



School of Electronics Engineering

Encryption and Decryption using 8086

By

**Kulvir Singh (19BCE2074)
Anitej Srivastava (19BCE0835)**

**Project Report
on
CSE2006 – Microprocessor & Interfacing**

Fall Semester 2021-22

**Submitted to
Faculty: Dr. Debashish Dash
Date: 27-11-2021
Slot : L3+L4**

| CONTENT | PAGE No. |
|---|-----------------|
| Abstract | 3 |
| Introduction | 4 |
| Literature Survey | 5 |
| Drawback of the existing work and the Proposed work | 8 |
| Block Diagram, pin diagram, flow chart | 9 |
| Implementation | 11 |
| Screenshots of the prototype | 13 |
| Results and Graphs | 15 |
| Conclusion | 15 |
| References | 16 |
| Appendix (code) | 17 |
| Plagiarism report | 22 |

1. Abstract

In today's world, we are very skeptical about our privacy. We always want our personal information and data to be protected and not visible to the general public. Therefore, we thought that it would be a great opportunity to work on such a topic which deals with data security. Information security is absolutely necessary. With the world going digital, all our personal data and information is going online. There was a time when we were afraid to use online banking but now almost everyone uses it. Such advancements have made life a lot easier and is only possible when the users feel a sense of security attached to it. Hence there is a significant importance of data and information security in today's world. There are various methods of securing data out of which one of them is the use of ciphers. These ciphers have two keen processes under the names of encryption and decryption.

Encryption and decryption of data at the assembly level will allow us to get a better understanding of security analysis of a system. Since we are working with microprocessors, we will be getting an insight at developing a root level security feature for various systems incorporating these microchips.

The entire script for encryption and decryption has been written on 8086 assembly language.

Moreover, we want to use this project to make a system secure. This project on 8086 can be extended to make protected files by encrypting user data.

This can be implemented to make a secure data transfer channel across 8086 systems. So therefore, in totality, we can say that there is an important use of encryption and decryption mechanism applied to 8086 microprocessor as the usages of this chip is plenty. Hence, to avoid malfunctions and mishaps, a protective layer is to be applied on this chip is vital.

2.Introduction

The project revolves around security and cryptography. The security of an application can be defined as the ability of an application to protect the data and configurations present on it. Any malicious code or software can ruin the normal working of the application and hence can cause the application to disrupt. The ability to protect the application from such attacks can be judged as the measure of unbreakable defence mechanism. To the strengthen the defence mechanism, cryptography and the use of symmetric and asymmetric ciphers can be explored.

In this project, we have designed such a security measure for coding strings into unreadable text and decoding it back. The proposed algorithm forms a symmetric cipher. Symmetric ciphers are those ciphers which require a key for encryption and decryption. The main feature about this key is that the key for both the encryption and decryption is same. Symmetric ciphers have been around for a while now and with software getting more advanced and complex, the cryptography techniques updated to asymmetric ciphers. Asymmetric ciphers are those ciphers which do not use the same key for encryption and decryption. These ciphers are said to be unbreakable and are hence used in real world problems to make application secure and unbreachable. Cryptanalysis is the term used to define the process of breaking down a cipher and getting the encrypted or cipher text back to its original plain text without the use or knowledge of the decryption or encryption key. Therefore it can be said that cryptanalysis of asymmetric ciphers is more difficult than that of symmetric.

With the basics of cryptography, we can develop an application for 8086 microprocessors to encrypt and encode a string using a cipher technique and decode the same to retrieve. The cipher being created is a mononumeric symmetric key cipher which will encode according to the ascii value of the characters making the brute force process of decryption very tough.

3.Literature Review

3.1. Fully encrypted high-speed microprocessor architecture: the secret computer in simulation.

**International Journal of Critical Computer-Based Systems
(2019)**

The paper discusses the architecture of an encrypted high-performance microprocessor designed on the principle that a nonstandard arithmetic generates encrypted processor states is described here. Data in registers, in memory and on buses exists in encrypted form. Any block encryption is feasible, in principle. It is proved here, for programs written in a minimal subset of instructions, that the platform is secure against 'Iago' attacks by the privileged operator or a subverted operating system, which cannot decrypt the program output, nor change the program's output to a particular value of their choosing.

