

Today's Outline

Review

Miller-Rabin strong pseudoprime test
Euler's criterion for roots modulo primes
Euler's phi-function

More on Euler's phi-function

Euler's theorem (generalizing Fermat's Little Theorem)

Proof that the decryption step in RSA really decrypts!

Primitive roots in general

Composite moduli, Sun-Ze's theorem

Square roots modulo $p \cdot q$

Factoring $n = pq$ by a square root oracle

Review

**Miller-Rabin strong pseudoprime test
base b :**

factor $n - 1 = 2^s \cdot m$ with m odd

replace b by $b^m \bmod n$

if $b = \pm 1 \bmod n$ **stop:** n is 3/4 prime
else continue

set $r = 1$

while $r < s$

replace b by $b^2 \bmod n$

if $b = -1 \bmod n$ **stop:** n is 3/4 prime

elseif $b = +1 \bmod n$ **stop:** n is **composite**

else replace r by $r + 1$ and **continue**

if we fall out of the loop, n is **composite**.

If n passes this test it is a
strong pseudoprime base b .

Euler's criterion for roots:

Theorem: Let e be a positive integer and p a prime with $p \equiv 1 \pmod{e}$. An integer b not 0 modulo p is an e^{th} power modulo p if and only if

$$b^{\frac{p-1}{e}} \equiv 1 \pmod{p}$$

A **primitive root** modulo a prime p is an integer g relatively prime to p such that no positive exponent ℓ smaller than $p - 1$ will make

$$g^\ell \equiv 1 \pmod{p}$$

From Fermat's Little Theorem we know that $g^{p-1} \equiv 1 \pmod{p}$. The content of this condition on g is that no *smaller* positive integer will make this true.

Theorem: Primitive roots modulo primes exist.

Testing for primitive roots:

Proposition: An integer g is a primitive root modulo a prime p if, for every prime q dividing $p - 1$,

$$g^{\frac{p-1}{q}} \not\equiv 1 \pmod{p}$$

A medium good formula for $\varphi(n)$

Rather than taking n steps to compute $\varphi(n)$, a prime factorization of n gives a formula for $\varphi(n)$.

Theorem: With prime factorization of n

$$n = p_1^{e_1} \dots p_t^{e_t}$$

with the p_i distinct primes and the e_i positive integers, then

$$\varphi(n) = (p_1 - 1)p_1^{e_1 - 1} \dots (p_t - 1)p_t^{e_t - 1}$$

Since factorization takes at worst \sqrt{n} steps, this formula is much better than brute force, but is still **infeasible** for $n \sim 10^{100}$, since we cannot expect to factor n of that size.

Examples of $\varphi(n)$

In all cases we use trial division to obtain a prime factorization and then use the formula:

$$\varphi(18) = \varphi(2 \cdot 3^2) = (2 - 1) \cdot (3 - 1)3^{2-1} = 6$$

$$\varphi(21) = \varphi(3 \cdot 7) = (3 - 1) \cdot (7 - 1) = 12$$

$$\varphi(24) = \varphi(2^3 \cdot 3) = (2 - 1)2^{3-1} \cdot (3 - 1) = 8$$

$$\varphi(30) = \varphi(2 \cdot 3 \cdot 5) = (2 - 1) \cdot (3 - 1) \cdot (5 - 1) = 8$$

Bigger examples:

$$\varphi(100) = \varphi(2^2 \cdot 5^2)$$

$$= (2 - 1)2^{2-1} \cdot (5 - 1)5^{2-1} = 40$$

$$\varphi(1000) = \varphi(2^3 \cdot 5^3)$$

$$= (2 - 1)2^{3-1} \cdot (5 - 1)5^{3-1} = 400$$

Euler's Theorem

Theorem: For b relatively prime to n ,

$$b^{\varphi(n)} = 1 \pmod{n}$$

For n prime, $\varphi(n) = n - 1$, so this gives a form of Fermat's Little Theorem.

This assertion is what guarantees that the decryption step in RSA really does decrypt.

The *smallest* exponent $\lambda(n)$ such that for all b prime to n we have $b^{\lambda(n)} = 1 \pmod{n}$ is **Carmichael's lambda** function of n . In general $\lambda(n) < \varphi(n)$. Detailed discussion of $\lambda(n)$ plays a role in understanding why the Fermat pseudoprime test may fail completely for Carmichael numbers, and why the Miller-Rabin test does not have an analogous problem.

