Preface

Notes from here are adapted from Professor Shishir Agrawal's notes for Math 187A and modified for my own understanding.

1 Introduction to Cryptography

We begin with some common definitions.

1.1 Terminology

Definition 1.1: Cipher

A **cipher**, or cryptosystem, is a cryptographic method for confidential communication.

Generally, a cryptographic method includes algorithms for *encryption* and *decryption*, which are inverse processes that convert between plainly readable information called *plaintext*[1] and unintelligible information called *ciphertext*.

Definition 1.2: Sender

A **sender**, often named "Alice" in abstract cryptographic discussions, *encrypts* her plaintext into ciphertext.

Definition 1.3: Receiver

A **receiver**, often named "Bob," *decrypts* (or deciphers) the ciphertext back into plaintext.

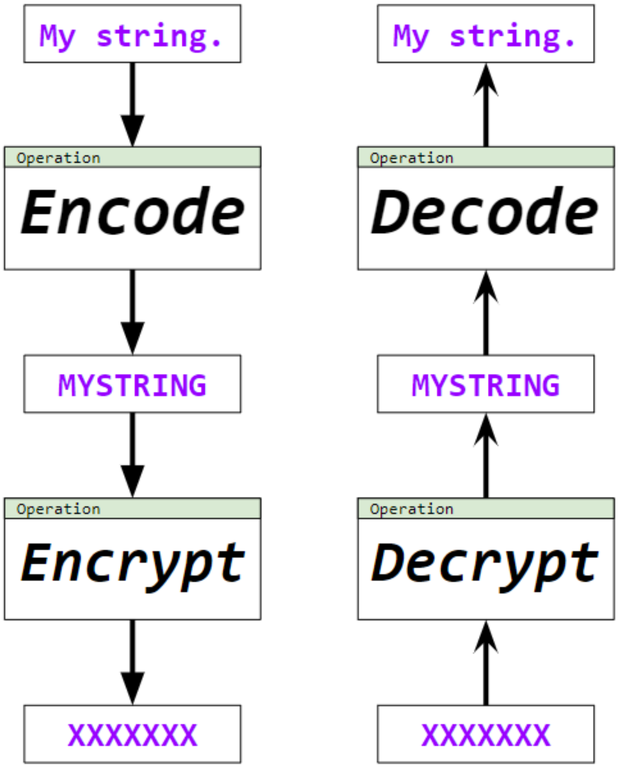
Often times, Bob will use a *key* to decrypt the message. This is sometimes known as a private key or decryption key.

Definition 1.4: Encoding

The (usually) preliminary step where a message is converted into a format which can then be encrypted is called **encoding**.

Note that encoded text is not secure; it is only secure after encryption. So, we can think of encoding as the pre-processing step. In other words, before we encrypt something, we might *encode* the text so it's easier to encrypt. It should also be noted that if a message had to be encoded before encryption, then it will also need to be decoded after decryption.

[1] In cryptography, we use *plaintext* and *ciphertext* instead of *plain text* and *cipher text*.



Definition 1.5: Adversary

An **adversary**, often named “Eve,” is one whose aim is to prevent the users of a cryptosystem from achieving their goal.

In our case here, an adversary can intercept a ciphertext. Thus, the adversary will not have Bob’s decryption key at the beginning. The idea is that, even if the adversary knows what cryptosystem was used to encrypt the message, if the adversary doesn’t have this decryption key, she should ideally not be able to decrypt the message. If she does manage to figure out the plaintext, she has *broken* the code.

Definition 1.6: Attack Model

An **attack model** specifies what Eve is allowed to do in order to break the code.

Some common attack models includes:

- Ciphertext-only attack: Eve must recover the plaintext using only the ciphertext.
- Known-plaintext attack: Eve may have access to some information about the plaintext (e.g., knowledge of portions of the plaintext), which can be used to recover the plaintext entirely.
- Chosen-plaintext attack: Eve can request or generate ciphertexts corresponding to any plaintext message of her choosing, and she can use this information to recover the plaintext.

Classical cryptography was mostly concerned with assuring security against the first two. Modern cryptography tries to assure security against the last.

2 Classical Cryptosystems

We begin with a definition:

Definition 2.1: n-gram

An n-gram is a sequence of n letters.

For example, a 1-gram is just a single letter; a 2-gram (i.e., *bigram*) is a pair of letters; and so on. Generally, we can group many classical cryptosystems into a few different encryption strategies.