3.2. Application of Genetic Algorithm in Cryptanalysis of Mono-alphabetic Substitution Cipher

**International Journal of Trend in Scientific Research and Development
(2018)**

This paper explores the cryptanalysis of Monoalphabetic Cipher and ways to improve it. Genetic algorithms are considered to be a tool for meta heuristic applications. In this work an attempt is made to carry out cryptanalysis, through genetic algorithms. The proposed work can be described as to design different Genetic Algorithm to crypt analyse the Mono- alphabetic Substitution Cipher using different type of genetic operator and then analyze and compare the results with respect to parameters considered for the best.

3.3. Nur Algorithm on Data Encryption and Decryption

**International Journal of Engineering & Technology
(2018)**

This paper has proposed a new & improved cryptographic algorithm called Nur algorithm.

The authors have implemented Nur algorithm using assembly programming language (MASM32). In Nur Aminuddin's Encryptor there are two data-reading techniques namely encryption technique (the technique of converting data from the original into unreadable code) and decryption technique (the techniques of reading unreadable codes become readable). Encryption technique is built by applying technique of modern cryptography which holds secrecy on the symmetric key, so the security of encryption depends only on the key and does not depend on whether the algorithm is known to people or not.

3.4. Masked Implementation of PIPO Block Cipher on 8-bit AVR Microcontrollers

**International Conference on Information Security Applications
(2021)**

This paper explores the PIPO lightweight cipher. The authors propose an efficient first-order masking technique using a 2-byte random mask by taking an advantage of PIPO block cipher. They present a new OR operation masking technique. Among functions of PIPO, the masked S-layer with 23 AND operations, 5 OR operations, and 46 XOR operations is used. Operations of PIPO block cipher are implemented in AVR assembly languages. The proposed implementation showed 1.5 times faster performance enhancements compared to the unprotected C implementation in the encryption process and 2.2 times faster performance enhancements compared to the unprotected optimized assembly implementation.

3.5. An involutive lightweight block cipher for 256-bit block size

**International Conference on Green Computing and Internet of Things
(2018)**

This paper and the authors address the need for securing specific details which have a great significance, As the heavily loaded world of information is coming out with rapid growth. Among the many proposed lightweight block ciphers out there, the main focus nowadays is on

their simple design specifications. In this paper, they have proposed a new involutive block cipher of 256-bit block size which is a member of the Extended LS-Designs.

3.6. Implementation of security module to protect programme theft in microcontroller-based applications

**Journal for Control, Measurement, Electronics, Computing and Communications
(2019)**

This paper addresses the problem of source code plagiarism which has become a serious threat for the development of small scale embedded industries and also the violations of intellectual property right which are a threat for the development of hardware system. In this paper, verification methods are presented to detect software plagiarism in the embedded application software without the implemented source code. All the approaches use side-channel information obtained during the execution of the suspicious code. The primary method is passive, i.e. no previous modification of the original code is required. It determines that the Hamming weights of the executed instructions of the suspicious device are used and uses string matching algorithms for comparisons with a reference implementation. The other method inserts additional code fragments as a watermark that can be identified in the power consumption of the executed source code. Proposed approaches are robust against code-transformation attacks.

4. Drawbacks in Existing Work

While all the papers and works mentioned have a huge output and significant results, they are not fully suited to our project. The 8086 architecture and design is fairly complex for the proposed algorithms to be applied in it. Hence, we need to come up with a simple and efficient coding technique which would be optimal for the processor to work with. Therefore, to optimise the time and space complexity of the system, an algorithm needs to be designed which suffices all the purposes of providing encoding and decoding, security and having minimal and optimal time and space usage.

5. Proposed Work

We have come up with an encryption technique which can be applied and coded in 8086 assembly language to yield optimal results. The technique has to be followed according to the algorithm explained below. The string is to be taken as input. For each character taken as an input, we will have to encode it using a table which would represent each letter with a number. This can be achieved by coding it with its ASCII values in the memory of 8086 and performing the translate operation to achieve the special digit associated to the character. The reverse process would be followed to get the original input word. Since it's a symmetric cipher, we can easily follow the reverse of the encryption process to get the original input. The key for this is defined and can be seen in the next segment of the paper. The entire process of input to encryption then displaying the encrypted code word, using the encrypted code word to decrypt and retrieve the original data is done in 8086 assembly language programming and has used shell scripting only. No third-party application is involved. The entire process is simple yet secure hence optimality and efficiency are high. Therefore, the proposed work is solving the drawbacks of the existing systems.

