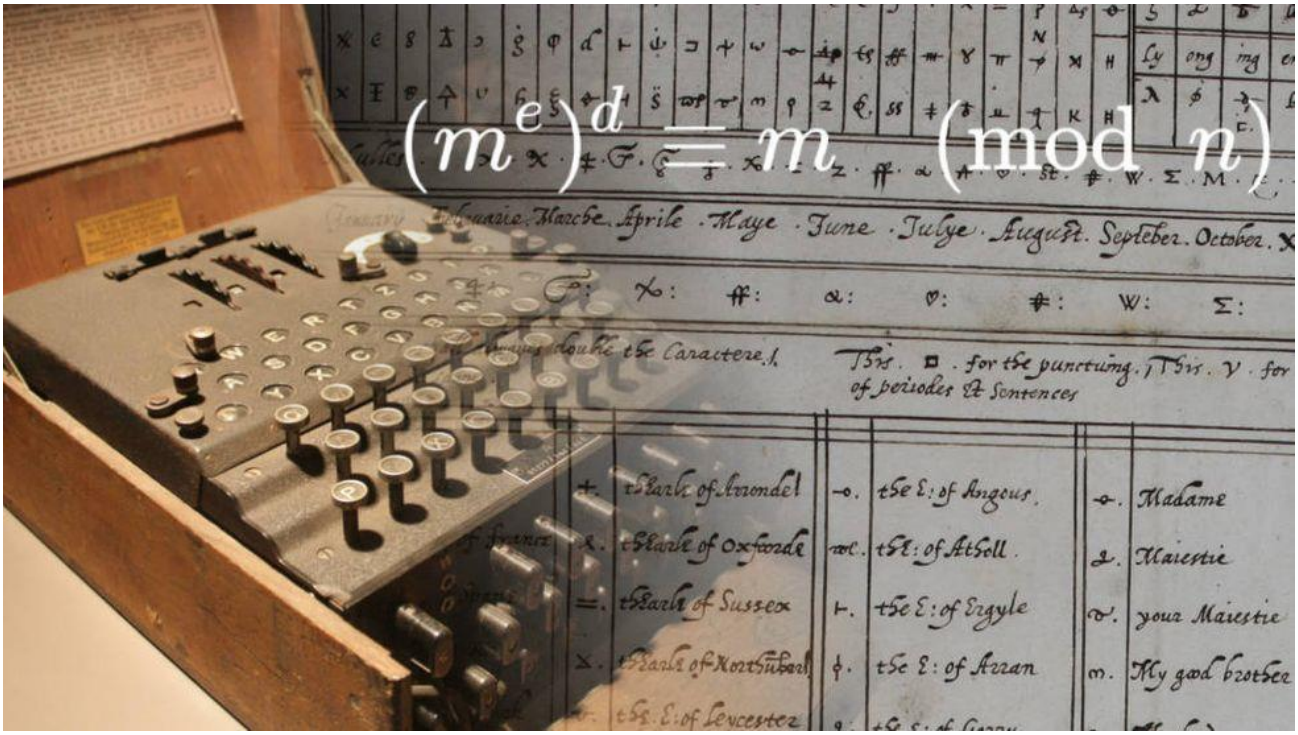
HISTORY OF CRYPTOGRAPHY

with curiosities



Summary

En kortfattad beskrivning över kryptografins historia med de enklaste och mest använda metoderna

Richard Frank

richard.frank@infotrust.se

Author: **Richard Frank**

Innehåll

History

Version	Author	Date	Description
0.1	Richard Frank	17 March 2023	Created first edition
0.2	Richard Frank	March 18, 2023	Revision

Author: **Richard Frank**

1 Introduction

Already the ancient Romans sent coded messages to each other and throughout history until today, encryption has played an extremely important role in society. Over the years, technology has evolved in history and this small compendium shows the most common methods in a more popular historical way and then takes a deep dive into how it works So it's time to close the doors, close the curtainsbecause now let's talk about ENCRYPTION

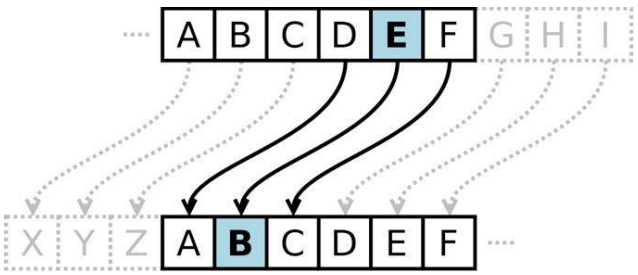
Before modern times, cryptology dealt only with the secrecy of messages (i.e. encryption) - conversion from an intelligible message to an incomprehensible one and vice versa, making the message unreadable to people without knowledge of the secret language (that is, the key needed for decrypting the message). Encryption has been used by, among others, spies, the military and diplomats to try to ensure confidentiality in communications. In recent decades, the field has expanded to include technologies for message integrity, sender/receiver authentication, digital signatures and secure payments.

2 Caesar and the first ciphers

The Romans were not the first to write with ciphers. A Mesopotamian potter in the 1500s BC protected the valuable formof a glaze with a crypto, and Hebrew scholars used simple ciphers a thousand years later. Julius Caesar's widespread use of simple displacement ciphers has given name to these so-called Caesar **ciphers**. The guidance is primarily aimed at those who will carry out the work in practice.

2.1 Displacement ciphers

In a simple ***offset cipher,*** the letters in a text are replaced by moving each letter a number of steps to the right or left of the alphabet. Breaking such is child's play, so over the years, cryptologists have invented more and more intricate ways to replace the letters.



2.2 Vigenère-chiffret

In 1553, Giovan Battista Bellaso invented the Vigenère cipher which was not completely cracked until 1863. In the 1800s, the first ***polygraphic*** ciphers were invented where the letters are not replaced separately but in groups. A little further down in the text, the Vigenère cipher is explained

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

2.3 Codebooks

"Nothing should be considered as advantageous as intelligence; nothing should be rewarded as abundantly as intelligence; nothing should be as confidential as intelligence."

Sun Tsu, The Art of War (300-t f.Kr.)

On Wednesday morning, October fifteenth, 1586, Queen Mary of Scotland stands trial for high treason. The charge relates to inciting the murder of Queen Elizabeth in order to conquer the English crown herself. Sir Francis Walsingham, First Secretary to Elizabeth, has already dispatched the other coup plotters. The historian William Camden recounts: "They were all cut down, their genitals cut off, the viscera cut out while they were still alive and in their right mind; Then they were dismembered." Maria is denied all forms of legal aid and is all alone at her bench. However, the situation was not entirely hopeless, as she had ensured that all correspondence with the insurgents was written in cipher. Even if Walsingham had seized the letters themselves, he should not be able to decipher them and therefore not use them as evidence. The nation's foremost expert in cracking codes was Thomas Phelippes. If he could interpret what was in the letters, she was as good as dead. If the cipher was powerful enough to hide what she had written, she had a chance of survival. It wasn't the first time that life and death hung on the strength of a cipher.

