Chapter 1 - Introduction and Classical Ciphers

Cryptography: is the scientific study of techniques for securing digital information, transactions, and distributed computations.

☒ Today, cryptography is everywhere!

- Cash machines, money transfer between banks
- Electronic cash, online banking, secure email
- Satellite TV, pay-per-view TV
- Immobilizer systems in cars
- Digital Rights Management (DRM), Cloud

☒ Cryptography has gone from an *art* form that dealt with secret communication for the military to a *science* that helps to secure systems for ordinary people all across the globe.

☒ This also means that cryptography is becoming a more and more central topic within computer science.

Cryptographic goals

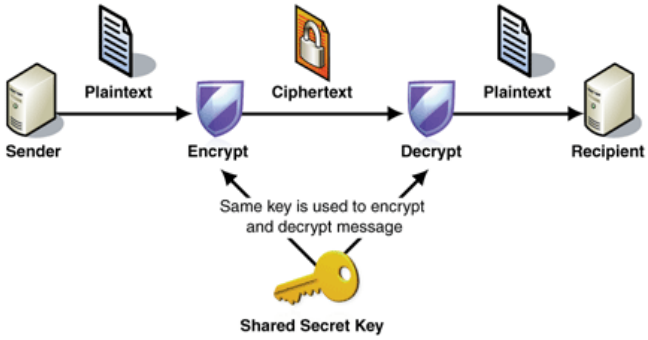
1. *Confidentiality*: Information can be made effectively unavailable or unreadable for unauthorized individuals, entities, and processes.
2. *Authentication*: The receiver of a message can verify the identity of the sender.
3. *Integrity*: Integrity ensures that data has not been altered or destroyed in an unauthorized manner.
4. *Non-Repudiation:* The receiver can prove that the message he or she received is precisely what the sender sent; the sender will have no means to deny any part of his or her participation.

Glossary

- **Plaintext:** An original message.
- **Ciphertext** : the coded message
- **Enciphering or encryption**: the process of converting from plaintext to ciphertext.
- **Deciphering or decryption** : restoring the plaintext from the ciphertext.
- **Cryptography**: encryption + decryption.
- **Cryptanalysis** :techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of.
  - Cryptanalysis is what the layperson calls "breaking the code."
- **Cryptology** : cryptography +cryptanalysis.

The Setting of Private-Key Encryption

- In the *private-key* setting, two parties share some secret information called a *key*, and use this key when they wish to communicate secretly with each other.
- A party sending a message uses the key to *encrypt* (or "scramble") the message before it is sent, and the receiver uses the same key to *decrypt* (or "unscramble") and recover the message upon receipt.
- The message itself is often called the *plaintext*, and the "scrambled" information that is actually transmitted from the sender to the receiver is called the *ciphertext*.



The syntax of encryption

A private-key encryption scheme, or cipher, is comprised of *three* **algorithms**:

1. *The key-generation algorithm* **Gen** is a *probabilistic* algorithm that outputs a key k chosen according to some distribution that is determined by the scheme.

2. *The encryption algorithm* **Enc** takes as input a key k and a plaintext m and outputs a ciphertext c. We denote the encryption of the plaintext m using the key k by $\mathbf{Enc}_k(m)$.

3. *The decryption algorithm* **Dec** takes as input a key k and a ciphertext c and outputs a plaintext m. We denote the decryption of the ciphertext c using the key k by $\mathbf{Dec}_k(c)$.

- The procedure for generating keys defines a *keyspace* $\mathcal{K}$ (i.e., the set of all possible keys).
- The encryption scheme is defined over some set of possible plaintext messages denoted $\mathcal{M}$ and called the *plaintext (or message) space*.
- Since any ciphertext is obtained by encrypting some plaintext under some key, $\mathcal{K}$ and $\mathcal{M}$ define a set of all possible ciphertexts that we denote by $\mathcal{C}$.
- Note that an encryption scheme is fully defined by specifying the three algorithms (**Gen, Enc, Dec**) and the plaintext space $\mathcal{M}$.
- The basic **correctness requirement** of any encryption scheme is that for every key k output by **Gen** and every plaintext message $m \in \mathcal{M}$, it holds that