6. Block Diagram and Flowchart

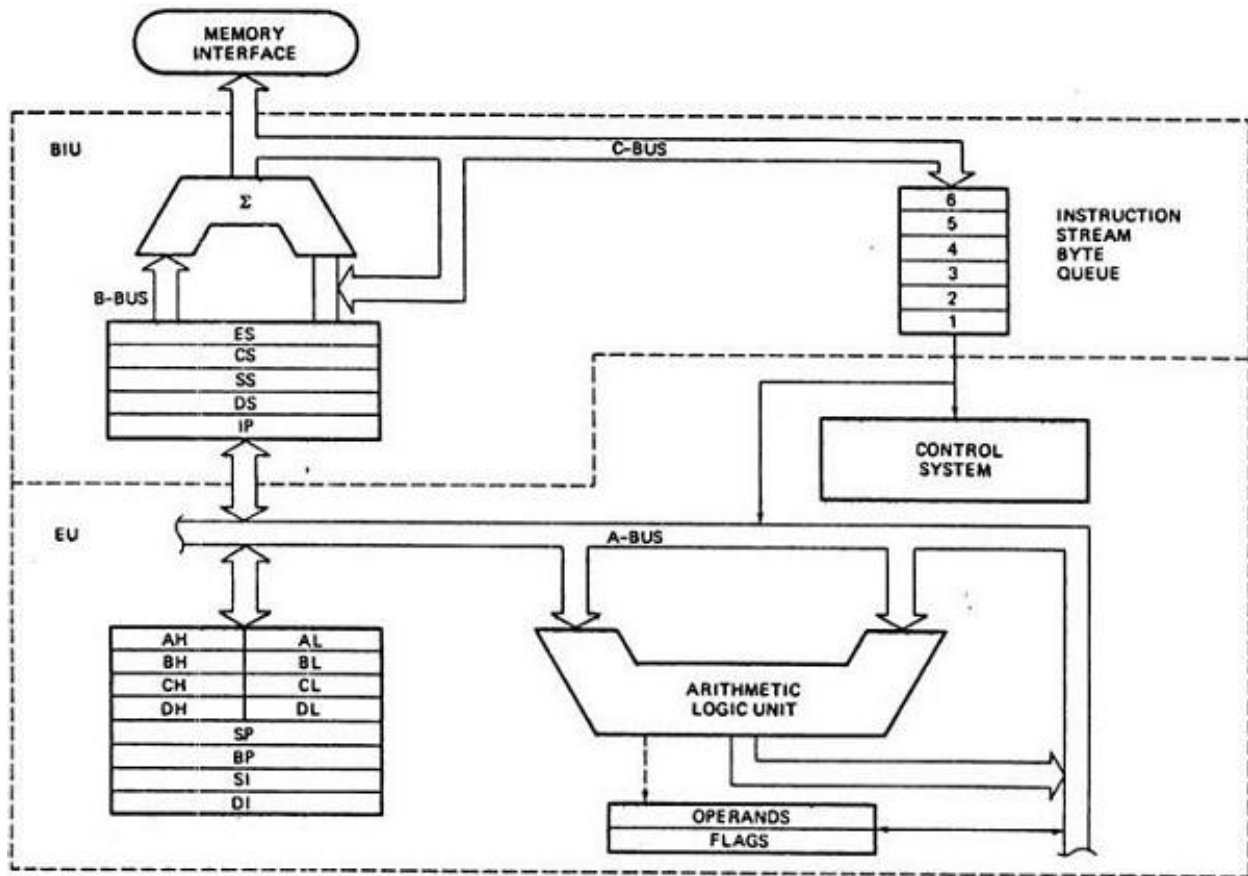


Figure 1: The 8086 Architecture. The entire architectural design of the microprocessor on which the coded algorithm would run.

