Alexander Perlis
c/o Department of Mathematics
The University of Arizona
Tucson, AZ 85721
Telephone: (520) 621–2137
Fax: (520) 621–8322
Email: aprl@math.arizona.edu
SWRIMS Webpage: http://www.math.arizona.edu/~rims

SUBSTITUTION CIPHERS MODULAR ARITHMETIC

Prerequisite material for workshop on Public-key Cryptography

Note: I rapidly tossed these notes together. I appreciate any corrections or general comments, such as an indication of which parts are difficult to understand. [This printout is 17 Nov 98 revision.]

WHAT ARE THESE NOTES?

The variety of topics touched upon at our INTRODUCTION TO CRYPTOLOGY workshops is not always the same; however, two "core" topics are always covered, and are considered prerequisite material for the more advanced PUBLIC-KEY CRYPTOGRAPHY workshops.

The two core topics are "substitution ciphers" and "modular arithmetic". Both are covered in these notes. The latter might be difficult to swallow on a first reading; I urge you to persist, as this material is of fundamental importance (not just in cryptology but also in many branches of mathematics and most areas of computer science.)

WHY SECRET CODES? WHAT IS CRYPTOLOGY?

Let us say that you wish to send messages in confidence to a friend in another location. You might send letters by post, or send telegrams, or simply send email. If your messages might contain sensitive information of use to an adversary, then you should take the most paranoid stance and assume that *everything* you send will be examined by your adversary. For example, say you wish to send your mother flowers on her birthday. Such orders may be placed electronically on the Internet, but to do so, you will have to provide your credit card number to the flower company. You might be concerned that a hacker could temporarily examine the message before passing it along, allowing them to learn your credit card number, but giving you no clue that it has been intercepted.† Traditional examples in which one wants to send secret messages include communication between friendly governments and internal military communication during wartime.

---

† By the way: consumer fear of giving out credit card numbers has been cited as a reason why online commerce isn't too popular; yet it is interesting to note that consumers have no qualms handing out their credit cards for ten minutes to a stranger, at a restaurant say, when it comes time to paying a bill. In addition, most credit card companies have policies limiting your liability (to $50 or sometimes even $0) in case your card number is used without your permission.

Cryptology is the study of making and breaking secret codes. The discipline further subdivides into *cryptography* (the making of codes) and *cryptanalysis* (the breaking of codes). The goal of the cryptographer is to design codes that are easy for authorized users to use but difficult for unauthorized users (the cryptanalysts) to break.

WHAT CRYPTOLOGY IS NOT

Some students at past workshops have come up with elaborate schemes of encrypting messages. For example, one might replace the letters of the alphabet by numbers, then convert these numbers to hexadecimal representation (that's base 16, the way computer programmers like to do things), then apply a substitution cipher (described below), perhaps apply another substitution cipher, finally write the numbers down in reverse order on a scrap of paper. Such scraps have been handed to me, at past workshops, with the challenge to "break the code"; the person handing me the paper usually has a grin that is smugly saying, "you won't be able to figure out the message". No kidding! Of course I won't figure it out. The security of this encryption scheme rests on being used once for a single short message. If the same scheme were to be used over and over, for many messages, then likely it *could* be broken. To avoid using it over and over, you would have to invent a new scheme for each message. But you would also have to secretly meet with the legitimate receiver of the message ahead of time and explain the elaborate scheme. If you are willing to do that for each message, then you really don't need cryptology in the first place: just exchange your secrets during your secret meeting!

The real goal of cryptology (more specifically, or cryptography) is to come up with schemes that are *easy to use* by the legitimate sender and receiver, but *hard to break* by unauthorized users, even if lots of messages are being sent using the same scheme. Perhaps it helps to imagine the following: you wish to design email software that others can use to exchange email securely. Right now, while writing the software, you can make decisions about how it will run. But once it is finished, thousands of people will be using your software, and you won't be able to make any more changes.

SUBSTITUTION CIPHERS

Let us say you wish to send: "THIS MESSAGE IS SECRET". One way to encrypt this message is to substitute letters according to some substitution scheme, for example:

A→C	E→Y	I→Z	M→O	Q→Q	U→H	Y→J
B→F	F→L	J→W	N→U	R→T	V→K	Z→G
C→D	G→X	K→A	O→E	S→P	W→R	
D→M	H→N	L→V	P→S	T→B	X→I	

Using this scheme (which I made up randomly by reordering the alphabet), the encrypted message is:

BNZP OYPPCXY _ _ _ _ _ _ _ _

To make sure you understand the method, fill in the remaining blanks! To decrypt the message, the authorized receiver (who also knows the particular substitution scheme used to encrypt the message) simply applies the process in reverse.

