CPSC 467: Cryptography and Computer Security

Michael J. Fischer

Lecture 12
October 7, 2013

Computing in $\mathbf{Z}_n$

Multiplication modulo n

Theorem

$\mathbf{Z}_n^*$ is closed under multiplication modulo n.

This says, if a and b are both in $\mathbf{Z}_n^*$, then $(ab \bmod n)$ is also in $\mathbf{Z}_n^*$.

Proof.

If neither a nor b share a prime factor with n, then neither does their product ab. □

Example: Multiplication in $\mathbf{Z}_{26}^*$

Let $n = 26 = 2 \cdot 13$. Then

$$\begin{aligned}\mathbf{Z}_{26}^* &= \{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\} \\ \phi(26) &= |\mathbf{Z}_{26}^*| = 12.\end{aligned}$$

Multiplication examples:

$$5 \times 7 \bmod 26 = 35 \bmod 26 = 9.$$

$$3 \times 25 \bmod 26 = 75 \bmod 26 = 23.$$

$$9 \times 3 \bmod 26 = 27 \bmod 26 = 1.$$

We say that 3 is the *multiplicative inverse* of 9 in $\mathbf{Z}_{26}^*$.

Example: Inverses of the elements in $\mathbf{Z}_{26}^*$.

x	1	3	5	7	9	11	15	17	19	21	23	25
x^{-1}	1	9	21	15	3	19	7	23	11	5	17	25
$\equiv_n$	1	9	-5	-11	3	-7	7	-3	11	5	-9	-1

Bottom row gives equivalent integers in range $[-12, \ldots, 13]$.

Note that $(26 - x)^{-1} = -x^{-1}$.

Hence, last row reads same back to front except for change of sign.

Once the inverses for the first six numbers are known, the rest of the table is easily filled in.

Finding modular inverses

Let $u \in \mathbf{Z}_n^*$. We wish to find u^{-1} modulo n.

By definition, u^{-1} is the element $v \in \mathbf{Z}_n^*$ (if it exists) such that

$$uv \equiv 1 \pmod{n}.$$

This equation holds iff $n \mid (uv - 1)$ iff $uv - 1 = qn$ for some integer q (positive or negative).

We can rewrite this equation as

$$uv - nq = 1. \tag{1}$$

u and n are given and v and q are unknowns. If we succeed in finding a solution over the integers, then v is the desired inverse u^{-1}.

Diophantine equations

A *Diophantine equation* is a linear equation in two unknowns over the integers.

$$ax + by = c \tag{2}$$

Here, a, b, c are given integers. A solution consists of integer values for the unknowns x and y that make (2) true.

We see that equation 1 fits the general form for a Diophantine equation, where

$$\begin{aligned} a &= u \\ b &= -n \\ c &= 1 \end{aligned} \tag{3}$$

Existence of solution

Theorem

The Diophantine equation

$$ax + by = c$$

has a solution over $\mathbf{Z}$ *(the integers) iff* $\gcd(a, b) \mid c$.

It can be solved by a process akin to the Euclidean algorithm, which we call the *Extended Euclidean algorithm*.

Extended Euclidean algorithm

The algorithm generates a sequence of triples of numbers $T_i = (r_i, u_i, v_i)$, each satisfying the invariant

$$r_i = au_i + bv_i \geq 0. \tag{4}$$

$$T_1 = \begin{cases} (a, 1, 0) & \text{if } a \geq 0 \\ (-a, -1, 0) & \text{if } a < 0 \end{cases}$$

$$T_2 = \begin{cases} (b, 0, 1) & \text{if } b \geq 0 \\ (-b, 0, -1) & \text{if } b < 0 \end{cases}$$

Extended Euclidean algorithm (cont.)

$$r_i = au_i + bv_i \geq 0. \tag{4}$$

T_{i+2} is obtained by subtracting a multiple of T_{i+1} from from T_i so that $r_{i+2} < r_{i+1}$. This is similar to the way the Euclidean algorithm obtains (a mod b) from a and b.