To encrypt and decrypt secret messages, both parties must know the key, and for hundreds of years, militaries and diplomats printed codebooks that were distributed and would be kept safe. If the enemy came across a codebook, they could not only decode, but also send new fake messages.

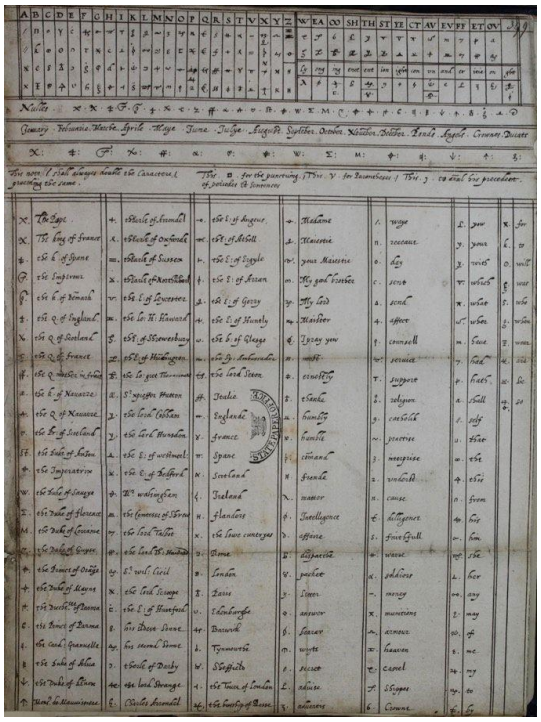


Figure 1: Mary Stuart's codebook – the Scottish Queen was executed after Queen Elizabeth's spymaster cracked the code and veiled the Babington plot.

2.4 The disposable crypto – the only unbreakable crypto

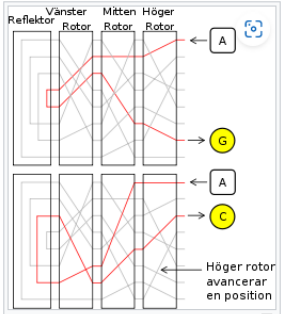
The only encryption system that has been proven impossible to crack is ***the disposable crypto.*** It involves using a series of randomly selected letters (or other characters) that indicate how each character in the plain text should be replaced. The length of the key must be at least as long as the

plain text, and a given key can only be used for a single message. For practical reasons, this is used very rarely.

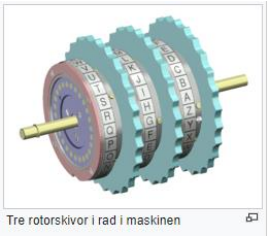
2.5 Crypto machines

During the first half of the 1900s, several countries' intelligence services developed cryptographic machines in which a number of rotating disks perform repeated displacements on each character. The Germans' ***Enigma*** is best known, because the British codebreakers headed by Alan Turing cracked it and shortened the war.

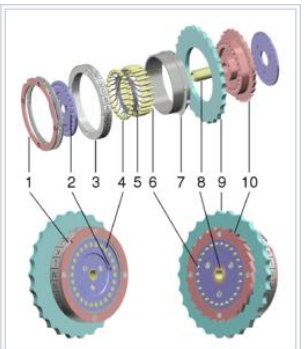
En tysk militär tre-rotors Enigma med (nedifrån och upp) instickskort, tangentbord, lampor och fingerskivor som sticker ut ur höljet



Kryptering av två lika tecken i följd med Enigma – spänning appliceras på rotorerna och fortsätter runt reflektorerna och tillbaka till rotorerna. Bokstaven A krypteras annorlunda för var gång den används, först till G, sedan till C. Detta beror på att den högra rotorn har stegats fram och nu skickar strömmen en helt ny väg.



Tre rotorskivor i rad i maskinen



Sprängskiss av en rotorskiva. 1) ring med markeringar 2) kontakt för "A" 3) alfabetiskt hjul 4) anslutningsytor 5) ledningar 6) bleck 7) hävarm med fjäderbelastning 8) nav 9) fingerskiva 10) kuggkrans för spärrhake

2.6 Mathematical help

After the advent of computers, development has been rapid and encryption has become common even in civilian life. Various cryptos have been developed that rely on computers to quickly convert the ones and zeros that make up a message. Today, the AES algorithm is the most widely used. It involves performing a series of operations on arrays in several rounds, and is generally considered secure in perpetuity as long as the keys are kept secret.

2.7 Steganography

Deliberately hiding the existence of a message is called steganography. Tattooing the message into the rind and then letting the hair grow out was one way the Greeks used. The Chinese wrote on thin silk that was wrapped into a ball and sprinkled with wax before being ingested with a few sips of water. (Both methods require that there is no rush.) In the 1400s, the Italian Giovanni Porta describes how invisible ink can be made from alum and vinegar. The solution penetrates the eggshell of a boiled egg, leaving legible text in the egg white. The microdot, reducing a text to a single millimeter-sized dot, was used by German agents during World War II. The first microdot was discovered by the FBI in 1941.However, steganography suffers from an inherent weakness - put any seizure on the message and it is revealed.

2.8 Cryptography (is not the same as Cryptology)

In parallel with steganography, cryptography developed. This hides the content of the message instead of the message itself. Cryptography can be divided into two branches, transposition and substitution.

Transposition is in practice an anagram in which the letters have simply been reversed. A three-letter word can only be written in six ways. E.g. comb, kma, akm, amk, mka, mak. A random

transposition of a longer message becomes in practice unimaginably difficult to interpret, it requires a system that both sender and receiver agree on.

For example, one can take every other letter and divide the message.

T l e e p l a m n a a n o s a o h e a p m d e a d t +i l x m e k n a t v r n a b k t v c d l u p e d l n e =

T l e e p l a m n a a n o s a o h e a p m d e a d t i l x m e k n a t v r n a b k t v c d l u p e d l n e

Of course, three lines are better than two and the letters can change places in pairs, first with two, third with four, etc.

2.9 Substitution (Displacement cipher)

The alternative to transposition is substitution, i.e. replacing one letter with another. A very early description can be found in the Kama Sutra ,300-t f.Kr., where number 45 out of 64 useful arts of learning for a woman is the art of writing secret messages. It is suggested to randomly pair each letter with another and then replace each letter in the original message with the new letter. Very advanced for that time....

2.10 Scytale

The world's oldest military cryptographic invention is the Scytale of the Spartans (400-t f.Kr.) A strip of leather or parchment is wrapped around a rod of a certain diameter. The sender writes lengthwise and then winds off the strip, which then becomes just a string of meaningless letters. The messenger can use the strip as a belt or bracelet until the recipient wraps the strip on a similar scytale.

The first documented military use of substitution ciphers is Julius Caesar. A text from 100 AD tells us that the emperor simply replaced each letter with the letter three steps

Since then, this type of substitution is called Caesar scrolling or Caesar crypto.

2.11 Crypto or Cipher

The definition of Krypto or Cipher is the name used for all forms of cryptographic substitution where each letter is replaced by a different letter or symbol.

