NOTES ON THE CHINESE REMAINDER THEOREM AND RSA

MATH 3320, LARRY ROLEN

1. Chinese Remainder Theorem

We have seen that $ax \equiv 1 \pmod{m}$ has a solution precisely when $(a, m) = 1$. More generally, the **linear equation** $ax \equiv b \pmod{m}$ has a solution if and only if $(a, m)|b$. To see this, note that if $ax + my = b$, then $(a, m)|b$ as all common divisors of a, m divide b, and conversely, if $(a, m)|b$, say $b = (a, m)t$, then by the Bezout Identity there are x, y for which $ax + my = (a, m)$ and so $a(xt) + m(yt) = b$ and hence $a(xt) \equiv b \pmod{m}$.

How does one characterize all solutions modulo m? If x_0 is a particular solution of $ax \equiv b \pmod{m}$, then it is not too difficult to check that this is unique modulo $m/(a, m)$. That is,

$$ax \equiv b \pmod{m} \iff x \equiv x_0 \pmod{\frac{m}{(a,m)}}.$$

What about a systems of linear equations, like

$$\begin{aligned} a_1 x &\equiv b_1 \pmod{m_1}, \\ &\vdots \\ a_n x &\equiv b_n \pmod{m_n}? \end{aligned}$$

By the above, each equation $ax_i \equiv b_i \pmod{m_i}$ either doesn't have a solution, or it can be reduced to an equation of the shape $x \equiv x_0 \pmod{m_i'}$. Thus, we may assume that all the a_i are 1. The next step we need to be able to solve such equations is to make the moduli **pairwise coprime** (that is $(m_i, m_j) = 1$ for $i \neq j$). We will see this by example, but in general, if two m_i are not coprime, then either they are inconsistent or can be rewritten in terms of coprime moduli.

Example. The system

$$\begin{aligned} x &\equiv 2 \pmod{5} \\ x &\equiv 3 \pmod{15} \end{aligned}$$

has no solution as $x \equiv 3 \pmod{15} \implies x \equiv 3 \pmod{5}$. If instead the equations were

$$\begin{aligned} x &\equiv 3 \pmod{5} \\ x &\equiv 3 \pmod{15}, \end{aligned}$$

then these would be the same thing as the single equation $x \equiv 3 \pmod{15}$, as that implies that $x \equiv 3 \pmod{5}$ automatically, so the first equation is redundant.

Thus, to solve **any** system of linear congruence equations in a single variable x, it suffices to use the following result.

Theorem 1.1 (Chinese Remainder Theorem)**.** *Suppose that $m_1, \dots, m_n$ are pairwise coprime, and set $M := \prod_{i=1}^{n} m_i$. Then for any integers $a_1, \dots, a_n$, the system*

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ &\vdots \\ x &\equiv a_n \pmod{m_n} \end{aligned}$$

has a **unique** *solution modulo M.*

The proof will illustrate how to compute the solution.

Proof. Let

$$M_i := M/m_i = m_1 \dots \widehat{m_i} \dots m_n.$$

By the coprimality condition, we have $(M_i, m_i) = 1$ for all i (the other m_j have no prime factors in common with m_i). Thus, there exist inverses N_i of M_i modulo m_i.

The idea is now based on the following observation: If $x_i = M_i N_i$, then

$$x_i \equiv \begin{cases} 1 \pmod{m_i}, \\ 0 \pmod{m_j} \end{cases} \quad i \neq j. \tag{1.1}$$

Thinking of this as a sort of "standard basis vector," we then think like in linear algebra of writing the "vector" $(a_1, \dots a_n)$ as a linear combination of the x_i, in linear algebra, we would say $(a_1, \dots, a_n) = a_1 e_1 + \dots + a_n e_n$.

In this situation, we form

$$x = \sum_{i=1}^{n} a_i M_i N_i.$$

By (1.1), we have that $x \equiv \pmod{a}_i$ for all i. This shows existence of a solution.

We must also uniqueness modulo M. Suppose that x_1, x_2 are simultaneous solutions of all the equations. Then in particular, $x_1 \equiv x_2 \pmod{m_i}$ for $i = 1, \dots, n$, and so $m_1, \dots, m_n | (x_1 - x_2)$. But the m_i are coprime, and so $M = m_1 \dots m_n | (x_1 - x_2)$, and hence $x_1 \equiv x_2 \pmod{M}$.

□

Remark. A useful way of thinking about this theorem is that

$$\mathbb{Z}_{m_1} \times \dots \times \mathbb{Z}_{m_n} \cong \mathbb{Z}_M,$$

where "$\cong$" means "essentially the same as." That is, specifying a number modulo each m_i is the same thing as specifying it modulo M.

Remark. The CRT is a very useful way for storing large numbers as a collection of smaller numbers on a computer, and is widely used in applications. For instance, if you want to work with integers which are at most 10^{20}, you can instead do arithmetic with integers modulo $M > 10^{20}$, which in turn can be specified by a bunch of congruence classes modulo smaller primes. This is very important in applications of the Fast Fourier Transform, and other essentials in everyday computer algorithms.

