APPLICATION OF ADDITIVE GROUP COSETS OF INTEGER SETS IN CRYPTOGRAPHIC ALGORITHMS

Sherly Wulandari[1], Suroto[2], Bambang Hendriya Guswanto[3], Lilik Muzdalifah[4], Glagah Eskacakra Setyowisnu[5], and Triyani[6]

Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Jenderal Soedirman, Purwokerto, Central Java

*Email: suroto@unsoed.ac.id

Abstract. Cryptography is a field of science that studies information and communication security by converting messages into an unreadable, unalterable form and confirming the identities of the sender and recipient. Cosets of a group are sets formed by the action of a subgroup on elements of the group, yielding equivalence classes that partition the group into several mutually disjoint subsets. This research discusses the application of additive group cosets, namely the set of all integers Z, to the design of cryptographic algorithms. In this research, the subgroup used to form the cosets is the $60\mathbb{Z}$ subgroup. Furthermore, the set of $\mathbb{Z}/60\mathbb{Z}$ cosets will be used in the design of cryptographic algorithms for encryption and decryption. The cryptographic algorithm design is implemented as an encryption-decryption program in C++, using the group cosets approach. The research results are a cryptographic algorithm design based on the coset set $\mathbb{Z}/60\mathbb{Z}$, implemented in C++. The use of cosets on $\mathbb{Z}/60\mathbb{Z}$ can strengthen the security of the cryptographic algorithm by dividing messages into equivalence classes, making it more difficult to recover messages without the correct key.

Keywords: Group, cosets, cryptography, encryption, decryption

1. Introduction

In mathematics, group theory is a branch of abstract algebra that has applications in cryptography. One important concept in group theory is the concept of group cosets, which are used to understand the algebraic structure of a system and its applications across various fields. Group cosets classify elements into equivalence classes based on their relations to subgroups, thereby enabling the analysis of more complex group structures. In cryptography, understanding group structures and group cosets is an important foundation for the development of secure and efficient encryption systems [1]. Group cosets themselves play a role in digital authentication and zero-knowledge proof protocols, and are used in code-based cryptography to obfuscate encryption structures. The discussion of the application of group in cryptography can be found in [2-6]. Then, the discussion of computational group and cryptography in [7], group theory in lattice-based cryptography in [8], and post-quantum group-based cryptography in [9]. In addition, the combination of the Vigenere cipher with the zig-zag method, as used by [10], remains vulnerable to frequency analysis, especially when the key length is limited or the pattern is repeated many times. This method also lacks a complex mathematical basis, as it relies only on substitution and transposition.

In cryptography discussions, one commonly used group is the additive group Z_{26}. The use of the additive group Z_{26} is based on the number of letters in the Latin alphabet, namely, **a** to **z**. Several cryptographic studies have used the Z_{26} group in developing encryption techniques.

The classical cryptography using the Hill Cipher method is more vulnerable to attacks due to its linear nature [11]. Furthermore, [12] discusses classical cryptography with the additive group Z_{26} cosets. The Z_{26} group only contains letters of the alphabet without being able to distinguish between uppercase and lowercase letters, and several other characters.

This research will discuss the use of additive group Z cosets by selecting the $60Z$ subgroup to expand the key space and increase ciphertext variety in [12]. This approach allows for a wider character set, including uppercase and lowercase letters and several symbols, and is expected to increase the system's resistance to cryptanalytic attacks. In addition, the use of an additive group Z provides flexibility in forming more cosets than Z_{26}.

2. Methods

The research method used in this study is quantitative, with two approaches: a literature study and a case study on cryptographic algorithms, focusing on the concept of additive group cosets of the set of integers Z. The steps of research are:

1. Explaining the cosets on the set of integers Z, namely $Z/60Z$;
2. Explaining some properties of cosets;
3. Applying the member cosets onto design cryptographic algorithms in the encryption;
4. Applying the member cosets onto design cryptographic algorithms in the decryption;
5. Creating codes for encryption using the C++ programming language;
6. Creating codes for decryption using the C++ programming language.