Of course, you can use any offset between one and twenty-eight and if we do not limit ourselves to that but do any repositioning, we get a fairly large number of cryptos. Slightly more than 4 X 10 raised to the power of 26 possible possibilities. If an enemy agent controls a key every second, he needs about a billion times the age of the universe to get anywhere.

Each cipher is a combination of a particular method, also called algorithm, and a key. Thus, in Caesar crypto, the algorithm is to shift the alphabet and the key tells you how many steps it should be shifted. A variation is to replace this simple "plus or minus" key with a key phrase or keyword.

Figure:

a	b	c	d	e	f	g	h	in	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	stream ä	island
R	E	V	O	L	U	T	In	N	P	Q	S	W	X	Y	Z	Stream Ä	Island	A	B	C	D	F	G	H	J	K	M

Double drawings are deleted and at the last letter of the keyword the alphabet continues as usual but without the letters of the keyword.

The advantage is that it is easy to memorize, which is important as everything written down on paper can fall into the wrong hands. On this path it was throughout the first millennium of our era. All scholars agreed that it was impossible to force this crypto thanks to the huge number of keys.

2.12 The first crypto analysts

The Qur'an consists of 114 chapters, each based on a revelation given by the Prophet Muhammad. The Quran is not written by Muhammad but has been written down by various scribes for a long time, on loose sheets and also transferred from one manuscript to another,Now a couple of questions immediately arise;I what order did Muhammad receive these revelations? Are all the texts really the prophet's own words? It takes a culture that has reached a fairly advanced scientific level to sort this out. Mathematicians, statisticians and linguists solved the problem by analyzing the syntax, phonetics and word frequencies of the texts. As a small added bonus, they happened upon a shortcut to crack a crypto that has been considered unforceable for almost 1000 years.

In the 800s, the Arab scientist Al-Kindi wrote a treatise entitled "Manuscript on the Decipherment of Cryptographic Messages". The text that solves a thousand-year-old riddle fits on a single page and can be summarized as follows:1. Find out what language it's written in.2. Find a plain text, preferably quite long.3. Analyze what the most common letters are and arrange these in a frequency table.4. Do the same frequency analysis of the cryptotext.5. Match the plaintext and cryptotext using the frequency table.

Al-Kindi's method is called frequency analysis and shows that it is completely unnecessary to try all the keys.

Figure: Table of relative letter frequency in Swedish. Figures in %.

a	9,3
b	1,3
c	1,3
d	4,5
e	9,9
f	2,0
g	3,3
h	2,1
in	5,5
j	0,7
k	3,2
l	5,2

m	3,5
n	8,8
o	4,1
p	0,7
q	0,007
r	8,3
s	6,3
t	8,7
u	1,8
v	2,4
w	0,03
x	0,1
y	0,6
z	0,02
stream	1,6
ä	2,1
island	1,5

In Swedish, e is the most common letter, followed by a, n and t. Thus, if the most common letter in the cipher is P, then it should be an e or an a. Of course, the shorter the text we have, the more difficult it is to say.

2.13 Atbash

When the Arabs had already learned to dismantle ciphers, Europeans were still struggling to learn how to write ciphered text. The only place where the study of secret writing methods was encouraged was in the monasteries. In the Old Testament, you can find pieces of text encrypted with atbash, a Hebrew form of substitutionskrypto where you count how many steps into the alphabet a letter has and then replace it with a letter that says the same number of steps from the end. "A" becomes "island" and "B" becomes "ä" and so on.More recently, books such as "The Bible Code" by M. Drosnin have sold in large editions. But the content must probably be seen as more speculative than scientific. Any text of sufficient length can be interpreted as predicting this or that. For example, "Moby Dick" contains predictions about the murder of no less than thirteen famous people.

2.14 Transitional period

Towards the end of the 1500s, King Philip the Second of Spain petitioned the Pope in Rome that the famous French cryptanalyst Francois Viete should be examined as he must be "in league with the devil". Fortunately, the Pope's own men had been able to crack the Spanish crypto for several years, so it was immediately rejected.

The countries that noticed the weakness of the monoalphabetic substitution crypto looked for ways to make it safer. There were three main ways.

2.14.1 To sprinkle in "zeros".

Zeros are signs or letters that mean nothing. We imagine that we replace each letter with numbers but that we use 1 to 99. It gives us 70 numbers that mean nothing and thus make analysis difficult.

2.14.2 At såmm than krata staaua.

Deliberately using alternative spelling and syntax also mixes up the cards considerably.

2.14.3 To use a code.

A character or letter can also replace an entire word. "H" can be a substitute for "murder." HFG can mean "assassinate the king tonight". A large number of code words require a book that would be devastating in the hands of the enemy, so they mostly relied on a so-called nomenclature, which is a cipher alphabet and a short list of codes.

The most skilled cryptanalysts were capable of defeating all these difficulties.

2.15 Vigenérekryptot

Already in the 1460s, Leon Alberti (painter, composer, architect) had written an essay on the subject of krypton. He suggested the use of several different cryptoalphabets which could be switched between during encryption. (A simple and obvious idea but still the best in 1000 years.)

Example

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

ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖNOPQRSTUVWXYZÅÄÖABCDEFGHIJKLMa b c de f gh i j k l mno pq r s t uv w x yzåäö

Thus, one jumps between the upper and lower crypto alphabets according to an agreed system.

The person who developed this concept to perfection was Blaise de Vigenére, who in 1586 published a treatise in which he perfected Alberti's thought. Vigenére proposed the use of a large number of cryptoalphabets arranged in a table. Even if that table were to be known by the enemy, it could still be used in combination with a key in the form of a code word that tells you exactly which alphabets to use. (See article three for a more detailed description.) This means that the Vigenére crypto is inaccessible via regular frequency analysis.

The encrypters had finally gotten the weapon they needed.

2.15.1 Was the Vigenére crypto used?

No, it was considered too complicated. Instead of the hard-to-use poly alphabetic substitution crypto, a simpler homophonic ditto was used. This means that each letter can be represented by several different characters. The second most common letter "a" has a relative frequency of approx. 9%, thus it should have eight or nine different designations, while the letter "b" (1.3%) should only have a maximum of two. This is a great opportunity to give each letter one or more values between 00-99. This crypto is also difficult to access for regular frequency analysis. But...

Each language has its peculiarities. In English, "q" is a relatively uncommon letter and thus should not be denoted by more than one character. The peculiarity of the letter q is that in English it is never followed by any other character than you. Furthermore, we know that you make up approx. 3% of English text. If we find any sign in the crypto that is always followed by the same three symbols, we can assume that we have found q and you.

2.15.2 The big cipher

An impressive example is "Le Grand Chiffré" invented by father and son Rossignol who invented an improved monoalphabetic cipher in the 1600-t. (Four hundred years later, "Rossignol" is still slang for worship in French.) Unfortunately, it fell out of use and the details were soon forgotten, with the result that no one could read Louis XIV's secret messages.

