CPSC 467b: Cryptography and Computer Security

Lecture 14

Michael J. Fischer

Department of Computer Science
Yale University

March 1, 2010

Quadratic Residues

Important facts about quadratic residues (review)

1. If p is odd prime, then $|\mathrm{QR}_p| = 2$, and for each $a \in \mathrm{QR}_p$, $|\sqrt{a}| = 2$.
2. If $n = pq$, $p \neq q$ odd primes, then $|\mathrm{QR}_n| = 4$, 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?

Important facts about quadratic residues (review)

1. If p is odd prime, then $|\mathrm{QR}_p| = 2$, and for each $a \in \mathrm{QR}_p$, $|\sqrt{a}| = 2$.
2. If $n = pq$, $p \neq q$ odd primes, then $|\mathrm{QR}_n| = 4$, 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.

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

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

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 implicit last lecture in the proof that $|\sqrt{a}| = 4$.

Details

Let $b \in \sqrt{a}$ modulo p and let $c \in \sqrt{a}$ modulo q. Then

$$a \equiv b^2 \pmod{p} \quad (3)$$
$$a \equiv c^2 \pmod{q} \quad (4)$$

By the Chinese Remainder Theorem, exists unique $d \in \mathbf{Z}_n$

$$d \equiv b \pmod{p} \quad (5)$$
$$d \equiv c \pmod{q} \quad (6)$$

Squaring (5) and (6) and combining with (3) and (4) gives

$$d^2 \equiv a \pmod{p} \quad (7)$$
$$d^2 \equiv a \pmod{q} \quad (8)$$

Hence, $d^2 \equiv a \pmod{n}$, so a is a quadratic residue modulo n.

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 11.
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.[1]

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

[1] 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 11 for factoring an RSA modulus given the encryption and decryption keys and to Shanks Algorithm given in Lecture 13 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}.

Digital Signatures

Overview of digital signatures

A *digital signature* is a string attached to a message used to guarantee the integrity and authenticity of the message.

It is a public-key analog to the private-keyed message authentication codes (MACs) discussed in lecture 7.

- Recall that Alice can protect a message m (encrypted or not) by attaching a MAC $\xi = C_k(m)$ to the message m.
- The pair (m, ξ) is an *authenticated message*.
- To produce a MAC requires possession of the secret key k.
- To verify integrity and authenticity, Bob, who also must know k, checks a received pair (m', ξ') by verifying that $\xi' = C_k(m')$. Assuming Alice and Bob are the only parties who share k, then Bob knows that m' came from Alice.

Digital signature scheme

A digital signature can be viewed as a 2-key MAC, just as a public key cryptosystem is a 2-key version of a classical cryptosystem.

Let $\mathcal{M}$ be a *message space* and $\mathcal{S}$ a *signature space*.

A *signature scheme* consists of a private *signing key* d, a public *verification key* e, a *signature function* $S_d : \mathcal{M} \to \mathcal{S}$, and a *verification predicate* $V_e \subseteq \mathcal{M} \times \mathcal{S}$.[2]

A *signed message* is a pair $(m, s) \in \mathcal{M} \times \mathcal{S}$. A signed message is *valid* if $V_e(m, s)$ holds, and we say that (m, s) is *signed with* e.

[2] As with RSA, we denote the private component of the key pair by the letter d and the public component by the letter e, although they no longer have same mnemonic significance.

Fundamental property of a signature scheme

Basic requirement:

The signing function always produces a valid signature, that is,

$$V_e(m, S_d(m)) \tag{9}$$

holds for all $m \in \mathcal{M}$.

Assuming e is Alice's public verification key, and only Alice knows the corresponding signing key d, then a signed message (m, s) that is valid under e identifies Alice with m (possibly erroneously, as we shall see).

RSA digital signature scheme

RSA can be used for digital signatures as follows:

- Alice generates an RSA modulus n and key pair (e, d), where e is public and d private as usual.
- Let $S_d(m) = D_d(m)$, and let $V_e(m, s)$ hold iff $m = E_e(s)$.
- Must verify that $V_e(m, S_d(m))$ hold for all messages m, i.e., must check that $m = E_e(D_d(m))$ holds.
- This is the reverse of the condition we required for RSA to be a valid cryptosystem, viz. $D_d(E_e(m))$ for all $m \in \mathbf{Z}_m$.
- RSA satisfies both conditions since

$$m \equiv D_d(E_e(m)) \equiv (m^e)^d \equiv (m^d)^e \equiv E_e(D_d(m)) \pmod{n}.$$

Commutative cryptosystems

A cryptosystem with this property that $D_d \circ E_e = E_e \circ D_d$ is said to be *commutative*, where "$\circ$" denotes functional composition.

Indeed, any commutative public key cryptosystem can be used for digital signatures in exactly this same way as we did for RSA.

Signatures from non-commutative cryptosystems

We digress slightly and ask what we could do in case E_e and D_d did not commute.

One could define $S_e(m) = E_e(m)$ and $V_e(m, s) \Leftrightarrow m = D_d(s)$. Now indeed every validly-signed message $(m, S_e(m))$ would verify since $D_d(E_e(m)) = m$ is the basic property of a cryptosystem.

To make use of this scheme, Alice would have to keep e private and make d public. Assuming Alice generated the key pair in the first place, there is nothing preventing her from doing this. However, the resulting system might not be secure.

Even if it is hard for Eve to find d from e, it might not be hard to find e from d.

Interchanging public and private keys

For RSA, it is just as hard to find e from d as it is to find d from e. That's because RSA is completely symmetric in e and d.
Not all cryptosystems enjoy this symmetry property.

For example, the ElGamal scheme discussed in Lecture 12 is based on the equation $b = g^y \pmod{p}$, where y is private and b public.

Finding y from b is the discrete log problem — believed to be hard.

Finding b from y, is straightforward, so the roles of public and private key cannot be interchanged while preserving security.[3]

[3] However, ElGamal found a different way to use the ideas of discrete logarithm to build a signature scheme, which we will discuss later.