Lecture Notes: Cryptography – Part 1

A cryptographer *encodes* messages (typically texts in some standard language; we will stick to English here) before they are transmitted (by courier, over radio, now over the internet, ...) so that even if the encrypted message is intercepted by a hostile party, its meaning will (hopefully) still remain secret; in principle, only "friends" of the original cryptographer, who know the secret recipe for *decoding* or *decrypting*, can decode the encrypted message and recover the original plain text.

Cryptography schemes have existed for thousands of years, and attempts to "break" coding schemes are almost as old. A "code breaker" seeks to detect patterns in the encrypted messages that will lead to sufficient understanding of the encryption scheme to enable the discovery of a decryption method.

In this module of the Math Alive course we shall discuss several cryptography schemes, starting with a 2000 year old scheme, and ending with a discussion of the public-key cryptography used, for instance, for transmitting credit card information over the internet.

Old standards: transliteration schemes or substitution ciphers

The oldest schemes replace the letters in the message one by one, following a fixed recipe. The "key" to such a transliteration scheme just lists all the letters in the alphabet and gives for each letter the corresponding crypto-letter. For instance,

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

indicates that every A will be replaced by a G, every B by an O, etc.

To decrypt schemes of this nature, one just needs to reverse the transliteration; in the example above one would replace every G in the encrypted message by A, every O by B, every K by C, etc. The decoding is thus again a transliteration. For the example above, the decoding transliteration is

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

One of the very oldest documented transliteration schemes was used by the Roman general Julius Caesar (1st century BC). His transliterations were particularly simple because they involved only a *shift* of the alphabet, such as in the following example:

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

This example was obtained by writing a second copy of the ordered alphabet for the transliteration, starting 6 steps to the right, with A underneath the letter G in the first line, and going on until T is written underneath Z; at that point, the writing gets "wrapped around" and the alphabet is completed, from U to Z, by writing underneath the letters A through F from the first line. Codes of this simple shift type are called *Caesar codes.*

Note that the corresponding decoding transliteration

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

which consists of moving 6 steps to the *left* (or equivalently, 20 steps to the right: after all, moving 26 steps is the same as not moving at all, or $+26 \simeq 0$, so that $-6 \simeq -6 + 26 \simeq 20$). This is again a Caesar code.

A variant on Caesar codes is obtained by not only shifting the starting point of the alphabet, but writing the alphabet in the opposite order, as in the following example

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

In this case the decoding scheme is the same as the encoding scheme: the transliteration replaces M by T and T by M; the same symmetry holds for all other letters. These "mirrored Caesar codes" are used as one element in more complex schemes to which we shall come back later.

If the language in which the plain text is written is known to the code-breaker, and if the messages contain a few sentences of text, then it is very easy to break the transliteration codes by the following trick.

In every language some letters are used more often than others. The table below gives the frequency for the 26 letters of the alphabet in English.

	STANDARD
LETTER	**FREQUENCY**
a	.0761
b	.0154
c	.0311
d	.0395
e	.1262
f	.0234
g	.0195
h	.0551
i	.0734
j	.0015
k	.0065
l	.0411
m	.0254
n	.0711
o	.0765
p	.0203
q	.0010
r	.0615
s	.0650
t	.0933
u	.0272
v	.0099
w	.0189
x	.0019
y	.0172
z	.0009

Frequencies of Occurence of Individual Letters in English Text

We see that the most frequently used letter is E: on average, 12.62% of the letters in an English text are Es; the least frequent letter is Z, occurring, on average, only 0.09% of the time. If a text is transliterated from English, we can check which letters occur most frequently. In a text encoded with the first transliteration example above, we would detect that the most frequent letter in the encoded text is A, indicating that A stands for the original letter E; checking for the next most frequent letter would lead to uncovering that X stands for T, etc. Once sufficiently many letters are decrypted, some of the text becomes legible, and guesses about its content can be used to help with the further decryption. If one observes that the transliteration is a simple alphabet shift, then it is even easier to complete the transliteration table after the first few letters.

Making transliteration more complex

Transliteration codes are easy to break, because known and recognizable patterns in the original text (frequency of letters) persist in the encrypted text, and can be exploited to construct a decoding scheme. Codebreaking attacks always try to detect patterns and use these to get a handle on the coding key, which then suggests a decoding key.