In detail, let $q_{i+1} = \lfloor r_i / r_{i+1} \rfloor$. Then $T_{i+2} = T_i - q_{i+1} T_{i+1}$, so

$$\begin{aligned} r_{i+2} &= r_i - q_{i+1} r_{i+1} = r_i \bmod r_{i+1} \\ u_{i+2} &= u_i - q_{i+1} u_{i+1} \\ v_{i+2} &= v_i - q_{i+1} v_{i+1} \end{aligned}$$

The sequence of generated pairs (r_1, r_2), (r_2, r_3), (r_3, r_4), ... is exactly the same as the sequence generated by the Euclidean algorithm. We stop when $r_t = 0$. Then $r_{t-1} = \gcd(a, b)$.

Extended Euclidean algorithm (cont.)

$$r_i = au_i + bv_i \geq 0. \tag{4}$$

From (4) it follows that

$$\gcd(a, b) = au_{t-1} + bv_{t-1} \tag{5}$$

Finding all solutions

Returning to the original equation,

$$ax + by = c \tag{2}$$

if $c = \gcd(a, b)$, then $x = u_{t-1}$ and $y = v_{t-1}$ is a solution.

If $c = k \cdot \gcd(a, b)$ is a multiple of $\gcd(a, b)$, then $x = ku_{t-1}$ and $y = kv_{t-1}$ is a solution.

Otherwise, $\gcd(a, b)$ does not divide c, and one can show that (2) has no solution.

Example of extended Euclidean algorithm

Suppose one wants to solve the equation

$$31x - 45y = 3 \tag{6}$$

Here, $a = 31$ and $b = -45$. We begin with the triples

$$\begin{aligned} T_1 &= (31, 1, 0) \\ T_2 &= (45, 0, -1) \end{aligned}$$

Computing the triples

The computation is shown in the following table:

i	r_i	u_i	v_i	q_i
1	31	1	0	
2	45	0	-1	0
3	31	1	0	1
4	14	-1	-1	2
5	3	3	2	4
6	2	-13	-9	1
7	1	16	11	2
8	0	-45	-31	

Extracting the solution

From $T_7 = (1, 16, 11)$, we obtain the solution $x = 16$ and $y = 11$ to the equation

$$1 = 31x - 45y$$

We can check this by substituting for x and y:

$$31 \cdot 16 + (-45) \cdot 11 = 496 - 495 = 1.$$

The solution to

$$31x - 45y = 3 \tag{6}$$

is then $x = 3 \cdot 16 = 48$ and $y = 3 \cdot 11 = 33$.

Generating RSA Encryption and Decryption Exponents

Recall RSA exponent requirement

Recall that the RSA encryption and decryption exponents must be chosen so that

$$ed \equiv 1 \pmod{\phi(n)}, \tag{7}$$

that is, d is e^{-1} in $\mathbf{Z}^*_{\phi(n)}$.

How does Alice choose e and d to satisfy (7)?

- Choose a random integer $e \in \mathbf{Z}^*_{\phi(n)}$.
- Solve (7) for d.

We know now how to solve (7), but how does Alice sample at random from $\mathbf{Z}^*_{\phi(n)}$?

Sampling from $\mathbf{Z}_n^*$

If $\mathbf{Z}_{\phi(n)}^*$ is large enough, Alice can just choose random elements from $\mathbf{Z}_{\phi(n)}$ until she encounters one that also lies in $\mathbf{Z}_{\phi(n)}^*$.

A candidate element e lies in $\mathbf{Z}_{\phi(n)}^*$ iff $\gcd(e, \phi(n)) = 1$, which can be computed efficiently using the Euclidean algorithm.[1]

[1] $\phi(n)$ itself is easily computed for an RSA modulus $n = pq$ since $\phi(n) = (p-1)(q-1)$ and Alice knows p and q.

How large is large enough?

If $\phi(\phi(n))$ (the size of $\mathbf{Z}^*_{\phi(n)}$) is much smaller than $\phi(n)$ (the size of $\mathbf{Z}_{\phi(n)}$), Alice might have to search for a long time before finding a suitable candidate for e.