Remark. The relation

$$\mathbb{Z}_{m_1} \times \ldots \times \mathbb{Z}_{m_n} \cong \mathbb{Z}_M,$$

"descends" to a relation

$$\mathbb{Z}^*_{m_1} \times \ldots \times \mathbb{Z}^*_{m_n} \cong \mathbb{Z}^*_M;$$

this can be proven by a careful check of what's going on in the proof. But the size of a Cartesian products of sets is the product of the sizes, and the sizes of each factor are the Euler Totient numbers. This is how one proves that $\varphi(n)$ is multiplicative.

Example. We solve the system

$$\begin{aligned} x &\equiv 0 \pmod 3 \\ x &\equiv 3 \pmod 4 \\ x &\equiv 4 \pmod 5 . \end{aligned}$$

Following the proof, we take $M = 3 \cdot 4 \cdot 5 = 60$, $M_1 = 20$, $M_2 = 15$, $M_3 = 12$. We have to find three inverses. First, we solve

$$20x \equiv 1 \pmod 3 \iff 2x \equiv 1 \pmod 3 \iff x \equiv 2 \pmod 3 .$$

We could use the Euclidean algorithm to compute the inverse, but since the modulus is so small, its just as easy to guess in this case (or to note that $2 \equiv -1 \pmod 3$ and the inverse of -1 is always -1). Similarly, we solve

$$15x \equiv 1 \pmod 4 \iff -x \equiv 1 \pmod 4 \iff x \equiv -1 \pmod 4 .$$

Finally, we solve

$$12x \equiv 1 \pmod 5 \iff 2x \equiv 1 \pmod 5 \iff x \equiv 3 \pmod 5 .$$

Thus, the solution is

$$x \equiv 0 \cdot 20 \cdot 2 - 3 \cdot 15 + 4 \cdot 3 \cdot 12 \equiv 99 \equiv 39 \pmod{60} .$$

Indeed, we have

$$3|39, \quad 4|(39-3), \quad 5|(39-4).$$

2. RSA

We now describe the Rivest-Shamir-Adleman (RSA) cryptosystem. This is a type of **public key cryptography**. This is in contrast to our earlier symmetric key systems. A website, or company, can publicly give a key, which **anyone** can use to encrypt messages. However, it is difficult to find the decryption key from the encryption key. This is very useful for instance if many people wish to send their credit card information to Amazon. RSA is a very popular, and still powerful method.

In RSA, the key is a pair (e, n), where e is an **exponent**, and n is a modulus. Further, n is chosen to be a **semiprime**, that is, a product of two primes

$$n = pq.$$

Finally, e is chosen so that

$$(e, \varphi(n)) = 1.$$

Encryption works as follows. Suppose we want to encode an English message. Turn letters into numbers as $A = 00$, $B = 01$, ... $Z = 25$. Then put the sequence of numbers into a number of blocks of the same size. On each block of plaintext P, we encrypt by sending

$$P \mapsto E(P)$$

where

$$E(P) = C \equiv P^e \pmod{n}, \quad 0 \leq C < n.$$

That is, we simply exponentiate. As multiplication is fast, this is also (relatively) fast).

To decrypt, one needs more than just the exponent e and the modulus n. One needs the inverse of $e \pmod{\varphi(n)}$. This exists since e was chosen to be coprime to $\varphi(n)$. Assuming for a moment we know this inverse, call it d, then we claim that we can decrypt the ciphertext C by exponentiating:

$$D(C) \equiv C^d \pmod{n}.$$

To see why this works, we want to show that

$$D(C) \equiv (P^e)^d \equiv P \pmod{n}.$$

But e, d are inverses modulo $\varphi(n)$. Thus, $ed = k\varphi(n) + 1$ for some integer k, and so

$$D(C) \equiv P^{ed} \equiv P^{k\varphi(n)+1} \pmod{n}.$$

Now if $(P, n) = 1$, then by Euler's Theorem, we immediately find that $D(C) \equiv P \pmod{n}$, and we're done. In rare cases, one may have $(P, n) > 1$. This isn't very likely as n is a product of two, usually **huge** primes. We could presumably avoid this by taking really big p and q. But the decryption still works, and for an interesting reason.

To do so, we need the Chinese Remainder Theorem. Since $n = pq$, we look mod p and mod q. If $P \not\equiv 0 \pmod{p}$,

$$D(c) \equiv p^{\varphi(n)k+1} \equiv p^{(p-1)(q-1)k+1} \equiv P \pmod{p},$$

where we used Fermat's Little Theorem. If $P \equiv 0 \pmod{p}$, then

$$C \equiv P^e \equiv 0 \pmod{p},$$

so $D(C) \equiv P \pmod{n}$ in this case too. The same argument works modulo q. Thus, by the Chinese Remainder Theorem,

$$D(C) \equiv P \pmod{pq} \implies D(C) \equiv P \pmod{n}.$$

Thus, the **public key** is the pair (e, n), and the **private key** is the pair (d, n).

