which is just the quadratic residuosity problem, assuming the ciphertexts are uniformly distributed.

One can show that every element of Q_n^{11} is equally likely to be chosen as the ciphertext c in case $m = 0$, and every element of Q_n^{00} is equally likely to be chosen as the ciphertext c in case $m = 1$. If the messages are also uniformly distributed, then any element of $Q_n^{00} \cup Q_n^{11}$ is equally likely to be the ciphertext.

Important facts about quadratic residues

1. If p is odd prime, then $|\mathrm{QR}_p| = |\mathbf{Z}_p^*|/2$, and for each $a \in \mathrm{QR}_p$, $|\sqrt{a}| = 2$.
2. If $n = pq$, $p \neq q$ odd primes, then $|\mathrm{QR}_n| = |\mathbf{Z}_n^*|/4$, and for each $a \in \mathrm{QR}_n$, $|\sqrt{a}| = 4$.
3. Euler criterion: $a \in \mathrm{QR}_p$ iff $a^{(p-1)/2} \equiv 1 \pmod{p}$, p odd prime.
4. If n is odd prime, $a \in \mathrm{QR}_n$, can feasibly find $y \in \sqrt{a}$.
5. If $n = pq$, $p \neq q$ odd primes, then distinguishing Q_n^{00} from Q_n^{11} is believed to be infeasible. Hence, infeasible to find $y \in \sqrt{a}$. Why?
   If not, one could attempt to find $y \in \sqrt{a}$, check that $y^2 \equiv a \pmod{n}$, and conclude that $a \in Q^{11}$ if successful.

The Legendre and Jacobi Symbols

Legendre symbol

Let p be an odd prime, a an integer. The *Legendre symbol* $\left(\frac{a}{p}\right)$ is a number in $\{-1, 0, +1\}$, defined as follows:

$$\left(\frac{a}{p}\right) = \begin{cases} +1 & \text{if } a \text{ is a non-trivial quadratic residue modulo } p \\ 0 & \text{if } a \equiv 0 \pmod{p} \\ -1 & \text{if } a \text{ is } \textit{not} \text{ a quadratic residue modulo } p \end{cases}$$

By the Euler Criterion, we have

Theorem

Let p be an odd prime. Then

$$\left(\frac{a}{p}\right) \equiv a^{\left(\frac{p-1}{2}\right)} \ (mod\ p)$$

Note that this theorem holds even when $p \mid a$.

Properties of the Legendre symbol

The Legendre symbol satisfies the following *multiplicative property*:

Fact

Let p be an odd prime. Then

$$\left(\frac{a_1 a_2}{p}\right) = \left(\frac{a_1}{p}\right)\left(\frac{a_2}{p}\right)$$

Not surprisingly, if a_1 and a_2 are both non-trivial quadratic residues, then so is $a_1 a_2$. Hence, the fact holds when

$$\left(\frac{a_1}{p}\right) = \left(\frac{a_2}{p}\right) = 1.$$

Product of two non-residues

Suppose $a_1 \notin \mathrm{QR}_p$, $a_2 \notin \mathrm{QR}_p$. The above fact asserts that the product $a_1 a_2$ *is* a quadratic residue since

$$\left(\frac{a_1 a_2}{p}\right) = \left(\frac{a_1}{p}\right)\left(\frac{a_2}{p}\right) = (-1)(-1) = 1.$$

Here's why.

- Let g be a primitive root of p.
- Write $a_1 \equiv g^{k_1} \pmod{p}$ and $a_2 \equiv g^{k_2} \pmod{p}$.
- Both k_1 and k_2 are odd since $a_1, a_2 \notin \mathrm{QR}_p$.
- But then $k_1 + k_2$ is even.
- Hence, $g^{(k_1+k_2)/2}$ is a square root of $a_1 a_2 \equiv g^{k_1+k_2} \pmod{p}$, so $a_1 a_2$ is a quadratic residue.

The Jacobi symbol

The *Jacobi symbol* extends the Legendre symbol to the case where the "denominator" is an arbitrary odd positive number n.

Let n be an odd positive integer with prime factorization $\prod_{i=1}^{k} {p_i}^{e_i}$. We define the *Jacobi symbol* by

$$\left(\frac{a}{n}\right) = \prod_{i=1}^{k} \left(\frac{a}{p_i}\right)^{e_i} \tag{1}$$

The symbol on the left is the Jacobi symbol, and the symbol on the right is the Legendre symbol.

(By convention, this product is 1 when $k = 0$, so $\left(\frac{a}{1}\right) = 1$.)

The Jacobi symbol extends the Legendre symbol since the two definitions coincide when n is an odd prime.

Meaning of Jacobi symbol

What does the Jacobi symbol mean when n is not prime?

- If $\left(\frac{a}{n}\right) = +1$, a might or might not be a quadratic residue.
- If $\left(\frac{a}{n}\right) = 0$, then $\gcd(a, n) \neq 1$.
- If $\left(\frac{a}{n}\right) = -1$ then a is definitely not a quadratic residue.