In general, $\mathbf{Z}^*_m$ can be considerably smaller than m.
Example:

$$m = |\mathbf{Z}_m| = 2 \cdot 3 \cdot 5 \cdot 7 = 210$$
$$\phi(m) = |\mathbf{Z}^*_m| = 1 \cdot 2 \cdot 4 \cdot 6 = 48.$$

In this case, the probability that a randomly-chosen element of $\mathbf{Z}_m$ falls in $\mathbf{Z}^*_m$ is only $48/210 = 8/35 = 0.228\ldots$.

A lower bound on $\phi(m)/m$

The following theorem provides a crude lower bound on how small $\mathbf{Z}_m^*$ can be relative to the size of $\mathbf{Z}_m$.

Theorem

For all $m \geq 2$,

$$\frac{|\mathbf{Z}_m^*|}{|\mathbf{Z}_m|} \geq \frac{1}{1 + \lfloor \log_2 m \rfloor}.$$

A lower bound on $\phi(m)/m$

Proof.

Write $m = \prod_{i=1}^{t} p_i^{e_i}$, where p_i is the i^{th} prime that divides m and $e_i \geq 1$. Then $\phi(m) = \prod_{i=1}^{t} (p_i - 1) p_i^{e_i - 1}$, so

$$\frac{|\mathbf{Z}_m^*|}{|\mathbf{Z}_m|} = \frac{\phi(m)}{m} = \frac{\prod_{i=1}^{t}(p_i - 1)p_i^{e_i-1}}{\prod_{i=1}^{t} p_i^{e_i}} = \prod_{i=1}^{t}\left(\frac{p_i - 1}{p_i}\right). \qquad (8)$$

A lower bound on $\phi(m)/m$

Proof (cont.)

To estimate the size of $\prod_{i=1}^{t}(p_i - 1)/p_i$, note that

$$\left(\frac{p_i - 1}{p_i}\right) \geq \left(\frac{i}{i+1}\right).$$

This follows since $(x-1)/x$ is monotonic increasing in x, and $p_i \geq i+1$. Then

$$\prod_{i=1}^{t}\left(\frac{p_i - 1}{p_i}\right) \geq \prod_{i=1}^{t}\left(\frac{i}{i+1}\right) = \frac{1}{2}\cdot\frac{2}{3}\cdot\frac{3}{4}\cdots\frac{t}{t+1} = \frac{1}{t+1}. \quad (9)$$

A lower bound on $\phi(m)/m$

Proof (cont.)

Clearly $t \le \lfloor \log_2 m \rfloor$ since $2^t \le \prod_{i=1}^{t} p_i \le m$ and t is an integer.

Combining this with equations (8) and (9) gives the desired result.

$$\frac{|\mathbf{Z}_m^*|}{|\mathbf{Z}_m|} \ge \frac{1}{t+1} \ge \frac{1}{1+\lfloor \log_2 m \rfloor}. \tag{10}$$

$\square$

Expected difficulty of choosing RSA exponent e

For n a 1024-bit integer, $\phi(n) < n < 2^{1024}$.

Hence, $\log_2(\phi(n)) < 1024$, so $\lfloor \log_2(\phi(n)) \rfloor \le 1023$.

By the theorem, the fraction of elements in $\mathbf{Z}_{\phi(n)}$ that also lie in $\mathbf{Z}^*_{\phi(n)}$ is at least

$$\frac{1}{1 + \lfloor \log_2 \phi(n) \rfloor} \ge \frac{1}{1024}.$$

Therefore, the expected number of random trials before Alice finds a number in $\mathbf{Z}^*_{\phi(n)}$ is provably at most 1024 and is likely much smaller.

Euler's Theorem

Repeated multiplication in $\mathbf{Z}_n^*$

If any element $x \in \mathbf{Z}_n^*$ is repeatedly multiplied by itself, the result is eventually 1. [2]

Example, for $x = 5 \in \mathbf{Z}_{26}^*$: 5, 25, 21, 1, 5, 25, 21, 1, . . .

