Healthcare Data Query Attacks on Block-Cipher Modes of Operation

Isaiah Sinclair
University of Guelph

ABSTRACT

Block ciphers, such as AES, are designed to encrypt data for a fixed block size. They use modes of operation such as Electronic Code Book (ECB) or Cipher Block Chaining (CBC) to encrypt data that exceeds the specified block size for the block cipher algorithm. Certain modes, such as ECB, have deterministic properties that pose security vulnerabilities [2, 3], as shown many years ago in the infamous "ECB Penguin" image [13]. This paper shows the security and run-time performance trade-offs between ECB and CBC, particularly for static health data that contains information about the diabetes status of individuals. It demonstrates a practical representation of how similar plaintext blocks in ECB can lead to similar ciphertext blocks, allowing an attacker to easily extract information about the data's contents from the ciphertext in some cases. The KPAs (Known Plaintext Attacks) conducted show a high degree of diabetes classification accuracy in certain scenarios for AES-ECB encrypted data, demonstrating some of its security flaws. In comparison, AES-CBC encryption did not provide enough information to classify individuals who had diabetes with any degree of accuracy, showing the robustness of its algorithm against these kinds of attacks.

ACM Reference Format:
Isaiah Sinclair. 2024. Healthcare Data Query Attacks on Block-Cipher Modes of Operation. *Final Project Report, SoCS, University of Guelph (CIS*4520),* 7 pages.

1 INTRODUCTION

Block ciphers, such as AES, are used in a wide variety of applications. To encrypt/decrypt data, they operate on the data in fixed-length blocks (i.e. 128 bits) and perform the block cipher algorithm on each block. This poses an issue when the data is not the exact size of the fixed block length: how can a block of data that is not the exact length of the fixed block length be encrypted/decrypted? In practice, this problem is addressed by splitting large data into multiple blocks of the specified fixed length. Any blocks that are under the fixed length are padded. Once the data is split into blocks, the block cipher algorithm is applied to each block to encrypt/decrypt the data. The way the block cipher algorithm is repeatedly applied to the data to encrypt/decrypt multiple blocks is called a "mode of operation".

There are a variety of block cipher modes, but they're generally broken up into two different types: deterministic and probabilistic. Given a key, a deterministic mode of operation (i.e. ECB) will encrypt repeated plaintext blocks into the same ciphertext. Alternatively, probabilistic modes (i.e. CBC) use techniques (such as the use of random initialization vectors) to provide a pseudo-random effect. This means that repeated plaintext blocks will be encrypted into different blocks even with the same plaintext.

ECB is a very simplistic algorithm, but its deterministic properties pose a variety of security vulnerabilities that can be exploited. Specifically, plaintext that is very repetitive will produce correspondingly repetitive ciphertext, given the deterministic qualities of ECB. Repeated ciphertext can indicate patterns that attackers can exploit to infer certain information about the encrypted plaintext. Figure 1 is an example of an infamous image that was popularized online, where someone encrypted an image of a penguin (left) using ECB (centre) [13]. As can be seen, the encrypted image of the penguin using ECB still resembles much of the properties of the original image, allowing the penguin to still be deciphered from the encrypted data.

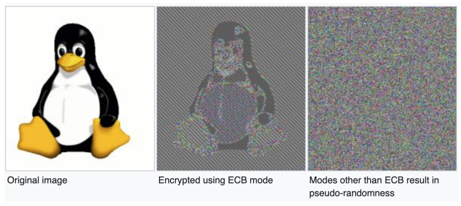

Figure 1: The infamous ECB Penguin image

With this in mind, CBC was created to introduce enough diffusion and confusion to prevent some of the security vulnerabilities apparent in deterministic modes. Along with the randomly generated key, it generates an additional random bit-string, an initialization vector to add another layer of pseudo-randomness. This initialization vector is then XORed with the first plaintext block, and the resulting data is input into the block cipher algorithm. The chaining effect of CBC is realized in the encryption of the next plaintext block, as the previous ciphertext (from the first block) is XORed with the second plaintext block, to chain the effects of the random IV that were present in the first ciphertext output. This ensures that the pseudo-random effect of the IV is maintained while increasing diffusion and confusion to the encrypted data across each block. As a result of these added features, CBC is more secure than ECB and is used in many modern cryptographic applications [2].

In past research, a wide variety of trade-offs between the use of ECB and CBC as modes of operation have already been described in theoretical explanations. These consistent of mathematical proofs showing the security improvements from CBC's additional features, algorithmic complexity and error propagation analysis, which is of

*CIS*4520, SoCS, University of Guelph, Guelph, ON, Canada*
© 2024 Copyright held by the owner/author(s).
ACM Conference Format.

major concern in this field. However, there is a void of research performed in analyzing the practical implications of the theoretically-proven trade-offs between ECB and CBC. Figure 1 above provides a visual representation of some of the security vulnerabilities of ECB, but this has not been taken further in many research projects. Specifically, the ability to query and derive information from the encrypted ECB plaintext has not been quantified.