.

3. Results And Discussion

3.1. Cosets in the Additive Group of Integers

In the additive group of the set of integers Z, subgroups can be formed by finding the elements that constitute the subgroup. In general, the subgroup of the additive group of the set of integers Z is nZ, where n is an integer. In this study, the subgroup used is

$$60Z = \{60z | z \in Z\} = \{..., -120, -60, 0, 60, 120, ...\}.$$

This study uses the $60Z$ subgroup because it will use 60 characters: capital letters, lowercase letters, and several frequently used important symbols: period (.), comma (,), exclamation mark (!), question mark (?), underscore (_), dash (-), at (@), and asterisk (*).

Since the set $60Z$ is a subgroup of Z, we can construct cosets of the subgroup $60Z$ in the additive group Z, namely the left and right cosets of $60Z$. Since the additive group Z is an Abel group, the subgroups in the additive group Z are normal subgroups. This means that the left cosets and the right cosets obtained from each subgroup are always equal. In this study, the cosets formed from each subgroup in the additive group Z are the left cosets. The general form of cosets of the subgroup $60Z$ in the additive group Z is

$$a + 60Z = \{a + s | s \in 60Z\}.$$

In general, the set of all integers Z is partitioned into 60 partitions as follows:

$$0 + 60Z, 1 + 60Z, 2 + 60Z, \ldots, 58 + 60Z \text{ and } 59 + 60Z.$$

The set of all cosets of the $60Z$ subgroup in the additive group Z is

$$Z/60Z = \{0 + 60Z, 1 + 60Z, 2 + 60Z, ..., 58 + 60Z, 59 + 60Z\}.$$

Since $Z/60Z$ is a group, then, the mapping

$$f: Z/60Z \rightarrow Z_{60}$$

where $f(a + 60Z) = \bar{a}$ for all $a + 60Z \in Z/60Z$ is a group isomorphism. Thus, the structure of $Z/60Z$ is isomorphic to Z_{60}, which means that every $a + 60Z \in Z/60Z$ corresponds one-to-one with $\bar{a} \in Z_{60}$.

3.2. Application of Additive Group Cosets Integers in Cryptographic Algorithm Design

The group used in this algorithm design is the additive group Z, while the selected subgroup is $60Z$. The steps in this research are as follows:

1. Select the plaintext, namely "Mynameis@Sherly_Wulandari."
2. Perform the encryption process.
   a. Encode the plaintext characters into decimal numbers.

Table 1. Character Representation in Decimal Numbers

a	b	c	...	z	A	B	C	...	Z	.	,	!	?	_	-	@	*
0	1	2	...	25	26	27	28	...	51	52	53	54	55	56	57	58	59

Then, for plaintext "Mynameis@Sherly_Wulandari", the value of characters.

Table 2. Plaintext characters in decimal numbers

Characters	M	y	n	a	M	e	i	s	@	S	h	E	r	l	y
Value	38	24	13	0	12	4	8	18	58	44	7	4	17	11	24
Characters	_	W	u	l	A	n	d	a	r	I					
Value	56	48	20	11	0	13	3	0	17	8					

   b. Inputting the encryption key selected from the relatively prime numbers 60, namely 1, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 49, 53, and 59 to obtain cosets. Among all numbers are relatively prime to 60. Without reducing the generalization, we choose $h = 7$. The cosets obtained for $h = 7$ is

$$\{0 \cdot 60, 1.60, 2 \cdot 60, \ldots, (h-1) \cdot 60\} = \{0, 60, 120, 180, 240, 300, 360\}.$$

   c. Adding the code of each character in the plaintext in step (a) with the elements in cosets in step (b).