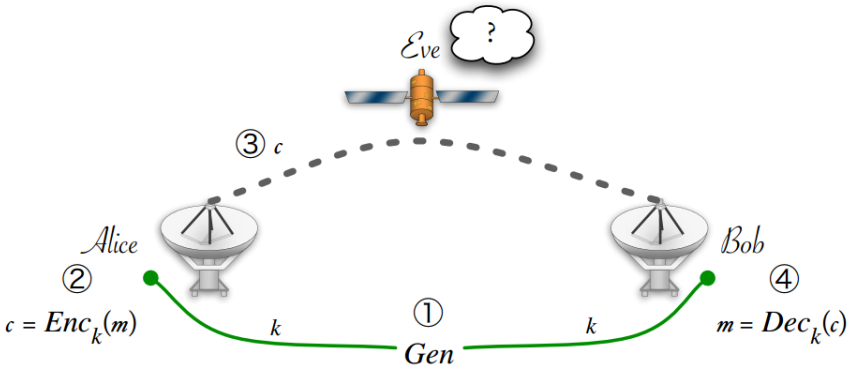
$$\mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m.$$

An encryption scheme would be used by two parties (Alice and Bob) who wish to **communicate** as follows.

1- **Gen**is run to obtain a key *k* that the parties share.
2- When one party wants to send a plaintext *m*to the other, he would compute

$$c := \mathbf{Enc}_k(m) \text{ and}$$

3- Send the resulting ciphertext *c* over the public channel to the other party.
4- Upon receiving *c*, the other party computes $m := \mathbf{Dec}_k(c)$ to recover the original plaintext.



Keys and Kerckhoffs' principle.

In the late 19th century, Auguste **Kerckhoffs** gave an encryption: "*the encryption scheme itself should not be kept secret, and so only the key should constitute the secret information shared by the communicating parties*".

- Some of the *advantages* of "open cryptographic design", where the algorithm specifications are made public, include:

1. Published designs undergo public scrutiny and are therefore likely to be stronger. Many years of experience have demonstrated that it is very difficult to construct good cryptographic schemes.
2. It is better that security flaws are revealed by "*ethical hackers*" and made public, than having the flaws be known only to *malicious parties*.
3. If the security of the system relies on the secrecy of the algorithm, then reverse engineering of the code (or leakage by industrial espionage) poses a serious threat to security. This is in contrast to the secret key which is not part of the code, and so is not vulnerable to reverse engineering.
4. Public design enables the establishment of standards.

Cryptanalysis and Brute-Force Attack

There are two general approaches to attacking a conventional encryption scheme:

1- Cryptanalysis: Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext–ciphertext pairs. This type of attack exploits the

characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

2- **Brute-force attack**: The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

Table 2.2 Average Time Required for Exhaustive Key Search

Key Size (bits)	Number of Alternative Keys	Time Required at 1 Decryption/μs	Time Required at 10^6 Decryptions/μs
32	$2^{32} = 4.3 \times 10^9$	$2^{31}\mu s = 35.8$ minutes	2.15 milliseconds
56	$2^{56} = 7.2 \times 10^{16}$	$2^{55}\mu s = 1142$ years	10.01 hours
128	$2^{128} = 3.4 \times 10^{38}$	$2^{127}\mu s = 5.4 \times 10^{24}$ years	5.4×10^{18} years
168	$2^{168} = 3.7 \times 10^{50}$	$2^{167}\mu s = 5.9 \times 10^{36}$ years	5.9×10^{30} years
26 characters (permutation)	$26! = 4 \times 10^{26}$	$2 \times 10^{26}\mu s = 6.4 \times 10^{12}$ years	6.4×10^6 years

