Secret Sharing

Aaron Kleinman

January 7, 2014

1 Modular Arithmetic

1.1 Addition and Multiplication

Arithmetic modulo m can be thought of as restricting the integers to the set $\{0, 1, 2, \cdots, m-1\}$ and "wrapping around" every time we get larger than m. Math with time is like doing modular arithmetic with $m = 12$. Computing days of the week is like doing modular arithmetic with $m = 7$.

Definition 1. Two integers a, b are said to be *equivalent modulo* m if $a - b$ is a multiple of m. We write this as

$$a \equiv b \pmod{m}.$$

Example 1. Assign the days of the week numbers from 0 to 6 sequentially, starting with Sunday. So Sunday is 0, Monday is 1, Tuesday is 2, etc. Suppose it's Wednesday, and you'd like to know what day of the week it will be 30 days from now. Then you can compute

$$3 + 30 = 33 \equiv 5 \pmod{7},$$

so the answer is Friday.

Example 2. Multiplication is similar. Suppose you'd like to know what day it will be five 30-day periods from now.

$$3 + 5 \cdot 30 = 153 \equiv 6 \pmod{7},$$

so the answer is Saturday. Note that we could have made our life a little easier by using the fact that $30 \equiv 2 \pmod{7}$. Then the calculation becomes

$$3 + 5 \cdot 30 \equiv 3 + 5 \cdot 2 \equiv 6 \pmod{7}.$$

It's often useful to reduce as you go along.

Problem 1. You have a secret number n, $0 \leq n < m$, that you'd like to send to your friend. The two of you agree to the following protocol: you'll compute the *ciphertext* $c = n + 3$, reduce (mod m), and send him c. How can your friend decode the message? Is this a good encryption scheme?

1.2 Division

What about division? When working with real numbers, dividing by a can be thought of as multiplying by the number b defined by the equation $ab = 1$. In modular arithmetic, we can define division similarly.

Definition 2. The number $b := a^{-1}$ is defined by the equation

$$ab \equiv 1 \pmod{m}.$$

If b exists we say it is the *multiplicative inverse* of a.

From now on, we will take $m = 10$ as our motivating example.

Problem 2. Which numbers have multiplicative inverses for $m = 10$? Formulate a conjecture about which numbers have multiplicative inverses for general m, and prove your conjecture.

Problem 3. Write out the numbers $0, 1, \ldots, 9$. Multiply them all by 3 (mod 10). What do you notice about resulting 10 numbers? Does the same thing happen if you replace 3 by 4 or 5? What is the relation to the previous problem?

Problem 4. Suppose you want to send your friend a secret number n, $0 \leq n < 10$. You encrypt your message by computing the ciphertext $c \equiv 3n \pmod{10}$, and then transmit c to your friend. How can your friend decrypt this message and recover n? Is this a good encryption scheme?

2 Encryption

Problem 5. There exist n such that $a^n \equiv 1 \pmod{10}$ for all a with $\gcd(a, 10) = 1$. Find the smallest such positive n. How does this relate to Problem 2?

Problem 6. Motivated by this, you now transform your message n into the ciphertext $c \equiv n^3 \pmod{m}$. How can your friend decrypt this message and recover n?

2.1 Diffie-Hellman

The Diffie-Hellman encryption scheme allows two people to come up with a shared secret in public. It proceeds as follows:

1. You and your friend publicly choose a prime p and number g, $1 < g < p$.
2. You secretly choose a number m, and your friend secretly chooses a number n.
3. You send the number $g^m \pmod{p}$ to your friend, and your friend sends the number $g^n \pmod{p}$ to you.
4. Your shared secret is $s = g^{mn} \pmod{p}$.

Problem 7. Why can both you and your friend compute s? Why is it hard for an attacker to figure out s, even if he knows the numbers g, p, g^m, g^n? How can you use your shared secret s to improve upon the encryption scheme in Problem 4?

2.2 RSA

Definition 3. Euler's totient function $\phi(n)$ is equal to the number of integers m, $1 \leq m \leq n-1$, such that $\gcd(m, n) = 1$.

Problem 8. Compute $\phi(m)$ for some small values of m. What is $\phi(p)$ for p prime? How about $\phi(p^n)$? Come up with an expression for $\phi(mn)$ in terms of $\phi(m)$ and $\phi(n)$ when $\gcd(m, n) = 1$. Can you prove it?

Theorem 1 (Euler's theorem)**.** *If* $\gcd(a, m) = 1$*, then*

$$a^{\phi(m)} \equiv 1 \pmod{m}.$$

RSA encryption was invented in 1977 and is still used widely today. Here's how it works:

1. Choose two distinct prime numbers p, q. Compute the product $m = pq$, and compute $\phi(m) = (p-1)(q-1)$.
2. Choose an integer e with $\gcd(e, \phi(m)) = 1$ and $1 < e < \phi(m)$.
3. Compute the multiplicative inverse d of e modulo $\phi(m)$, i.e. find d such that

$$de \equiv 1 \pmod{\phi(m)}.$$

The public key is the pair (m, e). The private key is d.

Suppose your friend wants to send you a number n, $0 \leq n < m$. She computes

$$c \equiv n^e \pmod{m}.$$

She then sends you c. To decode it, you compute

$$c^d \equiv n^{ed} \equiv n \pmod{m}.$$

Problem 9. Why does this work? How could this be broken? Why is it important that p and q be chosen randomly?