How many different substitution schemes are possible? An equivalent question is: how many different ways can you reorder the alphabet? Before answering this question, let's answer a simpler version of it. Say that our alphabet consists only of three letters, "A", "B", and "C". To write down a substitution scheme, start with

A→ _
B→ _
C→ _

and fill in the blanks. Each letter must go into the blanks exactly once. Thus there are three choices for which letter goes into the first blank; having made that choice, there are two choices for which letters goes into the second blank; having made that choice, there is one choice (which is really no choice at all!) for which letter goes into the last blank. This makes for a total of $3 \cdot 2 \cdot 1 = 6$ possibilities. Here they are:

A→A	A→ _	A→B	A→B	A→ _	A→ _
B→C	B→ _	B→C	B→A	B→ _	B→ _
C→B	C→ _	C→A	C→C	C→ _	C→ _

Find all six possibilities and fill in the blanks! In doing so, you might wonder, is the scheme "A→A, B→B, C→C" acceptable? Although this is not a *useful* encryption scheme (why not?), it is nonetheless a valid substitution scheme and thus ought to be counted when we count all possible substitution schemes.

Back to our English alphabet of 26 letters. How many substitution schemes are there? This time there are 26 blanks to be filled in. There are 26 choices for filling in the first blank; having made that choice, there are 25 choices for filling in the second blank; having made that choice, there are 24 choices for filling in the third blank; and so on. This makes for a total of $26 \cdot 25 \cdot 24 \cdots 3 \cdot 2 \cdot 1$ possibilities, which is a whopping big number, namely

$$26! = 26 \cdot 25 \cdot 24 \cdots 3 \cdot 2 \cdot 1 = 403291461126605635584000000.$$

Here I have used the shorthand notation "26!", which is spoken "twenty-six factorial"; the exclamation point means to write down the number, followed by all the counting numbers below it, and multiply everything together. Earlier, with the three-letter alphabet, we saw that the number of ways to rearrange the three-letter alphabet was given by "three factorial", which is just the number "six":

$$3! = 3 \cdot 2 \cdot 1 = 6.$$

BREAKING SUBSTITUTION CIPHERS

There are two ways to break (i.e., decrypt) a message encrypted with a substitution scheme. There is a stupid way and a smart way. The stupid way doesn't work because it takes too long (longer than the age of the universe); the smart way works so well that you can do it on the bus while riding to school. I'll talk about the stupid way first, then give a brief description of the smart way.

Say an adversary has intercepted your encrypted message

BNZP OYPPCXY ZP PYDTYB

and wishes to break it. They know only that you have employed some substitution scheme, but they do not know which of the 26! schemes has been used. One breaking method is to "try all possibilities". For example, a computer program could run through all possible substitution schemes and see whether any of them unscrambles the message. (By comparing against a dictionary of known English words, the computer program could reject most substitution schemes because they do not turn the message into something intelligible; there might be more than one substitution that gives back English words, but there will be very few such possibilities, and a human can make the final decision.) Trying all possibilities is also known as "performing an exhaustive search". How long would this take the world's fastest computer, or even all computers in existence today working together in parallel? To be on the safe side, let us say such a computer network can check one trillion possible substitutions per second. Thus the number of seconds required is

$$\frac{26!\ \text{SUBSTITUTIONS}}{1000000000000\ \text{SUBSTITUTIONS/SEC}} = 403291461126605.6\ \text{SEC},$$

which translates to 6721524352110 minutes, or 112025405868 hours, or 4667725244 days, or 12788288 years. Thus we are talking 12 million years to try all possibilities!

At this point you might feel convinced that using a substitution cipher is a secure way of encrypting your messages. But in fact, the method is totally insecure! The reason is simple: your adversary can use a smart way of breaking the message that is much faster than performing an exhaustive search! The idea is to make use of letter frequencies and other patterns to help guess which substitution scheme is being used. For example, in the encrypted message

`BNZP OYPPCXY ZP PYDTYB`

one notices that "P" occurs rather frequently, namely 5 times, and thus most likely represents one of the more common English letters, perhaps "E" or "S". Knowing how common various letters (also pairs of letters) are helps you make good guesses about the scheme. It would be boring to talk about this further here; you learn how to break substitution ciphers by trying a few examples. Newspapers often contain some puzzles along these lines, called "cryptoquips". Steal your parents' puzzle page from the newspaper and break a few of these messages on the way to school!