Attack scenarios

- ***Ciphertext-only attack***: This is the most basic type of attack and refers to the scenario where the adversary just observes a ciphertext C^* and attempts to determine the plaintext that was encrypted.
- ***Known-plaintext attack***: Here, the adversary learns one or more pairs of plaintexts/ciphertexts encrypted under the same key $(m, enc_k(m))$. The aim of the adversary is then to determine the plaintext that was encrypted to give some other ciphertext C^* (for which it does not know the corresponding plaintext).
- ***Chosen-plaintext attack***: In this attack, the adversary has the ability to obtain the **encryption** of any plaintext(s) $(m, enc_k(m))$ *of its choice*. It then attempts to determine the plaintext that was encrypted to give some other ciphertext C^*.
- ***Chosen-ciphertext attack***: The final type of attack is one where the adversary is even given the capability to obtain the **decryption** of any ciphertext(s) $(c, dec_k(c))$ of its choice. The adversary's aim, once again, is then to determine the plaintext that was encrypted to give some other ciphertext C^* (whose decryption the adversary is unable to obtain directly).

  - Note that the first two types of attacks are *passive* in that the adversary just receives some ciphertexts (and possibly some corresponding plaintexts as well) and then launches its attack.
  - In contrast, the last two types of attacks are *active* in that the adversary can adaptively ask for encryptions and/or decryptions of its choice.

Security Levels:

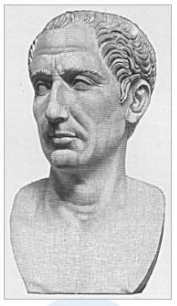
1- **Perfect security (strong):** the ciphertext *reveals absolutely nothing* about the plaintext even to an adversary with unlimited computational power. Example: vernam cipher
2- **Computational security** (**weak**): the level of most *modern* cryptographic constructions, where they can be broken given *enough time* and *computation*.

- The amount of computation needed to break these encryption schemes would take more than many lifetimes to carry out even using the fastest available supercomputers.
- For all practical purposes, this level of security *suffices*.
- All modern cryptographic constructions depend on their security on a hard problems (**assumptions**), so if we can solve the assumption, we can break the construction.
- *Example*: RSA depend on factorization problem.

Historical Ciphers and Their Cryptanalysis

The two basic building blocks of all encryption techniques are

- **Substitution** (substitute the letters with another ones)
- **Transposition** (transpose the letters without new ones).
- We can combine between them



Caesar's cipher.

- Julius Caesar encrypted by rotating the letters of the alphabet by 3 places: a was replaced with D, b with E, and so on.
- Of course, at the end of the alphabet, the letters wrap around and so x was replaced with A, y with B and z with C.
- Example:

```
plain:  meet me after the toga party
cipher: PHHW PH DIWHU WKH WRJD SDUWB
```

- Let us assign a numerical equivalent to each letter:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

- Then the algorithm can be expressed as follows. For each plaintext letter, substitute the ciphertext letter C:

$$C = E(3, p) = (p + 3) \bmod 26$$

- An immediate problem with this cipher is that the method is *fixed*: the key-generation algorithm **Gen** is trivial (that it, it does nothing) and there is no secret key to speak of.
- Interestingly, a variant of this cipher called **ROT-13** (where the shift is 13 places instead of 3) is widely used in various online forums. It is understood that this does not provide any cryptographic security.

The shift cipher and the sufficient key space principle.

- The shift cipher is similar to Caesar's cipher, but a secret key is introduced. Specifically, the shift cipher uses as the key k a number between 0 and 25.
- To encrypt, letters are rotated (as in Caesar's cipher) but by k places.
- Algorithm *Gen* outputs a random number k in the set {0, …, 25};
- Algorithm **Enc** takes a key k and a plaintext written using English letters and shifts each letter of the plaintext forward k positions (wrapping around from z to a);
- Algorithm **Dec** takes a key k and a ciphertext written using English letters and shifts every letter of the cipher text backward k positions (this time wrapping around from a to z).

- Encryption of a plaintext character *mi* with the key *k* gives the ciphertext character [*(mi+k) mod 26*], and decryption of a cipher text character *ci* is defined by [*(ci-k) mod 26*].

- Is the shift cipher secure? try to decrypt the following message that was encrypted using the shift cipher and a secret key *k*(whose value we will not reveal): OVDTHUFWVZZPISLRLFZHYLAOLYL

- Is it possible to decrypt this message without knowing *k*? Actually, it is completely trivial! The reason is that there are only 26 possible keys. Thus, it is easy to try every key, and see which key decrypts the ciphertext into a plaintext that "makes sense". Such an attack on an encryption scheme is called a *brute-force attack*.