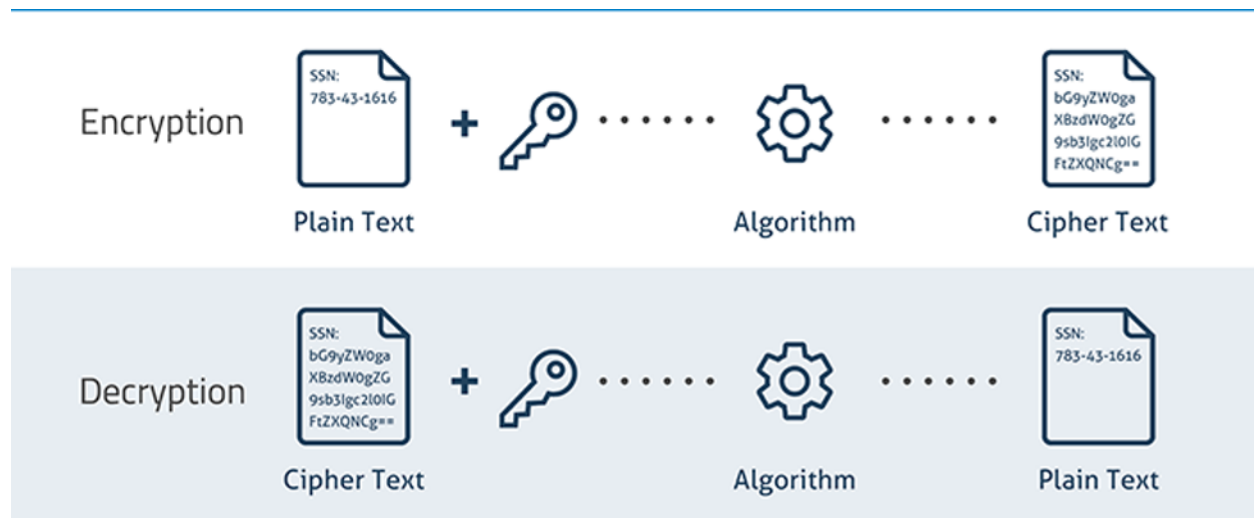


Figure 2: Symmetric Cipher Process Flow Diagram. The diagram shows how the encryption and decryption works for a symmetric cipher algorithm

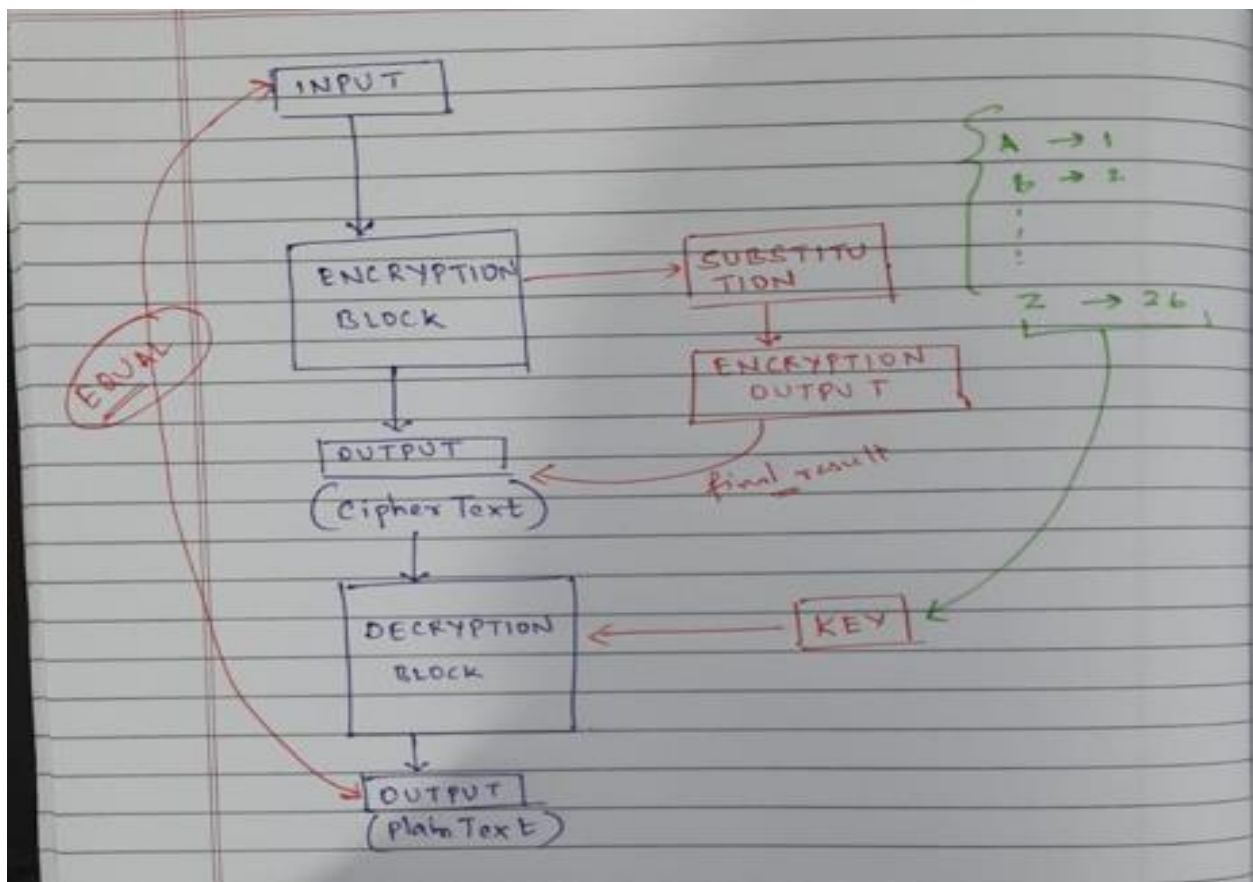


Figure 3: Program Flowchart. This is the final flowchart of the model. The entire encryption and decryption process can be seen and the plain text to cipher text conversion is also visible.

