Group S

[redacted]

MATH 485 Section 1

9 September 2020

OCRAI Message Encryption Report

Given the constraints for your secure messaging system, we have determined that the best method of encrypting and decrypting messages between employees is an approach similar to what is known as a *One-time pad*. The approach is a key-based algorithm that effectively obfuscates the message and any patterns that may allow an attacker to decipher the message.

The encryption algorithm takes a *plaintext* message and uses a predetermined or calculated *key*, shared between both the sender and receiver, to encrypt the *plaintext* into *ciphertext*, then decrypt it back into *plaintext* for the receiver. To begin, the algorithm takes the *plaintext* message and removes any spaces or punctuation, and for simplicity, capitalizes all characters. For example, the *plaintext* message, "*Hello, World!*", is converted into the *plaintext* message, "*HELLOWORLD*."

Hello, World! → *HELLOWORLD*

Initial Message Modified Message

This prevents an attacker from being able to make any guesses or inferences from the *ciphertext* based on patterns in the language of the message.

With the *plaintext* message refactored, the algorithm then uses the predetermined *key* to encrypt the updated *plaintext*. The algorithm accomplishes this by shifting each character in the *plaintext* by the number of characters determined by the corresponding character in the *key*. Before we can make sense of this, we give each character of the alphabet a number:

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
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

These numbers are used to determine how many shifts will be performed, so $A = 0$ shifts, $B = 1$ shift, $C = 2$ shifts, etc. We define a "shift" as an increment in the alphabetic value of the character. For example, the letter *A* shifted three times becomes the letter *D*. If the shift goes passed the letter *Z*, we loop back around to *A* and continue the shifting. So, the letter *Y* shifted three times becomes the letter *B*.

We then use the *key* to determine how many characters to shift each character of the *plaintext*. Using the example *key*, "*PASSWORD",* we determine that we must shift the first character of the *plaintext* 15 times (since *P*=15), the second character 0 times (*A*=0), the third character 18 times (*S*=18), and so forth. Because the messages are typically longer than the *key*, we start back at the beginning of the *key* when the end is reached and there is more *plaintext* to encrypt. We can visualize this by lining up the *key* with the *plaintext* as follows:

Plaintext:	*H*	*E*	*L*	*L*	*O*	*W*	*O*	*R*	*L*	*D*
Key:	*P*	*A*	*S*	*S*	*W*	*O*	*R*	*D*	*P*	*A*

Now we can see how the *key* is used to determine how many characters to shift the *plaintext* to give us the *ciphertext*:

Plaintext:	*H*	*E*	*L*	*L*	*O*	*W*	*O*	*R*	*L*	*D*
# of shifts:	+15	+0	+18	+18	+22	+14	+17	+3	+15	+0
=Ciphertext:	*W*	*E*	*D*	*D*	*K*	*K*	*F*	*U*	*A*	*D*

So, the *plaintext* message "*Hello, World!*" encrypted with the *key* "*PASSWORD"* yields the *ciphertext* "*WEDDKKFUAD.*"

This process can be simplified mathematically by thinking of the *plaintext* and *key* in terms of their numbers:

$$c_i \equiv p_i + k_j \ (mod\ 26)$$

where p_i is the numerical value of the i^{th} character in the *plaintext*, k_j is the numerical value of the j^{th} character in the *key*, and c_i is the numerical value of the i^{th} character in the *ciphertext.*

Decrypting the *ciphertext* is very straightforward. Using the shared *key*, the reverse of encryption is applied to the *ciphertext*. The *key* is aligned with the *ciphertext* and then the characters are shifted back based on the *key* values:

Ciphertext:	*W*	*E*	*D*	*D*	*K*	*K*	*F*	*U*	*A*	*D*
Key:	*P*	*A*	*S*	*S*	*W*	*O*	*R*	*D*	*P*	*A*
# of shifts:	-15	-0	-18	-18	-22	-14	-17	-3	-15	-0
=Plaintext:	*H*	*E*	*L*	*L*	*O*	*W*	*O*	*R*	*L*	*D*

Mathematically, this can be described using the same symbols as before with the following equation:

$$p_i \equiv c_i - k_j \ (mod\ 26)$$

There are a few other security factors to consider, such as how the *key* will be shared between users and the security of that *key*, but from an encryption standpoint and the requirements provided, this encryption algorithm will provide the security the company needs to safely transmit messages.