Table 3. Adding the plaintext character code with coset elements

38	$38 + \{0, 60, 120, 180, 240, 300, 360\} = \{38, 98, 158, 218, 278, 338, 398\}$
24	$24 + \{0, 60, 120, 180, 240, 300, 360\} = \{24, 84, 144, 204, 264, 324, 384\}$
13	$13 + \{0, 60, 120, 180, 240, 300, 360\} = \{13, 73, 133, 193, 253, 313, 373\}$
0	$0 + \{0, 60, 120, 180, 240, 300, 360\} = \{0, 60, 120, 180, 240, 300, 360\}$
12	$12 + \{0, 60, 120, 180, 240, 300, 360\} = \{12, 72, 132, 192, 252, 312, 372\}$
4	$4 + \{0, 60, 120, 180, 240, 300, 360\} = \{4, 64, 124, 184, 244, 304, 364\}$
8	$8 + \{0, 60, 120, 180, 240, 300, 360\} = \{8, 68, 128, 188, 248, 308, 368\}$
⋮	⋮
11	$11 + \{0, 60, 120, 180, 240, 300, 360\} = \{11, 71, 131, 191, 251, 311, 371\}$
0	$0 + \{0, 60, 120, 180, 240, 300, 360\} = \{0, 60, 120, 180, 240, 300, 360\}$

13	$13 + \{0, 60, 120, 180, 240, 300, 360\} = \{13, 73, 133, 193, 253, 313, 373\}$
3	$3 + \{0, 60, 120, 180, 240, 300, 360\} = \{3, 63, 123, 183, 243, 303, 363\}$
0	$0 + \{0, 60, 120, 180, 240, 300, 360\} = \{0, 60, 120, 180, 240, 300, 360\}$
17	$17 + \{0, 60, 120, 180, 240, 300, 360\} = \{17, 77, 137, 197, 257, 317, 377\}$
8	$8 + \{0, 60, 120, 180, 240, 300, 360\} = \{8, 68, 128, 188, 248, 308, 368\}$

d. Replacing each message character code with any character code in the same cosets.

Table 4. Replacing each message character code

38	$38 + \{0, 60, 120, 180, 240, 300, 360\} = \{38, 98, 158, 218, 278, 338, 398\}$	338
24	$24 + \{0, 60, 120, 180, 240, 300, 360\} = \{24, 84, 144, 204, 264, 324, 384\}$	384
13	$13 + \{0, 60, 120, 180, 240, 300, 360\} = \{13, 73, 133, 193, 253, 313, 373\}$	13
0	$0 + \{0, 60, 120, 180, 240, 300, 360\} = \{0, 60, 120, 180, 240, 300, 360\}$	60
12	$12 + \{0, 60, 120, 180, 240, 300, 360\} = \{12, 72, 132, 192, 252, 312, 372\}$	132
4	$4 + \{0, 60, 120, 180, 240, 300, 360\} = \{4, 64, 124, 184, 244, 304, 364\}$	184
8	$8 + \{0, 60, 120, 180, 240, 300, 360\} = \{8, 68, 128, 188, 248, 308, 368\}$	248
	⋮	⋮
11	$11 + \{0, 60, 120, 180, 240, 300, 360\} = \{11, 71, 131, 191, 251, 311, 371\}$	131
0	$0 + \{0, 60, 120, 180, 240, 300, 360\} = \{0, 60, 120, 180, 240, 300, 360\}$	180
13	$13 + \{0, 60, 120, 180, 240, 300, 360\} = \{13, 73, 133, 193, 253, 313, 373\}$	253
3	$3 + \{0, 60, 120, 180, 240, 300, 360\} = \{3, 63, 123, 183, 243, 303, 363\}$	303
0	$0 + \{0, 60, 120, 180, 240, 300, 360\} = \{0, 60, 120, 180, 240, 300, 360\}$	360
17	$17 + \{0, 60, 120, 180, 240, 300, 360\} = \{17, 77, 137, 197, 257, 317, 377\}$	17
8	$8 + \{0, 60, 120, 180, 240, 300, 360\} = \{8, 68, 128, 188, 248, 308, 368\}$	68

e. Converting the code value to a letter character by calculating the number of digits in the number 60. h. If the number of digits in the code number in the plaintext is less than the number of digits in the 60. h, then insert the number 0 in front of the number.