7. Implementation

7.a. Hardware Requirements

The only hardware components required are a laptop or desktop along with working input output devices like monitors and mouse to run the program on it.

7.b. Software Requirements

8086 Microprocessor Emulator, also known as EMU8086, is an emulator of the program 8086 microprocessor. It is developed with a built-in 8086 assembler. This application is able to run programs on both PC desktops and laptops. This tool is primarily designed to copy or emulate hardware. Emu8086 combines an advanced source editor, assembler, disassembler, software emulator (Virtual PC) with debugger, and step by step tutorials. This software is extremely helpful for those who study assembly language. It compiles the source code and executes it on emulator step by step.

Windows 10 Operating System is needed to the software emu8086, a major release of the Windows NT operating system developed by Microsoft. It is the successor to Windows 8.1, which was released nearly two years earlier, and itself was released to manufacturing on July 15, 2015, and broadly released for the general public on July 29, 2015.

7.c. Input/output module

The 8086-assembly language uses the interrupts to take user input and output. From the list of predefined interrupts of interrupt 21H or INT 21H, the value of AH to be selected for output functionality is 09H while the value of AH to be selected for input is 0AH. This will trigger the input and output operations accordingly. Before triggering this interrupt, the memory location of the variable to be input or output is to be retrieved using the LEA(used to load the address of the operand into the given register) operation and stored into DX general purpose register of the data segment of 8086.

7.d. Encryption

The encryption part of the code starts with storing the ascii values from a to z into the memory in hexadecimal form, therefore a loop is run starting from 61h and a counter of 26 is set. On every updation, the value is stored into the memory. Same for storing the numbers 1 to 26 into the memory. After performing the two loops, we take input string from the user to encrypt. After taking the input string as a buffer variable, we have achieved the length of the string. Once that is done, we now add a \$ symbol to the end of the input string to denote the end of the string. Once that is done, we load the offset of the ascii values to DI. The input string's first character is present at the SI location and is compared to the \$ symbol to know if we have reached the end of the string or not. If that's true, we exit the code else we continue. Moving forward we check if the character is a valid letter or not. Finally after passing that check, we encrypt using the XLATB function which would transfer the value from the ascii memory location to the input location. After performing this, print the number using a procedure PRINT_NUM_UN\$ which would print the decimal encoded version of the ascii value. Hence a mononumeric cipher is generated which prints the cipher text for the given input.

7.e. Decryption

For the decryption process, the modified buffer input is again processed similar to the encryption module but this time there is a bit of a change. We check if the character is valid or not by comparing it to the coded values of the character that is 1 to 27. This therefore becomes the key as it is used commonly for encryption as well as decryption. Ultimately we perform XLATB operation for each valid character to retrieve the original input string. The entire decryption process is the reverse for the encryption process.

