Independent Math Contractors Group J

[redacted]

We at Independent Math Contractors are happy to be trusted with your business. As we understand it, you require a secure method of sending sensitive information over text. You wish to give a normal message in plaintext to be encoded in a ciphertext, which you can send out without fear for its security. To provide you with maximum security, we have combined two well-known methods of encryption with one of our own invention, greatly reducing the possibility of a successful security breach in the near future. For your convenience, we have also created a computer program to perform the encryption so your employees can allocate their time to something more valuable.

The first part of our encryption process includes choosing a 9 letter keyword with no repeating letters. For the examples in this report, we will use the keyword COPYRIGHT. We used the keyword to create a matrix of the alphabet with the keyword in the top row. We then rearranged the columns of the matrix based on the alphabetical order of the keyword (as shown below). A basic substitution cipher was implemented by matching the positions of each letter in the old matrix with the new rearranged matrix. Since 9 does not evenly divide 26, if there was a gap in the matrix, the letter just matches the next available letter in the matrix. In the example below, this means that y=W and z=V.

C	O	P	Y	R	I	G	H	T
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	_

$\rightarrow$

C	I	G	H	O	P	R	T	Y
a	f	g	h	b	c	e	i	d
j	o	p	q	k	l	n	r	m
s	x	y	z	t	u	w	_	v

For the second step of encryption, we use an affine cipher. In an affine cipher, letters A-Z are numbered 0-25. To determine how the plaintext (for this stage) letters correspond to ciphertext, we perform the equation $NewValue = \alpha * OldValue + \beta$, mod 26, on the value assigned to our plaintext letter. (Mod 26 means we do normal arithmetic, but when we get to the number 26, we start over at 0, like counting on a clock).

To make our affine cipher more difficult to break, we change our values of α and β for each word. For the very first word, the α value equals the numerical value assigned to the first letter of the keyword. However, if the value of α is not relatively prime to the number of characters in the alphabet (in this case 26), then α is reduced (by subtracting 1) until it is relatively prime. If α is zero, then it wraps around to 26 and is reduced from there until it is relatively prime. For our example, with our keyword COPYRIGHT, the first letter is C, with a value of 2. Thus, the α value for the first word is 2. However, 2 is not relatively prime to 26, so it is reduced to 1, which is relatively prime. In a similar manner, β for each word equals the numerical value assigned to the last letter of the keyword. In our example, that is T, with a numerical value of 19, so β is 19. For the next word, we change the plaintext using another affine cipher. For this cipher, instead of using the numbers assigned to the first and last letters of the keyword, we assign ɑ and β based on the first and last letters, respectively, of the previous word. We continue to use affine ciphers based on the previous word for all remaining words until the entire plaintext sequence has been encoded using various affine ciphers.

Ex. If our "plaintext" from the first step is

Yib bayibm hiapebs rzdhpqw

The first word is coded with affine cipher $NewValue = 1 * OldValue + 19$, as explained above.

YIB

24 8 1

24*1+19 ≡ 17 (mod 26) | 8*1+19 ≡ 1 (mod 26) | 1*1+19 = 20 (mod 26)

17 1 20

RBU

The first word, encoded, is now rbu.

The second word is encrypted using the affine cipher $NewValue = 23 * OldValue + 1$ based on the former previous word yib. Remember that ɑ must be reduced (by subtracting 1) until it is relatively prime to 26.

B A Y I B M

1 0 24 8 1 12

24 1 7 3 24 17

Y B H D Y R

The second word, encoded, is now ybhdyr.

For the final stage of encryption, we manipulate the text on a word-by-word basis. To retain the original order of the words, we now use the encoded keyword as a kind of marker for each word. Every word will be surrounded by two letters from the keyword. This allows for a plaintext of up to 81 words, but more letters from the keyword could be used to surround the words if the plaintext is longer than 81 words. The first letter at the beginning of the word takes priority in ordering, and the other letters are used to increase the number of possible orderings of a word.

We choose the letters from the keyword thusly: for the first word in our text, we place the first letter of the keyword in front of it. We then place the last letter of the keyword at the end of the word. For the next word, we place the first letter of the keyword at the front of the word, but the second letter of the keyword as the last letter. The third word starts with the first letter and ends with the third letter, and so on, until the end of the keyword is reached. We repeat the same pattern until the keyword has been iterated through once. Then we use the second letter of the keyword at the front of the word and iterate through the keyword as the last letter again. We continue this pattern until all of the words have been labeled. This method could be extended for plaintexts greater than 81 words by adding two letters to the front and end of each word.

Ex. 1. A sequentially ordered set of surrounding letter groups would be as follows: c-c, c-o, c-p, … , c-t, o-c, o-o, o-p, … , o-t, p-c, p-o, p-p, etc.

Ex. 2. If our "plaintext" is rbu ybhdyr, and our keyword is COPYRIGHT, rbu, as the first word in our coded message, becomes **c**rbu**c** (bold added to make encryption steps clearer), and ybhdyr becomes **c**ybhdyr**o**.

<u>1</u> RBU <u>2</u> *underlined numbers indicate positional ordering priority

C O P Y R I G H T

1 2 3 4 5 6 7 8 9 *numbers below letters indicate letter ordering priority

Arranging alphabetically we get

crbuc cybhdyro

Which happens to be the same word order as in the "plaintext" because there were only two words.

We have now completely encrypted our plaintext. Each of our encryption examples uses less than 81 words. If a significantly longer plaintext needs to be encrypted, this part of the cipher could be extended to cover more words if two or three letters were used before and after each word, but such a strategy would have to be given in the key, which could be done by sending the number of letters before and after each word along with the keyword. Following the contract we received, 1 letter before and after should be plenty, so when there is no number given, we will assume it would be "1."

To decrypt a message, simply go through the above steps in reverse. First, rearrange the words in order of the first and last letters (based on the keyword). Then decrypt the affine cipher for the first word using the first and last letters of the keyword for α and β. Use the decrypted first word to get α and β for the second word. Continue until the whole text has completed the affine cipher decryption. Then use the keyword to create the alphabet matrix that is the key for the substitution encryption. Use the matrix to solve the substitution encryption for the whole text.

We have a few more notes to present for consideration. First, it would be most secure if the keyword chosen was not a word at all, but rather a scrambled bunch of non-repeating letters. This prevents potential attackers from simply unscrambling the keyword. Second, our method can handle 81 words in plaintext, but if you wish to send longer messages, you can just add a third letter position, follow the same pattern, and notify the recipient of the system change. Third, it is rather painful to go through this entire process by hand for a long message. With that under consideration, we have attached a program file that can perform the encryption for you.

We hope that you are satisfied with this improved way to send your sensitive information over text. The combination of multiple encryption methods provides a secure encryption that is difficult for an attacker to break. Thank you for entrusting us with your safety and good luck encrypting.

Independent Math Contractors

[redacted]

Link to program:

https://drive.google.com/file/d/1fdyYrUlqtBCh06FAiOnLmx7_UvnZthxp/view?usp=sharing