Let x^k denote the result of multiplying x by itself k times.
The *order of x*, written $\text{ord}(x)$, is the smallest integer $k \geq 1$ for which $x^k = 1$.

Theorem

$\text{ord}(x) \mid \phi(n)$. *(Recall, $\phi(n)$ is the size of $\mathbf{Z}_n^*$).*

[2] The first repeated element must be x. If not, then some $y \neq x$ is the first to repeat. The element immediately preceding each occurrence of y is yx^{-1}. But then yx^{-1} is the first to repeat, a contradiction. Hence, $x = x^{k+1}$ for some $k \geq 1$, so $x^k = x^{k+1}x^{-1} = xx^{-1} = 1$.

Euler's and Fermat's theorem

Theorem (Euler's theorem)
$x^{\phi(n)} \equiv 1 \pmod{n}$ for all $x \in \mathbf{Z}_n^*$.

Proof.
Since $\mathrm{ord}(x) \mid \phi(n)$, we have

$$x^{\phi(n)} \equiv (x^{\mathrm{ord}(x)})^{\phi(n)/\mathrm{ord}(x)} \equiv 1^{\phi(n)/\mathrm{ord}(x)} \equiv 1 \pmod{n}.$$

□

As a special case, we have

Theorem (Fermat's theorem)
$x^{(p-1)} \equiv 1 \pmod{p}$ for all x, $1 \le x \le p-1$, where p is prime.

An important corollary

Corollary

Let $r \equiv s \pmod{\phi(n)}$. Then $a^r \equiv a^s \pmod{n}$ for all $a \in \mathbf{Z}_n^*$.

Proof.

If $r \equiv s \pmod{\phi(n)}$, then $r = s + u\phi(n)$ for some integer u. Then using Euler's theorem, we have

$$a^r = a^{s+u\phi(n)} = a^s \cdot (a^u)^{\phi(n)} \equiv a^s \cdot 1 \equiv a^s \pmod{n},$$

as desired. □

Application to RSA

Recall the RSA encryption and decryption functions

$$\begin{aligned} E_e(m) &= m^e \bmod n \\ D_d(c) &= c^d \bmod n \end{aligned}$$

where $n = pq$ is the product of two distinct large primes p and q.

This corollary gives a sufficient condition on e and d to ensure that the resulting cryptosystem works. That is, we require that

$$ed \equiv 1 \pmod{\phi(n)}.$$

Then $D_d(E_e(m)) \equiv m^{ed} \equiv m^1 \equiv m \pmod{n}$ for all messages $m \in \mathbf{Z}_n^*$.

Messages not in $\mathbf{Z}_n^*$

What about the case of messages $m \in \mathbf{Z}_n - \mathbf{Z}_n^*$?

There are several answers to this question.

1. Alice doesn't really want to send such messages if she can avoid it.
2. If Alice sends random messages, her probability of choosing a message not in $\mathbf{Z}_n^*$ is very small — only about $2/\sqrt{n}$.
3. RSA does in fact work for all $m \in \mathbf{Z}_n$, even though Euler's theorem fails for $m \notin \mathbf{Z}_n^*$.

Why Alice might want to avoid sending messages not in $\mathbf{Z}_n^*$

If $m \in \mathbf{Z}_n - \mathbf{Z}_n^*$, either $p \mid m$ or $q \mid m$ (but not both because $m < pq$).

If Alice ever sends such a message and Eve is astute enough to compute $\gcd(m, n)$ (which she can easily do), then Eve will succeed in breaking the cryptosystem.

Why?

Why a random message is likely to be in $\mathbf{Z}_n^*$

The number of messages in $\mathbf{Z}_n - \mathbf{Z}_n^*$ is only

$$n - \phi(n) = pq - (p-1)(q-1) = p + q - 1$$

out of a total of $n = pq$ messages altogether.

If p and q are both 512 bits long, then the probability of choosing a bad message is only about $2 \cdot 2^{512}/2^{1024} = 1/2^{511}$.

