Tyler Fontana
CSCI 4130
Phani Vadrevu
October 20 2020

Homework Assignment 3

Question 1. [50 points]

Alice's adventures in Hacker-land continue. She now wants to use a better implementation of DES to keep messages hidden away from Oscar. But, since it's hard to remember a good key, she unfortunately decided to craft a Hexspeak key (https://en.wikipedia.org/wiki/Hexspeak). She shared this key with Bob. Oscar, having seen Alice in an array of various collegiate and national league sports apparel knew she was an ardent sports fan. With this knowledge and some sheer luck, he rightly guessed that her DES key is '**f007ba11ba5eba11**'! (Note that this is Hexspeak for 'footballbaseball').

Oscar has seen 5 messages being transmitted from Alice to Bob. He has the Cipher text and IVs for these messages. But, he doesn't know which mode was used for transmitting these messages. Can you help Oscar **decipher** them and figure out **what mode** each message was sent in?

1. **These are the Cyphers, IV's, and their lengths given in the first part of the assignment:**

A.

Cipher: 64 27 63 80 7a 44 6b 98 3c 2d c9 8f 22 de 55 08 40 29 56 19 48 98 6d 1f 02 2e 03 8a 62 3c b3 bf 51 70 d3 2e db cd 93 1b 13 45 e7 a2 0f 16 82 d2 4d 71 55 c3 ff 12 25 38 07 cb 10 c3 80 cf 73 be 6c 22 7d db 55 75 e6 df a5 fb eb 8d ec 81 6f 75 39 de a8 6b 92 cf d4 63 49 10 3d 14 4f 69 f3 2c 52 7f ab 28 cc cb 8c e9

IV: 191668e63c6d7c45 **Cipher Length:** 104 Bytes

B.

Cipher: 69 e7 02 52 7c 08 55 14 ea b6 02 38 e2 64 47 8d 52 da 7d 29 a3 07 71 4b c7 21 94 e7 9b 88 d8 9a 40 4f 41 44 88 e0 b0 42 5b 87 70 7c f1 de 53 39 fb 93 00 5c 2c 0c 36 f6 83 54 42 2d cd 85 6f 26 06 7c 00 03 1a 6d 7a dd 44 bf eb be 58 1e 55 e7 80 d8 57 8f a2 7f db a8

IV: f09d673319595818 **Cipher Length:** 88 Bytes

C.

Cipher: 05 f3 5d b4 83 26 64 39 cf b4 b4 c0 89 5e 21 94 95 87 c6 ff 0c 00 a9 29 70 fa ef 21 2d b8 2e 70 74 9a 98 05 37 6f e0 9d 44 6e 8d 10 20 cc 35 87 61 d9 6e 40 be 29 98 3c 5e bc 4c d5 47 50 18 ca 01 a3 73 fd cc a2 45 52 77 2b e2 86 a1 98 1b 41 0e 16 31 ff f9 dd 47 85 27 1d 72 c9 fc a0 71 1f de 46 ef 88 03 04 7b 66 82 ac 71 7b ee 35 e0 ea 59 6b a1 d5 e8 fb bb 1f 55 82 2f 6f 81 fa c3 7c dc f5 a2 a5 08 95 b9 8f 9b e4 5e eb 7c 96 20 c5 7a 8b bb 26 88 ef 03 80 35 ef f5 79 18 42 3b 16 11 2a 16 28 2b 51 ce bf e1 03 3c e4 c8 35 24 04 93 a0 a5 e1 e8 a4 b9 1d 93 f0 7f d3 61 13 0a cd 64 51

IV: 39db1a2d187e4e3a **Cipher Length:** 256 Bytes

D.

Cipher: b7 4f 7a af a1 b9 0b fd 66 8c a5 66 c6 bf 03 8c 1f 04 58 6c dd 16 6c 36 57 ce 1c 4d 05 5d 0b 7b 5e 95 eb e1 02 0e 11 68 73 63 51 0f 4e 1c 95 4f 45 ec ed 21 1f 92 a6 97 4f 7b d3 4c f5 64 ce 5f b6 9c 16 73 66 a7 53 5b 2f e3 e8 25 2f 70 3d db 32 6e 71 2b d9 a6 54 50 da 3c 2d ba e2 b9 7c ea 1a a9 f9 ab 1f b6 16 90 ca 62 cf 4b 86 ce 09 06 47 b3 57 5b f2 37 59 5e 1e a3 5a 91 7f a7 fd 5a f5 ec 0c 31 b5 78 b3 75 32 eb 47 20 e7 a7 0a 32 35 38 5a 1c 63 52 77 94

IV: bb2dda8bcc8d6b63 **Cipher Length:** 152 Bytes

E.

Cipher: e2 9f 0c d8 85 55 ca 72 fb 34 f1 a0 c7 cd 27 0f fb a5 64 75 2c 5a 9f 47 1c 0d db c6 98 31 20 da 63 eb 63 cd 7c cb e3 2f 65 ee df 15 8b b3 1c 31 dc 22 5b c0 bb 8d 46 4d e5 5b a8 e8 4e 8d 8f 5d 27 a4 c4 2d 88 1d 1a 26 40 ed 58 b7 c9 9a 11 d7 25 5c 18 15 43 c4 c7 ee e9 44 03 68 17 4c 6c 00 73 8f d2 63 20 e9 6c 3a 63 a0 c8 a5 90 7f 87 99 dd fb 9b d7 94 4c 96 bc 72 ef 8b f8 15 cc e5 36 15 ca 5b 7e 32 cc bc 99 38 a8 80 c6 cf 1e ca 2b d2 c9 ee 69 69 28 71 c1 8c 68 e6 e2 b3 f3 f7 17 9e 46 c6 0e fd 37 00 b2 c9 32 04 cf 11 82 fa 45 24 ac b9 3e 49 c5 ac 9b a4 ba 72 02 e1 e1 64 dd 74 30 40 cf db da b9 64 a8 2c a3 0f eb 87 5f 77 ee 94 33 fa 97 f9 48 9f 96 30 d5 cc 12 c6 76 9e 62 49 e6 35 c7 80 26 f4 1f 6b a0 fa 36 c2 35 57 d1 30 65 3a 9d 57 6b d3 d2 0d 74 f0 68 ec e6 5e fc f1 15 ac 38 9f 19 e7 ab 1e f3 31 63 80 d4 4a 51 f6 fd

IV: 746daff7ce2056cd **Cipher Length:** 275 Bytes

In order to help you get this up and running, here's one other message that Alice sent in 'ECB' mode. We have both plain text as well as cipher text for this here:

Ex

Cipher: 5e 6f f8 5f c9 fd af ee 51 d0 1a d7 ff 5f d9 ad 6c 7a 61 bd 49 a1 4f 69 4c 34 a0 42 d9 95 b8 16 d3 b1 b0 b1 53 39 97 91 11 9c d8 b4 ac fd 51 0e 62 2b 23 2f eb b9 1c c1 95 df 7b a3 68 05 ff 4d 8d 7d c4 02 15 cc 03 55 09 23 3d f6 08 d0 84 d2 28 ce 8a e1 52 3d a2 f5 32 8e 47 59 1b 4c 30 c3 4b 63 7f a4 fc 58 09 c5 7b d5 d5 99 0c a9 93 8d 27 af 6d a7 55 e9 8b 9e a2 98 04 61 1f 43 03 3b a8 8d 71 c1 a9 d8 48 df 9a 2c ad 0e 72 15 a2 0c 37 dc 18 8f 6c f7 c2 da ef cb 64 d5 54 a0 60 09 ef dc 30 20 c8 b7 f2 30 39 f9 a0 c1 cb cb a4 22 c9 ce d7 e9 5c 1c a0 15

IV: 913e3818958ab43e **Cipher Length:** 184 Bytes

Cipher Mode: ECB (*Electronic Code Book Mode*)

Formula: $Y_i = F (Plaintext_i , Key)$

Message: In 2012, the owner of New Orleans Hornets wanted to change their name to something more local. They changed their name to 'Pelicans' as the brown pelican is Louisiana's state bird. (Note that there's no new line character in the above message. It's just one long string with only spaces as the separators between words)

Note 1: In the above ECB, CBC, CFB, OFB and CTR modes are used. For CTR mode, the first half of the IV was used as the initial value and the counter was set to 0. For ECB mode, the IV was ignored.

For Java, here are some helpful links:
https://www.ibm.com/support/knowledgecenter/en/SSYKE2_7.0.0/com.ibm.java.security.component.70.doc/security-component/JceDocs/jce_spec.html

https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html
https://proandroiddev.com/security-best-practices-symmetric-encryption-with-aes-in-java-7616beaaade9
https://www.journaldev.com/1309/java-des-algorithm-program

Note 3: PKCS5 was used for padding. (This is the same as PKCS7 when the block size is 64 bits). Here's a simple explanation of this padding scheme:
https://cryptosys.net/pki/manpki/pki_paddingschemes.html
This is a very popular padding scheme and is supported by most crypto libraries including the ones cited above.

In order to Decrypt the five messages shown above and determine which Block Cipher Mode was used to Encode them, I created a Java GUI Application which allows for the user to input their chosen ***DES Security Key***, ***Initialization Vector (IV)***, and ***Encrypted Hexidecimal Message***. The Application allows the User to choose between the five possible Block Cipher Modes mentioned above by choosing the corresponding option from a JComboBox Drop Down List. *(**Note**: **The Initialization Vector (IV)** Input Text Field, will only become accessible to the User should they choose a Block Cipher Mode other than **Electronic Code Block (ECB)** Mode or the Default Selection Option.)*

The GUI Application is Composed of Seven Separate Classes, four of which pertain to the functioning and appearance of the GUI Window Form, two others which function as Object Creation Classes, and the Last Class which is responsible for holding the various Decryption Related Methods. The Classes, and a short description describing their functionality are provided below:

1

GUI – This Swing Class is responsible for creating the Main GUI Window Form and establishing the functions of a majority of its Components. This Class incorporates numerous Java Swing Components such as a *JFrame*, multiple *JLabel's*, two *JTextField's*, two *JButton's*, two *JTextArea's*, and one *JComboBox* in order to achieve a very user friendly approach to decrypting the given messages above. The JFrame Component mentioned is responsible for establishing the overall dimensional size of the GUI Window in addition to holding all of the other Components Previously Mentioned. Should the User exit from this Component (*JFrame* Window), it will cause the entire Application to close as well. The two *JTextField* Components are used for allowing the user to input their chosen 16 Character HEXIDECIMAL **Security Key** and **Initialization Vector (IV)** Phrases, depending on the Block Cipher Mode selected. One of the JTextArea Components is used to receive the User's Inputted Encrypted HEXIDECIMAL Message, while the other is used to display the Decrypted Result Message upon successfully finishing the Decryption Operation. The *JComboBox* Component is used to create a Selectable Drop Down Menu which allows the user to choose between the five Block Cipher Modes mentioned above. Finally, the two *JButton* Components, are used to either Invoke the Decryption Process or Clear the other User Response Fields depending on which Button Component is Clicked Upon. The *JLabel* Components are Responsible for displaying text and images throughout the Application.