Proof: Let G be the equivalence classes of integers mod n with representative prime to n . This is well-defined, as $\gcd(b, n) = 1$ if and only if $\gcd(b \pm n, n) = 1$. G has $\varphi(n)$ elements.

$$b^{\varphi(n)} \cdot \left(\prod_{g \in G} g \right) = \prod_{g \in G} bg \pmod n$$

The map $g \rightarrow bg$ is a *bijection* of $G \rightarrow G$ since b is prime to n and thus has a multiplicative inverse, so there is an inverse map. Thus,

$$b^{\varphi(n)} \cdot \left(\prod_{g \in G} g \right) = \prod_{g \in G} g \pmod n$$

$\prod_{g \in G} g$ is a product of things relatively prime to n , so it has a multiplicative inverse mod n . Thus,

$$b^{\varphi(n)} = 1 \pmod n$$

as asserted. ///

Decryption in RSA

Euler's theorem assures that the decryption step in RSA really does what it claims, namely, decrypt.

Let $n = p \cdot q$ be the RSA modulus, where p and q are distinct primes. Then

$$\varphi(n) = \varphi(pq) = (p - 1)(q - 1)$$

Given (public) encryption exponent e , the (secret) decryption exponent d is a multiplicative inverse e^{-1} of e mod $(p - 1)(q - 1)$.

For a plaintext $1 < x < n$, the encryption is

$$y = x^e \% n$$

The decryption is (allegedly)

$$x = y^d \% n$$

Proof: (that decryption in RSA works) Let

$$d \cdot e = 1 + \ell(p - 1)(q - 1)$$

Then modulo n

$$\begin{aligned} (x^e)^d &= x^{ed} = x^{1+\ell(p-1)(q-1)} \\ &= x \cdot (x^{(p-1)(q-1)})^\ell = x \cdot 1^\ell \pmod{n} = x \pmod{n} \end{aligned}$$

from Euler's theorem. Since we have equality modulo n , the reductions are necessarily equal as well. ///

Keep in mind that it is not known at this point in history how to compute $\varphi(n) = (p - 1)(q - 1)$ without knowing the factors p and q , so an adversary cannot compute (in reasonable time) the decryption exponent d from n and e alone.

Primitive roots in general

For an arbitrary modulus n , an integer g is a **primitive root** modulo n if no positive exponent ℓ smaller than $\varphi(n)$ has the property that

$$g^\ell = 1 \pmod{n}$$

By Euler's theorem we know that the exponent $\varphi(n)$ does have this property.

Theorem: Primitive roots exist for moduli n of the form

$n = p^e$ is a power of a prime $p > 2$

$n = 2p^e$ is twice a power of a prime $p > 2$

$n = 2$ or $n = 4$

No other moduli have primitive roots.

(The proof is hard, and we won't give it right now.)

Composite moduli, Sun-Ze's theorem

(Also called *Chinese remainder theorem*)

Theorem: Let m and n be relatively prime integers. Given a and b , there is an integer x such that *both*

$$\begin{cases} x &= a \pmod{m} \\ x &= b \pmod{n} \end{cases}$$

This x is *unique* mod mn in the sense that any other solution x' to that system satisfies

$$x' = x \pmod{mn}$$

In particular, let r, s be integers such that

$$rm + sn = 1 \quad (= \gcd(m, n))$$

Then

$$\boxed{x = rm \cdot b + sn \cdot a \pmod{mn}}$$

is *the* solution mod mn .

Proof: If x and x' are two solutions to the system, then $x - x' = 0 \pmod{m}$ and $x - x' = 0 \pmod{n}$, so $m|(x - x')$ and $n|(x - x')$. Since m and n are relatively prime, $mn|(x - x')$ (as we showed a week or two ago in class). Thus, $x = x' \pmod{mn}$, which is the asserted uniqueness.

Next, claim that with r, s such that $rm + sn = 1$ the integer

$$x = rm \cdot b + sn \cdot a$$

satisfies $x = a \pmod{m}$ and $x = b \pmod{n}$. From $rm + sn = 1$ we get $rm = 1 \pmod{n}$. Thus \pmod{n}

