Cryptography Project

[redacted]

There exists a problem in this world. If things of value are put in the open, people are likely to steal them. Therefore, we must take security measures to ensure that those things do not happen. Furthermore, in recent history, the problem of keeping valuable information secret has been getting harder and harder. Therefore, we have created a cryptographic system that, with limited processing power, encrypts a message of ASCII characters using a key of length nine, expanding the length of the original message by five.

Before encrypting, take the plain text message and convert it into its bits. Also, take each character of the key and convert it into the bits (but keep each character's bits separate). Reference the modified ASCII table in the appendix to do this. The first step in encryption uses the first five characters from the key. Take each character from the original message (suppose it is of the form "abcdefg", where each bit is a "1" or a "0"), and modify it like this with a copy of the keys:
key1': a*****g
key2': *b***f*
key3': **c****
key4': ***d***
key5': ****e**

Then put the modified keys back together in order (key1'key2'...key5'). Note that the asterisks are the original bits of the keys. The only thing that happens is that some of the bits are overwritten by the character from the plaintext. This lengthens the message by a factor of five.

The next step in encryption is to do a transposition. We use the next character from the key, which we transform into decimal according to the rules of binary (remember it should be a seven digit binary number). The character used here should not be divisible by seven, so that when the bits are turned back into characters, we will get a very different set of characters. This allows for over one hundred possible keys here.

For the sake of explaining the transposition, we will say that the amount is an arbitrary quantity "k." First, the end of the message is indicated with a character. Then the first "k" bits are moved to after that point in the message (they are deleted from the part before the special character and reproduced following it). The next "k" bits are left alone and a different character is used to indicate the end of the first "k" characters. Then the first "k" bits after the first special character are placed at the end of the message, deleting them again from the first part of the message. The same special character placed after the first "k" bits is placed after the next "k" bits.

This pattern repeats until the action would involve the character indicating the end of the message. If the action was going to leave the characters alone, nothing is done, and all the special characters are removed and the message is converted from bits to their ASCII designations (seven bits at a time, using the table in the appendix). If the action was going to move the characters, all characters before the end of message character are moved to the end, and all the special characters are removed and the bits are converted to their ASCII designations.

Once the table is used to change the bits into characters, we will do a separate transposition. We will use the next three characters of the key in decimal form. Since the length of the partially encrypted message is about six hundred, each character of the key should be less than one hundred. Therefore, the part of the key that matters in decryption has approximately 100*100*100*100 = 100^4 = 100,000,000 configurations. Suppose the decimal forms of each key are “a,” “b,” and “c.” Then we create an empty message, which we will call “ciphertext.” To transpose the message, we take the first “a” characters out of the original message, and put them at the end of “ciphertext.” Then we take the last “b” characters out of the original message, and put them at the end of “ciphertext.” Then we take the first “c” characters out of the original message, and put them at the end of “ciphertext.” We take the last “a” characters out of the original message, and put them at the end of “ciphertext.” We cycle through “a,” “b,” and “c” and alternate between taking characters from the start and the end. Once the number of remaining characters are less than the key currently in use, we put the rest at the end of “ciphertext.” We have now encrypted our message.

In decryption, each the order of the steps is reversed. We undo the second transposition first. Therefore, the first step in decryption uses the last three characters of the key, translated to decimal form. We will split the encrypted message into two smaller messages. We will distinguish them from each other by using the terms “front,” and “back.”

Suppose the three numbers from the key are “a,” “b,” and “c.” To undo this step, we take the first “a” characters from the ciphertext, and put them at the end of “front.” Then we take the next “b” characters and put them before the beginning of “back.” Then we take the next “c” characters from the ciphertext and put them at the end of “front.” Then we take the next “a” characters from the ciphertext and put them before the beginning of “back.” This pattern is continued until the length of the rest of the message is less than the current key being used. Then the partially decrypted message is acquired by putting the front, the rest of the message, and the back together in that order. Generally, we start with the first number from the key and take that many characters from the ciphertext, alternating between putting it at the end of “front” and the beginning of “back,” starting with “front.” Then the next key is used, and we cycle through the keys.

