**Project Report**

1. **ECB and CBC mode of encryption and decryption in AES**

**Electronic Cipher Block:**

1. ECB is broken as a generic cipher - it is not indistinguishable under chosen plaintext attack - as repeating blocks will confer information to an attacker. That particular piece of information doesn't let itself be quantified compared to the number of bits within the key. It's even completely separate from the key size by definition (AES-256-ECB will be as broken as AES-128-ECB).

2.ECB uses same key without initial vector for encrypting the data and thus it is not secure for generating a cipher text.

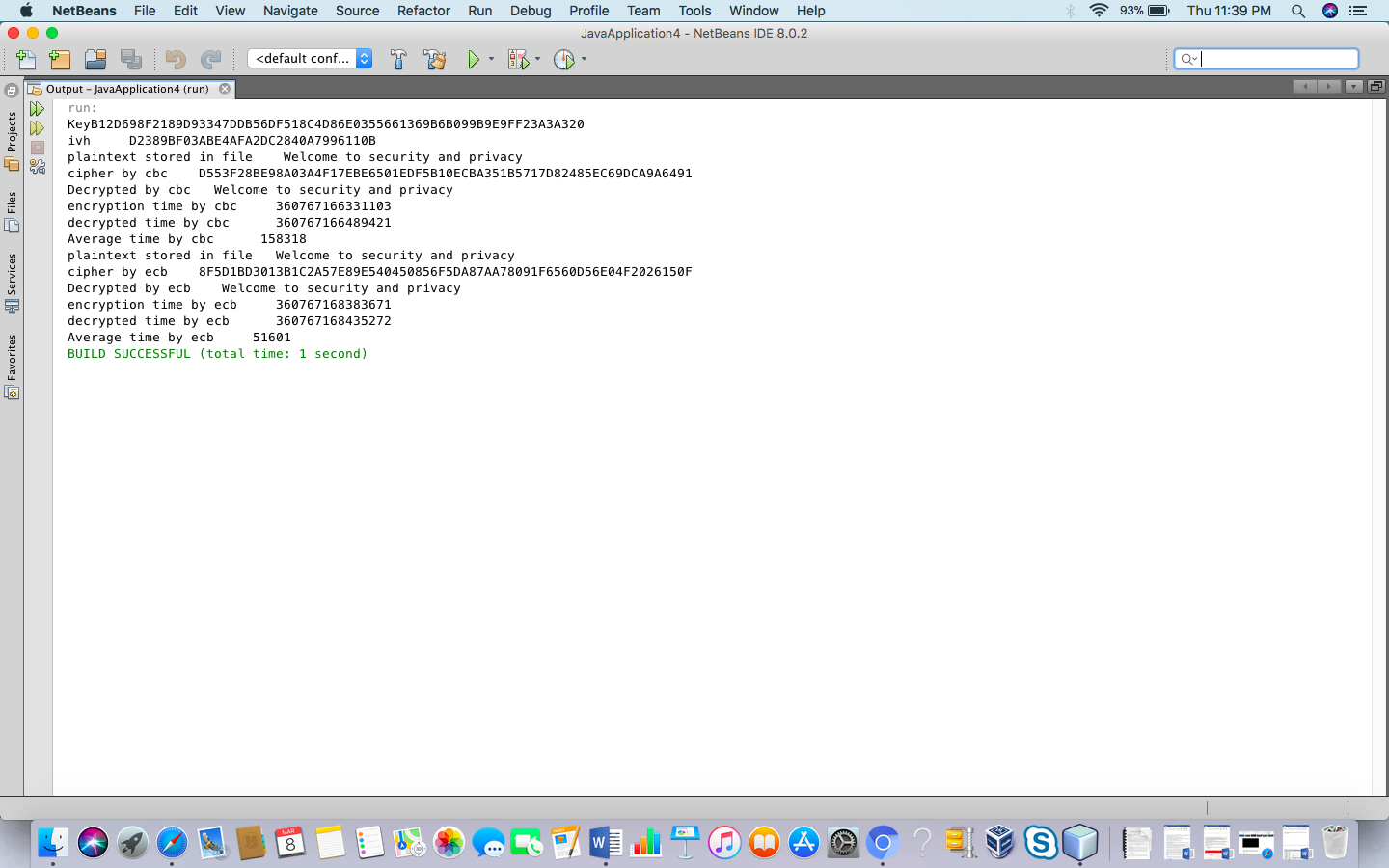
3.The main reason not to use [ECB mode](https://en.wikipedia.org/wiki/ECB_mode) encryption is that it's not [semantically secure](https://en.wikipedia.org/wiki/Semantic_security) — that is, merely observing ECB-encrypted cipher text can leak information about the plaintext (even beyond its length, which all encryption schemes accepting arbitrarily long plaintexts will leak to some extent).