Jacobi symbol $= +1$ for $n = pq$

Let $n = pq$ for p, q distinct odd primes. Since

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p}\right)\left(\frac{a}{q}\right) \tag{2}$$

there are two cases that result in $\left(\frac{a}{n}\right) = 1$:

1. $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = +1$, or
2. $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$.

Case of both Jacobi symbols $= +1$

If $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = +1$, then $a \in \mathrm{QR}_p \cap \mathrm{QR}_q = Q_n^{11}$.

It follows by the Chinese Remainder Theorem that $a \in \mathrm{QR}_n$.

This fact was implicitly used in the proof sketch that $|\sqrt{a}| = 4$.

Case of both Jacobi symbols $= -1$

If $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$, then $a \in \mathrm{QNR}_p \cap \mathrm{QNR}_q = Q_n^{00}$.

In this case, a is *not* a quadratic residue modulo n.

Such numbers a are sometimes called "pseudo-squares" since they have Jacobi symbol 1 but are not quadratic residues.

Computing the Jacobi symbol

The Jacobi symbol $\left(\frac{a}{n}\right)$ is easily computed from its definition (equation 1) and the Euler Criterion, given the factorization of n.

Similarly, $\gcd(u, v)$ is easily computed without resort to the Euclidean algorithm given the factorizations of u and v.

The remarkable fact about the Euclidean algorithm is that it lets us compute $\gcd(u, v)$ efficiently, without knowing the factors of u and v.

A similar algorithm allows us to compute the Jacobi symbol $\left(\frac{a}{n}\right)$ efficiently, without knowing the factorization of a or n.

Identities involving the Jacobi symbol

The algorithm is based on identities satisfied by the Jacobi symbol:

1. $\left(\frac{0}{n}\right) = \begin{cases} 1 & \text{if } n = 1 \\ 0 & \text{if } n \neq 1; \end{cases}$
2. $\left(\frac{2}{n}\right) = \begin{cases} 1 & \text{if } n \equiv \pm 1 \pmod 8 \\ -1 & \text{if } n \equiv \pm 3 \pmod 8; \end{cases}$
3. $\left(\frac{a_1}{n}\right) = \left(\frac{a_2}{n}\right)$ if $a_1 \equiv a_2 \pmod n$;
4. $\left(\frac{2a}{n}\right) = \left(\frac{2}{n}\right) \cdot \left(\frac{a}{n}\right)$;
5. $\left(\frac{a}{n}\right) = \begin{cases} \left(\frac{n}{a}\right) & \text{if } a, n \text{ odd and } \neg(a \equiv n \equiv 3 \pmod 4) \\ -\left(\frac{n}{a}\right) & \text{if } a, n \text{ odd and } a \equiv n \equiv 3 \pmod 4. \end{cases}$

A recursive algorithm for computing Jacobi symbol

```
/* Precondition: a, n >= 0; n is odd */
int jacobi(int a, int n) {
  if (a == 0)                         /* identity 1 */
    return (n==1) ? 1 : 0;
  if (a == 2)                         /* identity 2 */
    switch (n%8) {
    case 1: case 7: return 1;
    case 3: case 5: return -1;
    }
  if ( a >= n )                       /* identity 3 */
    return jacobi(a%n, n);
  if (a%2 == 0)                       /* identity 4 */
    return jacobi(2,n)*jacobi(a/2, n);
  /* a is odd */                      /* identity 5 */
  return (a%4 == 3 && n%4 == 3) ? -jacobi(n,a) : jacobi(n,a);
}
```

Useful Tests of Compositeness

Solovay-Strassen compositeness test

Recall that a test of compositeness for n is a set of predicates $\{\tau_a(n)\}_{a \in \mathbf{Z}_n^*}$ such that if $\tau(n)$ succeeds (is true), then n is composite.

The *Solovay-Strassen Test* is the set of predicates $\{\nu_a(n)\}_{a \in \mathbf{Z}_n^*}$, where

$$\nu_a(n) = \text{true iff } \left(\frac{a}{n}\right) \not\equiv a^{(n-1)/2} \pmod{n}.$$

If n is prime, the test always fails by the Euler Criterion. Equivalently, if some $\nu_a(n)$ succeeds for some a, then n must be composite.

Hence, the test is a valid test of compositeness.

Usefulness of Strassen-Solovay test

Let $b = a^{(n-1)/2}$. The Strassen-Solovay test succeeds if $\left(\frac{a}{n}\right) \not\equiv b \pmod{n}$. There are two ways they could fail to be equal:

1. $b^2 \equiv a^{n-1} \not\equiv 1 \pmod{n}$.
   In this case, $b \not\equiv \pm 1 \pmod{n}$. This is just the Fermat test $\zeta_a(n)$ from lecture 9.
2. $b^2 \equiv a^{n-1} \equiv 1 \pmod{n}$ but $b \not\equiv \left(\frac{a}{n}\right) \pmod{n}$.
   In this case, $b \in \sqrt{1} \pmod{n}$, but b might have the opposite sign from $\left(\frac{a}{n}\right)$, or it might not even be ± 1 since 1 has additional square roots when n is composite.