8. Screenshots of the Prototype

```
edit: C:\EMU8086\MySource\mono_cipher.asm
file edit bookmarks assembler help

NEW OPEN SAVE ASSEMBLE RUN

0001:19BCE2074 Kulvir Singh
0002:19BCE0035 Anitej Srivastava
0003
0004 |
0005 | include 'emu8086.inc' ; Includes some common functions
0006 |
0007 org 100h
0008
0009 ;----- Saving Tables -----
0010
0011 ; Stores letters from a to z (lower case)
0012 MOV CX, 26 ; Size of letters in the alphabet
0013 MOV AL, 61h ; ASCII code for letter 'a'
0014 MOV DI, 400h ; Hold the offset of memory location in the ES
0015 CLD ; DF = 0
0016 store_letters:
0017 STOSB ; Copies a byte from AL to a memory location in ES. DI is used to hold the offset of the memory location in the ES. After the c
0018 INC AL ; Increases AL value by 1, therefore changing the letter
0019 LOOP store_letters ; Loops if CX after decrementing by 1 not equal 0
0020
0021
0022 ; Store numbers from 1 to 26
0023 MOV CX, 26 ; Size of letters in the alphabet
0024 MOV AL, 1 ; Starting from number 1
0025 MOV DI, 400h ;
0026
0027 store_numbers:
0028 STOSB
0029 INC AL
0030 LOOP store_numbers
0031
0032 ;----- Starting the program -----
0033
0034 ; Displays welcome message
0035 LEA DX, welcome_msg
0036 MOV AH, 9 ; Selecting the sub-function
0037 INT 21h ; Function that outputs a string at DS:DX. String must be terminated by '$'
0038
0039 ; Displays welcome message
0040 LEA DX, welcome_msg2
0041 MOV AH, 9
0042 INT 21h
0043
0044
0045 start_program:
0046
```


```
edit: C:\EMU8086\MySource\mono_cipher.asm
file edit bookmarks assembler help

NEW OPEN SAVE ASSEMBLE RUN

0043
0044
0045 start_program:
0046
0047 ; Displays "Enter a message to encrypt: " message
0048 LEA DX, encrypt_msg
0049 MOV AH, 9
0050 INT 21h
0051
0052 ; Takes input from user
0053 LEA DX, buffer
0054 MOV AH, 0Ah ; Sub-function that stores input of a string to DS:DX
0055 INT 21h
0056
0057 ; Puts $ at the end to be able to print it later
0058 MOV BX, 0
0059 MOV BL, buffer[1]
0060 MOV buffer[BX + 2], '$'
0061
0062 ;----- Encrypting -----
0063 ; Displays "Encrypted message: " message
0064 LEA DX, encrypted_msg
0065 MOV AH, 9
0066 INT 21h
0067
0068 ; The encryption code
0069 MOV DI, 3FFh
0070 MOV BX, DI
0071 LEA SI, buffer[2]
0072
0073 next_char:
0074 CMP [SI], '$' ; Check if reached end of message
0075 JE end_msg
0076
0077 LODSB ; Loads first char into AL, then moves SI to next char
0078 CMP AL, 'a'
0079 JB next_char ; If char is invalid, skip it
0080 CMP AL, 'z'
0081 JA next_char
0082 XLATB ; Encrypt
0083
0084 forspace:
0085 MOV [SI-1], AL
0086 MOV AH, 0
0087 CALL PRINT_NUM_UN$ ; Using a procedure to print numbers
0088 DEFINE_PRINT_NUM_UN$
```



9. Results and Discussion

 emulator screen (159x51 chars)

```
Welcome to Mononumeric Substitution Encryption Program
=====
Enter a message to encrypt: kulvir
The encrypted message is: 11211222918
The decrypted message is: kulvir
Enter a message to encrypt: anitej
The encrypted message is: 114920510
The decrypted message is: anitej
Enter a message to encrypt: microprocessor
The encrypted message is: 139318151618153519191518
The decrypted message is: microprocessor
Enter a message to encrypt: _
```

As seen in the above screenshot, the program can successfully take an input string, follow the mononumeric cipher algorithm and encrypt it to display a proper cipher text or encrypted text. On continuing with the program, we can see that the cipher text is decoded to give the plaintext or the input string is retrieved accurately.

10. Conclusion

This project work brings out the necessity of security in our applications and provides a simple algorithm to implement the same. As seen in the report, we were successfully able to design the algorithm and code it into the 8086-assembly language. The program thus created encrypted and decrypted the text input by the user. The application for this can be extended and, on more developments, can be implemented in various 8086 applications that require a security bypass. The encoding algorithm and program thus developed is fairly simple and easy to use and has a scope applications and developments in the future.

11. References

- [1]Hyunjun KimMinjoo SimSiwoo EumKyungbae JangGyeongju SongHyunji KimHyeokdong KwonWai-Kong LeeHwajeong Seo, Masked Implementation of PIPO Block Cipher on 8-bit AVR Microcontrollers, Springer: 2021
- [2] Peter T. Breuer, Jonathan P Bowen, Fully encrypted high-speed microprocessor architecture: the secret computer in simulation, InderScience: 2019.
- [3] P. Muthu Subramanian, A Rajeswari, Implementation of security module to protect programme theft in microcontroller-based applications, Automatika: 2019
- [4] Piyush Kumar Mudgal, Rajesh Purohit, Mahendra Kumar Jangir, Application of Genetic Algorithm in Cryptanalysis of Mono-alphabetic Substitution Cipher, IEEEExplore: 2018
- [5] Nur Aminudin, Andino Maseleno, Shankar K, S Hemalatha, K Sathesh Kumar, Fauzi, Rita Irviani, Muhamed Muslihudin, Nur Algorithm on Data Encryption and Decryption, SPC: 2018
- [6] Sukanya Saha, Krishnendu Rarhi, Abhishek Bhattacharya, Pratyusa Mukherjee, An involutive lightweight block cipher for 256-bit block size, IEEE Xplore: 2018.
- [7] Website used for learning about 8086 programming and function - <https://www.elprocus.com/8086-assembly-language-programs-explanation/>
- [8] Website used for learning about ciphers and encoding - <https://www.geeksforgeeks.org/java-program-to-implement-the-monoalphabetic-cypher/>
- [9] Website used for learning about xlatb and predefined procedures - <https://www.youtube.com/>
- [10]Website used for learning about 8086 assembly language - <https://www.geeksforgeeks.org/microprocessor-tutorials/>