To make a coding scheme hard to break, it is thus important to disguise patterns as much as possible. One way to disguise the relative frequency of letters which persists in transliteration is to alternate between different transliteration schemes,, e.g. using one scheme for the 1st, 3rd, 5th, ... letters, and another for the 2nd, 4th, 6th, ... letters. Given a sufficiently long text, the slightly more complex patterns that persist through this scheme are still easy to detect, and every code-breaking expert would make very short work of such an alternating scheme.

The more transliterations are used, and the more randomly the different transliterations succeed each other, the harder it becomes to break the code. In fact, if you were to write a *completely random* sequence of the numbers 0 through 26, e.g. 23 5 9 0 2 17 21 13 14 ..., with as many entities as you have letters in the text to encode, and you would use these numbers, one by one, to determine the Caesar code shifts for the letters in your text (in this example, we shift the first letters 23 letters to the right in the alphabet, the second 5 steps, the third 9 steps, etc.), then your encoded message could *not* be decrypted, provided you *never* used the same encryption sequence again. What was just described is the *one-pad method*, which is known to be unbreakable. Should you use the same sequence of numbers over and over again, then the scheme starts leaving detectable patterns, and it can be broken; to be secure, you must use a new pad every time you want to encrypt anything. This makes it very cumbersome for your intended recipient to decrypt messages: somehow, the two of you must have agreed on all those encryption pads beforehand, so that your recipient has them all (and if you intend to have regular transmissions, there will be a lot of them, since each can be used only once).

Around WWII, several inventors developed coding machines that used constantly changing transliterations, with complex and (they hoped) hard to detect patterns in the ordering of the shifts of the consecutive transliterations. There were several major inventors, e.g. Edward Hebern (the ECM machine), Arthur Scherbius (the celebrated Enigma machine) and Boris Hagelin (the Hagelin machine). Although the Enigma machine is the most famous, Hagelin was the only one of the three to make serious money from his invention. Versions of his machine were in use in many military and diplomatic units, until well into the 50s; the company he started (the A.G. company, based in Switzerland), is still a large corporation today.

The Hagelin Cryptograph: One of the most popular cryptographs in the 1940's and 1950's

This machine uses ever-changing "mirrored Caesar codes": for each letter, in the message to be encoded, the machine first determines a shift of the alphabet, and then copies the alphabet (with shift) in reverse order.

The Hagelin machine changes the shift for every letter to be encoded via an ingenious system that tries to make the successive shifts very “random.” Several views of a Hagelin cryptograph and its machinery are shown in Figures 1 to 5, below.