2 **WarningMessageFrame** – This Swing Class is responsible for creating a Pop-Up Warning Message Window which is shown should the user click the Decrypt *JButton* before inputting all required response information. The message displayed tells the user which fields are missing responses in addition to highlighting the User Input Fields associated with the Missing Information. This GUI Window Form is Denoted by the "*Warning*" Title Displayed above the Window in addition to the Yellow Caution Emblem shown to the left of the displayed message.

3 **WarningMessageFrameTwo** – This Swing Class is responsible for creating a Pop Up Warning Message Window, which is triggered should the user input information containing Prohibited Characters *(i.e. Non-HEXIDECIMAL Characters)* or information of an Invalid Length *(i.e. Not 16 Characters Long. [Note: This Correctness Check only Applies to the **Secret Key** and **Initialization Vector (IV)** Input Fields.])* This Warning Message Window's size is significantly larger than the previously covered one. This is done seeing as the possible amount of User Errors and Returned Exception Information requires a larger window in order to be fully displayed and accurately shown to user. Like the previously mentioned Warning Message Class, this GUI Form is denoted by the same "*Warning*" Title above the window in addition to the Yellow Caution Emblem displayed on the left side of the Error Message. This Warning Message Window will also have a table displayed underneath the error message which highlights the Fields Containing Invalid Information for the User.

4 **ErrorMessageFrame** – This Swing Class is responsible for creating a Pop-Up Error Message Window which displays a message detailing why the Decryption Process has Failed. This Class's Functionality and Make-Up are nearly identical to the two previously mentioned Warning Message Classes. However, the Error Message Window is instead denoted by the "Error" Title shown on top of the Window Form in addition to the red and white failure icon shown to the left of the message.

*(**Note**: This Class, in addition to the other three mentioned GUI Classes, all contain Inner **TaskHandler** Classes, which are used to spawn and execute separate Event Queue Threads which are responsible performing heavy data manipulation tasks. The reason for this, is explained in depth via Comments shown above the **TaskHandler** Inner Classes in the Application's Source Code.)*

5 **BlockCipherDecrypt** – This Java Class Holds the various Block Cipher Decryption Methods for each mode in addition to the methods used for formatting the Input Strings and converting them into their Hexidecimal Byte Equivalent Forms. This Class uses the External **Bouncy Castle Security Jar** File in order to provide the Encryption and Decryption Security Processes which are used to decode the User's Inputted Encrypted Messages.

*(**Note**: This JAR File is part of the Java Maven Repository but not Automatically Included within the Netbeans IDE. Additionally, it will be included under the "External Jars" Directory located within the Application's Submitted Source Code. [File Name: **bcprov-jdk15on-1.66.jar**])*

6 **DecryptionReturnObject** – This Java Class is Responsible for Creating an Object Instance used to return the results of a Decryption Process. The Boolean Result Field will be set to false should the operation fail, while it will be set to true should the operation succeed. If the operation has failed, it is due to an exception occurring. Should an error happen, its corresponding Integer Number will be stored in the Error Type Field so that the Main GUI Window can pull the associated error details and display them to the User. Additionally, should the Decryption Operation be Successful, this Object will store the Decrypted Message String in its last field.

7 **StringToByteArrayConversionResult** – This Java Class is Responsible for Creating an Object Instance used to return the results of the String to Byte Array Conversion Process. The Decryption Process requires a Byte Array Containing Specific Byte Casted Hexidecimal Values. (i.e. in the format of: *"**((byte) 0xFD)**"*, in order to function properly. Additionally, there is a possibility that the User's Response Values may be poorly formatted, such as containing excess whitespace, which may not be fully removed even after invoking multiple String formatting methods upon the input value. Thus, to ensure that the Input Values are both properly formatted and return the desired Byte Array Conversion Values, we need to use this Custom Object Class in order to determine whether or not the Conversion Process was successful.

Screenshots of the DES Decryption Tool Application, including the Warning / Error Message Window implementations and Successful Decryption Attempts will be shown at the end of the PDF Document. The Next Section contains the Block Cipher Modes and Decrypted Values Used in Encrypting the Messages shown above.

A.

Cipher: 64 27 63 80 7a 44 6b 98 3c 2d c9 8f 22 de 55 08 40 29 56 19 48 98 6d 1f 02 2e 03 8a 62 3c b3 bf 51 70 d3 2e db cd 93 1b 13 45 e7 a2 0f 16 82 d2 4d 71 55 c3 ff 12 25 38 07 cb 10 c3 80 cf 73 be 6c 22 7d db 55 75 e6 df a5 fb eb 8d ec 81 6f 75 39 de a8 6b 92 cf d4 63 49 10 3d 14 4f 69 f3 2c 52 7f ab 28 cc cb 8c e9

IV: 191668e63c6d7c45 **Cipher Length:** 104 Bytes

Block Cipher Used: **ECB (*Electronic Code Book Mode*)**

Decrypted Message Value:

The NBA team Utah Jazz actually began as New Orleans Jazz in 1974. This explains their strange name!

B.

Cipher: 69 e7 02 52 7c 08 55 14 ea b6 02 38 e2 64 47 8d 52 da 7d 29 a3 07 71 4b c7 21 94 e7 9b 88 d8 9a 40 4f 41 44 88 e0 b0 42 5b 87 70 7c f1 de 53 39 fb 93 00 5c 2c 0c 36 f6 83 54 42 2d cd 85 6f 26 06 7c 00 03 1a 6d 7a dd 44 bf eb be 58 1e 55 e7 80 d8 57 8f a2 7f db a8

IV: f09d673319595818 **Cipher Length:** 88 Bytes

Block Cipher Used: CTR (*Counter Mode*)

Decrypted Message Value:

Tom Dempsey was born without toes on his right foot and no fingers on his right hand!

C.

Cipher: 05 f3 5d b4 83 26 64 39 cf b4 b4 c0 89 5e 21 94 95 87 c6 ff 0c 00 a9 29 70 fa ef 21 2d b8 2e 70 74 9a 98 05 37 6f e0 9d 44 6e 8d 10 20 cc 35 87 61 d9 6e 40 be 29 98 3c 5e bc 4c d5 47 50 18 ca 01 a3 73 fd cc a2 45 52 77 2b e2 86 a1 98 1b 41 0e 16 31 ff f9 dd 47 85 27 1d 72 c9 fc a0 71 1f de 46 ef 88 03 04 7b 66 82 ac 71 7b ee 35 e0 ea 59 6b a1 d5 e8 fb bb 1f 55 82 2f 6f 81 fa c3 7c dc f5 a2 a5 08 95 b9 8f 9b e4 5e eb 7c 96 20 c5 7a 8b bb 26 88 ef 03 80 35 ef f5 79 18 42 3b 16 11 2a 16 28 2b 51 ce bf e1 03 3c e4 c8 35 24 04 93 a0 a5 e1 e8 a4 b9 1d 93 f0 7f d3 61 13 0a cd 64 51 2c 51 32 3b 4d d7 29 cd d6 22 27 c9 5f 98 d3 b8 90 24 cc 44 03 b4 8f 5e 10 96 d4 1b ff 8c 43 9b 44 2a fa 89 4c 52 36 d3 20 98 b2 fc 97 04 52 73 22 06 39 2a 09 45 7a 91 b7 05 40 6b 01 42

IV: 39db1a2d187e4e3a

Cipher Length: 256 Bytes

Block Cipher Used: CBC (*Cipher Block Chaining Mode*)

Decrypted Message Value:

In 1970, New Orleans Saint's kicker Tom Dempsey made a 63-yard field goal as time expired to give the Saints a 19–17 win over the Detroit Lions. At that time, this was a record for the longest field goal. This record stood for more than 28 years!

D.

Cipher: b7 4f 7a af a1 b9 0b fd 66 8c a5 66 c6 bf 03 8c 1f 04 58 6c dd 16 6c 36 57 ce 1c 4d 05 5d 0b 7b 5e 95 eb e1 02 0e 11 68 73 63 51 0f 4e 1c 95 4f 45 ec ed 21 1f 92 a6 97 4f 7b d3 4c f5 64 ce 5f b6 9c 16 73 66 a7 53 5b 2f e3 e8 25 2f 70 3d db 32 6e 71 2b d9 a6 54 50 da 3c 2d ba e2 b9 7c ea 1a a9 f9 ab 1f b6 16 90 ca 62 cf 4b 86 ce 09 06 47 b3 57 5b f2 37 59 5e 1e a3 5a 91 7f a7 fd 5a f5 ec 0c 31 b5 78 b3 75 32 eb 47 20 e7 a7 0a 32 35 38 5a 1c 63 52 77 94

IV: bb2dda8bcc8d6b63

Cipher Length: 152 Bytes

Block Cipher Used: CFB (*Cipher Feedback Mode*)

Decrypted Message Value:

As of the beginning of the 2017 NFL season, 51 former LSU players were on active rosters in the NFL. This is the most of any college football program!

E.