<table>
<tr><th>Strategy</th><th>Description</th></tr>
<tr><td>Transposition</td><td>Involves rearranging units of plaintext according to some pattern. We'll see just one example of this type of cipher: rectangular transposition.</td></tr>
<tr><td>Substitution</td><td>Involves replacing units of plaintext with units of ciphertext. We can further group substitution ciphers into some subtypes:
<table>
<tr><th>Subtype</th><th>Description</th></tr>
<tr><td>Simple Substitution</td><td>In these ciphers, single letters of plaintext are replaced by ciphertext. The substitution scheme stays the same over the course of the entire message. Some examples we'll see include:
• Masonic cipher
• Caesar cipher
• Affine cipher
• Polybius square
In essence, though, there is a 1-1 relationship between the letters of the plaintext and the ciphertext alphabets.</td></tr>
<tr><td>Polygraphic Substitution</td><td>In these ciphers, groups of letters in the plaintext are replaced by ciphertext (a group of n letters is called an n-gram).The substitution scheme stays the same over the entire message. Some examples we'll see include:
• Hill cipher
• Playfair cipher
So, in essence, polygraphic substitution is just simple substitution but with <i>groups of letters</i> instead of individual letters.</td></tr>
<tr><td>Polyalphabetic Substitution</td><td>In these ciphers, single letters in the plaintext are replaced by ciphertext, and the substitution scheme changes over the course of the message. Some examples include:
• Vignere cipher
• One-time pad</td></tr>
</table>
In practice, however, most cryptosystems employ a combination of these strategies.</td></tr>
</table>

2.1 Rectangular Tranposition

Rectangular tranposition, known also as *regular columnar transposition*, is a tranposition cipher. The ciphertext is obtained by *permuting* the letters of the plaintext in a particular pattern. The pattern is determined by a secret *keyword*.

Roughly speaking, the steps to perform rectangular transposition are as follows:

1. Using the keyword, rank the letters based on alphabetical ranking.
2. Break up the message into groups of n, where n is the length of the keyword.
3. For each group, do the following:
   - Encrypting: If the ith letter of the keyword has rank j, move the ith letter in the group into the jth position.
   - Decrypting: If the ith letter of the keyword has rank j, move the jth letter of each group into the ith position.

Note that keywords with repeat letters do not work by themselves. We either need to agree not to use words with repeat letters, or remove duplicate letters from the keyword[2].

(Example: Encryption.) Suppose that Alice and Bob share the keyword `GUARD`, and that Alice wants to send the following message to Bob:

```
Hide! The baboons are coming for you.
```

First, we'll **encode** the message so that it's easier to encrypt. In our example, we'll remove all spaces and punctuation.

```
HIDETHEBABOONSARECOMINGFORYOU
```

Now that encoding is done, we still need to encrypt the message. Notice how the keyword `GUARD` has 5 letters; we can break the message up into 5-grams and then stack them into rows:

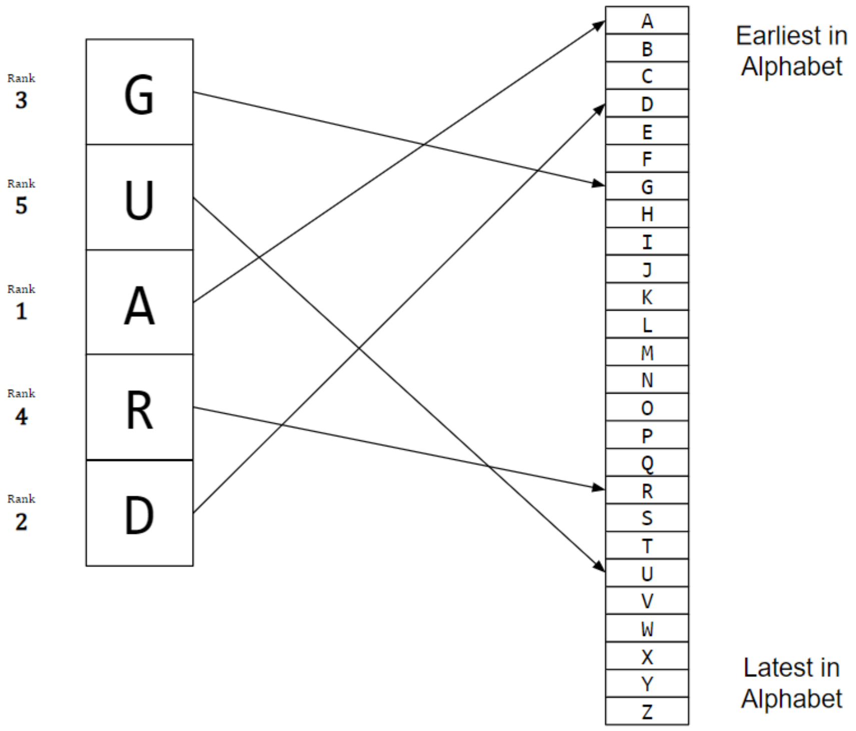
```
HIDET
HEBAB
OONSA
RECOM
INGFO
RYOU
```