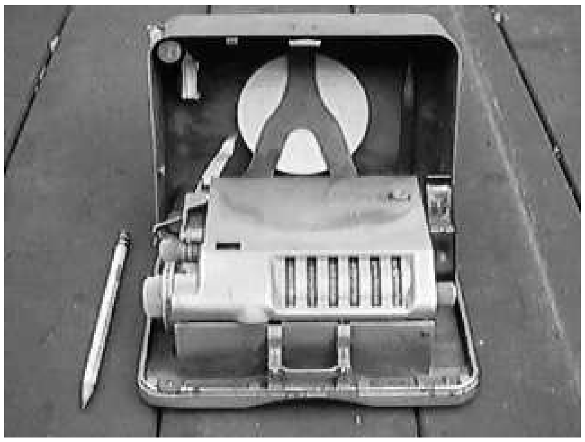
Figure 1

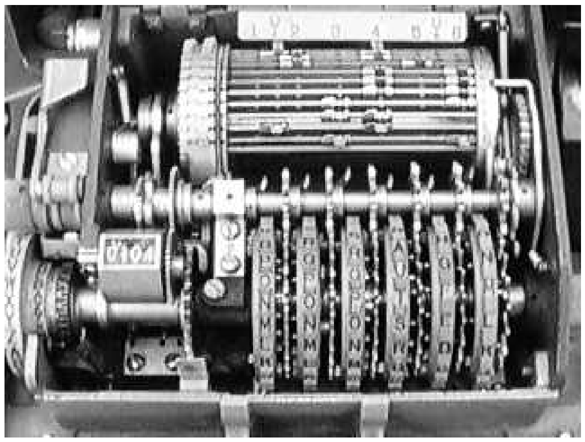
Figure 2

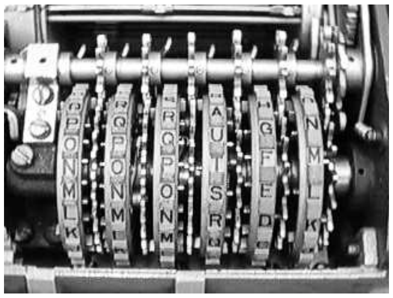
Figure 3

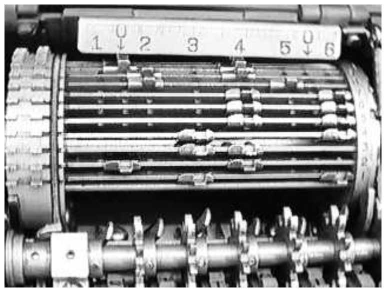
Figure 4

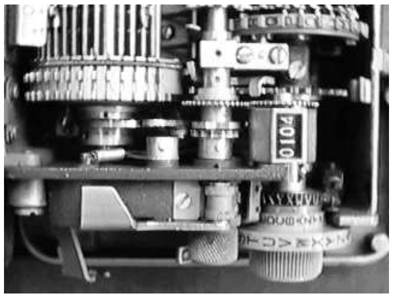
Figure 5

Description of the mechanism

The machine contains six rotors. Each of these rotors has pins sticking out. (In Figure 6 below, one such rotor is sketched, with only one pin shown, but in practice there are several. Figure 3 shows all six rotors, seen from the side – they are between the wheels with letters.) In some positions, these pins push against a little lever, in others they don’t. (These levers can be seen in Figures 3 and 4, above the cilindrical axis that is itself just above the rotors.) In one of its two positions, the lever moves some of the little riders (called “lugs” in Hagelin parlance) on a rotating drum; in the other position it does nothing. (The drum and its lugs can be seen clearly in Figures 2 and 4; the drum rotates each time the machine is cranked for the encoding of the next letter.) When any little rider (or lug) gets pushed out of place, it makes a ridge stick out on the rotating drum. This ridge fits into a gear wheel and makes the gears turn; this only happens when the ridge sticks out, and not otherwise. (This mechanism is shown in Figure 5.)

Finally, this gear wheel is connected with the spinning alphabet wheel. So the shift of alphabet is determined by the number of lugs that are moved; they, in turn, are determined by the levers