It would take until the late 1800s before a crypto expert in the French army managed to solve it. The cipher contained thousands of digits - but only in 587 different combinations. It then occurred to the expert Bazeriés that it could be letter pairs, so-called bigrams. There are 676 possibilities to pair the 26 letters of French. He knew what the most common syllables in French were and could now start applying frequency analysis at the letter-pair level. It took him three years, but he solved it.

2.15.3 The Man with the Iron Mask

At the Bastille sat a prisoner with an iron mask. This prisoner has tickled the imagination of many and, according to one theory, it would even be the king's own brother whom he imprisoned to avoid disputes over the throne. In a letter to Louis XIV, there was a paragraph referring to a man with a mask. A certain General Bulonde has shown cowardice bordering on treason and is now being kept locked up and wearing an iron mask. The letter is genuine and both the time and some other circumstances are correct. Now remember one thing dear friends; Just because it's written in cipher, it doesn't have to be true. Maybe it's a trap in a trap set over 200 years earlier.

2.15.4 How did Mary Stuart, Queen of Scotland, fare?

Thomas Phelippes was masterful when it came to frequency analysis and little by little he found the zeros and was able to start unearthing what the code words meant. He was also an incredibly good forger and forged an addendum in Mary's letter asking the rioters to tell her their names so she could reward them after the rebellion.
Mary was found guilty of her denial and sentenced to death. Historian Rickard Wingfield writes: Then she lay down on the log quite calmly and quietly and shouted out "In manus tuas Domine!" ("In your hands Lord" - Jesus' last words on the cross.) three or four times while she stretched out her arms and legs, and at the last moment one of the executioners held her lightly with one hand, the other gave her two blows with an axe before cutting off her head.

2.16 Mechanized safety:

Churchill, after his visit to Bletchley Park, was shocked by the bizarre mix of people who gave him such valuable information on a daily basis. In addition to mathematicians and linguists, there was a porcelain expert, a museum curator from Prague, a crossword fanatic, several bridge experts and a British chess grandmaster. Despite this, he felt great affection for the motley collection, calling them "the geese who lay golden eggs without cackling".

3 Deep dive into crypto values

3.1 Adult education (hands-on)

In this chapter we will go down into the details and try out different methods, but beforewe start in earnest, a small glossary is probably in place with cryptographic terms. Then we are on the same level when we begin to delve into the algorithms themselves. Much of what we find here are repetitions from chapter 2.

Alice – the person who wants to send a message
Bob – the one who receives a message
Eva or **Eve** – Whoever Alice and Bob think wants to intercept the message

The whole purpose of cryptography is to enable person **A** (commonly referred to as **Alice** in literature) to send a message to **person B** (who is called **Bob**). In addition, Alice must send this message to Bob with the expectation that **person E** (the *eavesdropper or enemy*, commonly referred to as **Eve**) can and will intercept the message somehow.

The requirement for cryptography is to modify the message using some algorithm to make it extremely difficult for Eve to read the message first of all, and secondly, to replace another message that Bob would assume to be a real message from Alice.

The algorithm that translates the message is called **a cipher or encryption algorithm**.

The original message that Alice writes is inthe text. When she applies an encryption algorithm, it will produce another so-called encryption algorithm. encrypted message called ciphertext.
Bob's job, when he receives the ciphertext, is to apply the inverse encryption algorithm, called the **decryption algorithm**, to access the original plaintext.

It may come as no surprise that the encryption and decryption algorithms are very closely related. In fact, they are known as encryption algorithm.

3.2 DES (Data Enc ryption Standard)

Modern algorithms are generally well-defined, well-known and well-studied. To provide additional security, the algorithms most often use a key (or sometimes password), i.e. they are themselves "encrypting".

This key is usually a large binary number and must remain secret between Alice and Bob. In fact, I'm sure you've heard of 40-bit or 56-bit encryption methods like ***D**ata **E**ncryption **S**tandard* (**DES**), where the "bit value" is just the length of the key in bits. For example, in standard DES, the key length is 56 bits long or 7 bytes. (1 byte = 8 bits)

To encrypt a message, Alice connects the key to the encryption algorithm, and to decrypt, Bob connects the same key to the decryption algorithm.

These encryption algorithms are known as *symmetric algorithms*. The security of the system is determined by the secret of the key.

Please note that the security of the system is not deterred by the secrecy of the algorithm. This is a common fallacy. If a system uses a secret algorithm, it usually means that theyn easily become the target of hacking[1]. In our time, with the abundance of public and commercial algorithms, it is simply not worth it to use an encryption algorithm that is secret.

Public **key** algorithms work in a completely different way. These algorithms use two keys: a **public key and a private key**. With this system, Bob publishes his public key to the world.

When Alice wants to send him a message, she encrypts her plaintext with this public key (i.e. Bob's public key) and sends the ciphertext. When Bob receives the message, he decrypts it with his own personal private key. The best known program for doing this type of encryption is **Pretty Good Privacy (PGP).**

Sometimes public key algorithms work the opposite way, Alice encrypts a message with her private key and Bob decrypts it with Alice's public key.

These types of algorithms are commonly known as **digital signatures**: the premise is that if Bob can decrypt a message with Alice's public key, the message may only have come from Alice.
The message has no secret content whatsoever. Like real signatures, digital signatures are added to the "document" and through a simple application of the public key, Bob can verify that the document really comes from Alice.

Document in this context can mean text, applications, or ActiveX components or whatever you want. For example, in most programming languages, we digitally sign the packages for our products to show that they come from us. We use an eg. company called VeriSign, EuroSign or Addtrust to keep our public key and with the help of a Windows program you can automatically check the provenance of packages.

3.3 Lede Fi (Eve or Eva)

In all this discussion, we've mostly talked about Alice or Bob. What about Eva? What should she do to try to read Alice's ciphertext?

Here we assume that Eve knows the encryption algorithm used by Alice and Bob, but she does not know the key. She would like to crack a message so that she can derive the key and then read all further ciphertext messages with impunity. Any attempt to crack a message is known as an attack and the easiest attack to understand is *brute-force attack*. With this attack, she tries only key after key after key until she eventually restores the plaintext. J u the longer the key, the longer it will take her.

In fact, the ratio is exponential. For an 8-bit key, Eve must try up to 28 or 256 different keys. For a 16-bit key, twice as long, there are 65,536 different keys.

For the DES algorithm, seven times as long as an 8-bit key, it is 7 * 1016 different keys. If she could check a billion keys per second, it would take her up to 2.28 years to find the key.

If Eve knew that Alice and Bob had the habit of using 7 ASCII characters for her DES key, she wouldn't have to try nearly as many keys to crack a message and really be able to crack it in about 8 seconds at the same rate.

[1] In The foundation is about using knowledge and programming to obtain information about, get into and in some cases manipulate a computer system – or even an individual computer

If she's worth her weight, she hires a crypto analyst to try to crack the message. [2]

Cryptoanalysis is the science (or art?) of analyzing encryption algorithms along with any ciphertext or plaintext or both to identify the key used to encrypt the plaintext.

