Group U

[redacted]

Professor Jenkins

MATH 485

10 September 2020

Encryption Report

Dear OCRAI, we understand the need for an encryption system for your company. We know that keeping private company information is very important in today's day and age of getting ahead of the competition. Especially when this privacy is reliant upon the employees, things can be easily leaked to the public and rival business. We have put together an encryption method that you can apply to your company's mobile application to use for peer-to-peer encryption with your employees. This system allows for punctuation and special symbols and preserves the case of each letter.

Our cryptosystem requires a private key composed of any letters, numbers, or special symbols available on a standard computer or mobile phone. Per your specifications, the key does not need to be of great length. Although you can technically use as little as one character, we recommend 6-10 characters for maximum security. This key can be set by an administrator as often as desired; setting a randomized key daily will increase encryption security. There will be no need to memorize this key as it is only being used by the encryption method for encrypting messages sent that day. The keysetter only needs to provide a novel key daily and the cryptosystem will handle the rest.

The system works by first converting the characters of the plaintext to their ASCII Decimal representations. For the curious, these can be found at https://www.ascii-code.com/. The column marked "DEC" contains the decimal for the character in the corresponding row. We

leave the first and last characters of the message alone, but then iterate through the plaintext message (starting at the second character) and the pass key at the same time. The decimal value of the first character in the key is either added to or subtracted from the decimal value of the second letter in the message, mod 127. If the decimal value of the first character of the message is even, the values are added. If it is odd, the value is subtracted. This is why the first character of the message is left unchanged by encryption. Continuing with this idea, the decimal value of the second character in the key is added to the third character of the message, mod 127, and so on. The pattern continues like this, starting again with the first character of the key when we reach the end of the pass key. We now multiply the resulting decimal value by the value given by the last character of the message, mod 127. This is why this last character of the plaintext is also left untouched by the encryption. These values are needed again for decryption. One issue with using ASCII decimals is that the values from 0-32 correspond to unprintable characters. To remedy this, if the final values fall in that range, our system adds 200 to them to bring them into the range of printable characters while still maintaining a one-to-one relationship from plaintext to ciphertext. These final decimal values are then converted back into characters to complete the encryption process.

We chose to perform calculations in mod 127 because 127 is a prime number, allowing us to multiply these decimal values by any factor mod 127. Also, all upper and lower case letters, numbers, and punctuation marks commonly used fall below 127.

To represent our algorithm mathematically, we collect an ASCII number, A, from the private key, and an ASCII number, B, from a character, X, in the plaintext. We call the ASCII numbers of the first and last characters of the message C and D respectively. Furthermore, we

define a function *parity(x)* that returns a 1 if *x* is even, and a -1 if *x* is odd. Then we illustrate our encryption as follows:

$$encrypted(X) = (B + parity(C)*A)*D \pmod{127} = Y$$

$$If\ Y < 33:\ Y = Y + 200.$$

Then *Y* is converted back to a character.

With all of this in mind, decryption can be performed quickly as long as you know the private key. If we consider an encrypted character with ASCII number, *Y*, we perform decryption with the following:

$$If\ Y > 200:\ Y = Y - 200$$

$$decrypted(Y) = Y*Dinv + parity(C)*A \pmod{127} = B.$$

Where *Dinv* is the modular multiplicative inverse of *D.* From here, *B* is converted back to the character *X* to complete decryption.

In conclusion, our encryption method effectively obfuscates any and all data that may find itself fallen into the wrong hands. It preserves case and allows for special symbols and punctuation. We believe that setting a unique key daily and hiding the encrypting variables within the plain text will greatly increase security in intercompany communication. You shouldn't have to worry about the security of your sensitive data; you can rely on our cryptosystem to safely encrypt your communications and keep them away from prying eyes.