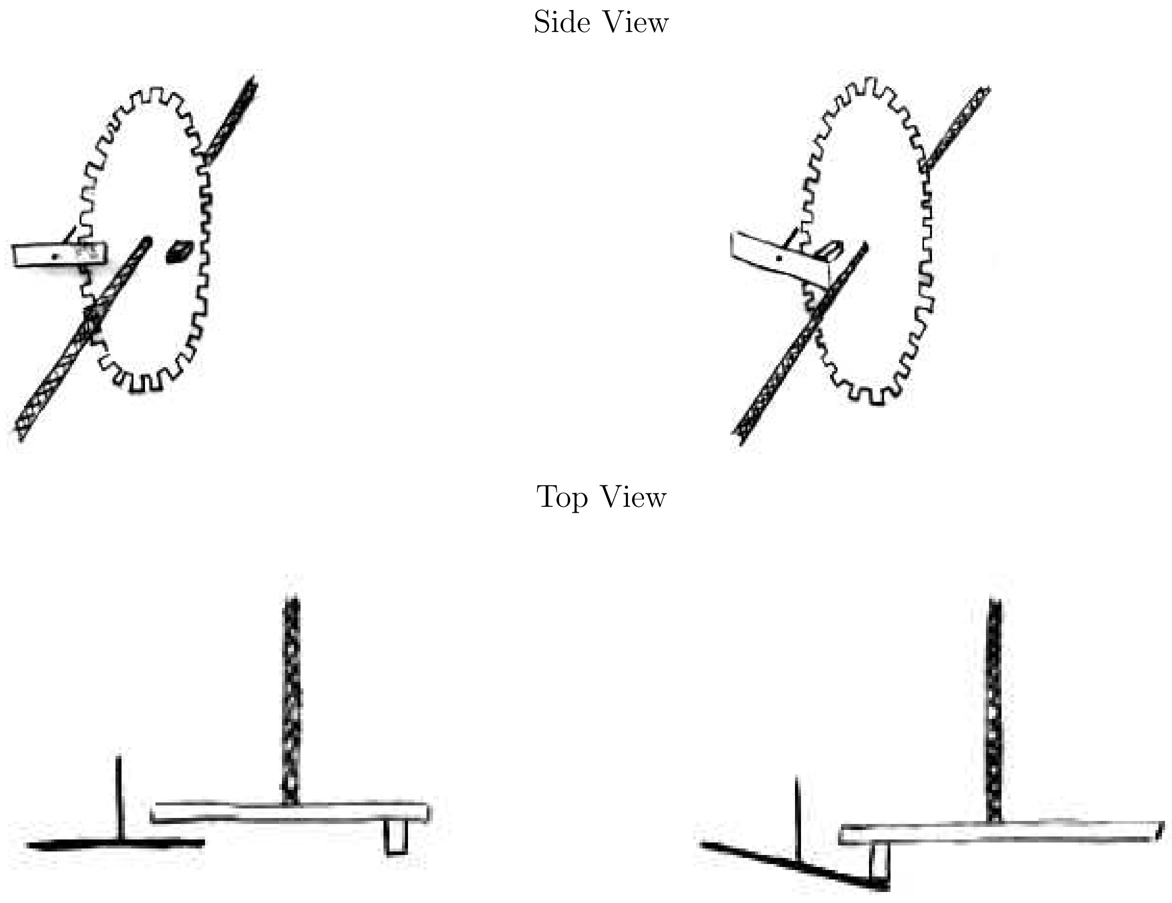

Figure 6

triggered by the “active” pins. The different rotors are turned by one gear step each after every letter encoded. This changes which pins are active or inactive, so that the number of lugs moved changes as well; therefore the alphabet shift will be different every time.

Complexity of the Hagelin encryption scheme

The six rotors have respectively 17, 19, 21, 23, 25, and 26 gears. The first rotors will therefore be in its original position after every 17 cranks, the second after every 19 cranks, etc. One can then ask after how many cranks (N) the rotors will all be back in their original position; N determines the period of the whole encryption scheme, i.e. the number of encryptions after which the pattern starts repeating.

N has to be a multiple of 17, 19, 21, 23, 25, and 26. Since these numbers are relatively prime (no pair has a common divisor), N must be their product:

$$N = 17 \times 19 \times 21 \times 23 \times 25 \times 26 = 101,405,850 \; .$$

This is a very large number, much larger than the number of letters in any text one would like to encrypt. However, even though it takes this very large number of cranks before the whole sustem repeats exactly, because it is made up of six rotors, each with its own system of pins that cycle back to their original position much faster, the whole mechanism leads to more patterns than a single rotating wheel with N gears would have (which of course cannot be constructed!). Thus there are more patterns than just this periodicity. In addition, if the machine, with the same settings, is used by many senders (e.g. in a military setting at war), then the code-breaker can use patterns present in many of their messages. As a result, this code can be (and was) broken especially if it was used unwisely. (For instance, if the internal settings governed by the pins and lugs, were not changed sufficiently frequently.). Still, the breaking takes quite a bit more work than if the alphabet shift did not change with every letter.

A newer scheme, used by HBO in the 80s

(Some explanation of this can also be found in For All Practical Purposes.)

HBO transmissions to their subscribers (some of whom are licensed local cable stations who then transmit the program, over cable, to *their* subscribers; others are individuals who pay their subscription directly to HBO and who capture the signal with their own antenna dish) have been encrypted since the 80s. The encryption of the movies and other programs distributed by HBO is the same for everyone, so that HBO has to broadcast only one encrypted version. The encryption is governed by a password, i.e. a sequence of 56 digits, each 1 or 0

$$11010100011010110100110111101100011\ldots$$

Because the subscribers of HBO pay their fee monthly, the password used by HBO changes monthly as well: subscribers who don't pay their fee for a given month won't receive the password for that month, and won't (in principle) be able to decode the HBO programs that month.