Cipher: e2 9f 0c d8 85 55 ca 72 fb 34 f1 a0 c7 cd 27 0f fb a5 64 75 2c 5a 9f 47 1c 0d db c6 98 31 20 da 63 eb 63 cd 7c cb e3 2f 65 ee df 15 8b b3 1c 31 dc 22 5b c0 bb 8d 46 4d e5 5b a8 e8 4e 8d 8f 5d 27 a4 c4 2d 88 1d 1a 26 40 ed 58 b7 c9 9a 11 d7 25 5c 18 15 43 c4 c7 ee e9 44 03 68 17 4c 6c 00 73 8f d2 63 20 e9 6c 3a 63 a0 c8 a5 90 7f 87 99 dd fb 9b d7 94 4c 96 bc 72 ef 8b f8 15 cc e5 36

15 ca 5b 7e 32 cc bc 99 38 a8 80 c6 cf 1e ca 2b d2 c9 ee 69 69 28 71 c1 8c 68 e6 e2 b3 f3 f7 17 9e
46 c6 0e fd 37 00 b2 c9 32 04 cf 11 82 fa 45 24 ac b9 3e 49 c5 ac 9b a4 ba 72 02 e1 e1 64 dd 74 30
40 cf db da b9 64 a8 2c a3 0f eb 87 5f 77 ee 94 33 fa 97 f9 48 9f 96 30 d5 cc 12 c6 76 9e 62 49 e6
35 c7 80 26 f4 1f 6b a0 fa 36 c2 35 57 d1 30 65 3a 9d 57 6b d3 d2 0d 74 f0 68 ec e6 5e fc f1 15 ac
38 9f 19 e7 ab 1e f3 31 63 80 d4 4a 51 f6 fd

IV: 746daff7ce2056cd **Cipher Length:** 275 Bytes

Block Cipher Used: **OFB (*Output Feedback Mode*)**

Decrypted Message Value:

Ron Maestri was the head coach of the Privateers baseball team for 16 years from 1972 to 1985. He is the most successful Privateer coach having taken the team to compete in NCAA Division 1 World Series. The Privateers are the first team in the state to win that honour!

Question 2. [50 points]

Chris is another friend of Alice. He also began to use DES to send messages to Alice and Bob. He sent these 2 messages using all the 5 different block cipher modes.
Hello there!!!

Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'
(Note that there's no new line character in the above message. It's just one long string with only spaces as the separators between words)

Unfortunately though for all the messages, Chris used the **same IV**. (Of course, ECB uses no IV and CTR uses only the first half of the IV). The key he used is also the same. Based on this information, can you figure out which of the following cipher texts corresponds to **which mode** and **which plain text message**?

1. **These Are the Encrypted Messages Corresponding with one of the two Plaintext Messages Above:**

A.

95 49 c3 87 bf ec 92 ec c7 f9 5d 5f 9a 8d

Cipher Length: 14 Bytes

B.

66 bd a4 a6 da 5b 7a 1b 2d 80 70 10 00 86 82 ae

Cipher Length: 16 Bytes

C.

95 49 c3 87 bf ec 92 ec a2 f5 06 4e f4 fa

Cipher Length: 14 Bytes

D.

92 43 df 98 f1 ec af a4 c8 fe 4b 0a 9b de 0e 8e 54 2c bc 2d 77 e8 d0 dc 94 70 66 23 44 7b 3c 16 45 75 d7 53 b0 cd cc 3c 5a 5f 30 25 b1 be 1a 31 01 4d 8c 2f 7c 68 9b e1 a3 43 f9 08 26 00 ac 2c 75 73 50 07 b8 70 6f 72 77 26 5a dd c2 11 47 42 ac e8 bf ba 41 77 37 8e 17 70 77 bd af 74 6b c7 ed 13 c2 12 db 06 ec 94 33 8e d3 99 fb fb 2a 32

Cipher Length: 112 Bytes

E.

01 7a 15 22 01 eb 04 d0 4f d2 02 9a da a6 6f cc d7 14 66 a1 ec a5 db 88 7f 86 23 af 6e 18 6b ff e7 d3 f4 f6 ee 6a 74 40 76 5b 79 46 fb 05 60 a5 e4 8b e1 6b 77 53 75 cc 3b f8 f8 50 55 73 e6 d1 6c 4b a0 7a 05 81 e6 28 66 bd a4 a6 da 5b 7a 1b 67 f0 35 76 ad ec 43 48 09 b9 72 e9 10 4a bc 76 13 b4 56 a5 6c b8 77 da 15 65 bd 10 f0 72 66 ef 01 25 6c 26 6e 8f bb f7

Cipher Length: 120 Bytes

F.

06 f0 39 5c 3d c7 6a bd a4 30 b3 07 cb 94 15 7c

Cipher Length: 16 Bytes

G.

cb a4 45 d5 8b 8e 94 76 e1 cf 52 03 51 e5 d9 3d e7 d7 2f 49 98 46 36 ec dc 98 23 dc 78 a4 71 56 1c b2 0e 37 f8 93 49 08 e7 af 77 68 ea 57 f1 17 0c 33 b2 73 27 d2 d4 5f fc 16 32 55 c6 a6 94 e2 11 41 bd ee b5 1f c0 6d 97 70 0e 04 d2 6d 11 c9 ba 15 e6 00 77 01 a3 a6 ce 4a 02 10 45 df b2 ca 11 bc 63 f6 04 40 ce 35 15 9d 5e dd ad 7f 26 95

Cipher Length: 112 Bytes

H.

cc ae 59 ca c5 8e a9 3e ee c8 44 56 50 b6

Cipher Length: 14 Bytes

I.

74 bb de cf 11 40 bf 97 1e 6b ce b8 6d 72 7f 53 b9 e3 7b b5 a0 f4 bb 96 3d 41 1a ea c5 4b 38 77 2e de 65 e5 27 c0 d2 4c 0a 02 eb 37 4f 87 3c 6f 02 45 7a 33 1f 56 89 39 74 b6 4f f4 09 92 42 76 59 2c 9e 93 c2 b2 00 fd fe 0b c8 6e a8 06 e2 b0 1a d3 09 e0 a6 b8 2f e8 30 74 43 15 47 5c e1 a0 75 06 0d d6 d8 39 d8 65 f3 fa fe c9 09 7d 5d 03 51 7b d4 38 4b 18 d8 12

Cipher Length: 120 Bytes

In order to Solve Question 2, there are a couple of Details we need to Consider:

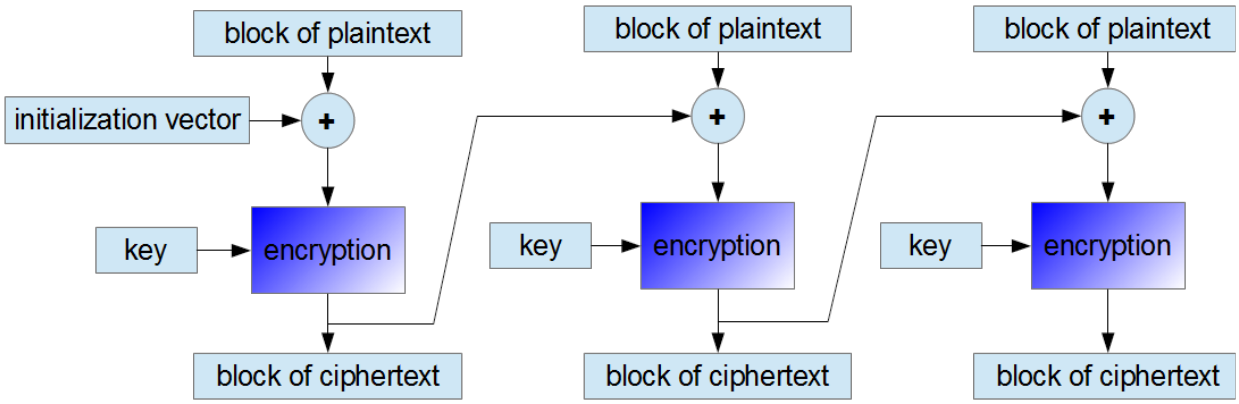
- **Every Two Character HEXIDECIMAL Value is Equivalent to a Single Byte.**
  - Ex.) HEXIDECIMAL: "0xFD" or "FD" = One Byte

- **Each Individual Byte Corresponds to a Single ASCII Character Value.**
  - Ex.) One Byte = One Regular Character: (A – Z) || (0 – 9) || (! - ?)

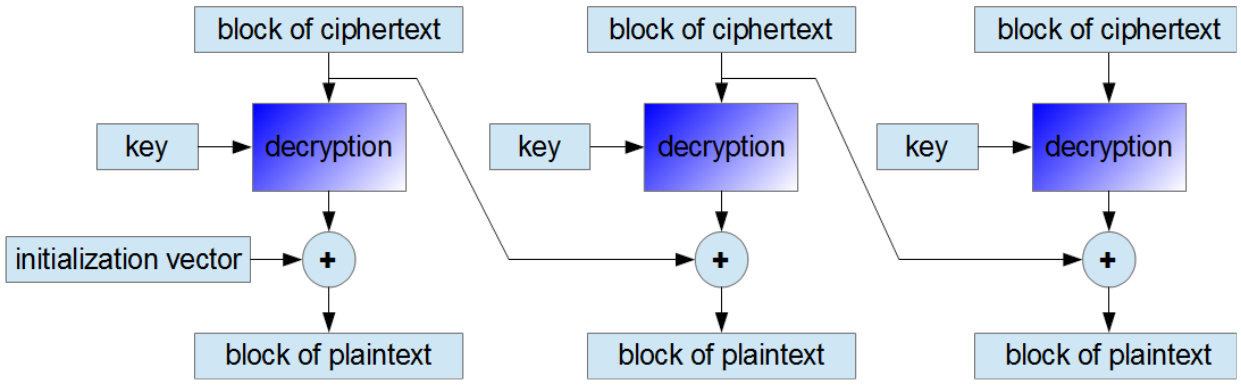
- ***Electronic Code Book Mode* (ECB) Encrypts Each 16 Character HEXIDECIMAL Phrase or 8 Byte Block Independently with the Same Key.**
  - Ex.) Say the Phrase: "***The Dog?***" maps to the Encrypted Value of "**E9A412F3378DAB20**".
  - This means that if the Phrase, "***The Dog?***" appears anywhere else in the Plaintext Version of the Message, then the phrase will map to the same HEXIDECIMAL value of "**E9A412F3378DAB20**" wherever else it appears in the Same Cipher Text or the Cipher Text of a Different Message.