$$x = rm \cdot b + sn \cdot a = 1 \cdot b + 0 \cdot a = b \pmod{n}$$

Symmetrically, $sn = 1 \pmod{m}$ and

$$x = rm \cdot b + sn \cdot a = 0 \cdot b + 1 \cdot a = a \pmod{m}$$

as desired. ///

For example, find x such that

$$\begin{cases} x &= 1 & \text{mod } 5 \\ x &= 2 & \text{mod } 7 \end{cases}$$

The extended Euclidean algorithm yields

$$3 \cdot 5 + (-2) \cdot 7 = 1$$

Thus, the formula gives

$$x = (3 \cdot 5) \cdot 2 + ((-2) \cdot 7) \cdot 1 = 30 - 14 = \boxed{16}$$

We can check that indeed

$$\begin{cases} 16 &= 1 & \text{mod } 5 \\ 16 &= 2 & \text{mod } 7 \end{cases}$$

For example, find x such that

$$\begin{cases} x &= 5 \pmod{101} \\ x &= 7 \pmod{157} \end{cases}$$

The extended Euclidean algorithm yields

$$\begin{aligned} 157 - 1 \cdot 101 &= 56 \\ 101 - 1 \cdot 56 &= 45 \\ 56 - 1 \cdot 45 &= 11 \\ 45 - 4 \cdot 11 &= 1 \\ 1 &= (1)45 + (-4)11 \\ &= (1)45 + (-4)(56 - 1 \cdot 45) \\ &= (-4)56 + (5)45 \\ &= (-4)56 + (5)(101 - 1 \cdot 56) \\ &= (5)101 + (-9)56 \\ &= (5)101 + (-9)(157 - 1 \cdot 101) \\ &= (-9)157 + (14)101 \end{aligned}$$

So

$$1 = (14)101 + (-9)157$$

Thus, the formula gives

$$\begin{aligned} x &= (14 \cdot 101) \cdot 7 + ((-9) \cdot 157) \cdot 5 \\ &= 9898 - 7065 = \boxed{2833} \pmod{101 \cdot 157} \end{aligned}$$

We can check that indeed

$$\begin{cases} 2833 = 2833 \% 101 = 5 \pmod{101} \\ 2833 = 2833 \% 157 = 7 \pmod{157} \end{cases}$$

Square roots mod pq

Proposition: The integer 1 has exactly two square roots modulo a prime $p > 2$, namely $\pm 1 \pmod p$.

Proof: Suppose that $x^2 = 1 \pmod p$. Then $p \mid (x^2 - 1)$, which is to say that $p \mid (x-1)(x+1)$. Since p is prime, if $p \mid ab$ then p divides a or b . Thus, either $x = 1 \pmod p$ or $x = -1 \pmod p$. ///

Proposition: Let b be an integer not divisible by the prime $p > 2$. The integer b^2 has exactly two square roots modulo p , namely $\pm b \pmod p$.

Proof: Suppose that $x^2 = b^2 \pmod p$. Then $p \mid (x^2 - b^2)$, which is to say that $p \mid (x - b)(x + b)$. Since p is prime, then p divides $x - b$ or $x + b$. Thus, either $x = b \pmod p$ or $x = -b \pmod p$. ///

Theorem: Let p and q be distinct primes both > 2 . There are exactly *four* square roots of 1 mod pq . Beyond the *obvious* square roots ± 1 there are two *unobvious* square roots distinct from these mod pq .

Proof: Since p and q are relatively prime, $x^2 = 1 \pmod{pq}$ is $pq \mid (x^2 - 1)$ which is equivalent to $p \mid (x^2 - 1)$ and $q \mid (x^2 - 1)$, which is equivalent to the system

$$\begin{cases} x^2 = 1 & \pmod{p} \\ x^2 = 1 & \pmod{q} \end{cases}$$

Thus, from the discussion of square roots of 1 mod p and mod q ,

$$\begin{cases} x = \pm 1 & \pmod{p} \\ x = \pm 1 & \pmod{q} \end{cases}$$

Let r, s be such that $rp + sq = 1$. By Sun-Ze

$$x_3 = (rp)(-1) + (sq)(+1)$$

$$x_4 = (rp)(+1) + (sq)(-1)$$

also satisfy $x^2 = 1 \pmod{pq}$. ///

For example, find the two non-obvious square roots of 1 modulo $3 \cdot 5$.

From $1 = 2 \cdot 3 + (-1) \cdot 5$ (from Euclid) the *obvious* square roots are also given by Sun-Ze, though this is boring:

$$x_1 = (2 \cdot 3)(+1) + ((-1) \cdot 5)(+1) = +1$$

$$x_2 = (2 \cdot 3)(-1) + ((-1) \cdot 5)(-1) = -1$$

The two unobvious square roots are given by *unmatching* choices of the plus-or-minuses:

$$x_3 = (2 \cdot 3)(+1) + ((-1) \cdot 5)(-1) = 11 \pmod{15}$$

$$x_4 = (2 \cdot 3)(-1) + ((-1) \cdot 5)(+1) = -11 \pmod{15}$$

It is not surprising that the two unobvious roots occur in a \pm pattern!