Table 5. Inserting the number 0

Value	338	384	13	60	132	184	248	318	418
Result	338	384	013	060	132	184	248	318	418
Value	44	67	124	197	251	324	416	48	80
Result	044	067	124	197	251	324	416	048	080
Value	131	180	253	303	360	17	68		
Result	131	180	253	303	360	017	068		

f. If the digit is 0 then it is changed to character a. If the digit is not zero, combine it with the number after it. When the combined value is more than 60 , then take the first number and convert it to a letter character according to the mapping rules. If the

combined value is less than or equal to 60, then the combined value is converted to letter characters according to the mapping rules.

Table 6. Conversion of code digits to letter characters

Value	33	8	38	40	13	0	6	0	13	21	8	42	48	31	8	41	8
Character	H	i	M	O	n	a	g	A	n	v	i	Q	W	F	i	P	i
Value	0	44	0	6	7	12	41	9	7	25	13	24	41	6	0	48	0
Character	a	S	a	g	h	m	P	J	h	z	n	y	P	g	a	W	a
Value	8	0	13	11	8	0	25	33	0	33	6	0	0	17	0	6	8
Character	i	a	n	l	i	a	z	H	a	H	g	a	a	r	a	g	i

3. Obtain the ciphertext whose characters have been encoded:
   "HiMOnaganviQWFiPiaSaghmPjhznyPgaWaianliazHaHgaaragi".

   At this stage, the encrypted plaintext has been transformed into a ciphertext that appears random and difficult to understand. This ciphertext will be further processed to facilitate analysis during decryption.
4. Perform the decryption process using the following flow:
   a. Convert each letter in the ciphertext to a decimal number.

Table 7. Converting Ciphertext Characters to Decimal Numbers

Characters	H	i	M	O	n	a	g	a	n	v	i	Q	W	F	I	P	i
Value	33	8	38	40	13	0	6	0	13	21	8	42	48	31	8	41	8
Characters	a	S	a	g	h	m	P	j	h	z	n	y	P	g	a	W	a
Value	0	44	0	6	7	12	41	9	7	25	13	24	41	6	0	48	0
Characters	i	a	n	l	i	a	z	H	a	H	g	a	a	r	a	g	i
Value	8	0	13	11	8	0	25	33	0	33	6	0	0	17	0	6	8

   b. Enter the value h for the decryption key, with the same value as in the encryption process $h = 7$.
   c. Partition the decimal ciphertext characters into message blocks of $60.h$.

Table 8. Partitioning Ciphertext Characters

Value	338	384	013	060	132	184	248	318	418
Value	044	067	124	197	251	324	416	048	080
Value	131	180	253	303	360	017	068		

   d. In a block, if there is a series of 0 digits on the left, then the digits taken are the remaining number of digits. However, if a block contains all the digits, 0 then take the digit number 0.

Table 9. Retrieval of number digits

Value	338	384	13	60	132	184	248	318	418
Value	44	67	124	197	251	324	416	48	80

Value	131	180	253	303	360	17	68

e. Check each decimal number to see if it lies in a specific coset. Next, convert it to a number less than or equal to 60 in that coset.

Suppose a sequence of numbers is written in an array $B[i]$, where i is the index of the sequence. The transformation algorithm is as follows:

$$for\ i\ from\ 1 \rightarrow length(B)$$

$$for\ j\ from\ 0 \rightarrow 59$$

$$if\ (B[i] - j)\ mod\ 60 = 0$$

$$then\ C[i] = j.$$

$C[i]$ is the result of transforming $B[i]$. Then, the value of as is obtained in the following table.