We then need to insert some random letters at the end of the message so every row has an equal number of letters. Let's use `Q`:

```
HIDET
HEBAB
OONSA
RECOM
INGFO
RYOUQ
```

Now, we begin the **encryption** process by rearranging the letters in each row based on the alphabetical ranking of the letters of the keyword `GUARD`.

[2] In this course, we won't consider words with repeat letters.



We note that the alphabetical rankings of the letters of this keyword are 3, 5, 1, 4, 2. We can see this as a *permutation*; that is,

$$1 \mapsto 3 \qquad 2 \mapsto 5 \qquad 3 \mapsto 1 \qquad 4 \mapsto 4 \qquad 5 \mapsto 2$$

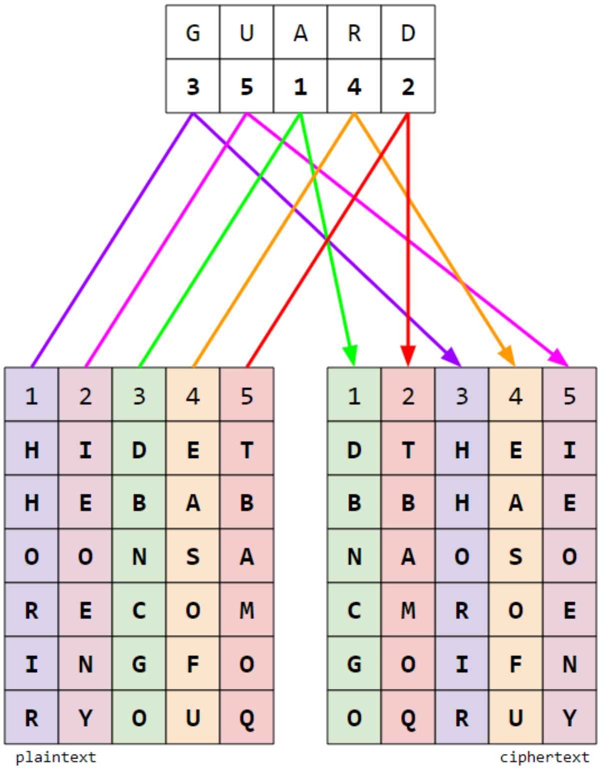
The idea for encryption is that, for each column i, we'll send that column to whatever is mapped by the permutation above. Going back to the stack of letters we have, we can label each individual column:

```
plaintext
position    1 2 3 4 5
            H I D E T
            H E B A B
            O O N S A
            R E C O M
            I N G F O
            R Y O U Q
```

The idea is that

- we can put all letters under position 1 in the plaintext stack to position **3** of the ciphertext stack,
- we can put all letters under position 2 in the plaintext stack to position **5** of the ciphertext stack,
- we can put all letters under position 3 in the plaintext stack to position **1** of the ciphertext stack,
- we can put all letters under position 4 in the plaintext stack to position **4** of the ciphertext stack,
- we can put all letters under position 5 in the plaintext stack to position **2** of the ciphertext stack.