3.4 Guessing games

Classically (in other words, in times before computers), encryption algorithms were very simple. In general, they always acted on letters of the alphabet, generating ciphertext that also consisted of alphabetic letters. There are two main types of classic encryption algorithms:

- Substitution ciphers
- Rationingcipher.

3.4.1 Substitution ciphers

A substitution cipher is both the simplest and oldest encryption algorithm. Each character in the plaintext is replaced by another character to produce the ciphertext. The most famous of these is known as Caesar ciphers. With this algorithm, each character is replaced by an **n** letter further away along the alphabet (with a cover at the end, obviously). It was rumored that the cipher that Julius Caesar was **n** was 3, i.e. A is replaced by D, B by E, X by A, Y by B and so on. Decryption is simple: replace each character in the ciphertext by counting backwards

For example, you can easily decrypt FDHVDU as CAESAR, knowing that n every 3. In fact, even if you didn't know the value of n, the cipher is simplicity to break: there are only 25 possible values of n (n = 0 makes no sense as an encryption algorithm!) Writinga cracker program to break the Caesar cipher is trivial. Before the computer age, the easiest way to crack a Caesar cipher was to write down the ciphertext and then extend the alphabet down from each ciphertext letter:

F D H V D U
———————
G E I W E V
H F J X F W
I G K Y G X
. . . . . .
B Z D R Z Q
C A E S A R

Eventually, you'd get a line that made sense; That would be the communication in plain language.

Another example of a Caesar cipher can be found on UNIX systems, especially for newsgroup messages. ROT13 is a Caesar cipher with n=13, and is easy to use by decrypting a message encrypted with ROT13 by applying ROT13 to the ciphertext. Consequently, its main use is not to encrypt messages securely, but to temporarily hide information so that casual viewers do not get upset by it (risqué jokes, for example) or that they only want to see after they had made a choice to do so (for example, spoilers for an adventure game or movie).

[2] The cryptanalysis process aims to study cryptographic systems to identify weaknesses and information leaks.

List 1 shows code that implements all Caesar ciphers: The N parameter defines how many letters the algorithm should advance through the alphabet to encrypt each character in the plaintext. To decrypt, the routine executes the same code internally, but forces N to 26-N. So, for example, if you were to encrypt with N=3, the routine would be decrypted with N=23. ROT13 is easy to implement: N = 13.

```
procedure AACaesarCipher(aEncrypt : boolean; N : integer; aInStream : TStream;
  aOutStream : TStream);
var
  BytesRead : longint;
  i         : integer;
  Ch        : byte;
  Buf       : array [0..255] of byte;
begin
  {force N in range 0..25}
  N := N mod 26;
  if (N < 0) then
    inc(N, 26);
  if not aEncrypt then
    N := 26 - N;
  {read through the input stream in blocks, encrypt the block, and
   write it to the output stream--only convert A-Z and a-z}
  BytesRead := aInStream.Read(Buf, sizeof(Buf));
  while (BytesRead > 0) do begin
    for i := 0 to pred(BytesRead) do begin
      Ch := Buf[i];
      if ((ord('A') <= Ch) and (Ch <= ord('Z'))) then
        Buf[i] := ((Ch - ord('A') + N) mod 26) + ord('A')
      else if ((ord('a') <= Ch) and (Ch <= ord('z'))) then
        Buf[i] := ((Ch - ord('a') + N) mod 26) + ord('a')
    end;
    aOutStream.Write(Buf, BytesRead);
    BytesRead := aInStream.Read(Buf, sizeof(Buf));
  end;
end;
```

The Caesar cipher is an example of a more generalized monoalphabetic cipher. Here we still replace each plaintext letter with the same ciphertext letter, but now we don't rotate the alphabet to do so, we just randomly generate the translation table. All monoalphabetic ciphers are very insecure because they can all be attacked by the letter frequency method: calculating the frequencies of ciphertext letters.

In English, the letters E, T, A, I and N dominate, so the first thing to try would be to translate the most common ciphertext letter into an E and see what we got. We continue in this vein, experimenting and trying to recognize individual words until we have completely decoded the ciphertext. It must be noted that if we can detect individual words in the ciphertext, it makes deciphering much easier so generally the ciphertext is sent without spaces or punctuation marks.

3.5 Unattended minute

A variation of this is the Vigenère cipher. This algorithm is a poly alphabetic substitution cipher where we use several simple substitution ciphers sequentially. This cipher also requires a password or key, a word that will be used to determine the allowances required.

Suppose Alice wanted to encrypt the string 'This is a message' with the password 'SECRET'. First, she makes everything uppercase, and then removes all non-alphabetic characters. She writes down this condensed message with the password (repeatedly, if necessary) underneath:

THISISAMESSAGE SECRETSECRETSE

The replacement depends on the current letter of the password line. The first letter of the password line is S. She writes down the alphabet and then rotates below the alphabet so that S appears under A:

ABCDEF... TUVWXYZ
STUVWX... LMNOPQR

She can now encode the T in plaintext as an L. The next letter in the password row is E, so she generates a new conversion table in the same way:

ABCDEFGH... VWXYZ
EFGHIJKL... ZABCD

From this we see that the next ciphertext letter is L again. She continues like this until the entire plaintext is encoded. Bob, when receiving the message, performs the same type of method to decrypt the message.

To make things easier for both Alice and Bob, Vigenère ciphers usually use a 26 * 26 table of characters where each line has the alphabet moved to the left by 1 from the previous one.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
BCDEFGHIJKLMNOPQRSTUVWXYZA
CDEFGHIJKLMNOPQRSTUVWXYZAB
DEFGHIJKLMNOPQRSTUVWXYZABC
EFGHIJKLMNOPQRSTUVWXYZABCD
FGHIJKLMNOPQRSTUVWXYZABCDE
GHIJKLMNOPQRSTUVWXYZABCDEF
HIJKLMNOPQRSTUVWXYZABCDEFG
IJKLMNOPQRSTUVWXYZABCDEFGH
JKLMNOPQRSTUVWXYZABCDEFGHI
KLMNOPQRSTUVWXYZABCDEFGHIJ
LMNOPQRSTUVWXYZABCDEFGHIJK
MNOPQRSTUVWXYZABCDEFGHIJKL
NOPQRSTUVWXYZABCDEFGHIJKLM
OPQRSTUVWXYZABCDEFGHIJKLMN
PQRSTUVWXYZABCDEFGHIJKLMNO
QRSTUVWXYZABCDEFGHIJKLMNOP
RSTUVWXYZABCDEFGHIJKLMNOPQ
STUVWXYZABCDEFGHIJKLMNOPQR
TUVWXYZABCDEFGHIJKLMNOPQRS
UVWXYZABCDEFGHIJKLMNOPQRST
VWXYZABCDEFGHIJKLMNOPQRSTU
WXYZABCDEFGHIJKLMNOPQRSTUV
XYZABCDEFGHIJKLMNOPQRSTUVW
YZABCDEFGHIJKLMNOPQRSTUVWX
ZABCDEFGHIJKLMNOPQRSTUVWXY
```

Figure 1: Complete Vigenère Exchange Table

Figure 1 shows this table. Remember that this is in the days before computers: although this system is called Vigenère cipher, it was first designed by Giovan Batista Belaso in 1553; Blaise de Vigenère proposed a small change where the plaintext follows the password, instead of the password being repeated indefinitely.

To encode, you will find the password word letter along the left and the plaintext letter at the top and the point where the specified row and column cross is the ciphertext letter. To decode, find the password letter along the page, follow the line until you reach the ciphertext letter, and then locate the column at the top to find the plaintext letter.

And Eva? What does she do? She knows Alice and Bob are using a Vigenère cipher, so she gets to work. The first thing to note is that she must have a ciphertext that is much longer than the key, since the first thing she tries to do is figure out the length of the key that was used for encryption.

To do this, she takes the ciphertext and compares it with the ciphertext that has been rotated by 1, 2, 3, etc. lettering. She then counts the number of coinciding letters at each position for each rotation.

Due to the fact that the plaintext is written in English, and some plaintext letters are more common than others, she will find that rotations that are a multiple of the key length will have more coincidences than rotations that are not. This gives a good indication of the length of the key. From this, she has a good basis for cracking the rest of the message and finding out the key.

Suppose she finds that the key length was 6. This means that the letters 1, 7, 13, etc., were all encoded with the same Caesar cipher. The letters 2, 8, 14, etc., were encoded with another Caesar cipher, and so on. Using a table of the frequencies of the letters in the English language, [3] she can now make assumptions about the ciphertext letters by matching frequencies of ciphertext to English language frequencies. Without too much trouble, she should be able to figure out every Caesar cipher, and thus the password, and thus the plain text. Obviously for this to have any chance of success, the ciphertext must be several times the length of the key or password used to produce it.