12. Appendix

Code

;19BCE2074 Kulvir Singh

;19BCE0835 Anitej Srivastava

include 'emu8086.inc' ; Includes some common functions

org 100h

;----- Saving Tables -----

; Stores letters from a to z (lower case)

MOV CX, 26 ; Size of letters in the alphabet

MOV AL, 61h ; ASCII code for letter 'a'

MOV DI, 400h ; Hold the offset of memory location in the ES

CLD ; DF = 0

store_letters:

STOSB ; Copies a byte from AL to a memory location in ES. DI is used to hold the offset of the memory location in the ES. After the copy, DI is automatically incremented or decremented to point to the next string element in memory.

INC AL ; Increases AL value by 1, therefore changing the letter

LOOP store_letters ; Loops if CX after decrementing by 1 not equal 0

; Store numbers from 1 to 26

MOV CX, 26 ; Size of letters in the alphabet

MOV AL, 1 ; Starting from number 1

MOV DI, 460h ;

store_numbers:

STOSB

INC AL

LOOP store_numbers

;----- Starting the program -----

; Displays welcome message

LEA DX, welcome_msg

MOV AH, 9 ; Selecting the sub-function

INT 21h ; Function that outputs a string at DS:DX. String must be terminated by '\$'

; Displays welcome message

LEA DX, welcome_msg2

MOV AH, 9

INT 21h

start_program:

; Displays "Enter a message to encrypt: " message

LEA DX, encrypt_msg

MOV AH, 9

INT 21h

; Takes input from user

LEA DX, buffer

mov AH, 0Ah ; Sub-function that stores input of a string to DS:DX

INT 21h

; Puts \$ at the end to be able to print it later

MOV BX,0

MOV BL, buffer[1]

MOV buffer[BX + 2], '\$'

;----- Encrypting -----

; Displays "Encrypted message: " message

LEA DX, encrypted_msg

MOV AH, 9

INT 21h

; The encryption code

MOV DI, 3FFh

MOV BX, DI

LEA SI, buffer[2]

next_char:

 CMP [SI], '\$' ; Check if reached end of message

 JE end_msg

 LODSB ; Loads first char into AL, then moves SI to next char

 CMP AL, 'a'

 JB next_char ; If char is invalid, skip it

 CMP AL, 'z'

 JA next_char

 XLATB ; Encrypt

for space:

 MOV [SI-1], AL

 MOV AH,0

 CALL PRINT_NUM_UN\$; Using a procedure to print numbers

DEFINE_PRINT_NUM_UN\$

```

        JMP next_char

;space: mov al,0h
;    jmp forspace

end_msg:

; Decryption:
MOV BX, DI
LEA SI, buffer[2]

next_num:
    CMP [SI], '$'
    JE end_nums

    LODSB ; Loads byte from DS:SI to AL, then increments SI by 1
    CMP AL, 1
    JB next_num
    CMP AL, 26
    JA next_num
    XLATB ; Decrypt
    MOV [SI-1], AL
    JMP next_num

end_nums:
LEA DX, decrypt_msg
MOV AH, 9
INT 21h

```

```
; Displays decrypted message
```

```
LEA DX, buffer + 2
```

```
MOV AH, 9
```

```
INT 21h
```

```
; Repeat the program
```

```
JMP start_program
```

```
welcome_msg db "Welcome to Mononumeric Substitution Encryption Program $"
```

```
welcome_msg2                                     db                                0Dh,0Ah,
```

```
"===== $"
```

```
encrypting_msg db 0Dh,0Ah, "Encrypting ... $"
```

```
encrypt_msg db 0Dh,0Ah, "Enter a message to encrypt: $"
```

```
encrypted_msg db 0Dh,0Ah, "The encrypted message is: $"
```

```
decrypt_msg db 0Dh,0Ah, "The decrypted message is: $"
```

```
buffer db 27,?, 27 dup(' ')
```

```
end
```