Example. Suppose that $p = 43$, $q = 59$. Then $n = 2537$. Pick $e = 13$, which is coprime to $\varphi(n) = 42 \cdot 58$. Suppose the plaintext is PUBLIC KEY CRYPTOGRAPHY. We turn this into numbers and break into blocks of 4, putting a final dummy letter $X = 23$ at the end to make the number of digits a multiple of 4:

1520 0111 0802 1004
2402 1724 1519 1406
1700 1507 2423.

Then encryption works as

$$C \equiv P^{13} \pmod{2537}.$$

For example, on the first block, we have

$$C \equiv (1520)^{13} \equiv 95 \pmod{2537}.$$

Note that specifying a block of 2 letters gives a number at most 2525, which is less than 2537, and so specifying mod 2537 uniquely determines the letters; this is the largest we can make our blocks with this choice of primes. Doing this operation on all blocks gives the ciphertex

0095 1648 1410 1299

0811 2333 2132 0370

1185 1957 1084.

The **private** information that is known only to select parties (for example Amazon) is equivalent to knowing the pair of primes p, q. If you know this, then you know $\varphi(n) = (p-1)(q-1)$. Computing an inverse of e modulo this number is fast, as it is an application of the Euclidean algorithm. The trouble is finding this value $\varphi(n)$ is **equivalent** to knowing the factorization of n into primes, and that problem is extremely hard, unless one has a quantum computer.

In reality, the primes used are 1024, 2048, etc. bits long. These are numbers with about $300-600$ digits. How can you find a good pair of primes? This is also a difficult problem. Finding a prime with certainty is very slow. However, there are very significant speedups for computing primes **probabilistically**. The most commonly used is the **Miller-Rabin test**. This is based on two key ideas. Firstly, if n is prime, then by Fermat's Little Theorem, we have

$$a^n \equiv a \pmod{n}$$

for all integers a (it still works if $n|a$, why?). We also make an observation on the number of roots of the polynomial equation $x^2 - 1 \pmod{n}$. If n is prime, then one has the roots $x \equiv \pm 1 \pmod{n}$, and these are the only ones, which can be shown by doing long division on polynomials (the key fact here is again that $\mathbb{Z}_p$ is a field: you can invert all non-zero elements). If n isn't prime, then by the CRT, you can factor n into primes and get more than two solutions. For example, if $n = 15 = 3 \cdot 5$, then there are two solutions to $x^2 - 1 \equiv 0 \pmod{3}$ and two solutions to $x^2 - 1 \equiv 0 \pmod{5}$, which build up to give 4 solutions of $x^2 - 1 \equiv 0 \pmod{15}$. Thus, n is prime iff $x^2 - 1$ has exactly two roots modulo n.

However, there are numbers that are not prime which still pass both of these tests. We can get a very high probability of finding a "real" prime by picking random tests and iterating. Specifically, suppose we want to test if n is prime. Pick a random $a \in \{1, \ldots, n-1\}$. If $a^{n-1} \equiv 1 \pmod{n}$, then we have passed the Fermat's Little Theorem test at a. As n is a candidate prime, it has to be odd (obviously we aren't considering the prime 2), and so we can consider $a^{(n-1)/2}$. This must be congruent to ± 1 if n is a prime, since it is a square root of 1 and those are the only two square roots modulo a prime. We can keep taking square roots until we get a number that's not 1. If its not $-1 \pmod{n}$, then n had to be composite.

It turns out that the probability that a composite n passes this test is less than 25%. However, one can perform this test many times in a row, and the probability of a false prime surviving decreases exponentially.

Although RSA is very secure in general (unless you have a good quantum computer), it can be broken in several ways. One example is the **common modulus attack**. For instance, suppose that Bob1 uses key (n, e_1) and Bob2 uses key (n, e_2). Then if $(e_a, e_b) = 1$, and Alice sends P to both of them, she sends $C_1 \equiv P^{e_1} \pmod{n}$ to Bob1 and $C_2 \equiv P^{e_2} \pmod{n}$ to Bob2. If Eve can

read C_1 and C_2, then Eve can decrypt the message by using the Euclidean Algorithm to find x, y such that

$$e_1 r + e_1 s = 1.$$

Then Eve can compute

$$C_1^r C_2^s \equiv (P^{e_1})^r (P^{e_2})^s \equiv P^{e_1 r + e_2 s} \equiv P \pmod{n}$$

and recover the original message. In general, it is very bad to pick common n with others. However, since so many RSA keys are being generated all the time, this can and does happen. Next, we will see another commonly used modern cryptoscheme, based on *elliptic curves*. These have the advantage than many people can use the same elliptic curve, so it doesn't suffer from this sort of "common modulus" attack, and they also have much smaller key sizes. The small key size is handy for things like passports and credit card chips.