The fact that there will be more than two square roots of 1 modulo pq gives a little explanation of how the Miller-Rabin strong pseudoprime test works.

If the Miller-Rabin test stops because at some point $b^2 = 1 \pmod n$, by the nature of the test b was neither $\pm 1 \pmod n$ (or we would have already stopped), so this b is a square root of 1 other than $\pm 1 \pmod n$. Thus, n is definitely composite.

The part that is *not* clear is why the Miller-Rabin test should be so efficient in finding these extra square roots *if* the number n is composite.

It is *not* the case that one should simply try to *guess* the unobvious square roots directly, since there might be just 2 of them, so one might as well try to guess the *factors* of n .

Theorem: (Generally) let p and q be distinct primes > 2 . Let b be not divisible by either p or q . There are exactly *four* square roots of $b^2 \pmod{pq}$, *obvious* square roots $\pm b$ and two *unobvious* square roots.

Proof: Since p and q are relatively prime, $x^2 = b^2 \pmod{pq}$ is $pq \mid (x^2 - b^2)$ which is equivalent to $p \mid (x^2 - b^2)$ and $q \mid (x^2 - b^2)$, which is equivalent to the system

$$\begin{cases} x^2 &= b^2 & \pmod{p} \\ x^2 &= b^2 & \pmod{q} \end{cases}$$

Thus, from the discussion of square roots of $b^2 \pmod{p}$ and \pmod{q} ,

$$\begin{cases} x &= \pm b & \pmod{p} \\ x &= \pm b & \pmod{q} \end{cases}$$

Let r, s be such that $rp + sq = 1$. By Sun-Ze

$$x_3 = (rp)(-b) + (sq)(+b)$$

$$x_4 = (rp)(+b) + (sq)(-b)$$

also satisfy $x^2 = b^2 \pmod{pq}$. ///

Factoring by a square root oracle

Let $n = pq$ with distinct primes p and q . A **square root oracle** modulo n is an entity which, when given an integer a which has at least one square root modulo n , returns one of those square roots.

We know neither the mechanism by which the oracle finds the square root, nor which one of the possibly several square roots it returns.

Theorem: If we have a square root oracle mod n then we can factor n . Specifically, pick *random* b mod n and give the oracle $b^2 \% n$. Let c be the square root of b^2 mod n returned by the oracle. Compute (by Euclid)

$$\gcd(n, b - c)$$

There is a $1/2$ probability that this will be either p or q ! (As opposed to being 1 or n). *Repeat as necessary.*

Proof: We do *not* assume that the oracle returns a random one from among the 4 square roots. That's why it's necessary to be able to pick random b 's to feed to the oracle, or the process may fail!

By now we know that there are exactly 4 square roots of b^2 modulo pq , given by all 4 choices of sign in the system

$$\begin{cases} x &= \pm b & \text{mod } p \\ x &= \pm b & \text{mod } q \end{cases}$$

Whatever square root c of b^2 the oracle returns, we can invert our viewpoint and say that our randomly chosen b is one of the four solutions of

$$\begin{cases} b &= \pm c & \text{mod } p \\ b &= \pm c & \text{mod } q \end{cases}$$

We have a $1/2$ probability that the \pm 's will be *different*, so that $p|(b - c)$ but $q \nmid (b - c)$, or *vice versa*. ///

For example, suppose we have a square root oracle for 15. Randomly pick $b = 4$. When you give the oracle $b^2 = 16 = 1 \pmod{15}$ to take the square root, it returns 11. Compute (by Euclid)

$$\gcd(4 - 11, 15) = 1$$

Failure. Try again. Pick random $b = 7$, and give the oracle $b^2 = 49 = 4 \pmod{15}$. It returns 2. Compute

$$\gcd(7 - 2, 15) = 5$$

which is a proper factor of 15. *Success.*

Suppose we have a square root oracle for 15857. Randomly pick $b = 901$. When you give the oracle $b^2 = 3094 \pmod{15857}$ to take the square root, it returns 14956. Compute (by Euclid)

$$\gcd(901 - 14956, 15857) = 1$$

Failure. Try again. Pick random $b = 1001$, and give the oracle $b^2 = 3010 \pmod{15857}$. It returns 7281. Compute

$$\gcd(1001 - 7281, 15857) = 157$$

so 157 is a proper factor of 15857. *Success.*

Since factoring is assumed to be hard, finding square roots modulo composites is hard.