3 Polynomial secret sharing

3.1 Introduction

The military has a nuclear launch code that it wants to give to certain employees. Because firing the missile mistakenly could have a catastrophic effect, though, the military wants to make sure that multiple employees agree before entering the launch code. What the military is looking for is a way to give a small piece of the code to each person, so that nobody actually can figure anything out about the code from their individual piece, but so that if a sufficiently large number of employees get together, they can compute the code.

Our motivating problem for the rest of the section is the following problem: Suppose we have n people, a parameter $k \leq n$, and secret number S that we want to share among the people in the following way: if any $k < n$ of the people get together, they can compute the secret S. However, no smaller group of people can learn anything at all about S. The only thing the participants are allowed to know is that S lies in some range, $0 \leq S < N$ for some large N.

Problem 10. The solution for $k = 1$ is easy! What is it? How about for $k = n$?

We'll return to the general case later.

3.2 Polynomial Interpolation

Recall that an n-dimensional polynomial is a function

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0.$$

Theorem 2. *An n-dimensional polynomial has at most n roots.*

Theorem 3. *Given n pairs of numbers $(x_1, y_1), (x_2, y_2), \ldots, (x_{n+1}, y_{n+1})$, there is exactly one polynomial $p(x)$ of degree $\leq n$ such that $p(x_i) = y_i$ for $i = 1, 2, \ldots, n$.*

Theorem 3 says that two points determine a line, three determine a quadratic, and so forth. Given $n + 1$ points, how can we recover the unique polynomial $p(x)$ that passes through them?

One way to do this is by solving a system of linear equations.

Example 3. We wish to find a quadratic polynomial $p(x)$ passing through the three points $(x_1, y_1) = (-2, 11)$, $(x_2, y_2) = (0, 1)$, $(x_3, y_3) = (1, -1)$. Writing $p(x) = a_2 x^2 + a_1 x + a_0$ and plugging in the points gives the system of linear equations

$$\begin{aligned} 4a_2 - 2a_1 + a_0 &= 11, \\ a_0 &= 1, \\ a_2 - 3a_1 + a_0 &= -1. \end{aligned}$$

Solving this system of equations gives $a_2 = 1$, $a_1 = -3$, $a_0 = 1$, so our polynomial is $p(x) = x^2 - 3x + 1$.

3.3 Lagrange Interpolation

There is another way to compute the $p(x)$ in Example 3.

Problem 11. Show that the quadratic $q(x) = x(x-1)$ satisfies $q(0) = q(1) = 0$, $q(-2) \neq 0$. Divide by a constant to get a polynomial $\Delta_1(x)$ such that $\Delta_1(x) = 1$ if $x = x_1$, $\Delta_1(x) = 0$ if $x = x_2$ or $x = x_3$. Find similar polynomials $\Delta_i(x)$ for $i = 2, 3$.

Problem 12. Using your polynomials from Problem 11, find $p(x)$.

Problem 13. More generally, suppose we wish to find a polynomial $p(x)$ passing through the points $(x_1, y_1), (x_2, y_2), \ldots, (x_{n+1}, y_{n+1})$. We construct the n Lagrange polynomials $\Delta_1, \Delta_2, \ldots, \Delta_n$ given by

$$\Delta_i(x) = \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)}.$$

Show that $\Delta_i(x_i) = 1$, and $\Delta_i(x_j) = 0$ for $j \neq i$. Then show that

$$p(x) = \sum_i y_i \Delta_i(x)$$

passes through the $n + 1$ points.

3.4 Polynomials Over Finite Fields

Because we can add and multiply modulo m, we can also define polynomials modulo m.

Example 4. Consider the polynomial $p(x) = x^2 - 2x + 3$. If we are working mod 7, then

$$p(4) \equiv 4^2 - 2 \cdot 4 + 3 \equiv 4 \pmod 7.$$

Amazingly, both Theorem 2 and Theorem 3 hold when we work $\pmod m$ for m prime! When we deal with polynomials modulo m, we say we're working "over F_m." The F stands for field.

From here on, we'll take $m = 5$ and work over F_5.

Problem 14. Find two linear polynomials that are not equal over the reals, but that are equal over F_5. Chose and plot two distinct linear polynomials modulo 5. At how many points do they intersect? Is this always true? Can you prove it?

Problem 15. How many distinct degree-n polynomials are there over F_m for m prime?

Now we return to the problem stated at the beginning of this section. Recall that we have a secret S, which is some number between 0 and a large prime m. We want to give information to each of n different people, so that any k of them can work together to compute S, but so that if fewer than k people collude, they can learn nothing at all about S. Shamir's Secret Sharing works as follows:

1. Choose a random degree $k-1$ polynomial $p(x) = a_{k-1}x^{k-1} + a_{k-2}x^{k-2} + \ldots + a_0$ such that $p(0) = S$.
2. Number the people $1, 2, ..., n$ and tell them their numbers. For each $1 \leq i \leq n$, compute $p(i) \pmod m$ and tell this to person i.

Now we're done. Any k individuals can collude to recover $p(x)$ and thus $p(0)$. But if $k-1$ individuals pool their information, they'll be left with k equations and $k-1$ unknowns, and thus can divine nothing.

Problem 16. How can you compute a Lagrange polynomial modulo m? Work a few small examples.

Problem 17. Why is it necessary to tell the people not only their computed value $p(i) \pmod m$, but also which numbered person they are?

4 Acknowledgements

Sections 1 and 2 are adapted from Matthias Beck's 2010 BMC presentation.