Strassen and Solovay show the probability that $\nu_a(n)$ succeeds for a randomly-chosen $a \in \mathbf{Z}_n^*$ is at least $1/2$ when n is composite.[6]

Hence, the Strassen-Solovay test is a useful test of compositeness.

[6] R. Solovay and V. Strassen, "A Fast Monte-Carlo Test for Primality", *SIAM J. Comput.* *6*:1 (1977), 84–85.

Miller-Rabin test – an overview

The Miller-Rabin Test is more complicated to describe than the Solovay-Strassen Test, but the probability of error (that is, the probability that it fails when n is composite) seems to be lower.

Hence, the same degree of confidence can be achieved using fewer iterations of the test. This makes it faster when incorporated into a primality-testing algorithm.

This test is closely related to the algorithm from Lecture 9 for factoring an RSA modulus given the encryption and decryption keys and to Shanks Algorithm given in this lecture for computing square roots modulo an odd prime.

Miller-Rabin test

The Miller-Rabin test $\mu_a(n)$ computes a sequence $b_0, b_1, \ldots, b_s$ in $\mathbf{Z}_n^*$. The test succeeds if $b_s \not\equiv 1 \pmod{n}$ or the last non-1 element exists and is $\not\equiv -1 \pmod{n}$.

The sequence is computed as follows:

1. Write $n - 1 = 2^s t$, where t is an odd positive integer.
2. Let $b_0 = a^t \bmod n$.
3. For $i = 1, 2, \ldots, s$, let $b_i = (b_{i-1})^2 \bmod n$.

An easy inductive proof shows that $b_i = a^{2^i t} \bmod n$ for all i, $0 \leq i \leq s$. In particular, $b_s \equiv a^{2^s t} = a^{n-1} \pmod{n}$.

Validity of the Miller-Rabin test

The Miller-Rabin test fails when either every $b_k \equiv 1 \pmod{n}$ or for some k, $b_{k-1} \equiv -1 \pmod{n}$ and $b_k \equiv 1 \pmod{n}$.

To show validity, we show that $\mu_a(n)$ fails for all $a \in \mathbf{Z}_n^*$ when n is prime.

By Euler's theorem, $b^s \equiv a^{n-1} \equiv 1 \pmod{n}$.

Since $\sqrt{1} = \{1, -1\}$ and b_{i-1} is a square root of b_i for all i, either all $b_k \equiv 1 \pmod{n}$ or the last non-1 element in the sequence $b_{k-1} \equiv -1 \pmod{p}$.

Hence, the test fails whenever n is prime, so $\mu_a(n)$ is a valid test of compositeness.

Usefulness of Miller-Rabin test

The Miller-Rabin test succeeds whenever $a^{n-1} \not\equiv 1 \pmod{n}$, so it succeeds whenever the Fermat test $\zeta_a(n)$ would succeed.

But even when $a^{n-1} \equiv 1 \pmod{n}$, the Miller-Rabin test succeeds if the last non-1 element in the sequence of b's is one of the two square roots of 1 that differ from ± 1.

It can be proved that $\mu_a(n)$ succeeds for at least 3/4 of the possible values of a. Empirically, the test almost always succeeds when n is composite, and one has to work to find a such that $\mu_a(n)$ fails.

Example of Miller-Rabin test

For example, take $n = 561 = 3 \cdot 11 \cdot 17$, the first Carmichael number. Recall that a *Carmichael number* is an odd composite number n that satisfies $a^{n-1} \equiv 1 \pmod{n}$ for all $a \in \mathbf{Z}_n^*$.
Let's go through the steps of computing $\mu_{37}(561)$.

We begin by finding t and s.
561 in binary is 1000110001 (a palindrome!).
Then $n - 1 = 560 = (1000110000)_2$, so $s = 4$ and $t = (100011)_2 = 35$.

Example (cont.)

We compute $b_0 = a^t = 37^{35} \bmod 561 = 265$ with the help of the computer.

We now compute the sequence of b's, also with the help of the computer. The results are shown in the table below:

$$\begin{aligned} b_0 &= 265 \\ b_1 &= 100 \\ b_2 &= 463 \\ b_3 &= 67 \\ b_4 &= 1 \end{aligned}$$

This sequence ends in 1, but the last non-1 element $b_3 \not\equiv -1 \pmod{561}$, so the test $\mu_{37}(561)$ succeeds. In fact, the test succeeds for every $a \in \mathbf{Z}^*_{561}$ except for $a = 1, 103, 256, 460, 511$. For each of those values, $b_0 = a^t \equiv 1 \pmod{561}$.

Optimizations

In practice, one computes only as many b's as are necessary to determine whether or not the test succeeds.

One can stop after finding b_i such that $b_i \equiv \pm 1 \pmod{n}$.

- If $b_i \equiv -1 \pmod{n}$ and $i < s$, the test fails.
- If $b_i \equiv 1 \pmod{n}$ and $i \geq 1$, the test succeeds.
  In this case, we know that $b_{i-1} \not\equiv \pm 1 \pmod{n}$, for otherwise the algorithm would have stopped after computing b_{i-1}.