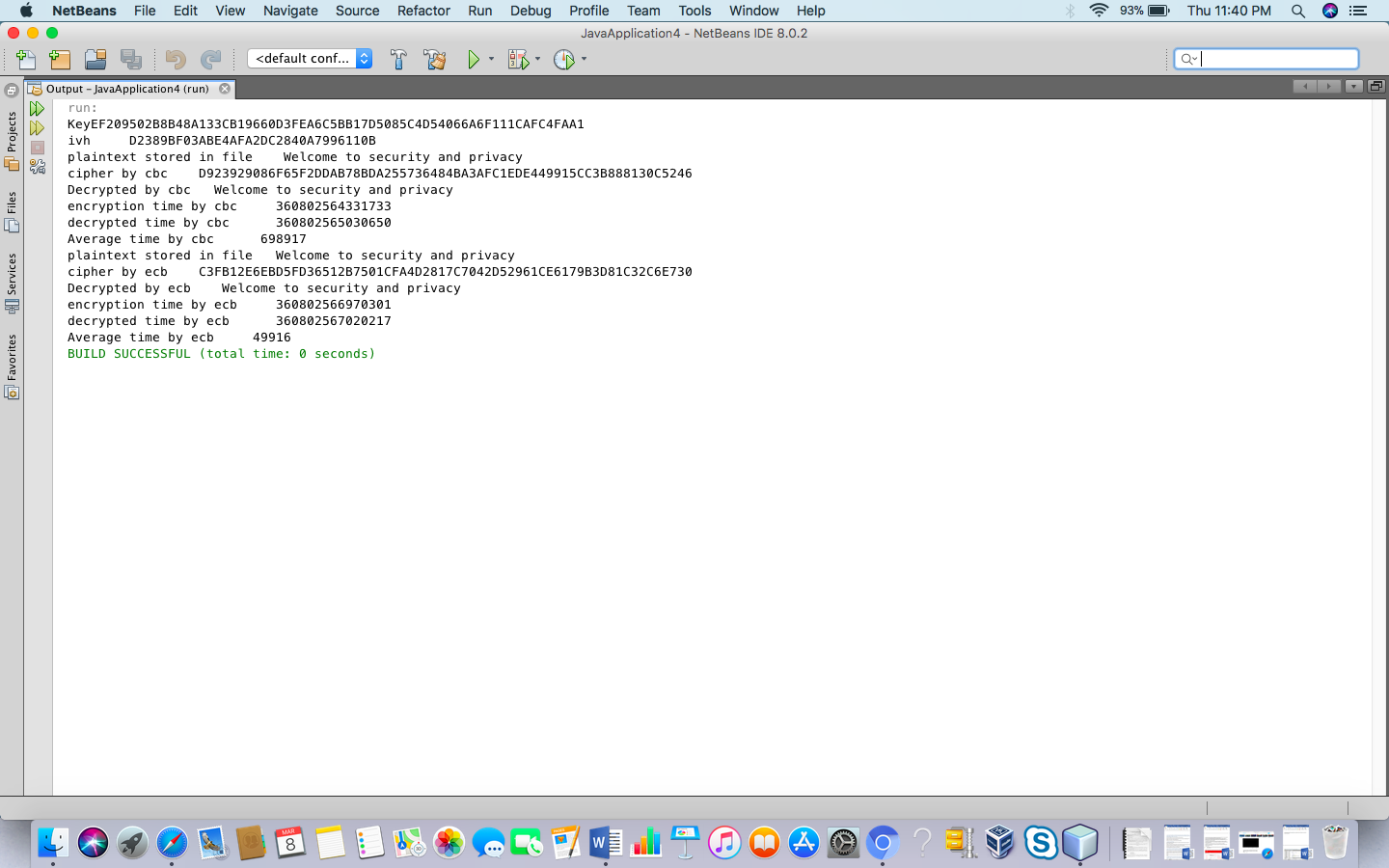
**CBC mode:**

1.CBC with a **static IV** is vulnerable for the first blocks that could be identical. CBC with a predictable IV is vulnerable as well. However, CBC when used correctly is secure for confidentiality of data at rest.

2.The key stays protected by the block cipher itself. If that's "cracking the cipher" then any mode is OK. However, the key only exists to protect the confidentiality of the plaintext. And if that is compromised the security of the system is compromised. So, the security of the cipher is inconsequential.