```
procedure AAVigenereCipher(aEncrypt : boolean;
  aKey : string; aInStream : TStream; aOutStream : TStream);
var
  BytesRead : longint;
  i, j      : integer;
  Ch        : byte;
  Buf       : array [0..255] of byte;
  OutBuf    : array [0..255] of byte;
  KeyValues : array [0..255] of byte;
  KeyLen    : integer;
  KeyInx    : integer;
begin
  {the Vigenere cipher is for uppercase alphabetic letters
    only; in calculating the key values assume the key is in
    such a state}
  KeyLen := 0;
  for i := 1 to length(aKey) do
    if ('a' <= aKey[i]) and (aKey[i] <= 'z') then begin
      KeyValues[KeyLen] := ord(aKey[i]) - ord('a');
      inc(KeyLen);
    end
    else if ('A' <= aKey[i]) and (aKey[i] <= 'Z') then begin
      KeyValues[KeyLen] := ord(aKey[i]) - ord('A');
      inc(KeyLen);
    end;
  if not aEncrypt then
    for i := 0 to pred(KeyLen) do
      KeyValues[i] := 26 - KeyValues[i];
  {read through the input stream in blocks, encrypt the
    block, and write it to the output stream--only convert
    and write A-Z and a-z)
  KeyInx := 0;
  BytesRead := aInStream.Read(Buf, sizeof(Buf));
  j := 0;
  while (BytesRead > 0) do begin
    for i := 0 to pred(BytesRead) do begin
      Ch := Buf[i];
      if ((ord('A') <= Ch) and (Ch <= ord('Z'))) then begin
        OutBuf[j] := ((Ch - ord('A') + KeyValues[KeyInx])
          mod 26) + ord('A');
        inc(j);
        KeyInx := (KeyInx + 1) mod KeyLen;
      end
      else if ((ord('a') <= Ch) and (Ch <= ord('z'))) then
        begin
        OutBuf[j] := ((Ch - ord('a') + KeyValues[KeyInx])
          mod 26) + ord('A');
        inc(j);
        KeyInx := (KeyInx + 1) mod KeyLen;
      end;
    end;
    aOutStream.Write(OutBuf, j);
    BytesRead := aInStream.Read(Buf, sizeof(Buf));
    j := 0;
  end;
end;
```

[3] Of course, different languages have different frequencies on the letters. Sweden has, among other things: A,E,N,T, etc

Listing 2 is an implementation of the Vigenère cipher. There are four parameters in the routine: if to encrypt or decrypt, the key, the plaintext as a stream and finally the ciphertext, also as a stream.

3.6 I can't go for it

There is a variant of the Vigenère cipher used throughout the computer industry. It's easy to implement, and not particularly hard to break either. I'm talking about XOR ciphers.

Alice and Bob agree on a key. This key does not have to be purely alphabetic characters, in fact it is better if it is not, and the longer this key is, the better. To encrypt plaintext, Alice starts with the first byte of the message and the first byte of the key. She XOR them together and outputs the resulting byte as the first byte of the ciphertext. She advances a character and does the same. When she runs out of bytes in the key, she starts from the beginning of the key again. To decrypt, Bob does exactly the same thing, and the plaintext will be restored, since XOR of a byte twice with the same value results in the original byte.