The process, visually, would look like:



G	U	A	R	D
3	5	1	4	2

1	2	3	4	5
H	I	D	E	T
H	E	B	A	B
O	O	N	S	A
R	E	C	O	M
I	N	G	F	O
R	Y	O	U	Q

plaintext

1	2	3	4	5
D	T	H	E	I
B	B	H	A	E
N	A	O	S	O
C	M	R	O	E
G	O	I	F	N
O	Q	R	U	Y

ciphertext

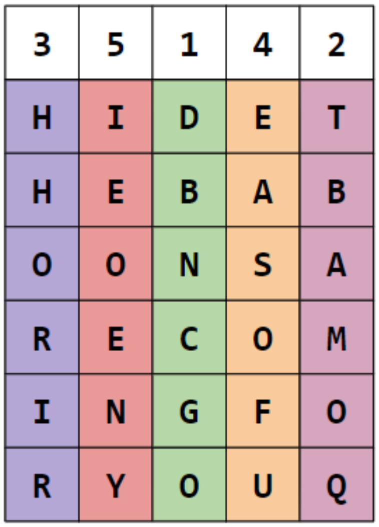
Therefore, the ciphertext stack would look like:

```
DTHEI
BBHAE
NAOSO
CMROE
GOIFN
OQRUY
```

Undoing the stacking gives us the ciphertext:

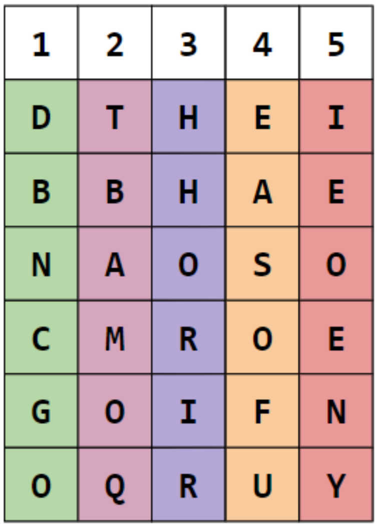
```
DTHEIBBHAENAOSOCMROEGOIFNOQRUY
```

Remark: An easy way to run through the process is to create two "groups," side-by-side. The first group will be the plaintext stack, and the second group will be the ciphertext text. Then, label each column of the first group with the **alphabetical ranking** of the keyword. Label each column of the second group with **12345**. Finally, map each column from the first group to the second group based on the label.



3	5	1	4	2
H	I	D	E	T
H	E	B	A	B
O	O	N	S	A
R	E	C	O	M
I	N	G	F	O
R	Y	O	U	Q

plaintext



1	2	3	4	5
D	T	H	E	I
B	B	H	A	E
N	A	O	S	O
C	M	R	O	E
G	O	I	F	N
O	Q	R	U	Y

ciphertext

Decrypting is merely the inverse of the encryption process.

(Example: Decryption.) Consider the above example again. Suppose Alice successfully sends the following ciphertext to Bob:

```
DTHEIBBHAENAOSOCMROEGOIFNOQRUY
```

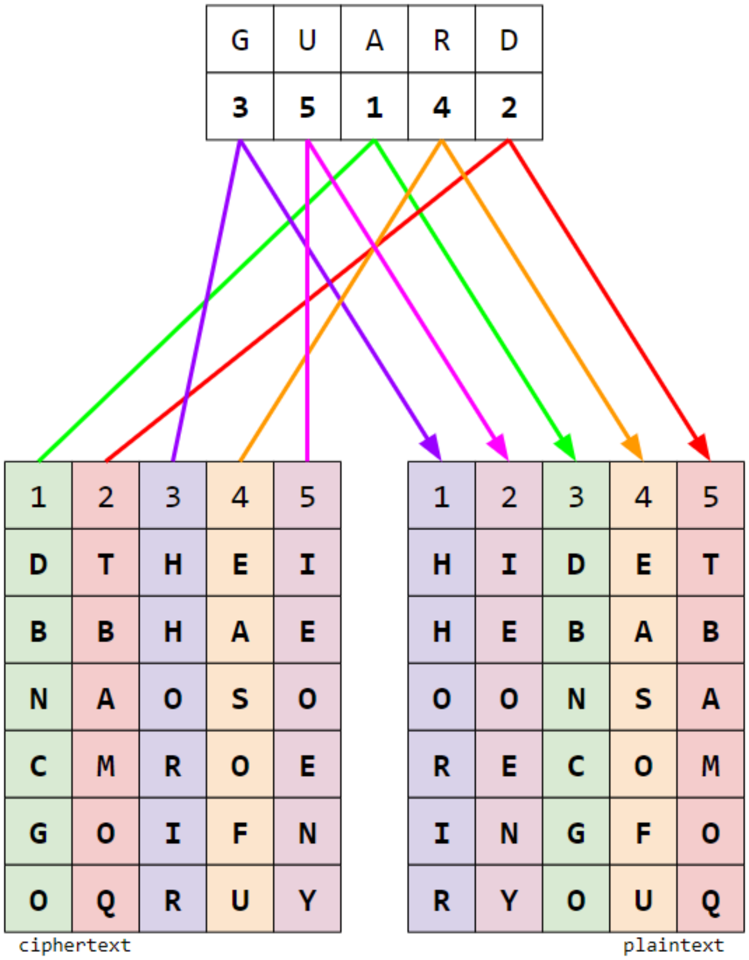
Bob knows that the keyword is `GUARD`. He can use this keyword to decrypt the message. He can begin by taking the letters of the ciphertext and stacking them into rows of 5, since `GUARD` has 5 letters:

```
DTHEI
BBHAE
NAOSO
CMROE
GOIFN
OQRUY
```