- ***Cipher Block Chaining Mode* (CBC) uses the Previous Cipher Block (*8 Bytes*) as the Input to next Encryption Algorithm After XOR Operation with Plaintext Block. A Picture of This Process is Shown Below:**

Encryption:



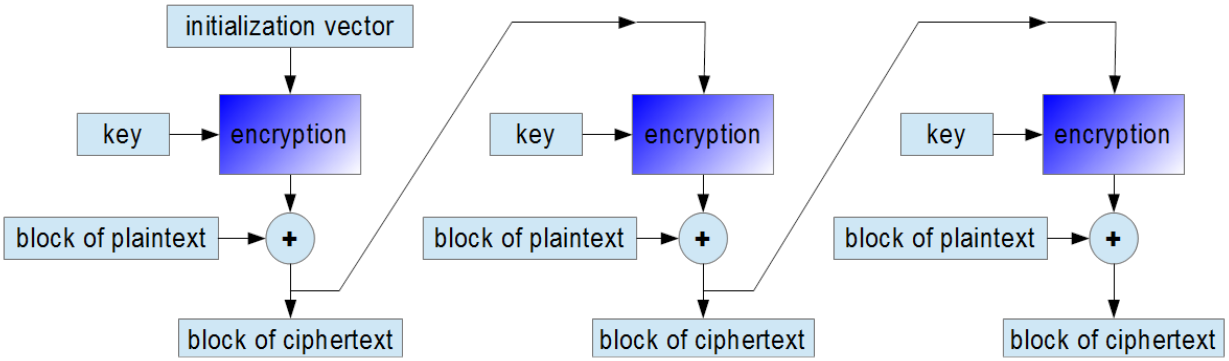
Decryption:



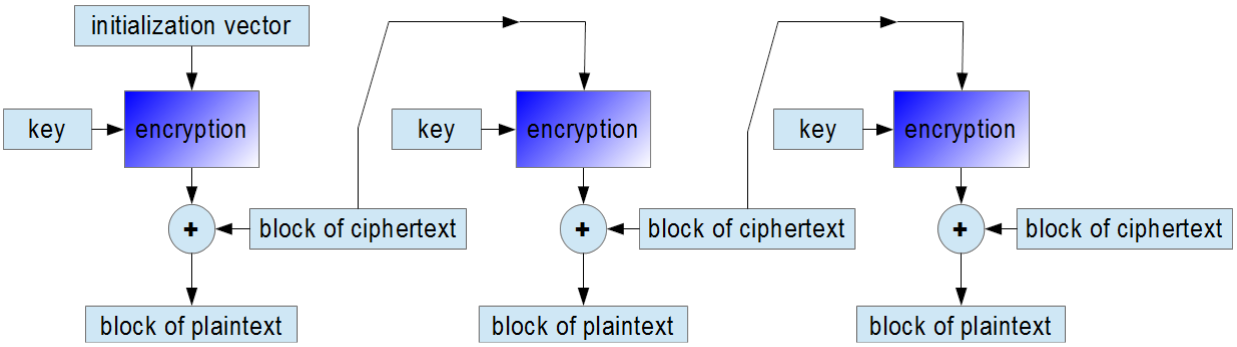
- **Both *Electronic Code Book Mode* (ECB) and *Cipher Block Chaining Mode* (CBC) are both Block Cipher Modes. This means that they require the Input Size to be a Block of 8 Bytes. If the input text is not a Multiple of 8 Bytes, then Padding will be used to fill in the rest of the missing bytes. Both ECB and CBC Modes require the use of Padding.**

- **In *Cipher Feedback Mode* (CFB), the cipher is given as feedback to the next block of encryption with some new specifications: first an initial vector IV is used for first encryption and output bits are divided as set of *s* and *b-s* bits the left hand side *s*-bits are selected and are applied an XOR operation with plaintext bits. The result given as input to a shift register and the process continues. A Picture of This Process is Shown Below:**

Encryption:

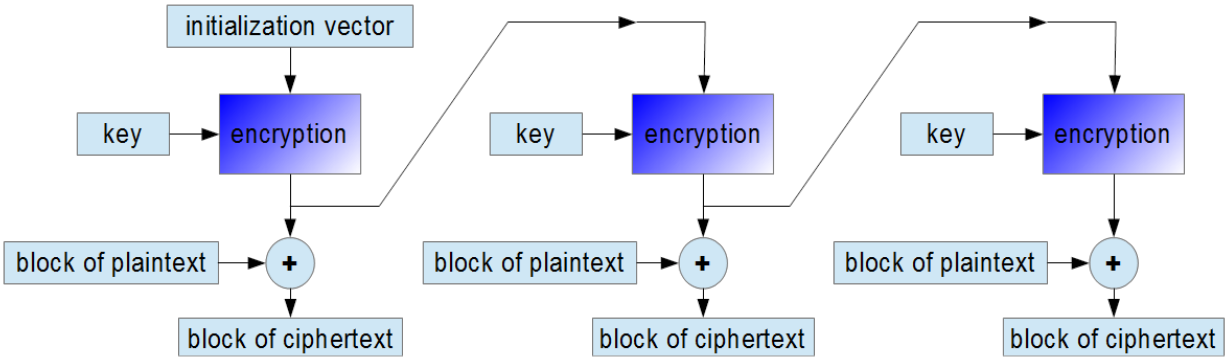


Decryption:

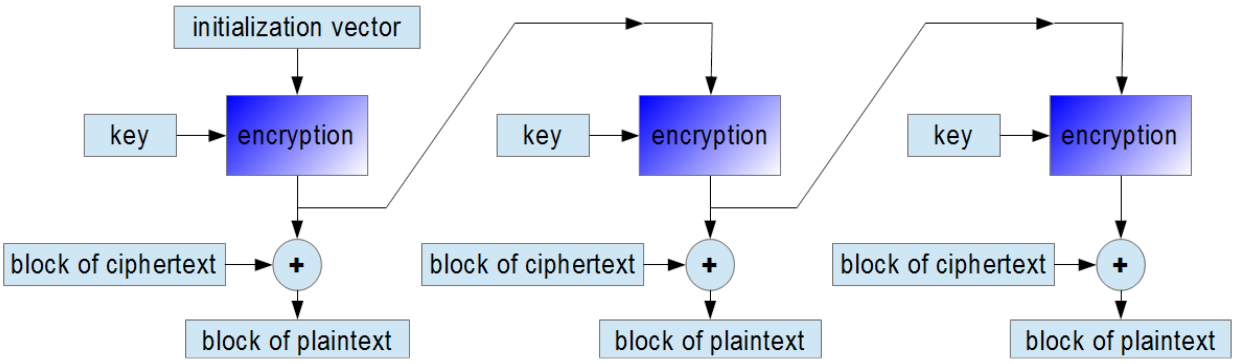


- **The Encryption Process of *Output Feedback Mode* (OFB) is shown below:**

Encryption:



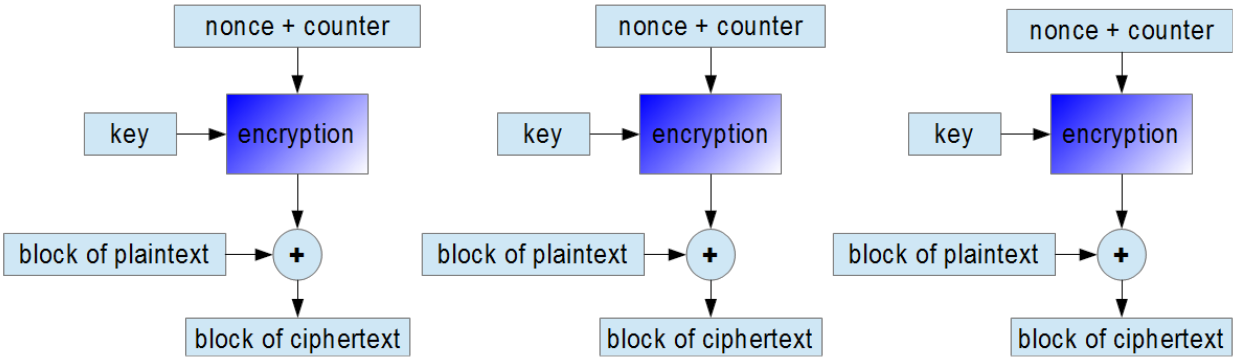
Decryption:



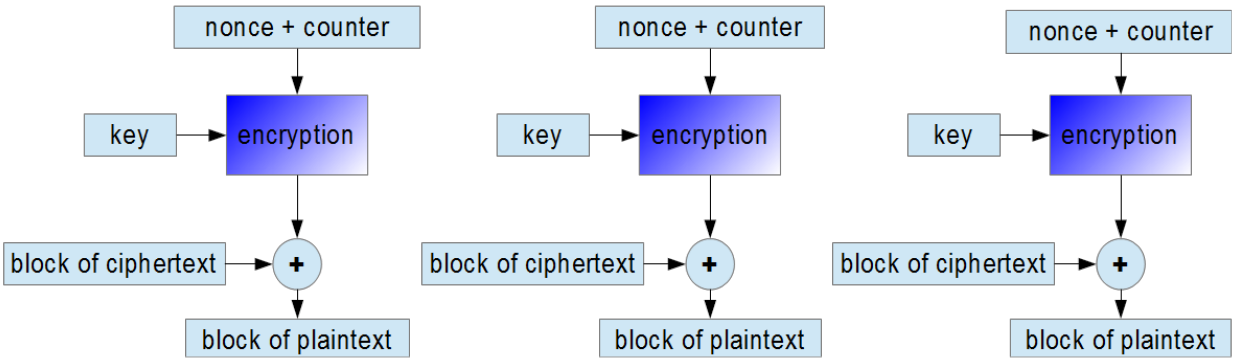
- **The Three Cipher Modes: *Cipher Feedback Mode* (CFB), *Output Feedback Mode* (OFB), and *Counter Mode* (CTR) are Stream Ciphers. This means that they Encrypt 8-bit Plaintext Phrases (*1 Byte*) at a time. This means that they also do not Require Padding and will Produce Ciphertext that is the Exact Same Length as the Original Input Plaintext.**

- **The Encryption Process of *Counter Mode* (CTR) is shown below:**

Encryption:



Decryption:



- **The First Plaintext Message "Hello there!!!" is comprised of 14 Bytes.**
- **The Second Plaintext Message "Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'" is comprised of 112 Bytes.**

Regardless of which Block or Stream Cipher Mode is used to Encrypt a Plaintext Value, a Single Byte will always correspond to a single ASCII Character Value. This means that we can immediately tell which of the provided messages will map to each of the provided plaintext phrases. Even with the inclusion of extra Padding Bytes for some of the Block Cipher Modes, the Lengths of the Two Plaintext Messages are Vastly Different, making it easy to determine which Cipher Text Messages Correspond to which Plaintext Phrase. The Messages containing 16 or less HEXIDECIMAL Bytes will map to the Plaintext Message: **"Hello there!!!"** while all of the messages containing more than 16 HEXIDECIMAL Bytes will map to the Plaintext Message: **"Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'"**

In this Next Section, I will provide the Plaintext Character to Byte Mapping Tables for each of the Various Messages. We will be able to use these Tables to Notice Hidden Patterns that may give us some clues regarding which Cipher Mode the Various Messages are Encrypted in. The Character Byte Mapping Tables are Provided Below:

*(**Note 1**: The **Purple** Entries Missing Both a HEXIDECIMAL Byte and Plaintext Value Symbolize Non-Present Byte Values within a Given 8 Byte Block.)*

*(**Note 2**: The **Blue** Entries Containing a HEXIDECIMAL Byte Value but Missing a Corresponding Plaintext Character Value Represent the Message's Padding should the Message Contain Any.)*

Message A:

95	49	C3	87	BF	EC	92	EC
H	e	l	l	o		t	h

C7	F9	5D	5F	9A	8D		
e	r	e	!	!	!		