```
procedure AAXORCipher(aKey : PByteArray; aKeyLen : integer; aInStream : TStream;
  aOutStream : TStream);
var
  Buf    : array [0..1023] of byte;
  KeyInx : integer;
  i      : integer;
  BytesRead : longint;
begin
  {read through the input stream in blocks, XOR the block with the key
   and write it to the output stream}
  if (aKey = nil) or (aKeyLen = 0) then
    raise Exception.Create('Cannot encrypt with XOR: the key is missing');
  KeyInx := 0;
  BytesRead := aInStream.Read(Buf, sizeof(Buf));
  while (BytesRead > 0) do begin
    for i := 0 to pred(BytesRead) do begin
      Buf[i] := Buf[i] xor aKey^[KeyInx];
      KeyInx := (KeyInx + 1) mod aKeyLen;
    end;
    aOutStream.Write(Buf, BytesRead);
    BytesRead := aInStream.Read(Buf, sizeof(Buf));
  end;
end;
```

The Listing 3 has this simple XOR cipher. Again, we pass in a key and two streams, one stream for plaintext and one for the ciphertext.

To crack this cipher, Eve proceeds in the same way as for the Vigenère cipher. She first tries to find coincidences between the ciphertext and the ciphertext that are displaced by different offsets. Those shifts that are multiples of the key length will have more coincidences than those that are not. From this we can derive the key length. Now we take the ciphertext and XOR it with itself offset by the key length. This will basically XOR text by itself and remove the key completely. From this we can then start applying some letter distributions to try to crack the XOR message.

This type of cipher is used in various applications everywhere. Perhaps the most serious use of XOR ciphers is with a Windows CE system: you set a password that is used to protect your portable device. When you turn on the device, you will be prompted to enter the password. Simple enough. Anyway, ActiveSync, the program you run so you can synchronize the data on your portable device with your desktop, uses the same password. When you start ActiveSync, you are prompted to enter your Windows CE password. You get the option to save the password on your desktop machine so you don't have to enter it every time you want to sync your machines. It saves the password, encrypted, in your registry.
However, the encryption it uses is a simple XOR cipher with 'susageP' as the key. Why this key? Well, the codename for Windows CE was 'Pegasus' and if you flip it over, you get 'susageP'

Actually, after realizing how bad the XOR cipher is, there is a way in which it can be used to produce an unbreakable cipher.

The unbreakable cipher is known as the one-timecode. (The Savings Bank uses this in so-called code cards) Classically, the one-timecode was a Vigenère cipher, but with a key that was completely random and as long as the plain text.

3.6.1 What happens is this:

Alice and Bob meet and create a block of pages (think "post-it blocks)", where each page has a sequence of letters on it randomly generated. Bcover is duplicated with Alice taking one and Bob taking the other. Now when Alice wants to send a message to Bob, she uses the standard Vigenère cipher, but where each letter in the plaintext is married to each letter consecutively from the top side of the post-it block. She continues to encrypt the plaintext and uses as many pages in the post-it block as required. When she's done, she sends the ciphertext and then destroys the pages she's used. Bob uses this block in the same way to decrypt the ciphertext and when he finishes he destroys the same pages. Since the key is completely random and also never used again (hence the one-timecodes), there is no way that Eva can get a grip on the original plaintext. She cannot use letter frequencies because the same letter is randomly encrypted with a different key letter each time. There is no attack to try to derive the key length, because the length of the key is the same as the length of the plaintext. Completely unbreakable.
In the XOR case, the same thing happens. A very large set of random bytes is generated and Alice and Bob both receive a copy. Alice encodes her plaintext by extracting out the same number of bytes from her random number table as there are bytes in the plaintext and then XORs them together. The ciphertext is sent and Alice destroys the random bytes she has used. Bob gets the ciphertext and uses the same number of random bytes as there are bytes in the ciphertext, destroying the used random bytes in the same way as Alice. Again, Eve has no chance to decrypt the ciphertext for the same reasons as in the classic case.

Keep in mind that if Alice and Bob created their random bytes by seeding one'slump number generator and then calling it to generate random bytes, Eve has a chance again. Since the entire sequence of random bytes generated by this method is deterministic (in other words, if Eve started with the same seed, she would get the same random bytes), she has a slot that she can open. All she has to do is try each seed sequentially, until the ciphertext is cracked. If you want, the disposable tablet suddenly becomes an encryption method with a 32-bit key and if Eve has a fast enough machine, she would be able to crack the ciphertext pretty quickly. Nevertheless, such randomized encryption algorithms are quite popular and there is quite a subset of the random number generator that are designed to be cryptographically secure.

3.7 Your imagination

After all this exhibition on substitution ciphers, we should take a look at transposition ciphers, the other classic method of plaintext encryption. A transpose cipher is one in which the letters in the plaintext are not converted in any way; It is only their order that is mixed up. The simplest of these ciphers is the columnar transposition cipher. Write the plain text on square paper, with one letter per box. Limit the width of the squared paper to a certain number of cells. Read the ciphertext one column at a time. So, for example, if the plain text was " ONCE MORE UNTO THE BREACH DEAR FRIENDS ", and our square paper was 10 boxes wide, we would get
ONCEMOREUN
TOTHEBREAC
HDEARFRIEN
DS
By reading the letters vertically, we get the cipher text 'OTHDNODS ...' To decrypt this message we must know the width of the squared paper. Count the letters in the ciphertext and divide this by the width to give you the number of letters to write vertically and so the ability to decode the ciphertext.
During World War I, the Germans designed the ADFGVX cipher, a mixture between a transposition cipher and a substitution cipher. It was a sophisticated encryption algorithm for its time. It is also a

"wordy" algorithm, since each letter of plaintext is converted into two letters of the final ciphertext. Create a 6 * 6 square, naming each row and column after the letters A, D, F, G, V, X, as in Figure 2.

Randomly put each letter of the alphabet in this square, along with the ten digits (36 characters in total). The first step is a substitution cipher: we replace each letter in the plain text with the row and column identifier depending on where the letter was found. For example, using Figure 2, the plaintext letter J would be replaced by DF, since J is at the intersection of row D and column F.

Now we need to decide on a password. Let's use the word SECRET. Discard all repeated letters in the password, so we have SECRT. Print the intermediate cipher text (the one consisting only of ADFGVX letter pairs) under the word SECRT as in the standard column transposition cipher. Here, of course, we have only five columns. Now we read the letters in the columns, not from the first to the last, but according to the position of the column heading in the alphabet. So for SECRT, we read the C column first, then the E column, the R column, the S column, and finally the T column. The result is a jump cap of As, Ds, Fs, Gs, Vs and Xs. To decrypt we need to know the password and also the replacement table.

To demonstrate the algorithm, Alice decides to send a message

THEYAREONTOYOU.

She performs the first substitution cipher and replaces T with FD, H with AD, and so on. She places the resulting letter pairs in a 5-column incorporation table under the word SECRT, as in Figure 3. She then reads the C column first: AXGFV, E column: DVGAX and so on, to make the cipher text AXGFVDVGAX ...

Bob can reconstruct the transposition table because he knows the password, and because he also knows the replacement table, he can figure out the original plaintext. And poor Eve? Well, there seems to be no rhyme or reason for the collection of these 6 letters, repeated and mixed seemingly randomly. But surprisingly, this cipher was actually broken with the fact that the two steps are completely disconnected from each other and the transfer password is not used in the substitution.

Code? Well, it's very messy, as you can imagine. List 4 shows the entire ADFGVX cipher code and as you can see, we cannot divide much code between encryption and decryption as in the previous three ciphers. This month's disk also has a routine for generating a random mix of the letters and numbers for the replacement table.