The next question HBO faces is how to transmit, each month, the password to only those customers who paid their subscription for that month. Rather than send out thousands of letters, HBO broadcasts the password as well, with an additional layer of encryption, different for each user. Each customer has a key, also a binary number of 56 digits. For example

$$010110100100111101\ldots$$

This key is used as a "shift" keypad to encrypt the password—exactly like in our alphabet examples, except that in our alphabet now there are only two digits, 0 and 1. A 0 in the keypad means no shift: 0 remains 0, 1 remains 1, or, in the double-line format we saw for letter encoding:

01
01

A 1 in the keypad means a circular shift: 0 becomes 1, 1 becomes 0, or:

01
10

The password encrypted with this key is thus obtained as follows:

$$\begin{array}{rl} \text{password} & 110101000110101101\ldots \\ \text{key} & \underline{010110100100111101\ldots} \\ \text{encrypted password} & 100011100010010000\ldots \end{array}$$

In order to recover the password, you just use your key to undo all the shifts one by one, which in this case is the same as just repeating the shifts! (that is, encryption = decryption, just as for the mirrored Caesar codes.)

> "Shifting" by 0 or 1 can also be viewed as a special kind of addition of binary numbers in which you "forget" the carry:

$$\begin{array}{rrrr} 0 & 0 & 1 & 1 \\ \underline{@\,0} & \underline{@\,1} & \underline{@\,0} & \underline{@\,1} \\ 0 & 1 & 1 & \cancel{1}0 \\ & & & \uparrow \\ & & & \text{forget the carry .} \end{array}$$

> You can check that the "encrypted password" above is obtained by such carry-less addition, which is often called *parity addition* or *XOR addition*. In the on-line lab and this week's homework, you can get some practice on working with binary numbers, with binary addition and with parity addition.

HBO sends out a whole sequence of [password @ key], with a different key for each paying customer. The customer's unscrambling box just tries its key on each of these 56-bit sequences, and tries to decrypt with the computed password candidate; when the right one is reached, the picture on the screen looks suddenly miraculously perfect; this can easily be detected automatically, and the unscrambler now knows the password for the next month.

The situation (in the video shown in class) is made slightly more complicated in that the key itself of every customer changes every month; that change is done via a pseudo random generator, of which customer and HBO have identical copies. Because the "seed" from which the pseudo random number generator starts is different for all customers, their keywords every month will be different as well. But the pseudo random number generator is chosen so that working "backwards" is not known to be possible. Otherwise, you could pay one month, get the password that month, use it to compute the keys for that month of all the other customers (you'll work on this in the homework!) and from there work backwards to find their "seeds". By installing such a pirated seed in your unscrambling box you would be in clover, without paying your fee, since your box would now generate, every month, the key of another subscriber who hopefully continues to pay his/her fee.

Note: it didn't take long before pirates found ways around the HBO scheme, despite the added security layer; it has been replaced since then by other schemes, about which HBO is not making any videos ...

How can it be easy to go forward from a seed, but hard to go backwards?

Let us look at an example in decimal numbers, instead of binary, and with passwords of four digits only. I have to tell you in a very simple way how to compute key$_1$, key$_2$, ... (a sequence of successive keys) in such a way that a snooper would not be able to guess key$_{n+1}$ even if he had intercepted key$_1$, key$_2$, ..., key$_n$. Imagine that I tell you to compute the decimals of π, and to take successive blocks of four digits, starting with the fifth:

$$3.1415\ \underbrace{9265}_{\text{key}_1}\ \underbrace{3589}_{\text{key}_2}\ \underbrace{79\ldots}_{\text{key}_3}\ \ldots$$

Since it is possible to compute millions of digits of π quite accurately, you can go on, happily computing key$_n$ every month, for a long time. But the different key$_n$'s are uncorrelated, and if you didn't know the number was π to start with, you'd be in trouble to guess the next one.

In practice, you wouldn't use π, because that is so obvious an example, but some other number that is not so universal but can also be computed with great accuracy.

In the on-line lab, you'll work through an HBO example. If you attempt the challenge questions, you'll find a variant on the HBO scheme where a password would get transmitted in a similar way, but with keys that stay constant, month after month. That system would work okay in the (fictitious) situation where:

1. it would take a long time (say one hour instead of a fraction of a second) to test whether a password guess is correct,
2. the transmission would contain many "decoys" as well.