Remember that one way to undo any transposition is to transpose a made up message and see where each character went, and then to pull the transposed message back accordingly. It works like the matching section of a test. This is not the most efficient way to undo the transposition, but it definitely works and is easy to understand.

Now each character from the partially decrypted text is transformed into the bits according to the ASCII table. The next step in decryption is to undo the second step of encryption. We will explain how this works using some mathematical terms. Assuming the sixth character of the key has a decimal form “k,” we will say that x is the length of the partially decrypted text divided by 2*k, and if the remainder “r” is greater than k, then add r - k to x. Therefore, if $r <= k$, the length of the partially decrypted text is given by $2kx + r$, and if $r > k$, it is $2k(x - r + k) + r$. The number x is important because it tells us the “halfway point” of the message. This tells us the number of bits that were “left alone” in the second step of encryption. Once again, we will split the partially decrypted message into two messages. “Front” is the first x bits of the partially decrypted message, and “back” is the rest of the partially decrypted message. We also create a separate, empty message called “final.” Then to reconstruct the original series of bits, we take the first k bits from “back,” and put them in “final.” Then we take the first k bits from “front” and put them at the end of“final.” Then we take the next k bits from “back” and put them at the end of “final,” the next k bits from “front” and put them at the end of “final.” So we go back and forth, taking the first k bits from “back,” then “front,” (removing them) and then placing them at the end of “final.”

The final step in decrypting is to undo the first part of encryption. We will take thirty-five bits at a time from the partially decrypted text. The original seven bits are the first bit, the ninth bit, the seventeenth bit, the twenty-fifth bit, the thirty-third bit, the thirteenth bit, and the seventh bit of the thirty-five bits in that order. Once we have all of the original bits, we can get the original message by taking seven bits at a time and transforming them back into characters using the modified ASCII table.

We have described a cryptosystem that encrypts mostly using transpositions. The message is transformed to bits through a modified ASCII system, the message is stretched to a message five times as long, and then the bits are moved around, and transformed back into characters, which are then extremely different because the blocks of seven bits were split up and put in different places. Then the characters are transposed again. The key needed to decrypt has a hundred million different configurations, which strong for a ten digit key. Therefore, the encrypted information will remain secure and, even if others knew the complex cryptosystem, it would be incredibly difficult to decrypt.

EXAMPLES, ENCRYPTION PART ONE:
We will use 1010011101010010000011011001 and a key of “stars” which we transform into:
s = 1110011
t = 1110100
a = 1100001
r = 1110010
s = 1110011
For future reference, the first seven digits of the plain text bits are 1010011, the next seven are 1010100, the third group of seven is 1000001, and the last seven are 1011001.
We work one byte at a time. For a byte “abcdefg,” we encode using the key “stars” as follows:
s* = a11001g
t* = 1b101f0
a* = 11c0001
r* = 111d010
s* = 1110e11
And then our resulting byte is encrypted by s* t* a* r* s*
In this case, we obtain
s* = 1110011
t* = 1010110
a* = 1110001
r* = 1110010
s* = 1110011
And we get 11100111010110111000111100101110011
We repeat for the bytes 1010100, 1000001, and 1011001.
s* = 1110010
t* = 1010100
a* = 1110001
r* = 1110010
s* = 1110111
11100101010100111000111100101110111

s* = 1110011
t* = 1010100
a* = 1100001
r* = 1110010
s* = 1110011
11100111010100110000111100101110011

s* = 1110011
t* = 1010100
a* = 1110001
r* = 1111010
s* = 1110011

11100111010100111000111110101110011

So our partially encrypted message that originally was 1010011101010010000011011001 becomes
11100111010110111000111100101110011111001010101001110001111001011101111110011101010011000
0111100101110011111001110101001110001111010101110011

EXAMPLES, DECRYPTION TO FIRST PART:
In order to decrypt, we split the series of bits into smaller sets of 35 bits.
11100111010110111000111100101110011
11100101010100111000111100101110111
11100111010100110000111100101110011
11100111010100111000111110101110011

We will work one set at a time. We split the first set into five sets of seven bits.
1110011
1010110
1110001
1110010
1110011
Remembering the positions of the original bits based on the general map:
a*****g
*b***f*
c**
d
****e**
The original bits were 1010011.
We repeat for the other sets of bits.

