Group O

Cryptographic System for OCRAI

11 September 2020

[redacted]

Introduction: The problem we are faced with is a serious one. Our business competitors have gained potential access to our company's classified information via missent text messages. Our goal with this report is to explain the solution our team has come up with in order to gain control over these security breaches of the past and prevent them from reoccurring in the future. Human error can be a serious cause of security problems, and our goal is to mitigate that as much as possible. By creating a secure encryption system where company messages are sent, we will prevent the accidental data leaks that can happen when someone mistakenly sends information to someone it was not meant for.

Encrypting in our system: Our nontrivial encryption method encrypts each single letter to another letter, in sequence through the whole text. It uses the original letter as well as where that letter is positioned in the text. This yields extra strength over simpler methods that always translate a given letter to another given letter, no matter where the original letter appears in the text. As such, our system is more resistant to letter frequency analysis[1] attacks.

Our random key is two positive integers: call them **a** and **b**. The greatest common divisor between **a** and 27 must be 1.

[1] An example of frequency analysis is as follows: "E" is very common in English text, so an attacker looks at the most common symbol in the encrypted text and assumes it is "E"

The first step in the encryption process is to convert all the letters (or spaces) into integers, according to the following table. If a letter is lower-case, find its upper-case version in the table. Discard any characters from the input text that are not in the table.

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	(space)
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	26

Then, each cipher letter's integer value **c** is computed with the formula

$$c = p * a + b + i \ (mod\ 27)$$

where p is the original letter's integer value, a and b are from the key, and i is the position in the text (starting from 0 for the first letter and increasing by 1 for each letter.) Mod 27 means that the final value of c is actually the remainder after dividing by 27.

After all c values are computed, find the corresponding letters in the table above and write them in sequence. Note that some "letters" will be a space, for the value 26.

For reference, "hello" with the key a=5, b=7 encrypts to "PBKLA" (note there are no repeated letters, despite the repeated "L" in "hello".)

Here is a brief summary of the operations to encrypt "hello":

```
H    E    L    L    O    (change the letters to uppercase)
↓    ↓    ↓    ↓    ↓    (translate from the table)
7    4   11   11   14
↓    ↓    ↓    ↓    ↓    (compute with the mod 27 formula)
15   1   10   11    0
↓    ↓    ↓    ↓    ↓    (translate back to letters)
P    B    K    L    A
```

If the resulting ciphertext is longer than the limit for a text message, split it into multiple messages in sequence. We assume that the cellular system will deliver the messages in order.

Decrypting in our system: If a message consists of multiple text messages, begin by combining the ciphertexts into one long ciphertext. Decryption proceeds similarly as encryption: convert all letters (or spaces) to the corresponding integers using the same table. Then compute each plaintext letter p with the formula

$$p = (c - b - i) * A \ (mod\ 27)$$

where c is the encrypted letter's integer value, b is from the key, i is the index as before, and A is the multiplicative inverse, mod 27, of the key's a. We won't waste the board members' time with a full explanation of the multiplicative inverse, but a short way to put it is that if one calculates a*A and then takes the remainder mod 27, the result will be 1 if the values are correct. The inverse can be calculated in SageMath with `inverse_mod(a, 27)`. For reference, for a = 5, A = 11. The "mod 27" part works as explained above. After all integer values have been calculated, convert them to letters (or spaces) as in the table above and put them together in sequence to obtain the plaintext message.

Conclusion: Our solution to the security breach issue is invaluable to the future of our informational security. By using the index of the plaintext letter for the encryption, we add another layer of protection against those who are attempting to gain access to our company's private information.