SUBSTITUTIONS AS FUNCTIONS

Any particular substitution scheme is very much like a mathematical function, only its inputs come from the 26-letter alphabet, and its outputs are again in the 26-letter alphabet. (In fancy language, the *domain* and *range* of the function are the English alphabet.) Because you also want to be able to decrypt a message, the substitution function cannot send two different input letters to the same output letter (do you see why not?). Thus different inputs go to different outputs, meaning that we are talking about *one-to-one* functions. Also, each possible output is hit precisely once by some input, so we're talking about *onto* functions. Sometimes these functions are also called *permutations*, because they *permute* (or "reorder") the alphabet. Since these functions are one-to-one and onto, you can apply the permutation "in reverse", i.e., you can make use of the *inverse* function.

COMPOSING SUBSTITUTIONS

Since we have seen that substitution ciphers are easy to break, you might wonder, can we increase the security of our system by first applying one substitution scheme to our message, then applying a second substitution scheme? That is, having applied the first scheme, we obtained

`BNZP OYPPCXY ZP PYDTYB`

and if we apply another scheme, we'll get something else. For example, if we apply the same scheme from earlier a second time, our double-encrypted message becomes:

FU _ S E _ SSD _ _ _ S S _ MB _ F

(As before, fill in the blanks.) Is this double-encrypted message any harder to break? The answer is "no", and the reason is as follows: composing two substitutions (performing one after the other) is the same as some other single substitution. In our example, the original message's "T" first got sent to "B", then that "B" got sent to "F"; this is the same as directly sending "T" to "F" via a single substitution. Even if we had composed two different substitutions, we would still get back a single substitution. Make sure you see why this is true. (Try writing down two arbitrary substitutions and then figure out the single composed substitution. By this I mean to make a list of arrows, as we did originally, showing where each letter gets sent.)

Do you remember the story about the student who invented this elaborate encryption scheme, writing the letters as numbers in hexadecimal, then applying a few different substitution schemes, then writing everything in reverse order? Even writing everything in reverse order is itself another substitution scheme, and thus the net effect of all these substitution schemes is some other unknown substitution scheme. In short, the elaborate encryption scheme came down to using numbers for letters and applying some substitution. It sounded elaborate, but any master of newspaper cryptoquips could break such a message in little time!

CONCLUSIONS ABOUT SUBSTITUTION CIPHERS

We have seen that there are many possible substitution ciphers. They cannot be broken using an exhaustive search (due to time constraints); but they can easily be broken by other means. By the way, almost all cryptosystems (methods of encrypting messages, including but by no means limited to the use of substitution ciphers) could be broken in theory using some type of exhaustive search, but time constraints prevent such an attack from being carried out in practice. The goal of the cryptographer (the person who designs a secret code system) is to design a system for which there is no better method of attack than exhaustive search. This is not the case for substitution ciphers, and so we conclude: substitution is not a secure system.

We have also seen that using substitutions to encrypt a message multiple times (by composing lots of substitutions) makes for a lot of extra work on the part of the authorized users of the system, but makes for zero extra work on the part of the adversary breaking the system. (The sender and receiver have to apply lots of extra substitutions, but an adversary is still going to look for patterns and often-used letters to quickly break the message.) By the way, there *do* exist cryptosystems for which applying the system multiple times *does* make the system more secure; substitution is not such a system.

You might wonder: since substitution ciphers are so bad, why bother studying them? It turns out that they form the heart of other more elaborate (and more secure) cryptosystems. Here follows one example. During WWII, Nazi Germany encrypted its internal military messages using a special mechanical typewriter-like device called Enigma. This device applied a certain substitution scheme to the first letter of the message. Then the device applied a *different* substitution scheme to the second letter. Yet a different substitution was applied to the third letter. And so on. I like to describe the Enigma as an "evolving substitution cipher". It would be hard for us to have a meaningful discussion of the Enigma if we didn't first understand substitution ciphers. In fact, it would be hard for us to have a meaningful discussion about most things related to cryptology without first understanding substitution ciphers. That's why we start by studying them.

(By the way, how the Enigma machine did its job mechanically using wheels and gears and such is itself a fascinating story, and I hope you'll make it a point to learn more about this! Also fascinating is the story of how the Enigma code was broken by the Allies, itself done using mechanical computers: first the number of possibilities was reduced using a combination of human analysis and guesswork; then all the remaining possibilities were checked over periods of hours by noisy mechanical machines.)