11100101010100111000111100101110111 becomes
1110010
1010100
1110001
1110010
1110111
And the original bits were 1010100.

11100111010100110000111100101110011 becomes
1110011
1010100
1100001
1110010
1110011
And the original bits were 1000001.

11100111010100111000111110101110011 becomes
1110011
1010100
1110001
1111010
1110011
And the original bits were 1011001.
Therefore, the original message was:
1010011 1010100 1000001 1011001
Which we can compare this with the original:
1010011 1010100 1000001 1011001
We have successfully encrypted and decrypted this message.

EXAMPLES, ENCRYPTION PART TWO:
Partially encrypted bits: 110100010101011101010111010010111110101010101010010
First we are going to put a marker at the end, we will use for the sake of being able to see it the character *.
Let's say the corresponding part of our key says "5," so we will select the first five bits and put them after the asterisk, deleting them from the left side of the asterisk.
00101010111010101110100101111101010101010010*11010
Notice that the “11010” has been moved from the front of the message to the part after the asterisk in its original order.
We will leave the next five where they are. We can put an ampersand after them to indicate that they are fixed. Then we select the five after them, which is "01011," and put them after "11010."
00101&1010101110100101111101010101010010*1101001011
Place an ampersand five characters after the first ampersand, select the five letters after that, and place them at the end.
00101&10101&1001011111010101010010*110100101101110
Just keep going.
00101&10101&10010&01010101010010*11010010110111011111
00101&10101&10010&01010&0010*1101001011011101111110101
Since the length of the rest of the characters before the asterisk is four, we're done. If we were to reach the asterisk while cut/pasting bits, we would cut/paste as many of the bits as we could - even if it was just the last two - three.
Since we're done, we remove the ampersands and the asterisk.
00101101011001001010001011010010110111011111110101

We will do another example.
Partially encrypted bits: 00101011101010110100010101101101010101110111010010111010
First we put the asterisk at the end.
00101011101010110100010101101101010101110111010010111010*
Let's say this time the key indicates each block should have a length of 3.

Each step of the process will be completed here but without much explanation.
010&1110101011010001010110110101010111011101001011101 0*001
010&010&1011010001010110110101010111011101001011101 0*001111
010&010&101&0001010110110101010111011101001011101 0*001111101
010&010&101&101&01101101010101110111010010111010*001111101000
010&010&101&101&011&010101011101110100101110 10*001111101000011
010&010&101&101&011&101&01110111010010111010*001111101000011010
010&010&101&101&011&101&101&11010010111010*001111101000011010011
010&010&101&101&011&101&101&100&10111010*001111101000011010011110
010&010&101&101&011&101&101&100&110&10*001111101000011010011110101
010&010&101&101&011&101&101&100&110&*001111101000011010011110101 10
010010101101011101101100110001111101000011010011110101 10

This step disrupts the bits in case of a frequency of resulting characters.

EXAMPLES, DECRYPTION TO SECOND PART:
We will demonstrate decryption by reproducing
"0010101110101011010001010110110101010111011101001011101 0" from
"010010101101011101101100110001111101000011010011110101 10" and knowing that the key is 3.
The first thing that we do is calculate how long the first part and the second part must have been (indicated by the two sides of the asterisk).
010&010&*101*&101&*011*&101&*101*&100&*110*&001&*111*&101&*000*&011&*010*&011&*110*&101&*10*
The sets of bits between asterisks would be moved to the end, and the sets of bits between ampersands would remain where they are. Therefore, for a set of bits this length, there should be nine sets of bits that stay on the left half of the bits, since there are nine sets of bits between ampersands. This is equivalent to the division mentioned in the paper. This is a more visual representation.
The first nine sets of bits are *010*&010&*101*&101&*011*&101&*101*&100&*110*, so we can remove all the special characters in this set of bits.
010010101101011101101100110
And notice that the remainder of the message is
&001&*111*&101&*000*&011&*010*&011&*110*&101&*10*, However, we will leave the special characters here. We will place the first half of the message in front.
010010101101011101101100110&001&*111*&101&*000*&011&*010*&011&*110*&101&*10*
Notice that the first set of bits surrounded by special characters is &001&. We will place this at the front of the message, delete the special characters, and place the marker (*) three bits after the end of 001.
001010(*)010101101011101101100110*111*&101&*000*&011&*010*&011&*110*&101&*10*
Therefore the (*) is placed after the first six bits. The next set enclosed by asterisks or ampersands is *111*. So we will place *111* after the (*) and remove the asterisks, and place another (*) three bits after the end of *111*.
001010(*)111010(*)101101011101101100110&101&*000*&011&*010*&011&*110*&101&*10*
We proceed according to this pattern.
001010(*)111010(*)101101(*)1010111011011001 10*000*&011&*010*&011&*110*&101&*10*