Message B:

66	BD	A4	A6	DA	5B	7A	1B
H	e	l	l	o		t	h

2D	80	70	10	00	86	82	AE
e	r	e	!	!	!		

Message C:

95	49	C3	87	BF	EC	92	EC
H	e	l	l	o		t	h

A2	F5	06	4E	F4	FA		
e	r	e	!	!	!		

Message D:

92	43	DF	98	F1	EC	AF	A4
O	o	p	s	!		I	

C8	FE	4B	0A	9B	DE	0E	8E
j	u	s	t		r	e	a

54	2C	BC	2D	77	E8	D0	DC
l	i	z	e	d		t	h

94	70	66	23	44	7B	3C	16
a	t		I		a	m	

45	75	D7	53	B0	CD	CC	3C
t	a	l	k	i	n	g	

5A	5F	30	25	B1	BE	1A	31
t	o		P	r	i	v	a

01	4D	8C	2F	7C	68	9B	E1
t	e	e	r	s	.		I

A3	43	F9	08	26	00	AC	2C
	s	h	o	u	l	d	n

75	73	50	07	B8	70	6F	72
’	t		s	a	y		‘

77	26	5A	DD	C2	11	47	42
H	e	l	l	o		t	h

AC	E8	BF	BA	41	77	37	8E
e	r	e	!	’	.		I

17	70	77	BD	AF	74	6B	C7
’	l	l		s	a	y	

ED	13	C2	12	DB	06	EC	94
‘	A	h	o	y	,		M

33	8E	D3	99	FB	FB	2A	32
a	t	e	y	s	!	!	’

Message E:

01	7A	15	22	01	EB	04	D0
O	o	p	s	!		I	

4F	D2	02	9A	DA	A6	6F	CC
j	u	s	t		r	e	a

D7	14	66	A1	EC	A5	DB	88
l	i	z	e	d		t	h

7F	86	23	AF	6E	18	6B	FF
a	t		I		a	m	

E7	D3	F4	F6	EE	6A	74	40
t	a	l	k	i	n	g	

76	5B	79	46	FB	05	60	A5
t	o		P	r	i	v	a

E4	8B	E1	6B	77	53	75	CC
t	e	e	r	s	.		I

3B	F8	F8	50	55	73	E6	D1
	s	h	o	u	l	d	n

6C	4B	A0	7A	05	81	E6	28
’	t		s	a	y		‘

66	BD	A4	A6	DA	5B	7A	1B
H	e	l	l	o		t	h

67	F0	35	76	AD	EC	43	48
e	r	e	!	’	.		I

09	B9	72	E9	10	4A	BC	76
’	l	l		s	a	y	

13	B4	56	A5	6C	B8	77	DA
‘	A	h	o	y	,		M

15	65	BD	10	F0	72	66	EF
a	t	e	y	s	!	!	’

01	25	6C	26	6E	8F	BB	F7

Message F:

06	F0	39	5C	3D	C7	6A	BD
H	e	l	l	o		t	h

A4	30	B3	07	CB	94	15	7C
e	r	e	!	!	!		

Message G:

CB	A4	45	D5	8B	8E	94	76
O	o	p	s	!		I	

E1	CF	52	03	51	E5	D9	3D
j	u	s	t		r	e	a

E7	D7	2F	49	98	46	36	EC
l	i	z	e	d		t	h

DC	98	23	DC	78	A4	71	56
a	t		I		a	m	

1C	B2	0E	37	F8	93	49	08
t	a	l	k	i	n	g	

E7	AF	77	68	EA	57	F1	17
t	o		P	r	i	v	a

0C	33	B2	73	27	D2	D4	5F
t	e	e	r	s	.		I

FC	16	32	55	C6	A6	94	E2
	s	h	o	u	l	d	n

11	41	BD	EE	B5	1F	C0	6D
’	t		s	a	y		‘

97	70	0E	04	D2	6D	11	C9
H	e	l	l	o		t	h

BA	15	E6	00	77	01	A3	A6
e	r	e	!	’	.		I

CE	4A	02	10	45	DF	B2	CA
’	l	l		s	a	y	

11	BC	63	F6	04	40	CE	35
‘	A	h	o	y	,		M

15	9D	5E	DD	AD	7F	26	95
a	t	e	y	s	!	!	’

Message H:

CC	AE	59	CA	C5	8E	A9	3E
H	e	l	l	o		t	h

EE	C8	44	56	50	B6		
e	r	e	!	!	!		

Message I:

74	BB	DE	CF	11	40	BF	97
O	o	p	s	!		I	

1E	6B	CE	B8	6D	72	7F	53
j	u	s	t		r	e	a

B9	E3	7B	B5	A0	F4	BB	96
l	i	z	e	d		t	h

3D	41	1A	EA	C5	4B	38	77
a	t		I		a	m	

2E	DE	65	E5	27	C0	D2	4C
t	a	l	k	i	n	g	

0A	02	EB	37	4F	87	3C	6F
t	o		P	r	i	v	a

02	45	7A	33	1F	56	89	39
t	e	e	r	s	.		I

74	B6	4F	F4	09	92	42	76
	s	h	o	u	l	d	n

59	2C	9E	93	C2	B2	00	FD
’	t		s	a	y		‘

FE	0B	C8	6E	A8	06	E2	B0
H	e	l	l	o		t	h

1A	D3	09	E0	A6	B8	2F	E8
e	r	e	!	’	.		I

30	74	43	15	47	5C	E1	A0
’	l	l		s	a	y	

75	06	0D	D6	D8	39	D8	65
‘	A	h	o	y	,		M

F3	FA	FE	C9	09	7D	5D	03
a	t	e	y	s	!	!	’

51	7B	D4	38	4B	18	D8	12

Message J:

92	43	DF	98	F1	EC	AF	A4
O	o	p	s	!		I	

8A	E0	8F	2A	6F	45	2D	6A
j	u	s	t		r	e	a

02	8C	C7	93	E3	2C	A2	6F
l	i	z	e	d		t	h

03	98	2D	D9	25	6F	B5	79
a	t		I		a	m	

92	60	D4	27	CD	55	07	90
t	a	l	k	i	n	g	

31	02	07	83	10	DF	3A	BB
t	o		P	r	i	v	a

A0	D8	D9	78	86	E8	F0	BE
t	e	e	r	s	.		I

4F	CA	B3	98	5D	52	D6	A6
	s	h	o	u	l	d	n

5A	1C	35	1B	C2	D8	90	DB
’	t		s	a	y		‘

38	69	31	1A	10	7D	76	6E
H	e	l	l	o		t	h

01	C2	C8	68	45	6C	4D	F1
e	r	e	!	’	.		I

20	B1	2F	4D	D9	7A	E0	55
’	l	l		s	a	y	

00	C6	D0	35	8C	B6	6A	6F
‘	A	h	o	y	,		M

A2	82	63	37	EF	63	F1	B3
a	t	e	y	s	!	!	’

In this next Section, we will use the Previously Provided Tables and Information in Order to Determine the Corresponding Plaintext Message and Cipher Mode Associated with the Given Ciphertext Message:

After Looking at the Character Mappings and Total Number of Bytes in Each of the Messages Above, we can Conclude the Following Information Regarding Which Ciphertext Messages Map to Which Plaintext Phrases:

Ciphertext	Plaintext
Message A	Hello there!!!
Message B	Hello there!!!
Message C	Hello there!!!
Message D	Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'

Message E	Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'
Message F	Hello there!!!
Message G	Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'
Message H	Hello there!!!
Message I	Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'
Message J	Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'

We can also conclude the following regarding which Cipher Mode each Message was Encrypted in (*Due to the Inclusion or Exclusion of Padding*):

Ciphertext	Cipher Mode
Message A	Stream Cipher
Message B	Block Cipher
Message C	Stream Cipher
Message D	Stream Cipher
Message E	Block Cipher
Message F	Block Cipher
Message G	Stream Cipher
Message H	Stream Cipher
Message I	Block Cipher

Message J	Stream Cipher

We then begin determining which Mode is being used between the Block and Stream Ciphers Present. We will start by first Determining which of the Two Block Cipher Modes Previously Found were Encrypted using the ***Electronic Code Book Mode*** **(ECB)**. As Previously Mentioned, the Defining Feature of **ECB** Mode is that it Encrypts and Decrypts (*8 Byte*) Blocks Individually. This means that the Exact Same Plaintext Message Phrase of Eight Characters will map to the Exact Same Block of Ciphertext Regardless of the Length of one of the Two Messages.

After taking this Detail into Consideration, we look at the Character Mapping Tables Provided Above to see if any of the Ten Messages Contain the Exact Same Ciphertext Byte Sequence for the Exact Same Plaintext Character Block.