Table 10. Results of the mod 60 algorithm

Value	38	24	13	0	12	4	8	18	58	44	7	4	17	11	24
Value	56	48	20	11	0	13	3	0	17	8					

f. Convert the decimal number obtained in the previous step to letters according to the mapping rules;

Table 11. Converting decimal numbers to letters

Value	38	24	13	0	12	4	8	18	58	44	7	4	17	11	24
Character	M	y	n	a	m	e	i	s	@	S	h	e	r	l	y
Value	56	48	20	11	0	13	3	0	17	8					
Character	_	W	u	l	a	n	d	a	r	i					

Then, we get “Mynameis@Sherly_Wulandari”.

3.3. **Implementation Using C++**

In this study, the programme was implemented using C++ and designed to encrypt and decrypt text using the additive group cosets Z. The program code and output generated from the encryption and decryption process using C++ are presented in full in the Appendix. In this study, a program was created to test the application of mathematical concepts to cryptographic algorithms. This program, implemented in C++, is designed to encrypt and decrypt text using the additive group cosets Z_{60}.

a. The programme output for encryption is as follows:

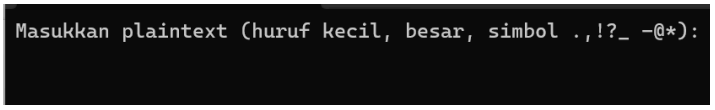

Figure 1. Input plaintext

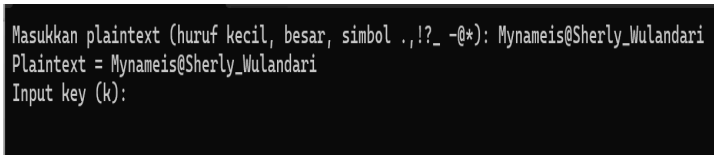

Figure *2*. Input key

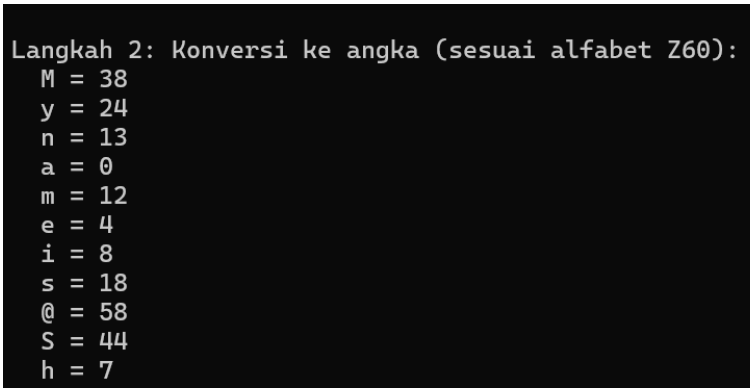

Figure 3. Plaintext processing into chipertext

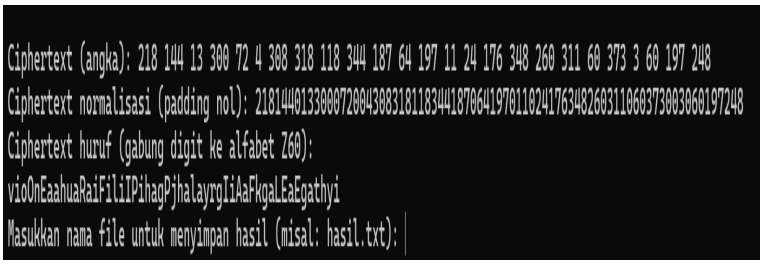

Figure 4. Save the filename of the chipertext

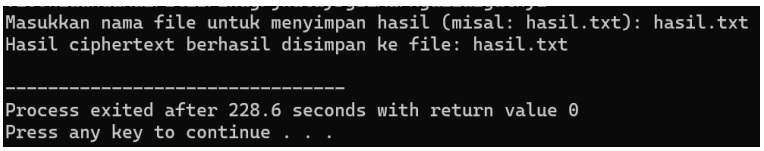

Figure 5. Save the filename of the result

b. The programme output for description is as follows:

Figure 6. Input the filename of the chipertext

Figure 7. Input key as in encryption

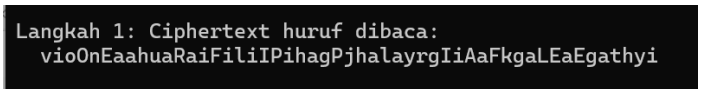

Figure 8. The result of descryption

All encrypted and decrypted data are presented to make it easier for readers to compare the program output with the manual calculations performed.

4. Conclusion

This research demonstrates that the application of the additive group Z cosets with the subgroup $60Z$ to a cryptographic algorithm can expand the key space, increase ciphertext variety, and support a wider range of character representations than the Z_{60} based method. Implementation in C++ demonstrates that the system can perform encryption and decryption accurately using a robust, fairly complex mathematical structure.

Suggestions for further research include exploring the use of non-trivial subgroups or more complex cosets to further enhance the security of the cryptographic system. This can also be achieved using other, more efficient programming languages, such as Python, which supports large-scale data processing, and should present a concise conclusion of the research and answer the objective of the study.

.

5. Acknowledgement

The authors were very grateful to LPPM Universitas Jenderal Soedirman for their funding support under the Riset Dasar Unsoed (Batch II) Tahun Anggaran 2025, No. 6.87/UN23.34/PT.01/V/2025.

References

[1]. Gençoğlu, M.T. Importance of Cryptography in Information Security. IOSR Journal of Computer Engineering, 2019. 21(1), 65–68

[2]. Gavirangaiah, K. Applications of Group Theory in Cryptographic Algorithms. IJFANS International Journal of Food and Nutritional Sciences. 2022. 11(11). 2011-2017

[3]. Asole, R.W. and Gaikwad, S.P. Applications of Group Theory in Cryptography. Gurukul International Multidisciplinary Research Journal (GIMRJ). 2025. IV(XIII). 236-239

[4]. Arora, P. Use of Group Theory in Cryptography. IJARIIE. 2016. 2 (6). 1767-1772

[5]. Blackburn, S.R., Cid, C., and Mullan, C. Group Theory in Cryptography.

arXiv:0906.5545v2 [math.GR] 25 Jan 2010.

[6]. Hasapis, S.D. and Panagopoulos, D. A Survey of Group-based Cryptography. Journal of Applied Mathematics & Bioinformatics. 2015. 5(3).73-96

[7]. Verma, M. and Kumar, S. Computational Group Theory and Cryptography: A Comprehensive Study. International Journal in Management and Social Science. 2021. 09(11). 1124-1130

[8]. John, M.N, Udoaka, O.G. and Udoakpan, I.U. Group Theory in Lattice-Based Cryptography. International Journal of Mathematics And its Applications. 2023. 11(4). 111-125

[9]. Kahrobaei, D., Flores, D, Noce, M., Habeeb, M.E. and Battarbee, C. Post-quantum Group-based Cryptography. Mathematical Surveys and Monographs. 2024. 278. 31-58

[10]. Fardianto, F.A. E., Yanto, F., Iskandar, I., & Pizaini. Kombinasi Algoritma Kriptografi Vigenere Cipher dengan Metode Zig-Zag dalam Pengamanan Pesan Teks. Jurnal CoSciTech (Computer Science and Information Technology), 2023. 4(1), 182–192.