001010(*)111010(*)101101(*)000101(*)01110110110 0110&011&*010*&011&*110*&101&*10*

001010(*)111010(*)101101(*)000101(*)011011(*)101101100110*010*&011&*110*&101&*10*

001010(*)111010(*)101101(*)000101(*)011011(*)010101(*)101100110&011&*110*&101&*10*

001010(*)111010(*)101101(*)000101(*)011011(*)010101(*)011101(*)100110*110*&101&*10*

001010(*)111010(*)101101(*)000101(*)011011(*)010101(*)011101(*)110100(*)110&101&*10*

001010(*)111010(*)101101(*)000101(*)011011(*)010101(*)011101(*)110100(*)101110(*)*10*

Now, since the last set of characters were changed (and therefore originally at the end of the message), we can just erase all special characters to get:

0010101110101011010001010110110101010111011101001011 1010

Note that this is equivalent to

0010101110101011010001010110110101010111011101001011 1010

Which I copy/pasted from the original, while the one above it I simply deleted the special characters. Therefore, we have achieved the original set of bits from the encrypted set and the key.

We will do it again for a slightly different case; we will use the first set of bits we encrypted this time. The original set of bits is 11010001010101110101011101001011111010101010 10010, the key is five, and the encrypted set of bits is 001011010110010010100010110100101101110111111 0101.

00101&10101&*10010*&01010&*00101*&10100&*10110*&11101&*11111*&0101&

This time we have run into an issue. Since the last four bits are between ampersands, and four is not five, we will have to re-do some of the symbols to adjust. However, we first count that there are four complete sets between ampersands, so the first four sets of bits are fine and we can delete the special characters.

00101101011001001010

Since there would be four more bits, we take the next four bits from the next set and place them at the end of this re-transposition.

001011010110010010100010

Now we take all the rest of the bits from the original encrypted message and delete the special characters.

1101001011011101111110101

We will take the first five of these bits and put them on the front of the re-transposition. We mark the spot five bits after the end of this addition with (*).

1101000101(*)10101100100101000 10

The rest of the characters to add are 01011011101111110101. Take the next five, place them after the last (*), and put another (*) five bits after the end of the recently inputted five bits.

1101000101(*)0101110101(*)10010010100010

Continue with the pattern.

011101111110101

1101000101(*)0101110101(*)0111010010(*)010100010

1111110101

1101000101(*)0101110101(*)0111010010(*)1111101010(*)0010

10101

1101000101(*)0101110101(*)0111010010(*)1111101010(*)101010010

Now remove all of the (*) and we are done.
1101000101010111010101110100101111101010101010010
Notice that it is equivalent to
1101000101010111010101110100101111101010101010010
Which we said it would be.

EXAMPLES, ENCRYPTION PART THREE
We will use three numbers from the key. So if the corresponding section of the key was 3, 4, 2, then it would work like this:
Plain text: THISISAMESSAGE
Our first step is to take the first three characters at the beginning of the plain text and to put them as they are into our cipher text.
Plain text: SISAMESSAGE
Cipher text: THI
Our next step is to take the last four characters at the end of the plain text and to put them as they are and attach them to the end of the cipher text.
Plain text: SISAMES
Cipher text: THISAGE
Then we take the first two characters at the beginning of the plain text and attach them to the end of the cipher text.
Plain text: SAMES
Cipher text: THISAGESI
Then we take the last three characters at the end of the plain text and attach them to the end of the cipher text.
Plain text: SA
Cipher text: THISAGESIMES
Since the next step would be to take the first four characters at the beginning of the plain text, but there are less than four characters left, we will take all of them and put them at the end.
Plain text:
Cipher text: THISAGESIMESSA
We have encrypted the message.
Some details to note: Each of the numbers in the key refers to how many characters are taken at a time. First it is three, then four, then two, then three, then four, two, and so on, repeating in this fashion.
The first time characters are taken from the plain text, they are taken from the beginning, and then they are taken from the end, and then they are taken from the beginning.
While this does not effectively scramble a message that much, because this is the last step, the message already does not make very much sense and is a meaningless jumble of characters. We will do a second example that demonstrates this.