- *Example*: the following figure shows the **brute force attack** on shift cipher encryption.

KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

- Clearly, any secure encryption scheme must not be vulnerable to such a brute-force attack; otherwise, it can be completely broken, irrespective of how sophisticated the encryption algorithm is.
- This brings us to a trivial, yet important, principle called the "*sufficient key space principle": Any secure encryption scheme must have a key space that is not vulnerable to exhaustive search*.
- For example the key space should be at least 2^{60} or 2^{70}.
- We emphasize that the above principle gives a **necessary** condition for security, not a **sufficient** one.
- In fact, we will see next an encryption scheme that has a very large key space but which is still insecure.

Mono-Alphabetic substitution.

- Mono-Alphabetic Substitution is simply a **permutation** of all the letters of the alphabet. In other words, each letter in the alphabet of the message space is mapped to a unique letter in the alphabet.
- The key space thus consists of all permutations of the alphabet, meaning that the size of the key space is 26! (or approximately 2^{88}) if we are working with the English alphabet. As an example, the **key**

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

in which *a* maps to *X*, etc., would encrypt the message "tellhimaboutme" to GDOOKVCXEFLGCD.

- A brute force attack on the key space for this cipher takes much longer than a lifetime, even using the most powerful computer known today.
- However, this *does not* necessarily mean that the cipher is secure.
- It is then possible to attack the mono-alphabetic substitution cipher *by utilizing statistical patterns* of the English language (of course, the same attack works for any language).
- The two properties of this cipher that are utilized in the attack are as follows:

1. In this cipher, the *mapping* of each letter is *fixed*, and so if *e* is mapped to *D*, then every appearance of *e* in the plaintext will result in the appearance of *D* in the ciphertext.

2. The probability distribution of individual letters in the English (or any other) language is known. That is, the average frequency counts of the different English letters are quite invariant over different texts. Of course, the longer the text, the closer the frequency counts will be to the average. However, even relatively short texts (consisting of only tens of words) have distributions that are "close enough" to the average. Specifically, since *e* is the most frequent letter in English, we will guess that the most frequent character in the ciphertext corresponds to the plaintext character *e*, and so on.

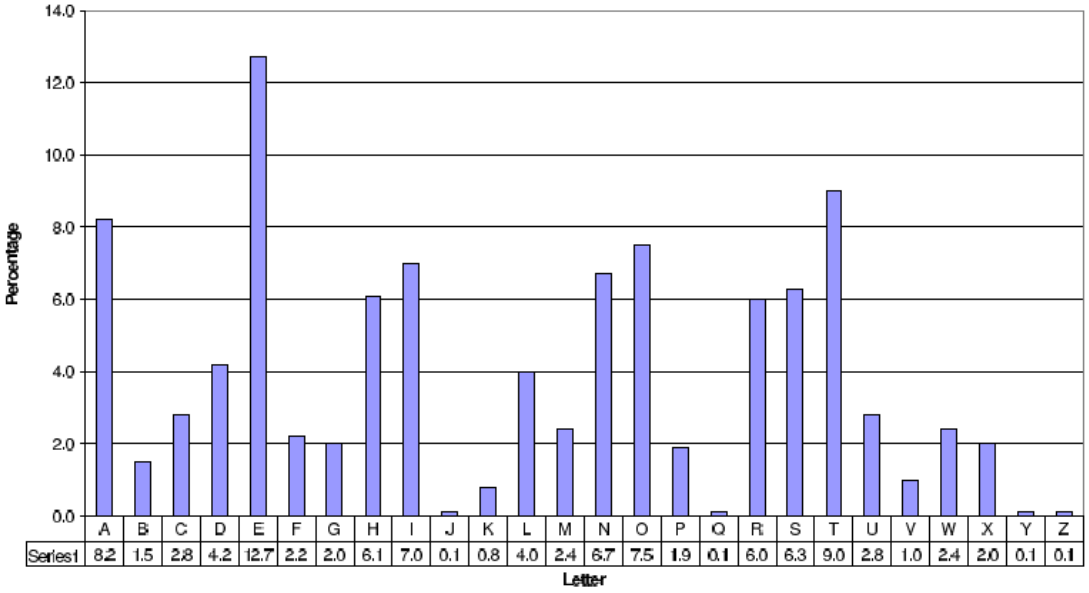


FIGURE 1.2: Average letter frequencies in the English language

- There are two approaches to hide the frequencies of the plaintext:
  - One approach is to encrypt **multiple letters** of plaintext,
  - and the other is to use **multiple cipher** alphabets.