We Immediately Notice that the **First** Cipher Text Block of ***Message B***:

Block 1							
66	**BD**	**A4**	**A6**	**DA**	**5B**	**7A**	**1B**
H	**e**	**l**	**l**	**o**		**t**	**h**

Block 2							
2D	**80**	**70**	**10**	**00**	**86**	**82**	**AE**
e	**r**	**e**	**!**	**!**	**!**		

Corresponds with the **Tenth** Cipher Text Block of ***Message E***:

Block 9							
6C	**4B**	**A0**	**7A**	**05**	**81**	**E6**	**28**
’	**t**		**s**	**a**	**y**		**‘**

Block 10							
66	**BD**	**A4**	**A6**	**DA**	**5B**	**7A**	**1B**
H	**e**	**l**	**l**	**o**		**t**	**h**

Block 11							
67	**F0**	**35**	**76**	**AD**	**EC**	**43**	**48**
e	**r**	**e**	**!**	**’**	**.**		**I**

One of the ways a Block Cipher is Characterized is by its use of Padding. In other words, (*And Specifically Related to this Problem*) the Amount of Ciphertext Bytes Given in the Message must be a Multiple of 16 Bytes or Two Blocks. This Means that, in order to Determine which other Two Messages Correspond to the Other Block Cipher Mode, ***Cipher Block Chaining Mode*** **(CBC)** we must simply determine which other two Messages Use Padding that haven't already been Determined as ***Electronic Code Book Mode*** **(ECB)**.

Immediately, we can Notice that ***Message F***:

Block 1							
06	F0	39	5C	3D	C7	6A	BD
H	e	l	l	o		t	h

Block 2							
A4	30	B3	07	CB	94	15	7C
e	r	e	!	!	!		

And ***Message I*** Both Contain Padding Bytes:

Block 14							
F3	FA	FE	C9	09	7D	5D	03
a	t	e	y	s	!	!	'

Block 15							
51	7B	D4	38	4B	18	D8	12

Seeing as there are no other Provided Messages that Contain Padding Bytes and the Phrase "**Hello th**" Doesn't Produce the Same Ciphertext Block Regardless of its Positioning, we can Safely Conclude that **Message F** and **Message I** are Both Encrypted in ***Cipher Block Chaining Mode*** **(CBC).**

Next, we will begin Determining Which of the Remaining Six Messages are Encrypted in the Various Stream Cipher Modes.

When Dealing with Data Encryption Standard (DES) 64-bit Stream Cipher Modes, the First Block of the Encrypted Ciphertext will Always Match Providing that it's the same Plaintext.

However, of the Remaining Six Messages, Only Four Messages Encrypt to the Same First Ciphertext Block as Another of the Messages Using the Same Plaintext Input.

Specifically, **Block 1** of ***Message A***:

Block 1							
95	49	C3	87	BF	EC	92	EC
H	e	l	l	o		t	h

Block 2							
C7	F9	5D	5F	9A	8D		
e	r	e	!	!	!		

Corresponds with the Ciphertext **Block 1** in ***Message C***:

Block 1							
95	49	C3	87	BF	EC	92	EC
H	e	l	l	o		t	h

Block 2							
A2	F5	06	4E	F4	FA		
e	r	e	!	!	!		

And, **Block 1** of ***Message D***:

Block 1							
92	43	DF	98	F1	EC	AF	A4
O	o	p	s	!		I	

Block 2							
C8	FE	4B	0A	9B	DE	0E	8E
j	u	s	t		r	e	a

Corresponds with the Ciphertext **Block 1** in ***Message J***:

Block 1							
92	43	DF	98	F1	EC	AF	A4
O	o	p	s	!		I	

Block 2							
8A	E0	8F	2A	6F	45	2D	6A
j	u	s	t		r	e	a

Both ***Cipher Feedback Mode*** **(CFB)** and ***Output Feedback Mode*** **(OFB)** are both required to be the 64-bit DES Versions seeing as the two Stream Modes are Nearly Identical In Their Encryption / Decryption Processes. The Only Difference between the two is that **CFB** Mode uses the newly Generated Ciphertext as the Input to its Next Block Encryption Process while OFB Mode uses the Output Feedback as Input to its Next Block Encryption Process Instead.

The ***Counter*** **(CTR)** Stream Cipher Mode instead, Encrypts each Individual Byte using a Counter Value that is Incremented During each Round of the Block Encryption Process. In this Mode, the Initial Counter Value is the Inputted Initialization Vector (IV).

Of the Three Stream Cipher Modes Previously Mentioned, the Only Mode that can Achieve a Completely Different Result for the First Block of Ciphertext would most likely be the ***Counter*** **(CTR)** Mode, seeing as the Counter Value (*Initialization Vector (IV)*) is constantly changing due to being Incremented in Each Round of Block Encryption.

Thus, the Two Messages Whose First Block of Ciphertext Don't Match With the First Ciphertext Block of the other Two Messages using the same Plaintext must be Encrypted Using the ***Counter*** **(CTR)** Mode.

More Specifically, ***Message G***:

Block 1							
CB	**A4**	**45**	**D5**	**8B**	**8E**	**94**	**76**
O	**o**	**p**	**s**	**!**		**I**	

Block 2							
E1	**CF**	**52**	**03**	**51**	**E5**	**D9**	**3D**
j	**u**	**s**	**t**		**r**	**e**	**a**

And ***Message H***:

Block 1							
CC	**AE**	**59**	**CA**	**C5**	**8E**	**A9**	**3E**
H	**e**	**l**	**l**	**o**		**t**	**h**

Block 2							
EE	**C8**	**44**	**56**	**50**	**B6**		
e	**r**	**e**	**!**	**!**	**!**		

Are Both Encrypted in the ***Counter*** **(CTR)** Mode.

We can Figure Out the Modes used in the other Stream Cipher Messages by Finding the Keystream of the Remaining Four Messages. The ***Output Feedback Mode*** **(OFB)** will produce the Same Keystream Ciphertext Block Values while the Keystream Ciphertext Block Values of the ***Cipher Feedback Mode*** **(CFB)** will be Completely Different from One Another, Regardless of the Plaintext Input or Message Length.

We can Calculate the Key Stream Values of each of the Four Messages By XORing the Ciphertext Byte Values of the Remaining Messages with their Corresponding HEXIDECIMAL Byte Values in the Plaintext to Byte Conversion Table Provided Below:

Plaintext Message 1 Byte Conversion Table:

48	65	6C	6C	6F	20	74	68
H	e	l	l	o		t	h

65	72	65	21	21	21		
e	r	e	!	!	!		

Plaintext Message 2 Byte Conversion Table:

4F	6F	70	73	21	20	49	20
O	o	p	s	!		I	

6A	75	73	74	20	72	65	61
j	u	s	t		r	e	a

6C	69	7A	65	64	20	74	68
l	i	z	e	d		t	h

61	74	20	49	20	61	6D	20
a	t		I		a	m	

74	61	6C	6B	69	6E	67	20
t	a	l	k	i	n	g	

74	6F	20	50	72	69	76	61
t	o		P	r	i	v	a

74	65	65	72	73	2E	20	49
t	e	e	r	s	.		I

20	73	68	6F	75	6C	64	6E
	s	h	o	u	l	d	n

27	74	20	73	61	79	20	27
’	t		s	a	y		‘

48	65	6C	6C	6F	20	74	68
H	e	l	l	o		t	h

65	72	65	21	27	2E	20	49
e	r	e	!	’	.		I

27	6C	6C	20	73	61	79	20
’	l	l		s	a	y	

27	41	68	6F	79	2C	20	4D
‘	A	h	o	y	,		M

61	74	65	79	73	21	21	27
a	t	e	y	s	!	!	’

Now we can XOR the Bytes Above with the Ciphertext Byte Values in Each Given Message in Order to Find the Keystreams. Tables Showing Each Calculated Keystream are Shown on the Next Page:

*(**Note**: The Keystream will only be Calculated Up to the Sixteenth Byte for Simplification Purposes.)*

Keystream Table: (XOR Operation With Plaintext):

(***Note**: XOR Operation Denoted By (⊕) Symbol*)

Message A (First 16 Bytes or Less):	
XOR Operation	**Keystream Output**
0x95 ⊕ 0x48	0xDD
0x49 ⊕ 0x65	0x2C
0xC3 ⊕ 0x6C	0xAF
0x87 ⊕ 0x6C	0xEB
0xBF ⊕ 0x6F	0xD0

0xEC ⊕ 0x20	0xCC
0x92 ⊕ 0x74	0xE6
0xEC ⊕ 0x68	0x84
0xC7 ⊕ 0x65	0xA2
0xF9 ⊕ 0x72	0x8B
0x5D ⊕ 0x65	0x38
0x5F ⊕ 0x21	0x7E
0x9A ⊕ 0x21	0xBB
0x8D ⊕ 0x21	0xAC

Message C (First 16 Bytes or Less):	
XOR Operation	**Keystream Output**
0x95 ⊕ 0x48	0xDD
0x49 ⊕ 0x65	0x2C
0xC3 ⊕ 0x6C	0xAF
0x87 ⊕ 0x6C	0xEB
0xBF ⊕ 0x6F	0xD0
0xEC ⊕ 0x20	0xCC
0x92 ⊕ 0x74	0xE6
0xEC ⊕ 0x68	0x84
0xA2 ⊕ 0x65	0xC7
0xF5 ⊕ 0x72	0x87

0x06 ⊕ 0x65	0x63
0x4E ⊕ 0x21	0x6F
0xF4 ⊕ 0x21	0xD5
0xFA ⊕ 0x21	0xDB

Message D (First 16 Bytes or Less):	
XOR Operation	**Keystream Output**
0x92 ⊕ 0x4F	0xDD
0x43 ⊕ 0x6F	0x2C
0xDF ⊕ 0x70	0xAF
0x98 ⊕ 0x73	0xEB
0xF1 ⊕ 0x21	0xD0
0xEC ⊕ 0x20	0xCC
0xAF ⊕ 0x49	0xE6
0xA4 ⊕ 0x20	0x84
0xC8 ⊕ 0x6A	0xA2
0xFE ⊕ 0x75	0x8B
0x4B ⊕ 0x73	0x38
0x0A ⊕ 0x74	0x7E
0x9B ⊕ 0x20	0xBB
0xDE ⊕ 0x72	0xAC
0x0E ⊕ 0x65	0x6B

0x8E ⊕ 0x61	0xEF

Message I (First 16 Bytes or Less):	
XOR Operation	**Keystream Output**
0x92 ⊕ 0x4F	0xDD
0x43 ⊕ 0x6F	0x2C
0xDF ⊕ 0x70	0xAF
0x98 ⊕ 0x73	0xEB
0xF1 ⊕ 0x21	0xD0
0xEC ⊕ 0x20	0xCC
0xAF ⊕ 0x49	0xE6
0xA4 ⊕ 0x20	0x84
0x8A ⊕ 0x6A	0xE0
0xE0 ⊕ 0x75	0x95
0x8F ⊕ 0x73	0xFC
0x2A ⊕ 0x74	0x5E
0x6F ⊕ 0x20	0x4F
0x45 ⊕ 0x72	0x37
0x2D ⊕ 0x65	0x48
0x6A ⊕ 0x61	0x0B

Upon Looking at the Calculated Keystreams Above, we notice that all of the Messages Contain the Same Keystream for the First Initial Block of 8 Bytes:

Keystream: All Messages (*First Block of 8 Bytes*)							
0xDD	0x2C	0xAF	0xEB	0xD0	0xCC	0xE6	0x84

However, the Second Block of the Keystream Differs for All of the Messages Except Two. Message A and Message D Both Contain the Same Keystream Byte Sequence for the Second Block of Eight Bytes:

*(**Note**: The Keystream Bytes Highlighted **Red**, are only Present Within **Message D**.*

Keystream: Message A & Message D (*Second Block of 8 Bytes*)							
0xA2	0x8B	0x38	0x7E	0xBB	0xAC	0x6B	0xEF

Seeing as these Message A and Message D both share the Same Keystream After the First Block of Eight Bytes, it Means that these Two Messages Are Encrypted Via the ***Output Feedback Mode*** **(OFB)** Stream Cipher.

