MATH 236 - Differential Equations and Linear Algebra

Dr. Reitenbach

The Mathematics of Encryption

Clayton Johnson, Christopher Vandermeer, and Caden Anderson

October 28, 2019

Let A be the $4x4$-matrix, also referred to as the encryption key, as follows:

$$A = \begin{bmatrix} 14 & 12 & 26 & 15 \\ 8 & 0 & 7 & 8 \\ 6 & 13 & 20 & 7 \\ 7 & 18 & 25 & 8 \end{bmatrix}$$

Additionally, we will translate plaintext into matrix form using the following character-integer mapping:

Space	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	13

N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

1: It is easy to see that $\det(A) = 7$. Find the multiplicative inverse of the number 7 modulo 27, i.e., find an integer $x \in \{0, 1, 2, ..., 26\}$ such that $7x \equiv 1$. This number x can be found by trial and error, or by using methods from number theory.

Since we have $7x \equiv 1$, we know there is an x such that, when multiplied by 7 and then taken modulo 27, the result is 1. Using this idea, since the remainder is, by definition, the difference between $7x$ and 27, we state $7x - 1 \equiv 27$. Thus, we have that

$$\begin{aligned} 7x - 1 &\equiv 27, \\ 7x &\equiv 28, \\ x &\equiv 4. \end{aligned}$$

2: Find the inverse of the matrix A modulo 27, i.e. find a matrix B with *integer entries* in $\{0, 1, 2, ..., 26\}$ such that $AB \equiv BA \equiv I_4$. This matrix B is called the decryption key.

To have a matrix B such that $AB \equiv BA \equiv I_4$, we would expect $B = A^{-1}$. Thus,

$$\begin{aligned} B &\equiv A^{-1}, \\ B &\equiv \begin{bmatrix} -18 & 19 & 18 & -1 \\ \frac{33}{7} & -\frac{34}{7} & -\frac{41}{7} & \frac{8}{7} \\ -\frac{40}{7} & \frac{41}{7} & \frac{48}{7} & -\frac{8}{7} \\ 23 & -24 & -24 & 2 \end{bmatrix}. \end{aligned}$$

Since, for modulo 27, $4 \cdot 7 \equiv 1$, we may multiply the matrix by $4 \cdot 7$ or 28 (see Problem 1):

$$B \equiv 4 \cdot 7 \begin{bmatrix} -18 & 19 & 18 & -1 \\ \frac{33}{7} & -\frac{34}{7} & -\frac{41}{7} & \frac{8}{7} \\ -\frac{40}{7} & \frac{41}{7} & \frac{48}{7} & -\frac{8}{7} \\ 23 & -24 & -24 & 2 \end{bmatrix},$$

$$\equiv \begin{bmatrix} 4 \cdot 7 \cdot -18 & 4 \cdot 7 \cdot 19 & 4 \cdot 7 \cdot 18 & 4 \cdot 7 \cdot -1 \\ 4 \cdot 33 & 4 \cdot -34 & 4 \cdot -41 & 4 \cdot 8 \\ 4 \cdot -40 & 4 \cdot 41 & 4 \cdot 48 & 4 \cdot -8 \\ 4 \cdot 7 \cdot 23 & 4 \cdot 7 \cdot -24 & 4 \cdot 7 \cdot -24 & 4 \cdot 7 \cdot 2 \end{bmatrix},$$

$$\equiv \begin{bmatrix} -504 & 532 & 504 & -28 \\ 132 & -136 & -164 & 32 \\ -160 & 164 & 192 & -32 \\ 644 & -672 & -672 & 56 \end{bmatrix} \bmod(27),$$

$$B \equiv \begin{bmatrix} 9 & 19 & 18 & 26 \\ 24 & 26 & 25 & 5 \\ 2 & 2 & 3 & 22 \\ 23 & 3 & 3 & 2 \end{bmatrix}.$$

To double check our work, we expect $AB \equiv BA \equiv I_4$:

$$AB \equiv \begin{bmatrix} 811 & 675 & 675 & 1026 \\ 270 & 190 & 189 & 378 \\ 567 & 513 & 514 & 675 \\ 729 & 675 & 675 & 838 \end{bmatrix} \bmod(27),$$

$$\equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

Thus, $AB \equiv I_4$.

$$BA \equiv \begin{bmatrix} 568 & 810 & 1137 & 621 \\ 729 & 703 & 1431 & 783 \\ 216 & 459 & 676 & 243 \\ 378 & 351 & 729 & 406 \end{bmatrix} \bmod(27),$$

$$\equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

Thus, $BA \equiv AB \equiv I_4$.