Suppose we have the following:
Partially encrypted text: aSjfI(43$0- \?/BnT,wQPty091Vcbr2[:'os3UhjblK5.!eI
Key: 592

This means that we will take five characters, nine characters, two characters, and repeat.
Partially encrypted text: aSjfI(43$0- \?/BnT,wQPty091Vcbr2[:'os3UhjblK5.!eI

Partially encrypted text: (43$0- \?/BnT,wQPty091Vcbr2[:'os3UhjblK5.!eI
Cipher text: aSjfI
Partially encrypted text: (43$0- \?/BnT,wQPty091Vcbr2[:'os3Uh
Cipher text: aSjfIjblK5.!eI
Partially encrypted text: 3$0- \?/BnT,wQPty091Vcbr2[:'os3Uh
Cipher text: aSjfIjblK5.!eI(4
Partially encrypted text: 3$0- \?/BnT,wQPty091Vcbr2[:'
Cipher text: aSjfIjblK5.!eI(4os3Uh
Partially encrypted text: nT,wQPty091Vcbr2[:'
Cipher text: aSjfIjblK5.!eI(4os3Uh3$0- \?/B
Partially encrypted text: nT,wQPty091Vcbr2[
Cipher text: aSjfIjblK5.!eI(4os3Uh3$0- \?/B:'
Partially encrypted text: Pty091Vcbr2[
Cipher text: aSjfIjblK5.!eI(4os3Uh3$0- \?/B:'nT,wQ
Partially encrypted text: Pty
Cipher text: aSjfIjblK5.!eI(4os3Uh3$0- \?/B:'nT,wQ091Vcbr2[
Partially encrypted text: y
Cipher text: aSjfIjblK5.!eI(4os3Uh3$0- \?/B:'nT,wQ091Vcbr2[Pt
Partially encrypted text:
Cipher text: aSjfIjblK5.!eI(4os3Uh3$0- \?/B:'nT,wQ091Vcbr2[Pty

EXAMPLES, DECRYPTION TO THIRD PART:
Now we will decrypt these messages. In the plain text "THISISAMESSAGE" there are 14 characters.
We will create a dummy text that is 14 characters long as follows:
"*1* *2* *3* *4* *5* *6* *7* *8* *9* *10* *11* *12* *13* *14*"
Notice that if we encrypt this with the key, it will tell us where each number goes. Therefore, to decrypt, we take the letter at each position and put it in the order of its corresponding number.
This can also be used to encrypt.

Dummy text: *1* *2* *3* *4* *5* *6* *7* *8* *9* *10* *11* *12* *13* *14*
Encrypted dummy text:

Dummy text: *4* *5* *6* *7* *8* *9* *10* *11* *12* *13* *14*
Encrypted dummy text: *1* *2* *3*

Dummy text: *4* *5* *6* *7* *8* *9* *10*
Encrypted dummy text: *1* *2* *3* *11* *12* *13* *14*

Dummy text: *6* *7* *8* *9* *10*
Encrypted dummy text: *1* *2* *3* *11* *12* *13* *14* *4* *5*

Dummy text: *6* *7*
Encrypted dummy text: *1* *2* *3* *11* *12* *13* *14* *4* *5* *8* *9* *10*

Dummy text:
Encrypted dummy text: *1* *2* *3* *11* *12* *13* *14* *4* *5* *8* *9* *10* *6* *7*

Now we need the encrypted text. We will line it up with our encrypted dummy text.
T H I S A G E S I M E S S A
01 02 03 11 12 13 14 04 05 08 09 10 06 07
Therefore, in re-organizing our message, we do the following:
T H I S I S A G E M E S S A
01 02 03 04 05 11 12 13 14 08 09 10 06 07