Playfair Cipher

- The best-known *multiple-letter* encryption cipher is the Playfair, which treats *digrams* in the plaintext as single units and translates these units into ciphertext digrams.
- The Playfair algorithm is based on the use of a 5 × 5 matrix of letters constructed using a key.
- *Example*: Suppose the key is: ***monarchy***

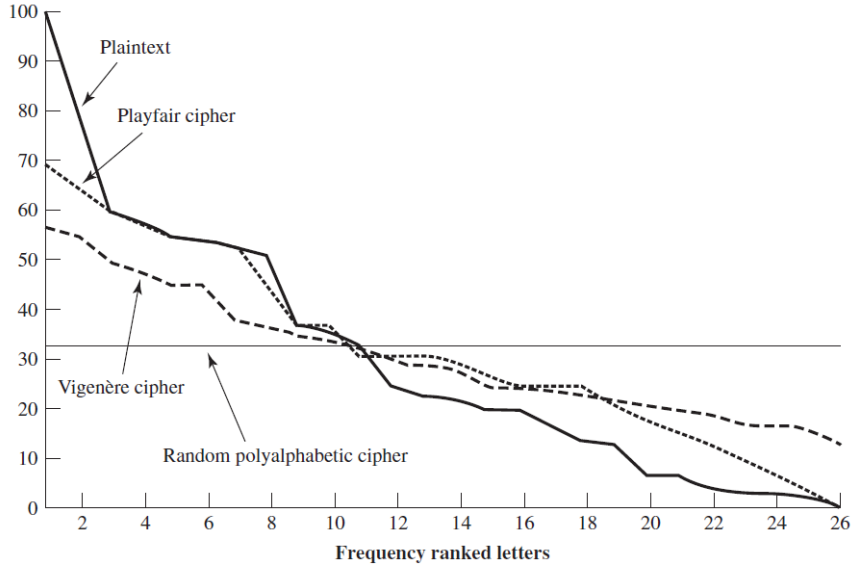
M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

- The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order.
- The letters I and J count as one letter.
- Plaintext is *encrypted* two letters at a time, according to the following **rules**:

1- Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that *balloon* would be treated as *ba lx lo on*.
2- Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, *ar* is encrypted as *RM*.

3- Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, *mu* is encrypted as CM.
4- Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

- **Example**: cipher text of " attack today" by key" monarchy" is : RS SR DE PR BR SZ
- The Playfair cipher is a great advance over simple monoalphabetic ciphers. For one thing, whereas there are only 26 letters, there are 26 × 26 = 676 digrams, so that identification of individual digrams is more difficult.



- As the figure shows, the Playfair cipher has a flatter distribution than does plaintext, but nevertheless, it reveals plenty of structure for a cryptanalyst to work with.

Hill Cipher

- Another interesting *multi-letter* cipher based on linear algebra. It is invented by Lester S. Hill in 1929.
- Each letter is represented by a number modulo 26. Often the simple scheme A = 0, B = 1, ..., Z = 25 is used.
- To encrypt a message, each block of n letters (considered as an n component vector) is multiplied by an ***invertible*** n × n matrix, against modulus 26.
- Not every matrix has inverse.
- To decrypt the message, each block is multiplied by the *inverse* of the matrix used for encryption.
- The matrix used for encryption is the cipher ***key***, and it should be chosen randomly from the set of invertible n × n matrices (modulo 26).
- Example: Consider the message 'ACT', and the **key** : GYBNQKURP.