```
procedure AAADFGVXCipher(aEncrypt : boolean; aKey : string;
  const aSubstTable: TaaADFGVXTable; aInStream : TStream;
  aOutStream : TStream);
const
  ADFGVX : array [0..5] of char = 'ADFGVX';
var
  BytesRead : longint;
  i, j      : integer;
  Ch        : char;
  Buf       : array [0..255] of char;
  DblBuf    : array [0..511] of char;
  CleanKey  : string;
  KeyLen    : integer;
  KeyInx    : integer;
  PosCh     : integer;
  MinInx    : integer;
  ColLen    : integer;
  Row, Col  : integer;
  SubstTextSize : longint;
  InStreamSize  : longint;
  MemStream     : TMemoryStream;
begin
  {clean up the key so that it consists only of unique
    uppercase characters}
  CleanKey := '';
  for i := 1 to length(aKey) do begin
    Ch := upcase(aKey[i]);
    if ('A' <= Ch) and (Ch <= 'Z') then
      if (Pos(Ch, CleanKey) = 0) then
        CleanKey := CleanKey + Ch;
  end;
  {===ENCRYPTION===}
  if aEncrypt then begin
    {read the entire input stream, converting into
      letterpairs into a temporary stream}
    MemStream := TMemoryStream.Create;
    try
      BytesRead := aInStream.Read(Buf, sizeof(Buf));
      while (BytesRead > 0) do begin
        j := 0;
        for i := 0 to pred(BytesRead) do begin
          Ch := upcase(Buf[i]);
          if (('A' <= Ch) and (Ch <= 'Z')) or
             (('0' <= Ch) and (Ch <= '9')) then begin
            PosCh := Pos(Ch, aSubstTable) - 1;
            DblBuf[j] := ADFGVX[PosCh div 6];
            DblBuf[j+1] := ADFGVX[PosCh mod 6];
            inc(j, 2);
          end;
        end;
        MemStream.Write(DblBuf, j);
        BytesRead := aInStream.Read(Buf, sizeof(Buf));
      end;
      {now read the letters in each column according to the
        order of the letters in the cleaned key}
      KeyLen := length(CleanKey);
      for KeyInx := 1 to KeyLen do begin
        {find the smallest letter in the key, this is the
          column we'll be reading next}
        MinInx := 1;
        for i := 2 to KeyLen do
          if (CleanKey[i] < CleanKey[MinInx]) then
            MinInx := i;
        CleanKey[MinInx] := #127; {so we don't see it again}
        dec(MinInx); {it's easier with a 0-based number}
        {starting off with the MinInx'th letter in the
          temporary stream, copy it and every KeyLen'th
          letter after that to the output stream}
        SubstTextSize := MemStream.Size;
        PosCh := MinInx;
        j := 0;
        while (PosCh < SubstTextSize) do begin
          MemStream.Position := PosCh;
          inc(PosCh, KeyLen);
          MemStream.Read(DblBuf[j], 1);
          inc(j);
          if (j = sizeof(DblBuf)) then begin
            aOutStream.Write(DblBuf, sizeof(DblBuf));
            j := 0;
          end;
        end;
        if (j > 0) then
          aOutStream.Write(DblBuf, j);
      end;
    finally
      MemStream.Free;
    end;
  end else begin
  {===DECRYPTION===}
    {first create memory stream to use as an intermediary,
     set its size to the size of the input stream}
    InStreamSize := aInStream.Size;
    MemStream := TMemoryStream.Create;
    try
      MemStream.SetSize(InStreamSize);
      {now read the letters in each column according to the
        order of the letters in the cleaned key}
      KeyLen := length(CleanKey);
      for KeyInx := 1 to KeyLen do begin
        {find the smallest letter in the key, this is the
          column we'll be reading next}
        MinInx := 1;
        for i := 2 to KeyLen do
          if (CleanKey[i] < CleanKey[MinInx]) then
            MinInx := i;
        CleanKey[MinInx] := #127; {so we don't see it again}
        dec(MinInx); {it's easier with a 0-based number}
        {calculate the length of this column}
        ColLen := InStreamSize div KeyLen;
        if ((InStreamSize-(ColLen*KeyLen)) > MinInx) then
          inc(ColLen);
        {copy the column from the input stream to the
          temporary stream, starting off by copying to the
          MinInx'th letter in the temporary stream, and
          every KeyLen'th letter after that; we stop at the
          end of the column}
        PosCh := MinInx;
        while (ColLen > 0) do begin
          if (ColLen > sizeof(Buf)) then
            BytesRead := aInStream.Read(Buf, sizeof(Buf))
          else
            BytesRead := aInStream.Read(Buf, ColLen);
          dec(ColLen, BytesRead);
          for i := 0 to pred(BytesRead) do begin
            MemStream.Position := PosCh;
            inc(PosCh, KeyLen);
            MemStream.Write(Buf[i], 1);
          end;
        end;
      end;
      {now read the temporary stream as letter pairs,
        converting them into the original characters for the
        output stream}
      MemStream.Position := 0;
      BytesRead := MemStream.Read(DblBuf, sizeof(DblBuf));
      j := 0;
      Row := 0;
      Col := 0;
      while (BytesRead > 0) do begin
        for i := 0 to pred(BytesRead) do begin
          if not Odd(i) then begin
            case DblBuf[i] of
              'A' : Row := 0;
              'D' : Row := 1;
              'F' : Row := 2;
              'G' : Row := 3;
              'V' : Row := 4;
              'X' : Row := 5;
            end;
            case DblBuf[i+1] of
              'A' : Col := 0;
              'D' : Col := 1;
              'F' : Col := 2;
              'G' : Col := 3;
              'V' : Col := 4;
              'X' : Col := 5;
            end;
            Buf[j] := aSubstTable[Row * 6 + Col + 1];
            inc(j);
            if (j = sizeof(Buf)) then begin
              aOutStream.Write(Buf, sizeof(Buf));
              j := 0;
            end;
          end;
        end;
        if (j > 0) then
          aOutStream.Write(Buf, j);
        BytesRead := MemStream.Read(DblBuf, sizeof(DblBuf));
      end;
    finally
      MemStream.Free;
    end;
  end;
end;
```

Code listing2

3.8 Looking for a good sign

As a final note in this article about classic ciphers, I would like to mention the ENIGMA machine. This was the encryption device used by the German army and navy during World War II. I would like to discuss this quickly for two reasons. First, it was a very sophisticated encryption machine. And secondly, the mining of its encryption fell to a team of mathematicians in Bletchley Park in Buckinghamshire, including one of the fathers of modern computing, Alan Turing. The ENIGMA machine was essentially a three-rotor machine. A rotor is a wheel that is coupled to perform a monoalphabetic substitution. The wheel was divided into 26 segments, one for each letter, and was designed in such a way that each segment of the face was connected to a different segment on the back. If each segment of the back was connected to a light, applying electric current to a segment of the face would cause the light for the encrypted letter to light up. The ENIGMA machine connected its three rotors in series, so each rotor applied a substitution. To ensure that the same substitution was not used over and over again, the rotors would be rotated by a number of segments after one letter was encrypted. In addition, the ENIGMA machine used the concept of a reflective rotor that caused the circuit to reflect through the three rotors again, backwards, doubly encrypting each

plaintext letter. There was a keyboard through which the plaintext could be typed for encryption (or the ciphertext for decryption), the action of pressing a key would perform the rotations, and the resulting ciphertext letter was read by a set of 26 lights as current was applied through the rotors. To make it even harder to crack, the initial position of the rotors can be defined before encrypting or decrypting the message. There was also a a merge table as if mapped with letter pairs before encryption. All in all, a terrifying encryption machine.

Breaking the ENIGMA encryption relied on complex mathematics from the most brilliant mathematicians in England, and also a lot of luck. The Allies had managed to get an ENIGMA machine and some of the codebooks (these books detailed which rotors would be used on which days, the initial positions of the rotors, and so on), and they had a stroke of luck in that the operators disliked changing rotors (apparently a tedious process), so one of the encryption possibilities was denied. Nevertheless, the British team, including Alan Turing, managed to break the encryption by building a machine called The Bombe, which helped analyze the ciphertext. Given the importance of Turing's work during the war, his subsequent treatment by the British government in 1952, when he was arrested for a homosexual liaison and stripped of his security clearance, was nothing short of scandalous and probably led to his death by suicide in 1954.