

MCS 320 Project One : The Chinese Remainder Theorem in Cryptography due Friday 13 February 2003, at 2PM.

The goal of this project is to use Maple to illustrate the application of modular arithmetic (using the Chinese Remainder Theorem) to a cryptography problem on how to protect a secret with multiple passwords. This document is made from a Maple worksheet. A good way to start this project is to download the worksheet from the course web site.

Doing experiments in Maple we will learn about the Chinese Remainder Theorem and see how to define a cryptosystem, either to protect a secret number (like your social security number) or to gain access via multiple personalized keys (like year of birth, last four digits of a social security number, zip code). We will study how secure the system is, estimating the computing time of an attack.

0. Sharing a Secret

Many systems are protected by multiple passwords. Examples from a wide variety of circumstances are easy enough to give.

For example, when calling customer service of a bank, before giving access to your account, the operator could ask for the last four digits of your social security number, your zip code, and your date of birth. To approve a motion proposed at a virtual meeting of the board of directors of a multinational company, sufficiently many members must phone in their key to the central office. Before launching a missile from a battle ship, sufficiently many senior officers on board must provide their key to the launching system before the launch goes through.

In the cryptography world, this is known as “Secret Sharing”, as several parties (who all mutually distrust each other) are sharing a secret access code. In this project we will design a cryptosystem, working in two ways: (1) Given a secret, we generate keys to compute the secret. (2) Given personalized keys, we generate a secret. In both ways, the secret can only be computed if sufficiently many keys are known. The computation happens via the Chinese Remainder Theorem, explained next.

1. The Chinese Remainder Theorem

We introduce the theorem with an example.

```
[> n := 5:           # number of parties involved
[> t := 3:           # threshold to the secret
```

We will set up a system that will grant access to the secret when t or more keys are known.

```
[> s := 500000:     # the secret unknown to all
[> m := [97,98,99,101,103]: # n moduli sorted in ascending order
```

There are three conditions on the moduli which need to be satisfied. We verify these three conditions.

1. The moduli must be relative prime: $\gcd(m[i], m[j]) = 1$, for all $i \neq j$.

```
[> mul(mul(gcd(m[i],m[j]),i=1..n),j=1..n) = mul(m[k],k=1..n);
```

The condition is satisfied when the multiplication of all possible gcd's equals the product of all moduli.

2. The secret must lie between the product of the last $t-1$ moduli and the first t moduli.

```
[> first_t := mul(m[i],i=1..t);
[> last_tm1 := mul(m[i],i=n-t+2..n);
[> last_tm1 < s and s < first_t;
```

3. The difference between the product of the first t moduli with the product of the last $t-1$ moduli should be at least three times the latter product.

```
[> difference := first_t - last_tm1;
[> difference >= 3*last_tm1;
```

When these three conditions are satisfied, we can share the secret taking moduli:

```
[> a := [seq(s mod m[k],k=1..n)];
```

The k -th number in a , $a[k]$ is only known to the k -th party. The keys to the secret are in the vector a , so we refer to the numbers in a as the keys. The Chinese Remainder Theorem says that we need at least t keys to gain access to the secret.

Theorem With definitions and assumptions from above, the secret

1. can be computed easily from any t keys in a ; and
2. cannot be computed from less than t keys in a .

The Chinese Remainder Theorem belongs to the Maple library as the command **chrem**. For our example, we can see that any three keys give access to the secret:

```
[> chrem([a[1],a[3],a[4]],[m[1],m[3],m[4]]) = s;
```

But knowing only two keys does not reveal the secret:

```
[> chrem([a[3],a[4]],[m[3],m[4]]);
```

2. Taking Shares of a Secret

With the Chinese Remainder Theorem we can generate keys to grant access to a secret. It takes at least t ($t =$ threshold) keys to get to the secret. In this case, the secret is not “shared” in the sense that several parties know the secret. Instead, from the secret, several numbers are generated, so we take shares of the secret number. With sufficiently many shares of the secret, we can recover the secret.

Assignment One. The first assignment is to generate keys to take shares of a number which is nine digits long, such as your social security number. Given your social security number, the task is to generate three keys, so that it takes at least two keys to compute the social security number. The secret cannot be computed by fewer than two keys. In the last part of this assignment, we examine the security of this system.

The print out of your worksheet with the solution must contain the following:

1. The secret number (i.e.: your social security number) and three moduli. Show that the moduli satisfy all conditions for the Chinese Remainder Theorem.
2. The keys generated from the secret modulo the m 's. Show that knowing two or more keys reveals the secret, while knowing just one key does not give the secret. Be exhaustive in this demonstration: check all possible combinations.
3. Assuming the moduli are public knowledge, an attack on the keys would be to enumerate all possible values and then call **chrem**. From timing one execution of **chrem**, calculate how long it could take to run such an enumeration (1) to find just one key (when the other key is known), and (2) to find the two keys needed for the secret.

3. Generating an Access Code

With the Chinese Remainder Theorem we can generate a secret based on given keys. The secret serves as an access code. Access is granted only if sufficiently many keys are given. Compared to the previous section, the keys here are given numbers. Therefore, the keys have a meaning to their owner.

Assignment Two. In this second assignment, we take three numbers to act as keys to an access code. Given are your year of birth, the last four digits of your social security number, and the zip code of your address. (If you live on the street, or spend more time at UIC than at home, you can take UIC's zip code.)

The print out of your worksheet with the solution must contain the following:

1. The three moduli and the secret number. As before, the moduli must satisfy all three conditions, and the keys must be the three given numbers (years of birth, last four digits of social security number, and the zip code).
2. Show that at least two keys need to be known to get to the secret number. Just as before, be exhaustive and show all possible combinations of the keys.

3. Examine the security of the system as before. From one run of `chrem`, estimate how much time it would take to enumerate all possible choices of keys. Distinguish between finding just one key (when the other key is known) and the case of finding both keys to get to the secret.

4. The deadline is Friday 13 February 2003, at 2PM.

The solutions to the project will be collected at the beginning of our class meeting on Friday 13 February at 2PM. If you cannot come to class that day, then you must arrange to hand in your solution before the deadline. Otherwise, your solution will be discounted with 10 points if it is turned in on the same day before 5PM, and will no longer be accepted afterwards.

The solution consists of the print out of one single worksheet, organized with sections according to the two assignments. Also write proper documentation for the calculations.

To avoid any misunderstanding, this project must be solved *individually*. Under no circumstances is it allowed to copy or to collaborate. Regardless of who copied from whom, all caught in the act of plagiarism will be penalized.

If you have questions, comments, or difficulties, feel free to come to my office for help.