Additionally, through Process of Elimination, we can also Conclude that Message C and Message I are Both Encrypted Via the ***Cipher Feedback Mode*** **(CFB)** Stream Cipher.

The Final Results of All Our Determinations are Displayed Below:

Final Results:

A.

95 49 c3 87 bf ec 92 ec c7 f9 5d 5f 9a 8d

Cipher Length: 14 Bytes

Plaintext: Hello there!!!

Cipher Mode: ***Output Feedback Mode*** **(OFB)**

B.

66 bd a4 a6 da 5b 7a 1b 2d 80 70 10 00 86 82 ae

Cipher Length: 16 Bytes

Plaintext: Hello there!!!

Cipher Mode: ***Electronic Code Book Mode*** **(ECB)**

C.

95 49 c3 87 bf ec 92 ec a2 f5 06 4e f4 fa

Cipher Length: 14 Bytes

Cipher Mode: ***Cipher Feedback Mode*** **(CFB)**

Plaintext: Hello there!!!

D.

92 43 df 98 f1 ec af a4 c8 fe 4b 0a 9b de 0e 8e 54 2c bc 2d 77 e8 d0 dc 94 70 66 23 44 7b 3c 16 45 75 d7 53 b0 cd cc 3c 5a 5f 30 25 b1 be 1a 31 01 4d 8c 2f 7c 68 9b e1 a3 43 f9 08 26 00 ac 2c 75 73 50 07 b8 70 6f 72 77 26 5a dd c2 11 47 42 ac e8 bf ba 41 77 37 8e 17 70 77 bd af 74 6b c7 ed 13 c2 12 db 06 ec 94 33 8e d3 99 fb fb 2a 32

Cipher Length: 112 Bytes

Cipher Mode: ***Output Feedback Mode*** **(OFB)**

Plaintext: Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'

E.

01 7a 15 22 01 eb 04 d0 4f d2 02 9a da a6 6f cc d7 14 66 a1 ec a5 db 88 7f 86 23 af 6e 18 6b ff e7 d3 f4 f6 ee 6a 74 40 76 5b 79 46 fb 05 60 a5 e4 8b e1 6b 77 53 75 cc 3b f8 f8 50 55 73 e6 d1 6c 4b a0 7a 05 81 e6 28 66 bd a4 a6 da 5b 7a 1b 67 f0 35 76 ad ec 43 48 09 b9 72 e9 10 4a bc 76 13 b4 56 a5 6c b8 77 da 15 65 bd 10 f0 72 66 ef 01 25 6c 26 6e 8f bb f7

Cipher Length: 120 Bytes

Cipher Mode: ***Electronic Code Book Mode*** **(ECB)**

Plaintext: Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'

F.

06 f0 39 5c 3d c7 6a bd a4 30 b3 07 cb 94 15 7c

Cipher Length: 16 Bytes

Cipher Mode: ***Cipher Block Chaining Mode*** **(CBC)**

Plaintext: Hello there!!!

G.

cb a4 45 d5 8b 8e 94 76 e1 cf 52 03 51 e5 d9 3d e7 d7 2f 49 98 46 36 ec dc 98 23 dc 78 a4 71 56 1c b2 0e 37 f8 93 49 08 e7 af 77 68 ea 57 f1 17 0c 33 b2 73 27 d2 d4 5f fc 16 32 55 c6 a6 94 e2 11 41 bd ee b5 1f c0 6d 97 70 0e 04 d2 6d 11 c9 ba 15 e6 00 77 01 a3 a6 ce 4a 02 10 45 df b2 ca 11 bc 63 f6 04 40 ce 35 15 9d 5e dd ad 7f 26 95

Cipher Length: 112 Bytes

Cipher Mode: ***Counter Mode*** **(CTR)**

Plaintext: Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'

H.

cc ae 59 ca c5 8e a9 3e ee c8 44 56 50 b6

Cipher Length: 14 Bytes

Plaintext: Hello there!!!

Cipher Mode: ***Counter Mode*** **(CTR)**

I.

74 bb de cf 11 40 bf 97 1e 6b ce b8 6d 72 7f 53 b9 e3 7b b5 a0 f4 bb 96 3d 41 1a ea c5 4b 38 77 2e de 65 e5 27 c0 d2 4c 0a 02 eb 37 4f 87 3c 6f 02 45 7a 33 1f 56 89 39 74 b6 4f f4 09 92 42 76 59 2c 9e 93 c2 b2 00 fd fe 0b c8 6e a8 06 e2 b0 1a d3 09 e0 a6 b8 2f e8 30 74 43 15 47 5c e1 a0 75 06 0d d6 d8 39 d8 65 f3 fa fe c9 09 7d 5d 03 51 7b d4 38 4b 18 d8 12

Cipher Length: 120 Bytes

Plaintext: Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'

Cipher Mode: ***Cipher Block Chaining Mode*** **(CBC)**

J.

92 43 df 98 f1 ec af a4 8a e0 8f 2a 6f 45 2d 6a 02 8c c7 93 e3 2c a2 6f 03 98 2d d9 25 6f b5 79 92 60 d4 27 cd 55 07 90 31 02 07 83 10 df 3a bb a0 d8 d9 78 86 e8 f0 be 4f ca b3 98 5d 52 d6 a6 5a 1c 35 1b c2 d8 90 db 38 69 31 1a 10 7d 76 6e 01 c2 c8 68 45 6c 4d f1 20 b1 2f 4d d9 7a e0 55 00 c6 d0 35 8c b6 6a 6f a2 82 63 37 ef 63 f1 b3

Cipher Length: 112 Bytes

Plaintext: Oops! I just realized that I am talking to Privateers. I shouldn't say 'Hello there!'. I'll say 'Ahoy, Mateys!!'

Cipher Mode: ***Cipher Feedback Mode*** **(CFB)**

*****The Following Pages Contain Screenshots of the DES Decryption Tool GUI Application*****

*****Please Note that you will need to import the External JAR File Mentioned Earlier and Remove the Package Declaration Line from All Java Classes In Order to Have the Application Properly Function Outside of the Netbeans IDE. The Starting Class of the Application is the GUI.java File and the JAR File can be found in the External Jars Directory Folder.*****


DES Decryption Tool
DES Decryption Tool
Key:
16 Digit HEXIDECIMAL Phrase
Mode:
Initialization Vector (IV):
16 Digit HEXIDECIMAL Phrase
Enter the Encrypted HEXIDECIMAL Phrase Below:
Encrypted HEXIDECIMAL Formatted Text
Decrypted Text
Decrypt
Clear


DES Decryption Tool
DES Decryption Tool
Key:
16 Digit HEXIDECIMAL Phrase
Mode:
Initialization V
Warning Screen
Multiple Fields Contain Incorrect or Missing Information.
Encrypted HEXI
Decrypted Text
Decrypt
Clear


DES Decryption Tool
DES DECRYPTION TOOL
Key:
16 Digit HEXIDECIMAL Phrase
Mode:
Initialization Vector (IV):
16 Digit HEXIDECIMAL Phrase
Enter the Encrypted HEXIDECIMAL Phrase Below:
Encrypted HEXIDECIMAL Formatted Text
Decrypted Text
Decrypt
Clear
Multiple Fields Contain Missing Information.


DES Decryption Tool
DES Decryption Tool
Key:
f007ba11ba5eba11
Mode:
ECB (Electronic Code Block)
CBC (Cipher Block Chaining)
CFB (Cipher Feedback)
OFB (Output Feedback)
CTR (Counter)
Encrypted HEXIDECIMAL Formatted Text
Decrypted Text
Decrypt
Clear


DES Decryption Tool
DES Decryption Tool
Key:
f007ba11ba5eba11
Mode:
CBC (Cipher Block Chaining)
Initialization V
Warning Screen
Please Enter a Valid Response
Within the "Initialization Vector (IV)"
Input Field.
64 27 63 80 7a
1f 02 2e 03
8a 62 3c b3 bf
5 c3 ff 12
25 38 07 cb 10 c3 80 cf 73 be 6c 22 7d db 55 75 e6 df a5 fb eb 8d ec 81 6f 75 39 de
a8 6b 92 cf d4 63 49 10 3d 14 4f 69 f3 2c 52 7f ab 28 cc cb 8c e9
Decrypted Text
Decrypt
Clear


DES Decryption Tool
DES DECRYPTION TOOL
Key:
f007ba11ba5eba11
Mode:
CBC (Cipher Block Chaining)
Initialization Vector (IV):
16 Digit HEXIDECIMAL Phrase
Enter the Encrypted HEXIDECIMAL Phrase Below:
64 27 63 80 7a 44 6b 98 3c 2d c9 8f 22 de 55 08 40 29 56 19 48 98 6d 1f 02 2e 03
8a 62 3c b3 bf 51 70 d3 2e db cd 93 1b 13 45 e7 a2 0f 16 82 d2 4d 71 55 c3 ff 12
25 38 07 cb 10 c3 80 cf 73 be 6c 22 7d db 55 75 e6 df a5 fb eb 8d ec 81 6f 75 39 de
a8 6b 92 cf d4 63 49 10 3d 14 4f 69 f3 2c 52 7f ab 28 cc cb 8c e9
Decrypted Text
Decrypt
Clear
Please Enter a Valid Response Within the "Initialization Vector (IV)" Input Field.