T H I S I S A S A G E M E S
01 02 03 04 05 06 07 11 12 13 14 08 09 10

T H I S I S A M E S S A G E
01 02 03 04 05 06 07 08 09 10 11 12 13 14

And we have reproduced our original plain text, "THISISAMESSAGE."
Additionally, we will demonstrate a different system to do this on the other encrypted message:
aSjfIjblK5.!eI(4os3Uh3$0- \?/B:'nT,wQ091Vcbr2[Pty
Notice that the first five characters must still be the same, but the next nine characters came from the end. We will split these two sets of characters into their own sequences.
(4os3Uh3$0- \?/B:'nT,wQ091Vcbr2[Pty
aSjfI
jblK5.!eI
The next two should follow the first five, and the five after that should precede the last nine.
3$0- \?/B:'nT,wQ091Vcbr2[Pty
aSjfI(4
os3UhjblK5.!eI
The next nine should follow the first seven characters, and the two after that should precede the last fourteen.
nT,wQ091Vcbr2[Pty
aSjfI(43$0- \?/B
:'os3UhjblK5.!eI
The next five should follow our first split set of characters, and the nine after that precede the second.
Pty
aSjfI(43$0- \?/BnT,wQ
091Vcbr2[:'os3UhjblK5.!eI
The next two should follow the first set, and the remaining character (since it is less than the amount we are taking) precedes the second set.

aSjfI(43$0- \?/BnT,wQPt
y091Vcbr2[:'os3UhjblK5.!eI
Now we place the second half after the first half:
aSjfI(43$0- \?/BnT,wQPty091Vcbr2[:'os3UhjblK5.!eI
And we can compare with the original character sequence:
aSjfI(43$0- \?/BnT,wQPty091Vcbr2[:'os3UhjblK5.!eI
The result is the same! Therefore, we have encrypted and successfully decrypted.

Sample problem
Encrypt the message “Is this on?” (without the quotes) with the key abcdeäåéü.
You should get “Ö{ÉôJ[ëgÆl@rfmH¡Å>[É£¢[0H¡é'l@Ä/n9köÉ!¥{ÉôlaJpn|@üg{V¡,”

Appendix
Modified ASCII Table

¡ 0000000	É 0010000	(space) 0100000	0 0110000	@ 1000000	P 1010000	` 1100000	p 1110000
ü 0000001	æ 0010001	! 0100001	1 0110001	A 1000001	Q 1010001	a 1100001	q 1110001
é 0000010	Æ 0010010	“ 0100010	2 0110010	B 1000010	R 1010010	b 1100010	r 1110010
â 0000011	ô 0010011	# 0100011	3 0110011	C 1000011	S 1010011	c 1100011	s 1110011
ä 0000100	ö 0010100	$ 0100100	4 0110100	D 1000100	T 1010100	d 1100100	t 1110100
à 0000101	ò 0010101	% 0100101	5 0110101	E 1000101	U 1010101	e 1100101	u 1110101
å 0000110	û 0010110	& 0100110	6 0110110	F 1000110	V 1010110	f 1100110	v 1110110
ç 0000111	ù 0010111	‘ 0100111	7 0110111	G 1000111	W 1010111	g 1100111	w 1110111
ê 0001000	ÿ 0011000	(0101000	8 0111000	H 1001000	X 1011000	h 1101000	x 1111000
ë 0001001	Ö 0011001	) 0101001	9 0111001	I 1001001	Y 1011001	i 1101001	y 1111001
è 0001010	Ü 0011010	* 0101010	: 0111010	J 1001010	Z 1011010	j 1101010	z 1111010
ï 0001011	¢ 0011011	+ 0101011	; 0111011	K 1001011	[1011011	k 1101011	{ 1111011
î 0001100	£ 0011100	, 0101100	< 0111100	L 1001100	\ 1011100	l 1101100	| 1111100
ì 0001101	¥ 0011101	- 0101101	= 0111101	M 1001101	] 1011101	m 1101101	} 1111101
Ä 0001110	á 0011110	. 0101110	> 0111110	N 1001110	^ 1011110	n 1101110	~ 1111110
Å 0001111	í 0011111	/ 0101111	? 0111111	O 1001111	_ 1011111	o 1101111	ñ 1111111