13. Plagiarism Report



Date: November, 27 2021

PLAGIARISM SCAN REPORT



Excluded Url : None

Content Checked For Plagiarism

1. Abstract In today's world, we are very skeptical about our privacy. We always want our personal information and data to be protected and not visible to the general public. Therefore, we thought that it would be a great opportunity to work on such a topic which deals with data security. Information security is absolutely necessary. With the world going digital, all our personal data and information is going online. There was a time when we were afraid to use online banking but now almost everyone uses it. Such advancements have made life a lot easier and is only possible when the users feel a sense of security attached to it. Hence there is a significant importance of data and information security in today's world. There are various methods of securing data out of which one of them is the use of ciphers. These ciphers have two keen processes under the names of encryption and decryption. Encryption and decryption of data at the assembly level will allow us to get a better understanding of security analysis of a system. Since we are working with microprocessors, we will be getting an insight at developing a root level security feature for various systems incorporating these microchips. The entire script for encryption and decryption has been written on 8086 assembly language. Moreover, we want to use this project to make a system secure. This project on 8086 can be extended to make protected files by encrypting user data. This can be implemented to make a secure data transfer channel across 8086 systems. So therefore, in totality, we can say that there is an important use of encryption and decryption mechanism applied to 8086 microprocessor as the usages of this chip is plenty. 2. Introduction The project revolves around security and cryptography. The security of an application can be defined as the ability of an application to protect the data and configurations present on it. Any malicious code or software can ruin the normal working of the application and hence can cause the application to disrupt. The ability to protect the application from such attacks can be judged as the measure of unbreakable defence mechanism. To the strengthen the defence mechanism, cryptography and the use of symmetric and asymmetric ciphers can be explored. In this project, we have designed such a security measure for coding strings into unreadable text and decoding it back. The proposed algorithm forms a symmetric cipher. Symmetric ciphers are those ciphers which require a key for encryption and decryption. The main feature about this key is that the key for both the encryption and decryption is same. Symmetric ciphers have been around for a while now and with software getting more advanced and complex, the cryptography techniques updated to asymmetric ciphers. Asymmetric ciphers are those ciphers which do not use the same key for encryption and decryption. These ciphers are said to be unbreakable and are hence used in real world problems to make application secure and unbreachable. Cryptanalysis is the term used to define the process of breaking down a cipher and getting the encrypted or cipher text back to its original plain text without the use or knowledge of the decryption or encryption key. Therefore it can be said that cryptanalysis of asymmetric ciphers is more difficult than that of symmetric. With the basics of cryptography, we can develop an application for 8086 microprocessors to encrypt and encode a string using a cipher technique and decode the same to retrieve. The cipher being created is a mononumeric symmetric key cipher which will encode according to the ascii value of the characters making the brute force process of decryption very tough. The encryption part of the code starts with storing the ascii values from a to z into the memory in hexadecimal form, therefore a loop is run starting from 61h and a counter of 26 is set. On every updation, the value is stored into the memory. Same for storing the numbers 1 to 26 into the memory. After performing the two loops, we take input string from the user to encrypt. After taking the input string as a buffer variable, we have achieved the length of the string. Once that is done, we now add a \$ symbol to the end of the input string to denote the end of the string. Once that is done, we load the offset of the ascii values to DI. The input string's first character is present at the SI location and is compared to the \$ symbol to know if we have reached the end of the string or not. If that's true, we exit the code else we continue. Moving forward we check if the character is a valid letter or not. Finally after passing that check, we encrypt using the XLATB function which would transfer the value from the ascii memory location to the input location. After

performing this, print the number using a procedure PRINT_NUM_UN\$ which would print the decimal encoded version of the ascii value. Hence a mononumeric cipher is generated which prints the cipher text for the given input. 7.e. Decryption For the decryption process, the modified buffer input is again processed similar to the encryption module but this time there is a bit of a change. We check if the character is valid or not by comparing it to the coded values of the character that is 1 to 27. This therefore becomes the key as it is used commonly for encryption as well as decryption. Ultimately we perform XLATB operation for each valid character to retrieve the original input string. The entire decryption process is the reverse for the encryption process. This project work brings out the necessity of security in our applications and provides a simple algorithm to implement the same. As seen in the report, we were successfully able to design the algorithm and code it into the 8086-assembly language.