Such a low-probability event will likely never occur during the lifetime of the universe.

RSA works anyway

For $m \in \mathbf{Z}_n - \mathbf{Z}_n^*$, RSA works anyway, but for different reasons.

For example, if $m = 0$, it is clear that $(0^e)^d \equiv 0 \pmod{n}$, yet Euler's theorem fails since $0^{\phi(n)} \not\equiv 1 \pmod{n}$.

We omit the proof of this curiosity.

Generating RSA Modulus

Recall RSA modulus

Recall the RSA modulus, $n = pq$. The numbers p and q should be random distinct primes of about the same length.

The method for finding p and q is similar to the "guess-and-check" method used to find random numbers in $\mathbf{Z}_m^*$.

Namely, keep generating random numbers p of the right length until a prime is found. Then keep generating random numbers q of the right length until a prime different from p is found.

Generating random primes of a given length

To generate a k-bit prime:

- Generate $k - 1$ random bits.
- Put a "1" at the front.
- Regard the result as binary number, and test if it is prime.

We defer the question of how to test if the number is prime and look now at the expected number of trials before this procedure will terminate.

Expected number of trials to find a prime

The above procedure samples uniformly from the set $B_k = \mathbf{Z}_{2^k} - \mathbf{Z}_{2^{k-1}}$ of binary numbers of length exactly k.

Let p_k be the fraction of elements in B_k that are prime. Then the expected number of trials to find a prime is $1/p_k$.

While p_k is difficult to determine exactly, the celebrated *Prime Number Theorem* allows us to get a good estimate on that number.

Prime number function

Let $\pi(n)$ be the number of numbers $\leq n$ that are prime.

For example, $\pi(10) = 4$ since there are four primes ≤ 10, namely, 2, 3, 5, 7.

Prime number theorem

Theorem
$\pi(n) \approx n/(\ln n)$, where $\ln n$ is the natural logarithm $\log_e n$.

Notes:

- We ignore the critical issue of how good an approximation this is. The interested reader is referred to a good mathematical text on number theory.
- Here $e = 2.71828\ldots$ is the base of the natural logarithm, not to be confused with the RSA encryption exponent, which, by an unfortunate choice of notation, we also denote by e.

Likelihood of randomly finding a prime

The chance that a randomly picked number in $\mathbf{Z}_n$ is prime is

$$\frac{\pi(n-1)}{n} \approx \frac{n-1}{n \cdot \ln(n-1)} \approx \frac{1}{\ln n}.$$

Since $B_k = \mathbf{Z}_{2^k} - \mathbf{Z}_{2^{k-1}}$, we have

$$\begin{aligned} p_k &= \frac{\pi(2^k-1) - \pi(2^{k-1}-1)}{2^{k-1}} \\ &= \frac{2\pi(2^k-1)}{2^k} - \frac{\pi(2^{k-1}-1)}{2^{k-1}} \\ &\approx \frac{2}{\ln 2^k} - \frac{1}{\ln 2^{k-1}} \approx \frac{1}{\ln 2^k} = \frac{1}{k \ln 2}. \end{aligned}$$

Hence, the expected number of trials before success is $\approx k \ln 2$.
For $k = 512$, this works out to $512 \times 0.693 \ldots \approx 355$.

Primitive Roots

Using the ElGamal cryptosystem

To use the ElGamal cryptosystem, we must be able to generate random pairs (p, g), where p is a large prime, and g is a primitive root of p.

We now look at primitive roots and how to find them.

Primitive root

We say g is a *primitive root* of n if g generates all of $\mathbf{Z}_n^*$, that is, $\mathbf{Z}_n^* = \{g, g^2, g^3, \ldots, g^{\phi(n)}\}$.

By definition, this holds if and only if $\mathrm{ord}(g) = \phi(n)$.

Not every integer n has primitive roots.

By Gauss's theorem, the numbers having primitive roots are $1, 2, 4, p^k, 2p^k$, where p is an odd prime and $k \geq 1$.