MODULAR ARITHMETIC

This topic is completely separate from the previous one, so take a moment to clear your mind of the previous material.

Modular arithmetic involves the kind of math done in elementary school. All of the following discussion concerns integers (the counting numbers together with their negatives and the number zero); there will be no fractions and no decimals. In particular, when we talk about dividing two numbers, there will be both an integer quotient and an integer remainder. For example, 27 divided by 12 gives a quotient of 2 and a remainder of 3; we write

$$26 = 12 \cdot 2 + 3.$$

Note: the remainder is always smaller than the number by which we are dividing (why is that?).

You already know a particular example of modular arithmetic, but you've probably called it "clock arithmetic" (if you've named it at all). Say it is presently 3 o'clock. What time will it be in 10 hours? It will be 13 o'clock, of course, and that's the same as 1 o'clock! Starting again at 3 o'clock, what time will it be in 36 hours? If you're clever, you'll observe that 36 is a multiple of 12, and adding 12 hours doesn't change the time on our clock, thus adding 36 hours also does not change the time! That is, adding 36 is the same as adding 0. On the other hand, what if we add 37 hours? Since 37 is 1 greater than a multiple of 12, isn't adding 37 the same as adding 1? If this makes sense so far, then answer the following: what gets added to the clock if we wait 243 hours? To figure this out, you really want to remove as many 12s as possible from 243; in other words, you want to know the remainder when 243 is divided by 12; since $243 = 12 \cdot 20 + 3$, adding 243 hours is the same as adding 3 hours.

Even before, when we started at 3 o'clock and added 10 hours to obtain 13 o'clock, in changing 13 o'clock back to 1 o'clock, you might have thought that you were simply subtracting 12, but in the back of your mind you *knew* that there was only a single 12 to be subtracted, and so one might argue that what you were *really* doing was *dividing* by 12 and keeping the remainder; doesn't 13 divided by 12 leave a remainder of 1?

The point of all this is that anytime you are doing anything involving a clock, you aren't really interested in the numbers themselves so much as in their remainders upon division by 12. These remainders are all that matter.

Now mathematicians like to use fancy language and fancy symbols. (Really they aren't trying to be fancy at all; what seems fancy at first turns out to make life a lot easier down the line.) Rather than saying that 13 o'clock is the same as 1 o'clock, a mathematician would say that "13 is congruent to 1 when working modulo 12", and one writes:

$$13 \equiv 1 \pmod{12}.$$

The "modulo 12" part tells you that in everything we're doing, we will always be dividing by 12 and only be talking about the remainder. Thus one can also say that 13 is congruent to 25 (modulo 12), since both 13 and 25 give the same remainder upon division by 12. (Although 13 and 25 end up being the same thing in the world of "modulo 12", rather than calling them "equal", we use the word "congruent" to remind ourselves that we're working in an unusual world.)

How many different numbers are there in the "modulo 12" world? Since upon division by 12 the possible remainders are 0, 1, 2, . . . , and 11, there are in fact 12 different numbers in our world. (Do you see that 12 o'clock might better be called "0 o'clock", since the remainder upon dividing 12 by 12 is 0?)

Now rather than doing clock arithmetic with a 12-hour clock, why not do clock arithmetic with some other sized clock? But let's call it "modular arithmetic" instead. The idea is to pick some number m, called the *modulus*, which remains fixed throughout our discussion. Then no matter what you're doing, you always divide everything by m and hold on to the remainder.

Let's set our modulus to be $m = 7$. Thus we are about to be in the world of "working modulo 7", which is the same as working on a clock that has a total of 7 hours, labeled 0, 1, 2, ..., through 6. In this world, the number 16 is the same as the number 2, and both are the same as the number 9, and these are even the same as the number -5. The point here is that all four numbers leave the same remainder upon division by 7. (You might wonder about -5. By writing $-5 = 7 \cdot (-1) + 2$, one could argue that the remainder is 2. Of course, you can also write $-5 = 7 \cdot 0 - 5$, so that the remainder is -5, and you can write $-5 = 7 \cdot 1 - 12$, so that the remainder might be -12. But you can always *choose* the quotient so that remainder lies precisely between 0 and 6, and this is what you should *always* do!) So we have

$$16 \equiv 2 \equiv 9 \equiv -5 \pmod{7}.$$

By the way, there's another way to think about modular arithmetic. Rather than saying that "two numbers are congruent (modulo m) when they leave the same remainder upon division by m", one could instead say that "two numbers are congruent (modulo m) when their difference is *divisible* by m". One way to see this is to look at a numberline. On the list below, mark all the numbers that are congruent to 1 (modulo 7):

$$\ldots \quad -7 \quad \underline{-6} \quad -5 \quad -4 \quad -3 \quad -2 \quad -1 \quad 0 \quad \underline{1} \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad \underline{8} \quad 9 \quad 10 \quad 11 \quad 12 \quad 13 \quad 14 \quad \underline{15} \quad 16 \quad \ldots$$