Bob also knows the alphabetical ranking of the letters of `GUARD` (which is the same rankings as described above). In particular, the alphabetical ranking is `35142`. So, we need to do the following:

- The letters in position 1 of the ciphertext stack needs to be moved to position **3**,
- the letters in position 2 of the ciphertext stack needs to be moved to position **5**,
- the letters in position 3 of the ciphertext stack needs to be moved to position **1**,
- the letters in position 4 of the ciphertext stack needs to be moved to position **4**,
- the letters in position 5 of the ciphertext stack needs to be moved to position **2**.

The process, visually, would look like:



G	U	A	R	D
3	**5**	**1**	**4**	**2**

1	2	3	4	5
D	T	H	E	I
B	B	H	A	E
N	A	O	S	O
C	M	R	O	E
G	O	I	F	N
O	Q	R	U	Y

ciphertext

1	2	3	4	5
H	I	D	E	T
H	E	B	A	B
O	O	N	S	A
R	E	C	O	M
I	N	G	F	O
R	Y	O	U	Q

plaintext

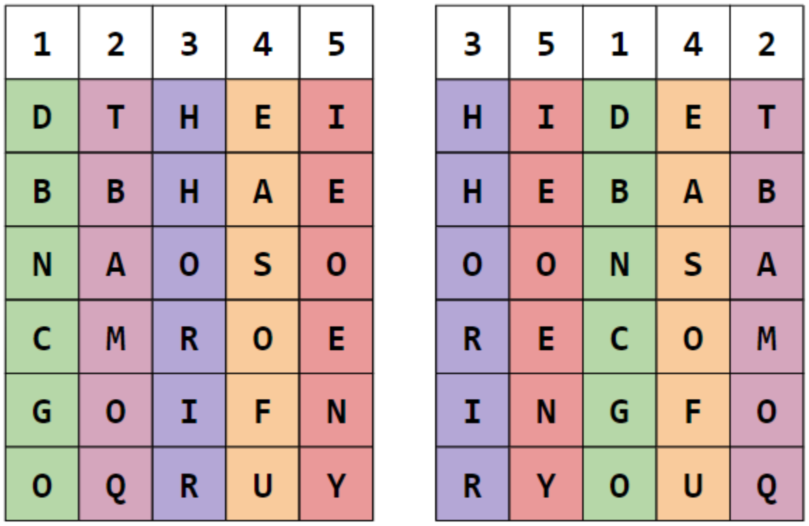
Undoing the stacking gives us:

```
HIDETHEBABOONSARECOMINGFORYOUQ
```

At this point, Bob needs to make an educated guess as to what the encoded message says (recall that we had to encode the message before encrypting it). By removing the `Q` and correctly punctuating the message, we get

```
Hide! The baboons are coming for you.
```

Remark: We can easily decrypt an encrypted word by doing the inverse of what we did above. Create two "groups," side-by-side. The first group will be the ciphertext stack, and the second group will be the plaintext text. Then, label each column of the first group with **12345**. Label each column of the second group with the **alphabetical ranking** of the keyword. Finally, map each column from the first group to the second group based on the label.