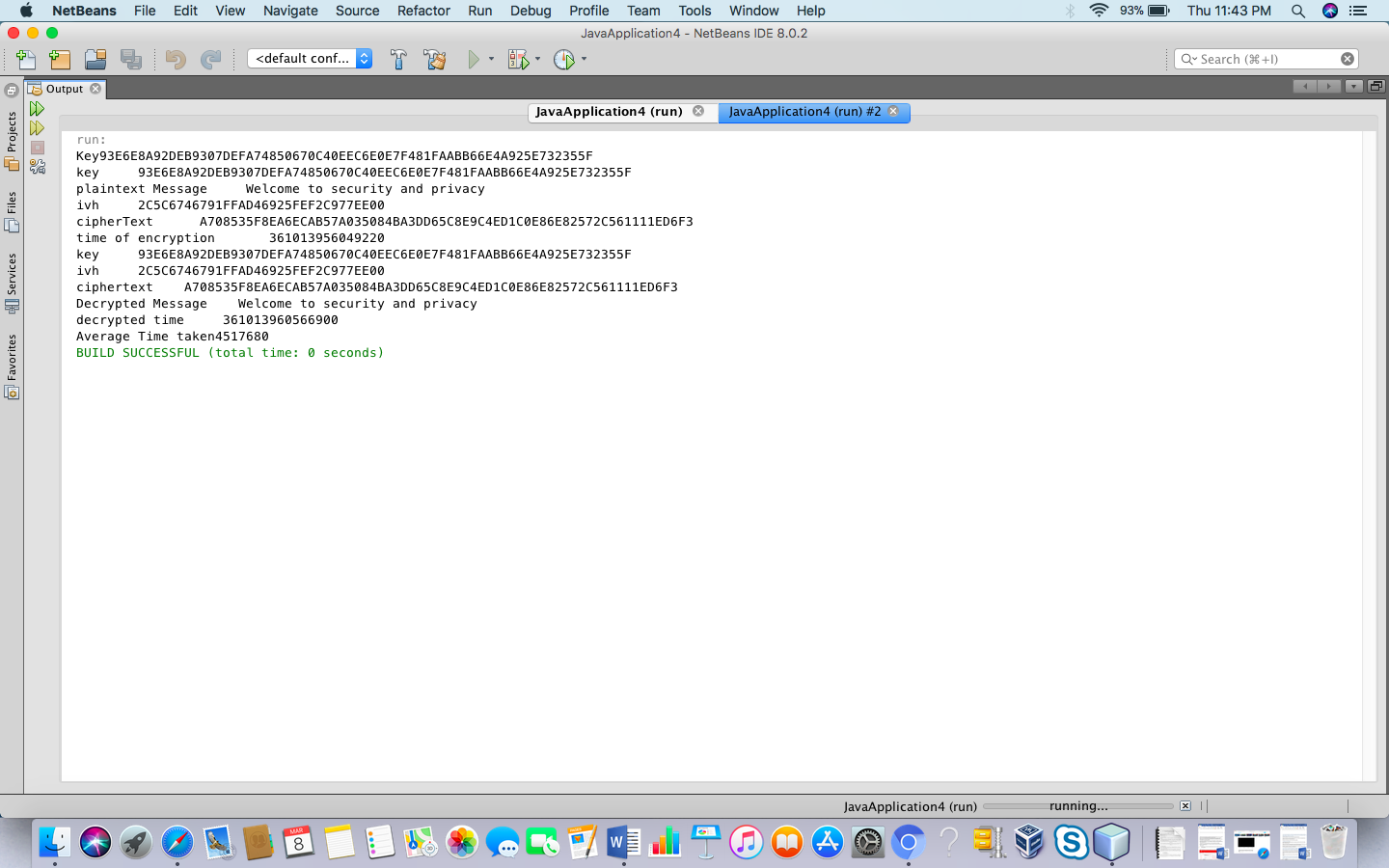
The following screenshots data cipher text and decrypted plaintext for CBC and ECB modes respectively.





The time taken in ECB mode is generally less as compared to that in CBC mode, in most of the cases as shown above.

1. Total time taken by whole program (including keyGen (), Enc() and decryption()) is 1t o 2 seconds



This program shows encryption and decryption time in Nano seconds and difference between the two in average time.

1. The project was giving illegal key size exception for 256-bit key length in AES but after downloading new Java Cryptographic extension package for java 8 and replacing security jar files of US policy jar and local policy jar it started working.

The cryptographic packages used in this project are:

1. java.security.SecureRandom : to generate secure random number which generate random 128 bits for initial parameter and random 256 bits for key
2. javax.crypto.Cipher : It is used to generate cipher text from the key being generated in keyGen function and encrypts it using the instance of AES.
3. javax.crypto.KeyGenerator : It is the package which is used to generate key.
4. javax.crypto.SecretKey: It is the package which is used to store bits of generated key and converts it into secretkey used to generate cipher text by taking instance of AES.
5. javax.crypto.spec.IvParameterSpec: This package is used to generate iv parameter to make CBC mode working.
6. SecretKeySpec: It is used to retrieve bytes of secret key stored in file and converts it into real key by taking the instance of AES algorithm.

Configurations

This project is being made in java 8 and the OS used is MAC.

The Operating System used is MAC

java version "1.8.0\_144"

Java(TM) SE Runtime Environment (build 1.8.0\_144-b01)

Java HotSpot(TM) 64-Bit Server VM (build 25.144-b01, mixed mode).

The parameters you require are:

1. KeyGenerator key; it is the key generated by KeyGeneration package.
2. Cipher c; it is cipher package to generate cipher text.
3. SecretKey sec; it is for generating secret key which is passed as parameter to encrypt and decrypt data
4. byte [] encrypted; it is used to store cipher text bytes.
5. IvParameterSpec ivParameterSpec; It is used to generate iv parameter for CBC mode.
6. byte [] decrypted; it is stored to store decrypted data.
7. SecretKeyspec s= It is used to converts bytes stored in file into actual key by the help of aes algorithm.
8. Dataconverter : to convert key , iv and cipher bytes to hexadecimal and again back to bytes.