3: Decode the following secret message back to plaintext:
"XKGYHYKPTF D TJIPOPVXMQSHQJP" (note the spaces)

Since $AP \equiv C$, where A is the encryption matrix (provided), P is the plaintext message converted to a matrix using the mapping on page 1, and C is the encoded matrix of the message

to be sent, and we have $B = A^{-1}$:

$$\begin{aligned} AP &\equiv C, \\ BAP &\equiv BC, \\ I_4P &\equiv BC, \\ P &\equiv BC. \end{aligned}$$

Thus, we have the scheme for decrypting a given encoded messsage: $BC \equiv P$. Using this scheme, we find that

$$\begin{aligned} P \equiv BC &= \begin{bmatrix} 9 & 19 & 18 & 26 \\ 24 & 26 & 25 & 5 \\ 2 & 2 & 3 & 22 \\ 23 & 3 & 3 & 2 \end{bmatrix} \begin{bmatrix} 24 & 8 & 20 & 0 & 16 & 24 & 8 \\ 11 & 25 & 6 & 20 & 15 & 13 & 17 \\ 7 & 11 & 0 & 10 & 16 & 17 & 10 \\ 25 & 16 & 4 & 9 & 22 & 19 & 16 \end{bmatrix}, \\ &\equiv \begin{bmatrix} 1201 & 1161 & 398 & 794 & 1289 & 1263 & 941 \\ 1162 & 1197 & 656 & 815 & 1284 & 1434 & 964 \\ 641 & 451 & 140 & 268 & 594 & 534 & 432 \\ 656 & 324 & 486 & 108 & 505 & 680 & 297 \end{bmatrix} \bmod(27), \\ &\equiv \begin{bmatrix} 13 & 0 & 20 & 11 & 20 & 21 & 19 \\ 1 & 9 & 8 & 5 & 15 & 3 & 19 \\ 20 & 19 & 5 & 25 & 0 & 3 & 0 \\ 8 & 0 & 0 & 0 & 19 & 5 & 0 \end{bmatrix}, \\ P &\equiv \text{'MATH IS THE KEY TO SUCCESS '} \end{aligned}$$

4: Write a pair of MATLAB M-files, encode.m and decode.m, where `encode('text')` converts plaintext to ciphertext and `decode('text')` converts ciphertext to plaintext.

The group has opted for using the Python language to create these scripts. It will be emailed to Dr. Reitenbach. However, if one desires to find it another way, the code may be found at `github.com/JohnsonClayton/Mathematics-of-Encryption`. Additional usage tricks are at this website and are copied below. Please contact Clayton Johnson for any concerns regarding the Python script.

This program was created to encrypt and decrypt given messages based off an encryption scheme discussed in the assignment. The encryption keys and decryption keys are hard-coded within the program; however if another pair is used to replace these the script will not break. The given message can be of n length. The program is intended for use from a command line as follows:

```
$ python encryption_master.py --encode 'this is a test message'
encoding THIS IS A TEST MESSAGE
Your message has been encoded:  DXXMHYKPOZIGZMFCXSPWWBZD
```

and decryption works similarly:

```
$ python encryption_master.py --decode 'DXXMHYKPOZIGZMFCXSPWWBZD'
decoding DXXMHYKPOZIGZMFCXSPWWBZD
Your message has been decoded:  THIS IS A TEST MESSAGE
```

5: Some encryption schemes can be cracked by frequency analysis; for example, if letters are mapped to numbers without using matrix multiplication it is easy to detect which number represents the letter "E" since it is the most commonly used letter in English. Run `encode` with a few sample messages to convince yourself that our encryption scheme cannot be cracked by frequency analysis. Explain why not.

Some frequency analysis may be found at the aforementioned `github` link in the `README`. That information will be restated here.

Using the first paragraph of the `Wikipedia` page for Encryption for a plaintext frequency analysis, we created Table 1. Also, in the table we can see the frequency analysis for the encrypted version of the same text.

From this table, we can clearly see a large difference between how the frequency of characters varies between the plain and ciphertext. For the plaintext, we see 13% of the characters are 'E', with seven other characters following at above 5% representation. Conversely, for the cipher text, we see that only four characters are represented as more than 5%, however they are on the edge of that classification. Additionally, while the plaintext character frequencies decrease substantially over the text, starting at 13.49% and descending to 0.00% (not shown), the ciphertext's character proportions see little variation, from 5.4% to a low of 2.2%. It is also worth noting that the ciphertext represents all characters in this situation, while the plaintext is missing some.

This small example demonstrates the inability or ineffectiveness of frequency analysis to break the encryption scheme we have established here due to the fact that outliers in the cipher text rarely exist, if at all; and where outliers do exist, they would not represent a one-to-one relationship with the plaintext.

Plaintext		Ciphertext	
E	13.49%	B	5.4%
T	9.34%	F	5.29%
N	8.95%	Z	5.29%
I	8.04%	P	5.18%
R	7.26%	T	4.63%
O	7.13%	I	4.41%
A	6.61%	N	4.41%
S	6.23%	E	4.07%
C	4.8%	O	4.07%
P	3.76%	Y	4.07%
H	3.63%	V	3.96%
D	3.63%	S	3.96%
Y	2.98%	D	3.85%
L	2.85%	J	3.74%
U	2.59%	X	3.63%
G	2.08%	G	3.63%
M	1.69%	R	3.63%
B	1.3%	A	3.52%
F	1.17%	W	3.41%
W	0.78%	Q	3.19%

Table 1: Frequency Analysis of Plaintext and Ciphertext using `dcode.fr`. Ordered in descending order of frequency. Lowest frequencies are dropped.