Now if you pick any two marked numbers, the distance between them is either 7 or twice 7 or three times 7, and so on. In other words, the difference between any two marked numbers is a multiple of 7, which is another way of saying that the difference is divisible by 7.

This new way of thinking might make you more comfortable with negative numbers. Earlier we said that $9 \equiv -5 \pmod{7}$. One way to check that this is true is to notice that the difference between 9 and -5, namely 14, is divisible by 7.

MODULAR MULTIPLICATION

Let's say you are working modulo $m = 5$, and you want to multiply 62 and 48. One way to do this is to first compute $62 \cdot 48$ (in the world of usual integers) to obtain 2914, then reduce 2976 modulo 5 (i.e., take the remainder) to obtain 1. But a better approach (which keeps your intermediate results smaller) is to *reduce at every step along the way.* Thus you first reduce modulo 5; that is, replace 62 and 48 by 2 and 3; then multiply to obtain 6; then reduce again to obtain 1. Thus we have seen that

$$62 \cdot 48 \equiv 2 \cdot 3 \equiv 6 \equiv 1 \pmod{5}.$$

It doesn't matter whether you wait to reduce until the very end, or reduce at each step along the way. You'll get the same final answer.

Make up a few arithmetic problems for yourself, and work them out in the modular world, either by reducing at the end, or by reducing at each step along the way. If your results are inconsistent, then either you have misunderstood something, or you're making a mistake. You should talk to your teacher and get the problem resolved now. If you can't do basic arithmetic in the modular world, then you'll have trouble with everything else (all the cool public-key cryptography) that makes use of modular arithmetic.

MODULAR EXPONENTIATION

Working modulo $m = 11$, let us say you wish to compute $14^{13} \pmod{11}$. Here you have to be a little careful: the exponent 13 does *not* live in the world of modulo 11; that is, 14^2 and 14^{13} are *not* the same thing in this world. Recall that $14^{13} \pmod{11}$ means to compute

$$\underbrace{14 \cdot 14 \cdot 14 \cdots 14}_{13 \text{ copies}} \pmod{11}.$$

Now each 14 *does* live in the modular world, so they can certainly be replaced by 3 (since $14 \equiv 3 \pmod{11}$), and furthermore, as you are multiplying all these numbers together, you can reduce modulo 11 at any time to keep the intermediate result small. Thus one approach is to compute

$$\begin{aligned} 3^2 &\equiv 3 \cdot 3 \equiv 9 \pmod{11} \\ 3^3 &\equiv 9 \cdot 3 \equiv 5 \pmod{11} \\ 3^4 &\equiv 5 \cdot 3 \equiv 4 \pmod{11} \\ &\vdots \\ 3^{11} &\equiv 1 \cdot 3 \equiv 3 \pmod{11} \\ 3^{12} &\equiv 3 \cdot 3 \equiv 9 \pmod{11} \\ 3^{13} &\equiv 9 \cdot 3 \equiv 5 \pmod{11} \end{aligned}$$

Thus at the end we see that

$$14^{13} \equiv 3^{13} \equiv 5 \pmod{11}.$$

By the way: there is an even better way to obtain the same result, called "repeated squaring"; this better method is of utmost importance, and will be demonstrated at the PUBLIC-KEY CRYPTOGRAPHY workshop.

CONCLUSIONS ABOUT MODULAR ARITHMETIC

Although modular arithmetic might seem strange at first, it is really just clock arithmetic; the clocks might be the usual 12-clocks, or they might be clocks of other sizes. Always keep in mind that you are more interested in the remainders than in the intermediate numbers.

Although the notation and language of modular arithmetic might appear bizarre, please make an effort to practice it, as fluency with this language will make future work much simpler. (Also: we'll use this language freely at the PUBLIC-KEY CRYPTOGRAPHY workshop.)