In particular, every prime has primitive roots.

Number of primitive roots

The number of primitive roots of p is $\phi(\phi(p))$.

This is because if g is a primitive root of p and $x \in \mathbf{Z}^*_{\phi(p)}$, then g^x is also a primitive root of p. Why?

We need to argue that every element h in $\mathbf{Z}^*_p$ can be expressed as $h = (g^x)^y$ for some y.

- Since g is a primitive root, we know that $h \equiv g^\ell \pmod{p}$ for some ℓ.
- We wish to find y such that $g^{xy} \equiv g^\ell \pmod{p}$.
- By Euler's theorem, this is possible if the congruence equation $xy \equiv \ell \pmod{\phi(p)}$ has a solution y.
- We know that a solution exists iff $\gcd(x, \phi(p)) \mid \ell$.
- But this is the case since $x \in \mathbf{Z}^*_{\phi(p)}$, so $\gcd(x, \phi(p)) = 1$.

Primitive root example

Let $p = 19$, so $\phi(p) = 18$ and $\phi(\phi(p)) = \phi(2) \cdot \phi(9) = 6$.

Let $g = 2$. The subgroup S of $\mathbf{Z}_p$ generated by g is given by the table:

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
g^k	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1

Since $S = \mathbf{Z}_p^*$, we know that g is a primitive root.

Now let's look at $\mathbf{Z}_{\phi(p)}^* = \mathbf{Z}_{18}^* = \{1, 5, 7, 11, 13, 17\}$.

The complete set of primitive roots of p (in $\mathbf{Z}_p$) is then

$$\{2, 2^5, 2^7, 2^{11}, 2^{13}, 2^{17}\} = \{2, 13, 14, 15, 3, 10\}.$$

Lucas test

Theorem (Lucas test)

g is a primitive root of p if and only if

$$g^{(p-1)/q} \not\equiv 1 \pmod{p}$$

for all $1 < q < p-1$ *such that* $q \mid (p-1)$.

Clearly, if the test fails for some q, then

$$\mathrm{ord}(g) \le (p-1)/q \; < \; p-1 \; = \; \phi(p), \qquad \text{Why?}$$

so g is not a primitive root of p.

Conversely, if $\mathrm{ord}(g) < \phi(p)$, then the test will fail for $q = (p-1)/\mathrm{ord}(g)$.

Problems with the Lucas test

A drawback to the Lucas test is that one must try all the divisors of $p-1$, and there can be many.

Moreover, to find the divisors efficiently implies the ability to factor. Thus, it does not lead to an efficient algorithm for finding a primitive root of an arbitrary prime p.

However, there are some special cases which we can handle.

Special form primes

Let p and q be odd primes such that $p = 2q + 1$.
Then, $p - 1 = 2q$, so $p - 1$ is easily factored and the Lucas test easily employed.

There are lots of examples of such pairs, e.g., $q = 41$ and $p = 83$.

How many primitive roots does p have?
We just saw the number is

$$\phi(\phi(p)) = \phi(p-1) = \phi(2)\phi(q) = q - 1.$$

Hence, the density of primitive roots in $\mathbf{Z}_p^*$ is

$$(q-1)/(p-1) = (q-1)/2q \approx 1/2.$$

This makes it easy to find primitive roots of p probabilistically — choose a random element $a \in \mathbf{Z}_p^*$ and apply the Lucas test to it.

Density of special form primes

We defer the question of the density of primes q such that $2q+1$ is also prime but remark that we can relax the requirements a bit.

Let q be a prime. Generate a sequence of numbers $2q+1$, $3q+1$, $4q+1$, … until we find a prime $p = uq+1$.

By the prime number theorem, approximately one out of every $\ln(q)$ numbers around the size of q will be prime.

While that applies to randomly chosen numbers, not the numbers in this particular sequence, there is at least some hope that the density of primes will be similar.

If so, we can expect that u will be about $\ln(q)$, in which case it can easily be factored using exhaustive search. At that point, we can apply the Lucas test as before to find primitive roots.