1	2	3	4	5
D	T	H	E	I
B	B	H	A	E
N	A	O	S	O
C	M	R	O	E
G	O	I	F	N
O	Q	R	U	Y

ciphertext

3	5	1	4	2
H	I	D	E	T
H	E	B	A	B
O	O	N	S	A
R	E	C	O	M
I	N	G	F	O
R	Y	O	U	Q

plaintext

(Exercise: Encryption.) *Encrypt the message* `There is always hope.` *using the keyword* `CRASH`*.*

First, we encode the message so that we can easily encrypt it:

```
THEREISALWAYSHOPE
```

Noting that `CRASH` has length 5, we break the now encoded message into groups of 5 letters (5-grams):

```
THERE
ISALW
AYSHO
PE
```

Let's now add nonsense letters at the end of the last row so every row has 5 letters:

```
THERE
ISALW
AYSHO
PEABC
```

Now, we note the alphabetical ranking of each letter in `CRASH`:

$$C \mapsto 2 \quad R \mapsto 4 \quad A \mapsto 1 \quad S \mapsto 5 \quad H \mapsto 3.$$

Using the streamlined way discussed above, we have

```
2 4 1 5 3 | 1 2 3 4 5
T H E R E | E T E H R
I S A L W | A I W S L
A Y S H O | S A O Y H
P E A B C | A P C E B
```

Unstacking the new rows gives us the ciphertext:

```
ETEHRAIWSLSAOYHAPCEB
```

(Exercise: Decryption.) *Decrypt the message* `ETIHGFREAFRSLAESOXOE` *using the keyword* `CRASH`*.*

Begin by grouping the letters into 5-grams, since `CRASH` has length 5:

```
ETIHG
FREAF
RSLAE
SOXOE
```

Recall that the alphabetical ranking of each letter in `CRASH` is `24153`. Using the streamlined way discussed above, we have

```
1 2 3 4 5    2 4 1 5 3
E T I H G -> T H E G I
F R E A F -> R A F F E
R S L A E -> S A R E L
S O X O E -> O O S E X
```

Unstacking the new rows gives us the plaintext:

```
THEGIRAFFESARELOOSEX
```

Decoding the message gives us:

```
The giraffes are loose.
```

(Exercise.) Encrypt the message `Meet at the trolley station.` using keyword `UCSD`.

Encoding, grouping the resulting letters into groups of 4, and adding a nonsense letter gives us:

```
MEET
ATTH
ETRO
LLEY
STAT
IONX
```

Noting that the alphabetical ranking of `UCSD` is `4132`, we can use the streamlined way discussed above to get the encrypted result:

```
4 1 3 2    1 2 3 4
M E E T -> E T E M
A T T H -> T H T A
E T R O -> T O R E
L L E Y -> L Y E L
S T A T -> T T A S
I O N X -> O X N I
```

Unstacking the result gives us:

```
/ETEMTHTATORELYELTTASOXNI
```

(Exercise.) Alice and Bob share the keyword ZEUS. Alice uses rectangular tranposition to encrypt the following nonsense message:

MTSQAGXY

What is the corresponding ciphertext?

Encoding, grouping the resulting letters into groups of 4, and adding a nonsense letter gives us:

```
MTSQ
AGXY
```

Noting that the alphabetical ranking of ZEUS is 4132, we can use the streamlined way discussed above to get the encrypted result:

```
4 1 3 2    1 2 3 4
M T S Q -> T Q S M
A G X Y -> G Y X A
```

Unstacking the result gives us:

```
TQSMGYXA
```

(Exercise.) The following message was encrypted using rectangular transposition with the keyword SNAKE. What is the plaintext?

DSUEMSEDIAJQQDA

SNAKE has alphabetical ranking 54132. With this in mind, stacking the letters of the encrypted message into groups of 5 and then running the streamlined process gives us:

```
1 2 3 4 5    5 4 1 3 2
D S U E M -> M E D U S
S E D I A -> A I S D E
J Q Q D A -> A D J Q Q
```

Unstacking the result gives us:

```
MEDUSAISDEADJQQ
```

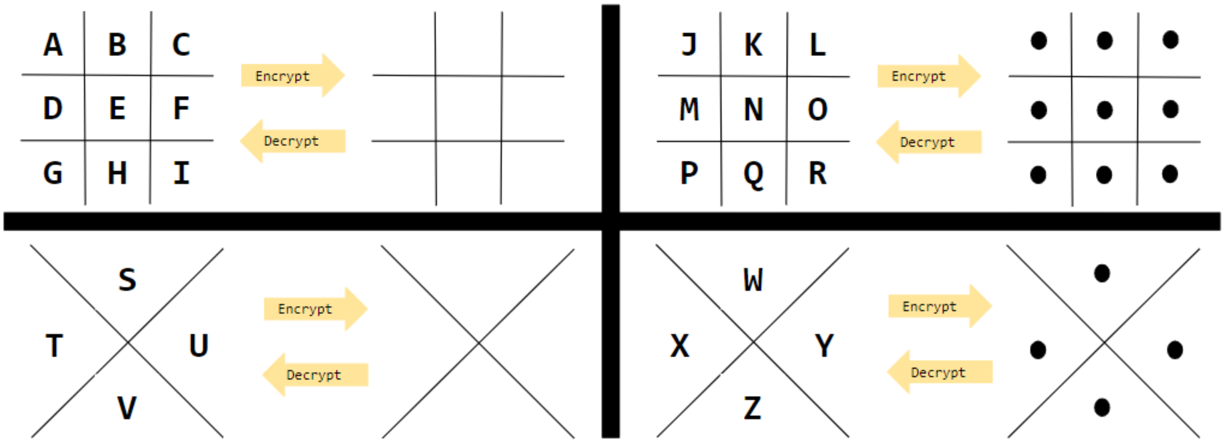
Decoding gives us:

```
Medusa is dead.
```