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}$$

Since 'A' is 0, 'C' is 2 and 'T' is 19, the message is the vector:

$$\begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix}$$

Thus the enciphered vector is given by:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} = \begin{pmatrix} 67 \\ 222 \\ 319 \end{pmatrix} \equiv \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \pmod{26}$$

which corresponds to a ciphertext of 'POH'. Now, suppose that our message is instead 'CAT', or:

$$\begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix}$$

This time, the enciphered vector is given by:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \\ 19 \end{pmatrix} \equiv \begin{pmatrix} 31 \\ 216 \\ 325 \end{pmatrix} \equiv \begin{pmatrix} 5 \\ 8 \\ 13 \end{pmatrix} \pmod{26}$$

- In order to decrypt, we turn the ciphertext back into a vector, then simply multiply by the *inverse* matrix of the key matrix (IFKVIVVMI in letters). We find that, modulo 26, the inverse of the matrix used in the previous example is:

$$\begin{pmatrix} 6 & 24 & 1 \\ 13 & 16 & 10 \\ 20 & 17 & 15 \end{pmatrix}^{-1} \equiv \begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \pmod{26}$$

Taking the previous example ciphertext of 'POH', we get:

$$\begin{pmatrix} 8 & 5 & 10 \\ 21 & 8 & 21 \\ 21 & 12 & 8 \end{pmatrix} \begin{pmatrix} 15 \\ 14 \\ 7 \end{pmatrix} \equiv \begin{pmatrix} 260 \\ 574 \\ 539 \end{pmatrix} \equiv \begin{pmatrix} 0 \\ 2 \\ 19 \end{pmatrix} \pmod{26}$$

which gets us back to 'ACT', just as we hoped.

Invertible Matrix

We define the inverse of a square matrix M^{-1} by the equation $MM^{-1}=I$, where I is the identity matrix. I is a square matrix that is all zeros except for ones along the main diagonal from upper left to lower right.

Example:

$$\mathbf{A} = \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} \qquad \mathbf{A}^{-1} \bmod 26 = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

$$\mathbf{A}\mathbf{A}^{-1} = \begin{pmatrix} (5 \times 9) + (8 \times 1) & (5 \times 2) + (8 \times 15) \\ (17 \times 9) + (3 \times 1) & (17 \times 2) + (3 \times 15) \end{pmatrix}$$
$$= \begin{pmatrix} 53 & 130 \\ 156 & 79 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- Given a matrix A, its inverse is given by

$$A^{-1} = \frac{1}{\det(A)} \operatorname{adj}(A)$$

- where det(A),|A|, is the **determinant** of A, and adj(A) is the **adjoint** of A.
- The following example show how to compute the inverse of a square matrix A.

$$\begin{pmatrix} 0 & -3 & -2 \\ 1 & -4 & -2 \\ -3 & 4 & 1 \end{pmatrix}$$

Step 1: replace every entry by its minor

$$\begin{pmatrix} (-4)\times 1-(-2)\times 4 & 1\times 1-(-2)\times(-3) & 1\times 4-(-4)\times(-3) \\ (-3)\times 1-(-2)\times 4 & 0\times 1-(-2)\times(-3) & 0\times 4-(-3)\times(-3) \\ (-3)\times(-2)-(-2)\times(-4) & 0\times(-2)-(-2)\times 1 & 0\times(-4)-(-3)\times 1 \end{pmatrix} = \begin{pmatrix} 4 & -5 & -8 \\ 5 & -6 & -9 \\ -2 & 2 & 3 \end{pmatrix}.$$

Step 2: change some of the signs

We now change the signs of some of the minors, according to the pattern

$$\begin{pmatrix} + & - & + \\ - & + & - \\ + & - & + \end{pmatrix},$$

thus creating what's called the **matrix of cofactors**. In our case, this is

$$\begin{pmatrix} 4 & 5 & -8 \\ -5 & -6 & 9 \\ -2 & -2 & 3 \end{pmatrix}.$$

Step 3: transpose

$$\begin{pmatrix} 4 & -5 & -2 \\ 5 & -6 & -2 \\ -8 & 9 & 3 \end{pmatrix}.$$

Step 4: divide by the determinant

Finally, we divide by the determinant of the original matrix. In our case, the determinant is

$$\det\begin{pmatrix} 0 & -3 & -2 \\ 1 & -4 & -2 \\ -3 & 4 & 1 \end{pmatrix} = 0\times\det\begin{pmatrix} -4 & -2 \\ 4 & 1 \end{pmatrix} + 3\times\det\begin{pmatrix} 1 & -2 \\ -3 & 1 \end{pmatrix} - 2\times\det\begin{pmatrix} 1 & -4 \\ -3 & 4 \end{pmatrix} = 1,$$

so the inverse is simply

$$A^{-1} = \begin{pmatrix} 4 & -5 & -2 \\ 5 & -6 & -2 \\ -8 & 9 & 3 \end{pmatrix}.$$