DES Decryption Tool
DES Decryption Tool
Key:
f007ba11ba
Mode:
ECB (Electronic Code Block)
Initialization V
Warning Screen
The Response You've Entered into the 'Key' Field Doesn't Meet the 16 Hexidecimal Character Requirement. Please Correct this before Attempting to Continue.
64 27 63 80 7a
1f 02 2e 03
8a 62 3c b3 bf
5 c3 ff 12
25 38 07 cb 10 c3 80 cf 73 be 6c 22 7d db 55 75 e6 df a5 fb eb 8d ec 81 6f 75 39 de
a8 6b 92 cf d4 63 49 10 3d 14 4f 69 f3 2c 52 7f ab 28 cc cb 8c e9
Decrypted Text
Decrypt
Clear


DES Decryption Tool
DES DECRYPTION TOOL
Key:
f007ba11ba5eba1*
Mode:
ECB (Electronic Code Block)
Initialization V
Warning Screen
The Response You've Entered into the 'Key' Field Contains Prohibited Characters. Please only use the Characters (0 – 9) & (A – F) in your Response.
64 27 63 80 7a
1f 02 2e 03
8a 62 3c b3 bf
5 c3 ff 12
25 38 07 cb 10 c3 80 cf 73 be 6c 22 7d db 55 75 e6 df a5 fb eb 8d ec 81 6f 75 39 de
a8 6b 92 cf d4 63 49 10 3d 14 4f 69 f3 2c 52 7f ab 28 cc cb 8c e9
Decrypted Text
Decrypt
Clear


DES Decryption Tool
DES DECRYPTION TOOL
Key:
f007ba11ba
Mode:
ECB (Electronic Code Block)
Initialization Vector (IV):
16 Digit HEXIDECIMAL Phrase
Enter the Encrypted HEXIDECIMAL Phrase Below:
64 27 63 80 7a 44 6b 98 3c 2d c9 8f 22 de 55 08 40 29 56 19 48 98 6d 1f 02 2e 03
8a 62 3c b3 bf 51 70 d3 2e db cd 93 1b 13 45 e7 a2 0f 16 82 d2 4d 71 55 c3 ff 12
25 38 07 cb 10 c3 80 cf 73 be 6c 22 7d db 55 75 e6 df a5 fb eb 8d ec 81 6f 75 39 de
a8 6b 92 cf d4 63 49 10 3d 14 4f 69 f3 2c 52 7f ab 28 cc cb 8c e9
Decrypted Text
Decrypt
Clear
'Key' Field Response: Invalid Length.


DES Decryption Tool
DES DECRYPTION TOOL
Key:
f007ba11ba5eba11
Mode:
CBC (Cipher Block Chaining)
Initialization Vector (IV):
39db1a2d187e4e3a
Enter the Encrypted HEXIDECIMAL Phrase Below:
05 f3 5d b4 83 26 64 39 cf b4 b4 c0 89 5e 21 94 95 87 c6 ff 0c 00 a9 29 70 fa ef 21
2d b8 2e 70 74 9a 98 05 37 6f e0 9d 44 6e 8d 10 20 cc 35 87 61 d9 6e 40 be 29 98
3c 5e bc 4c d5 47 50 18 ca 01 a3 73 fd cc a2 45 52 77 2b e2 86 a1 98 1b 41 0e 16
31 ff f9 dd 47 85 27 1d 72 c9 fc a0 71 1f de 46 ef 88 03 04 7b 66 82 ac 71 7b ee 35
e0 ea 59 6b a1 d5 e8 fb bb 1f 55 82 2f 6f 81 fa c3 7c dc f5 a2 a5 08 95 b9 8f 9b e4
5e eb 7c 96 20 c5 7a 8b bb 26 88 ef 03 80 35 ef f5 79 18 42 3b 16 11 2a 16 28 2b
In 1970, New Orleans Saint's kicker Tom Dempsey made a 63-yard field goal as time expired to give the Saintsa 19-17 win over the Detroit Lions. At that time, this was a record for the longest field goal.This record stood for more than 28 years!
Decrypt
Clear


DES Decryption Tool
DES DECRYPTION TOOL
Key:
f007ba11ba5eba11
Mode:
CTR (Counter)
Initialization Vector (IV):
f09d673319595818
Enter the Encrypted HEXIDECIMAL Phrase Below:
69 e7 02 52 7c 08 55 14 ea b6 02 38 e2 64 47 8d 52 da 7d 29 a3 07 71 4b c7 21
94 e7 9b 88 d8 9a 40 4f 41 44 88 e0 b0 42 5b 87 70 7c f1 de 53 39 fb 93 00 5c 2c
0c 36 f6 83 54 42 2d cd 85 6f 26 06 7c 00 03 1a 6d 7a dd 44 bf eb be 58 1e 55 e7
80 d8 57 8f a2 7f db a8
Tom Dempsey was born without toes on his right foot and no fingers on his right hand!
Decrypt
Clear


DES Decryption Tool
DES DECRYPTION TOOL
Key:
f007ba11ba5eba11
Mode:
OFB (Output Feedback)
Initialization Vector (IV):
bb2dda8bcc8d6b63
Enter the Encrypted HEXIDECIMAL Phrase Below:
b7 4f 7a af a1 b9 0b fd 66 8c a5 66 c6 bf 03 8c 1f 04 58 6c dd 16 6c 36 57 ce 1c 4d
05 5d 0b 7b 5e 95 eb e1 02 0e 11 68 73 63 51 0f 4e 1c 95 4f 45 ec ed 21 1f 92 a6
97 4f 7b d3 4c f5 64 ce 5f b6 9c 16 73 66 a7 53 5b 2f e3 e8 25 2f 70 3d db 32 6e
71 2b d9 a6 54 50 da 3c 2d ba e2 b9 7c ea 1a a9 f9 ab 1f b6 16 90 ca 62 cf 4b 86 ce
09 06 47 b3 57 5b f2 37 59 5e 1e a3 5a 91 7f a7 fd 5a f5 ec 0c 31 b5 78 b3 75 32
eb 47 20 e7 a7 0a 32 35 38 5a 1c 63 52 77 94
As of the beginning of the 2017 NFL season, 51 former LSU players were on active rosters in the NFL.This is the most of any college football program!
Decrypt
Clear


DES Decryption Tool
DES Decryption Tool
Key:
f007ba11ba5eba11
Mode:
CBC (Cipher Block Chaining)
Initialization V
Error
The Input Data Doesn't Match the Required Block Size. Please Check Your Response and Try Again.
7f 87 99 dd fb
cc bc 99 38
a8 80 c6 cf 1e ca 2b d2 c9 ee 69 69 28 71 c1 8c 68 e6 e2 b3 f3 f7 17 9e 46 c6 0e fd
37 00 b2 c9 32 04 cf 11 82 fa 45 24 ac b9 3e 49 c5 ac 9b a4 ba 72 02 e1 e1 64 dd
74 30 40 cf db da b9 64 a8 2c a3 0f eb 87 5f 77 ee 94 33 fa 97 f9 48 9f 96 30 d5 cc
12 c6 76 9e 62 49 e6 35 c7 80 26 f4 1f 6b a0 fa 36 c2 35 57 d1 30 65 3a 9d 57 6b
d3 d2 0d 74 f0 68 ec e6 5e fc f1 15 ac 38 9f 19 e7 ab 1e f3 31 63 80 d4 4a 51 f6 fd
Decrypted Text
Decrypt
Clear


DES Decryption Tool
DES DECRYPTION TOOL
Key:
f007ba11ba5eba11
Mode:
CFB (Cipher Feedback)
Initialization Vector (IV):
746daff7ce2056cd
Enter the Encrypted HEXIDECIMAL Phrase Below:
18 15 43 c4 c7 ee e9 44 03 68 17 4c 6c 00 73 8f d2 63 20 e9 6c 3a 63 a0 c8 a5 90
7f 87 99 dd fb 9b d7 94 4c 96 bc 72 ef 8b f8 15 cc e5 36 15 ca 5b 7e 32 cc bc 99 38
a8 80 c6 cf 1e ca 2b d2 c9 ee 69 69 28 71 c1 8c 68 e6 e2 b3 f3 f7 17 9e 46 c6 0e fd
37 00 b2 c9 32 04 cf 11 82 fa 45 24 ac b9 3e 49 c5 ac 9b a4 ba 72 02 e1 e1 64 dd
74 30 40 cf db da b9 64 a8 2c a3 0f eb 87 5f 77 ee 94 33 fa 97 f9 48 9f 96 30 d5 cc
12 c6 76 9e 62 49 e6 35 c7 80 26 f4 1f 6b a0 fa 36 c2 35 57 d1 30 65 3a 9d 57 6b
d3 d2 0d 74 f0 68 ec e6 5e fc f1 15 ac 38 9f 19 e7 ab 1e f3 31 63 80 d4 4a 51 f6 fd
Ron Maestri was the head coach of the Privateers baseball team for 16 years from 1972 to 1985. He is the most successful Privateer coach having taken the team to compete in NCAA Division 1 World Series. The Privateers are the first team in the state to win that honour!
Decrypt
Clear


DES Decryption Tool
DES DECRYPTION TOOL
Key:
f007ba11ba5eba11
Mode:
ECB (Electronic Code Block)
Initialization Vector (IV):
16 Digit HEXIDECIMAL Phrase
Enter the Encrypted HEXIDECIMAL Phrase Below:
64 27 63 80 7a 44 6b 98 3c 2d c9 8f 22 de 55 08 40 29 56 19 48 98 6d 1f 02 2e 03
8a 62 3c b3 bf 51 70 d3 2e db cd 93 1b 13 45 e7 a2 0f 16 82 d2 4d 71 55 c3 ff 12
25 38 07 cb 10 c3 80 cf 73 be 6c 22 7d db 55 75 e6 df a5 fb eb 8d ec 81 6f 75 39 de
a8 6b 92 cf d4 63 49 10 3d 14 4f 69 f3 2c 52 7f ab 28 cc cb 8c e9
The NBA team Utah Jazz actually began as New Orleans Jazz in 1974. This explains their strange name!
Decrypt
Clear