2.2 Masonic Cipher

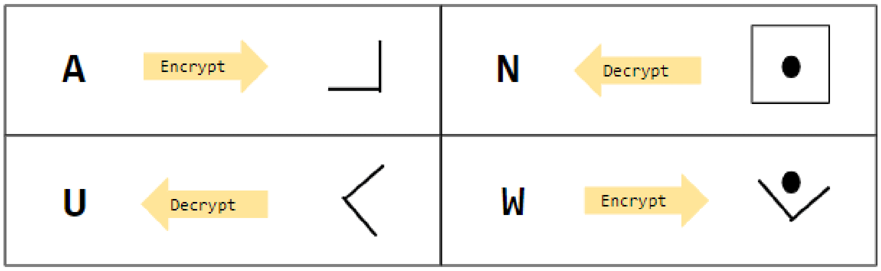
The masonic cipher (also known as the *pigpen cipher* or *tic-tac-toe cipher*) is a simple substitution cipher that replaces individual letters with certain geometric shapes.

For example, consider the following diagram, which represents a Masonic cipher for the English letters:



The idea is that we can replace a letter (e.g., A) with a corresponding geometric shape (e.g., the backwards L represented by the top-left part of the grid.)

Some other examples based on the above cipher are shown below:

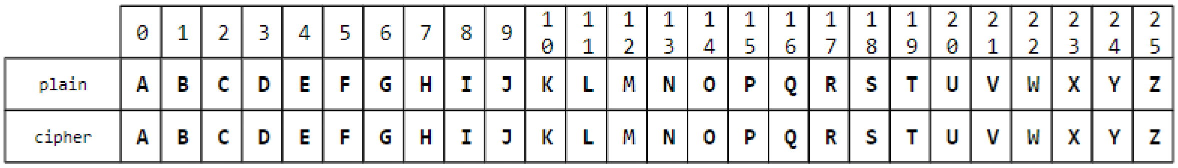


Note that there is *no key* associated with this cipher. There is only a decryption function (which is just mapping the geometric shape back to the letter). Therefore, the adversary, who knows that a message was encrypted using a masonic cipher, can recover the plaintext easily.

2.3 Caesar Cipher

The Caesar cipher, also known as a *shift cipher*, is a simple substitution cipher that *shifts* a letter by some amount n. Hence, the key for this cipher is an integer n. The idea is that we initially assign each letter an integer, perhaps by their alphabetical ranking (e.g., A is 0, B is 1, and so on.) If we want to shift the letters by some number, we can just "move" the letters by that amount. If a letter gets a new integer that's greater than 25, we can "wrap" the letter back.

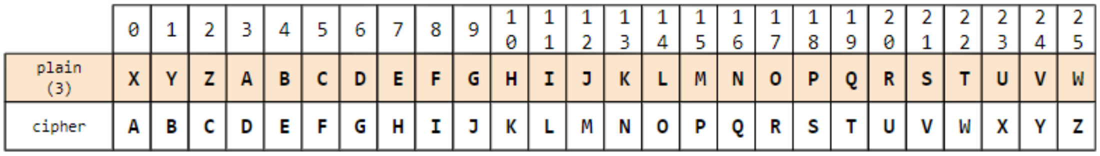
Consider the following diagram, which shows the correspondence between the plaintext alphabet and the ciphertext alphabet.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
plain	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
cipher	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

In this particular diagram, when we apply a shift, we apply the shift to the *plain* row. By doing this, we can translate whatever plaintext we have to ciphertext.