- The basic Hill cipher is vulnerable to a *known-plaintext* attack because it is completely linear. An opponent who intercepts *n* plaintext- ciphertext character pairs can set up a linear system which can (usually) be easily solved;

The Vigenere (poly-alphabetic shift) cipher.

- The attack on the mono-alphabetic substitution cipher can be thwarted by mapping different instances of the same plaintext character to different ciphertext characters.
- This has the effect of "smoothing out" the probability distribution of characters in the ciphertext.
- For example, consider the case that *e* is sometimes mapped to G, sometimes to P, and sometimes to Y.
- Then, the ciphertext letters G, P, and Y will most likely not stand out as more frequent, because other less-frequent characters will be also be mapped to them. Thus, counting the character frequencies will not offer much information about the mapping.
- The Vigenere cipher works by applying *multiple shift ciphers* in sequence.
- An encryption of the message "tellhimaboutme" using the key "cafe" would work as follows:

Plaintext:	`tellhimaboutme`
Key:	`cafecafecafeca`
Ciphertext:	`WFRQKJSFEPAYPF`

- This is exactly the same as encrypting the first, fifth, ninth, and so on characters with the shift cipher and key $k = 3$, the second, sixth, tenth, and so on characters with key $k = 1$, the third, seventh, and so on characters with $k = 6$ and the fourth, eighth, and so on characters with $k = 5$.
- Thus, it is a repeated shift cipher using different keys.
- Notice that in the above example l is mapped once to R and once to Q. Furthermore, the ciphertext character F is sometimes obtained from *e* and sometimes from *a*.
- Thus, the character frequencies in the ciphertext are "***smoothed***", as desired.

-	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
...	...	...																								

Breaking the Vigenere cipher.

- If the length of the key is known *t*, then the task is relatively easy.
- Divide the ciphertext into *t* parts. Let *k = k1, …,k t* be the key, and let *c1, c2,* …be the ciphertext characters.
- Then, for every j ($1 \le j \le t$), the set of characters *cj, cj+t, cj+2t,* … are encrypted by *kj* key.
- What remains is to check which of the 26 possible keys is the correct one, for each *j*.
- Checking all possible keys would require a brute force search through 26^t different possible keys.
- Nevertheless, we can still use the statistical attack method described earlier.
- That is, for every set of the ciphertext characters relating to a given key (that is, a given value of *j*), it is possible to build the frequency table of the characters and then check which of the 26 possible shifts gives the "right" probability distribution.
- Since this can be carried out separately for each key, the attack can be carried out very quickly; all that is required is to build *t* frequency tables (one for each of the subsets of the characters) and compare them to the real probability distribution.

Vernam cipher (one-time pad)

- This method provide a **perfect security**, which reveal **absolutely nothing** to the attacker.
- It choose a keyword that is *as long as* the plaintext and has no statistical relationship to it, and works on binary data (bits) rather than letters.
- The system can be expressed as follows:

$$c_i = p_i \oplus k_i$$

where

p_i = ith binary digit of plaintext
k_i = ith binary digit of key
c_i = ith binary digit of ciphertext
$\oplus$ = exclusive-or (XOR) operation

- Decryption simply involves the same bitwise operation:

$$p_i = c_i \oplus k_i$$

One-time pad encryption drawbacks

1- The key is required to be as *long as the message*.
2- One time pad encryption isonly "secure" if used *once* (with the same key).
- If two messages m, $m_0$are encrypted using the same key k. An adversary who obtains $c = m \oplus k$and $c_0 = m_0 \oplus k$can compute: $c \oplus c_0 = m \oplus m_0$.

