Group T

Progressive Key Cipher Encryption

[redacted]

Introduction

Information governs the world around us; it is used in marketing, strategy planning, product development, business management, job hiring, and even restaurant recommendations. This in turn makes information-based security breaches a major threat to individuals and organizations alike. Our goal for this project is to minimize the damage dealt to any party in the event of a zero-day attack or an unintentional leak of information.

How Our System Works

We call our cryptosystem the "Progressive Key Cipher". We start with the key $K = K_1K_2K_3 \cdots K_n$, which is an alphabetic string of length $2 \leq n \leq 10$. Let P be a string of alphabetic plaintext. For our purposes we will ignore non-alphabetic characters—we encrypt the plaintext as if they were not there. Define the bijective mapping μ from the English alphabet to the integers modulo 26 by assigning each letter to the equivalence class of the number representing its position in alphabetical order (e.g. $\mu(A) = 0, \mu(B) = 1, \mu(C) = 2, \dots, \mu(Z) = 25$). We now outline the procedure by which we encode the plaintext using the key:

1. Break P into chunks of characters $P_1, P_2, P_3, \dots, P_k$ of length $n - 1$. It is okay if the last chunk does not have $n - 1$ characters.
2. Apply the key to the first chunk $P_1 = C_1C_2C_3 \cdots C_{n-1}$ by mapping each character C_i to the character $\mu^{-1}\big(\mu(C_i) + \mu(K_i)\big)$. So we shift the i'th character of the chunk by the i'th letter of the key.
3. Now that we have applied the key to the first chunk, we use the n'th character of the key to alter the key for the second chunk. To do this, we add the position of the final character in the key to the position of each character in the key (including the final character) and reduce modulo 26. In other words, C_i becomes $\mu^{-1}\big(\mu(K_i) + \mu(K_n)\big)$. Set K equal to the new key.
4. Apply K to the next chunk of plaintext as in step 2, and afterward generate the new key using the process described in step 3.
5. Repeat step 4 until you have encrypted all of the chunks.

Let E be a string of Ciphertext and let K be the original key. Decryption is essentially the above process, but reversed:

1. Break E into chunks of $n - 1$ characters as before.

2. Decrypt the first chunk $E_1 = C_1C_2C_3 \cdots C_{n-1}$ by subtracting the corresponding K_i alphabetic position from the C_i alphabetic position. In other words, plaintext C_i will be equal to $\mu^{-1}(\mu(C_i) - \mu(K_i))$.
3. Generate the key for the next chunk exactly as in step 3 of the encryption process, and decrypt the next chunk using the new key following the process in step 2.
4. Repeat until all chunks are decrypted.

A Worked Example

To demonstrate our cryptosystem, we will work through a simple example. Suppose our key text is "CODE", and our plaintext is "CRYPTO FUN". We begin by breaking the plaintext into chunks of 3 letters. We have "CRY-PTO-FUN". We start with the first chunk, "CRY". Adding the position values of corresponding letters in the key "COD", we obtain cyphertext "EFB" for our first chunk. The following table outlines these computations in greater detail.

Plaintext Character	**C**	**R**	**Y**
Position in Alphabet (0-25)	2	17	24
Key Character	**C**	**O**	**D**
Position in Alphabet (0-25)	2	14	3
New Position in Alphabet (0-25)	2 + 2 = **4** (mod 26)	17 + 14 = **5** (mod 26)	24 + 3 = **1** (mod 26)
Cyphertext Character	**E**	**F**	**B**

Now we change the key. To change the key, we look at the last letter of the key, "E", which has position 4 in the alphabet. We then add this to the position of the first 3 letters to obtain our new, shifted key, "GSHI".

Key Character	**C**	**O**	**D**	**E**
Position in Alphabet (0-25)	2	14	3	4
New Position in Alphabet (0-25)	2 + 4 = **6** (mod 26)	14 + 4 = **18** (mod 26)	3 + 4 = **7** (mod 26)	4 + 4 = **8** (mod 26)
New Key Character	**G**	**S**	**H**	**I**

Now we apply the new key to the next chunk, "PTO". Adding the corresponding alphabetic positions of "GSH", we obtain the encrypted chunk "VLV".

Plaintext Character	**P**	**T**	**O**
Position in Alphabet (0-25)	15	19	14

Key Character	**G**	**S**	**H**
Position in Alphabet (0-25)	6	18	7
New Position in Alphabet (0-25)	15 + 6 = **21** (mod 26)	19 + 18 = **11** (mod 26)	14 + 7 = **21** (mod 26)
Cyphertext Character	**V**	**L**	**V**

To make the key for the next chunk, we add the position of the last letter ("I") of the new key, "GSHI", to the position of all the letters in the key.

Key Character	**G**	**S**	**H**	**I**
Position in Alphabet (0-25)	6	18	7	8
New Position in Alphabet (0-25)	6 + 8 = **14** (mod 26)	18 + 8 = **0** (mod 26)	7 + 8 = **15** (mod 26)	8 + 8 = **16** (mod 26)
New Key Character	**O**	**A**	**P**	**Q**

Then we apply this key, "OPAQ", to the next chunk, "FUN". That gives us "TUV". Hence, our encrypted ciphertext is "EFBVLV TUV". To decrypt this text, simply reverse the operations shown in the table.

Conclusion

Our cryptosystem aims to minimize the risk of a security breach or leakage by preventing an unauthorized agent from extracting important information quickly. The pattern is difficult to detect, and is resistant to low-level frequency analysis attacks. Unless a more sophisticated attack is employed, an unauthorized agent will not be able to break the cipher in a short amount of time. Changing the key frequently will further ensure the confidentiality of the encrypted data. Thus, we can confidently say that this achieves our goal for this cryptosystem—to minimize the damage dealt to any agent at the event of a zero-day attack or an unintentional leak of information. It is imperative to recognize the danger of putting the safeguard of information solely on one system, so we strongly recommend the use of other security measures in addition to our cryptosystem.