(Example.) If we shift each letter by 3 (i.e., $n = 3$), we have



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
plain (3)	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
cipher	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

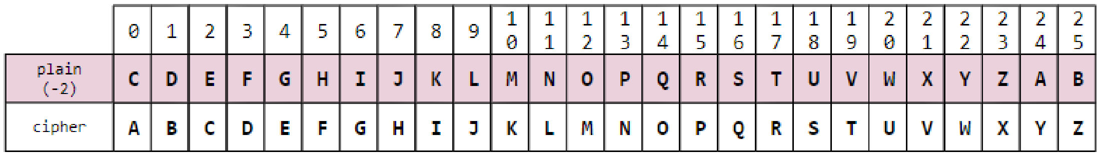
Notice how A now corresponds to 3. Recall that A's original position was 0; if we shift each letter by 3, we essentially add 3 to A's original position to get the new position

$$0 + 3 = 3.$$

The same idea applies to any other letter. One key thing to notice is how X, Y, and Z were *wrapped back* to the beginning. In any case, let's see how translation would work in this case:

- To convert a letter from plaintext to ciphertext, look for the letter in the(shifted) plaintext row and then look at the corresponding ciphertext column. For example, R in plaintext would become U in ciphertext.
- To convert a letter from ciphertext to plaintext, look for the letter in the ciphertext row and then look at the corresponding (shifted) plaintext column. For example, U in ciphertext becomes R in plaintext.

(Example.) If we shift each letter by -2 (i.e., $n = -2$), we have



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
plain (-2)	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
cipher	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

As with rectangular tranposition, we should encode the message by removing any non-alphabetic characters and capitalizing everything.

(Exercise.)

- *Using a shift of 3, encrypt the message* `Meet at La Jolla Shores.`

Encoding the message gives us MEETATLAJOLLASHORES. Then, we can use the example above (with the shift of 3) to give us the proper correspondence.

```
    plain       M E E T A T L A J O L L A S H O R E S
    cipher      P H H W D W O D M R O O D V K R U H V
```

This gives us PHHWDWODMROODVKRUHV.

- *Using a shift of 3, decrypt the message PHHWDWVXQJRGODZQ*

Using the example above (with the shift of 3), we have

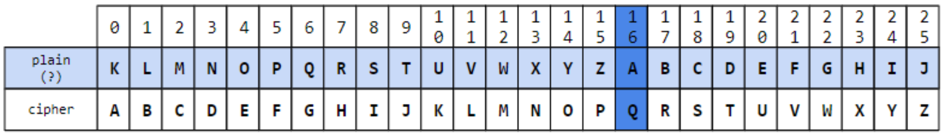
```
cipher     P H H W D W V X Q J R G O D Z Q
plain      M E E T A T S U N G O D L A W N
```

Decoding this gives us `Meet at Sun God Lawn.`

(Exercise.) *You are Eve. You have just intercepted the following message that Alice was trying to send to Bob:* `Q TQDM IB QPWCAM`*. You know that Alice used a Caesar cipher, but she didn't remove spaces before encrypting: she left the spaces in her original message as-is. What is the original message?*

`Q` itself could be a word; specifically, it could either be `A` or `I`. We can try to figure out what the message by guessing which word the first word could be.

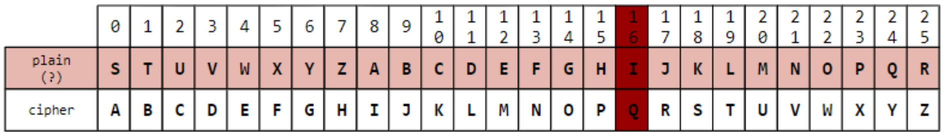
- If `Q` maps to `A`, then we have



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
plain (?)	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
cipher	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

  Partially decrypting the ciphertext gives us `A DANW`, but `DANW` is meaningless. Therefore, it cannot be `A`.

- If `Q` maps to `I`, then we have



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
plain (?)	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
cipher	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

  Decrypting this gives us:

  `I LIVE AT IHOUSE`

Therefore, the message is `I LIVE AT IHOUSE`. The shift was 8.

(Exercise.) Alice encrypts the following message using a Caesar cipher with a shift of 1.

`Zeus is hiding in a cave`

What is the corresponding ciphertext?

```
plain      ZEUSISHIDINGINACAVE
cipher     AFVTJTIJEJOHJOBDBWF
```

Essentially, we just move all letters forward by 1.