[11]. Hasugian, A. H. Implementasi Hill Cipher dalam Penyandian Data. Pelita Informatika Budi Darma. 2013. 4(2). 115-122

[12]. Yulianti, N., & Wijaya, A. Desain Algoritma Kriptografi Klasik dengan Konsep Koset Grup dalam Teori Aljabar. Unnes Journal of Mathematics. 2022. 11(1). 16–26

Appendix.

```
#include <iostream>
#include <vector>
#include <string>
#include <fstream>
#include <cstdlib>
#include <ctime>
using namespace std;
const string Z60 =
 "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ.,!?_-@*";
int charToNum(char ch) {
   size_t pos = Z60.find(ch);
   return (pos != string::npos) ? (int)pos : -1;}
vector<int> encryptZ60(const string& plaintext, int k) {
   srand(time(NULL));
   vector<int> cipher;
   for (char ch : plaintext) {
      int val = charToNum(ch);
      if (val == -1) { cipher.push_back(-1); continue; }
      vector<int> koset;
      for (int j = 0; j < k; ++j) koset.push_back(val + 60 * j);
      cipher.push_back(koset[rand() % k]);}
```

```
    return cipher;}
string normalize(const vector<int>& cipher, int k) {
    int len = 1, max_val = 60 * k - 1, tmp = max_val;
    while (tmp /= 10) len++;
    string out = "";
    for (int v : cipher) {
        if (v == -1) out += string(len, '_');
        else {
            string s = to_string(v);
            while ((int)s.size() < len) s = "0" + s;
            out += s;}}
    return out;}
string digitsToZ60(const string& digits) {
    string out = ""; size_t i = 0;
    while (i < digits.size()) {
        if (digits[i] == '_') { out += ' '; i++; }
        else if (digits[i] == '0') { out += Z60[0]; i++; }
        else {
            if (i + 1 < digits.size() && digits[i+1] != '_') {
                int val = (digits[i] - '0') * 10 + (digits[i+1] - '0');
                if (val < (int)Z60.size()) { out += Z60[val]; i += 2; continue; }}
            out += Z60[digits[i] - '0']; i++;}}
    return out;}
int main() {
    string plaintext; int k;
    cout << "Plaintext: "; getline(cin >> ws, plaintext);
    cout << "Key (k): "; cin >> k;
    vector<int> cipher = encryptZ60(plaintext, k);
    string norm = normalize(cipher, k);
    string cipher_letters = digitsToZ60(norm);
    cout << "Ciphertext huruf: " << cipher_letters << endl;
    string file;
    cout << "Simpan ke file: "; cin >> file;
    ofstream out(file); out << cipher_letters;}
```

b. The programme code for descryption is as follows:

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
const string Z60 =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ.,!?_-@*";
int charToNum(char ch) {
    size_t pos = Z60.find(ch);
```

```
    return (pos != string::npos) ? (int)pos : -1;}
string decodeZ60Digits(const string& cipher_letters, int digit_len) {
    string digits = "";
    for (char ch : cipher_letters) {
        if (ch == ' ') { digits += string(digit_len, '_'); continue; }
        int val = charToNum(ch);
        if (val == -1) continue;
        if (val < 10) digits += char('0' + val);
        else {
            digits += char('0' + val / 10);
            digits += char('0' + val % 10);}}
    return digits;}
string decryptZ60(const string& cipher_letters, int digit_len, int k) {
    string digits = decodeZ60Digits(cipher_letters, digit_len), plaintext = "";
    for (size_t i = 0; i + digit_len <= digits.size(); i += digit_len) {
        string block = digits.substr(i, digit_len);
        if (block.find('_') != string::npos) { plaintext += ' '; continue; }
        int idx = stoi(block) % 60;
        plaintext += Z60[idx];}
    return plaintext;}
int main() {
    string filename, cipher_letters;
    cout << "File ciphertext: "; getline(cin >> ws, filename);
    ifstream file(filename); if (!file) return 1;
    getline(file, cipher_letters); file.close();
    int k;
    cout << "Key (k): "; cin >> k;
    int max_val = 60 * k - 1, digit_len = 1;
    while (max_val /= 10) digit_len++;
cout << "Plaintext: " << decryptZ60(cipher_letters, digit_len, k) << endl;}
```