3- The one-time pad encryption scheme is only secure against a *ciphertext-only attack*.
- An adversary who obtains the encryption $c$of a known message $m$can compute the key $k = c \oplus m$and then decrypt any subsequent ciphertexts computed using this same key.

TRANSPOSITION TECHNIQUES

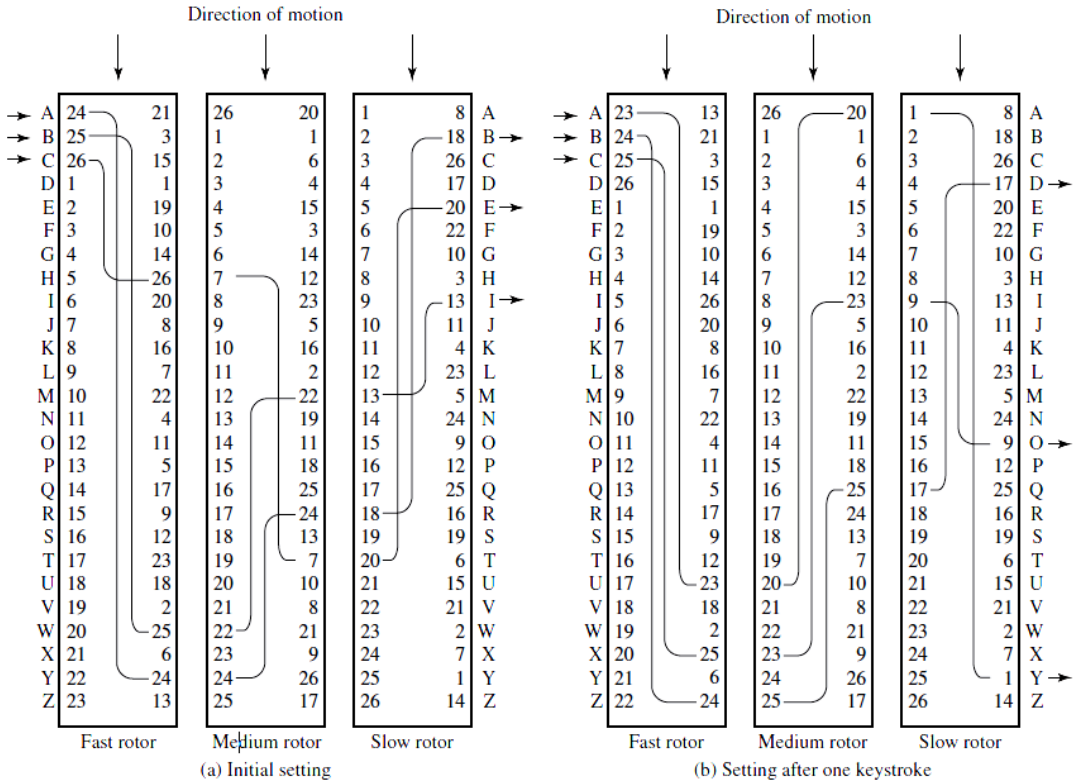
One scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the *key* to the algorithm.
Example:

```
Key:          4 3 1 2 5 6 7
Plaintext:    a t t a c k p
              o s t p o n e
              d u n t i l t
              w o a m x y z
Ciphertext:   TTNAAPTMTSUOAODWCOIXKNLYPETZ
```

- We can permute the ciphertext again with the same key.

ROTOR MACHINES

- The machine consists of a set of independently rotating cylinders through which electrical pulses can flow.
- Each cylinder has 26 input pins and 26 output pins, with internal wiring that connects each input pin to a unique output pin.
- For simplicity, only three of the internal connections in each cylinder are shown.
- After 26 letters of plaintext, the first cylinder would be back to the initial position. Thus, we have a poly-alphabetic substitution algorithm with a period of 26.
- The result is that there are 26 * 26 * 26 = 17,576 different substitution alphabets used before the system


Direction of motion
Fast rotor
Medium rotor
Slow rotor
(a) Initial setting
Direction of motion
Fast rotor
Medium rotor
Slow rotor
(b) Setting after one keystroke