Additionally, the focus of this paper around health data is driven by the increasing need for healthcare companies to follow good security practices. As hospitals and other healthcare institutions digitalize information, data will need to be encrypted and transmitted with the best security practices considered. There has not been a focus on the applications of ECB and CBC to encrypted health-related data in this field of research.

This research paper seeks to demonstrate the practical implications of the security vulnerabilities of ECB, in comparison to CBC. It uses a simplified example of data that may be encrypted in a healthcare setting, specifically CSV data indicating if individuals have diabetes or not [5], and is converted to a string to be encrypted. The data is encrypted using AES-128, with an implemented version of ECB and CBC used as the modes of operation. The two modes of operation are compared by the amount of correct classifications that can be made about the plaintext, from the encrypted data, via a KPA. Specifically, the data will be encrypted, and using a given plaintext and ciphertext pair for a single diabetic we will estimate the total amount of individuals who have diabetes in the dataset. This research demonstrates how given certain levels of determinism in the data, the amount of diabetics can be predicted with up to 100% accuracy, in certain scenarios. This provides sufficient evidence to suggest that health data should use a probabilistic mode of operation, such as CBC, as opposed to ECB despite its slower run-time and error propagation issues.

2 RESEARCH PROBLEMS

Given the research already performed in this field of cryptography, I seek to accomplish the following goals in this research:

(1) Provide a practical demonstration of the security vulnerabilities of ECB in comparison to CBC, using health data.
(2) Demonstrate how information about the plaintext can be queried, extracted and estimated from the ciphertext, using a KPA.
(3) Demonstrate the differences in run-time performance using empirical and theoretical methods.
(4) Provide a recommendation for the best mode of operation to encrypt static healthcare data, between ECB and CBC.

Each of these specific goals answers a general gap within this field of research: while ECB has been shown to be inferior to CBC using theoretical analysis, its vulnerabilities and benefits have not been analyzed as thoroughly using empirical techniques. Accomplishing these goals means that a practical demonstration will be provided of the vulnerabilities of ECB and potentially CBC, as well as other trade-offs associated with the two modes. Using the information provided from the empirical results of my research, I will be able to provide a recommendation pertaining right mode to encrypt health data such as the diabetic status of patients in a hospital.

In constructing these tests for research purposes, a major technical challenge has been constructing an optimal test that can aptly demonstrate the security vulnerabilities of the modes. My tests show some, but not all of the trade-offs between ECB and CBC, and thus other research should be referenced to obtain an understanding of the full scope of the trade-offs between the two modes.

3 RELATED WORKS

In literature, modes of operation have been compared and contrasted in a variety of ways. Bujari and Aribas explained how ECB, CBC, OFB (Output Feedback Mode), CFB (Cipher Feedback Mode) and CTR (Counter Mode) work and analyzed their security, run-time, and error propagation differences. ECB was found to be the most insecure, given how the same plaintext blocks produce the same ciphertext blocks, which makes this algorithm very deterministic. This makes it possible to generate a code book that maps plaintext blocks to ciphertext blocks. It is also vulnerable to substitution attacks which are manipulations in the ciphertext in order to deceive the receiver of the encrypted data. However, a few benefits of using ECB were cited: since the encryption of blocks is completely independent, errors cannot be propagated to different blocks. Additionally, encryption and decryption can be parallelized, which provides a significant run-time performance advantages over the other modes.

In contrast, CBC's chaining properties spread the influence of each plaintext block over many ciphertext blocks during encryption, introducing a higher level of diffusion. The use of the IV adds a pseudo-random effect that makes the mode more secure and probabilistic. Substitution effects do not occur if the IV is properly chosen and is used once. The downside of using chaining is that bit-errors in one plaintext block are propagated across multiple blocks [2]. Also, "block-wise adaptive" attacks can be performed if the message is not atomic [4]. Encryption time is also higher than ECB, given that encryption can't be parallelized while decryption can be [2].

While the trade-offs between different modes are important, the block cipher algorithms that they're used with are just as important to analyze. Riman and Abi-Char compared AES, DES, 3DES and EDES (Educated-DES), and compared their run-time performances and security differences. They found that the new block cipher, EDES was more secure against attacks than the traditional block ciphers has faster encryption times. It is important to note that its decryption algorithm was not compared as it was not implemented at the time of the article being written [11].

To address the issues with ECB and CBC, there have been some suggested improvements for specific use cases. To address the issues with the high determinism in the ciphertext produced by ECB, Al-Wattar proposed an alternate method for image encryption using this mode. Specifically, pixel values would be tied to their corresponding DNA bases, which would add another layer of complexity to AES-ECB image encryption. This added enough diffusion and confusion to the image to not be able to identify the Figures in the image. However, this solution is only limited to images, not AES-ECB encryption in general [1]. To address the error propagation issues with CBC, Vaidehi et al. proposed a fault detection and correction model. This is particulary important, as protecting against faults is particularly important in certain use cases, such as satellites [12].

Lastly, there have been some creative ways to evaluate the security of modes of operation. Malzoemoff et al. developed an automated approach (modelled with a directed acyclic graph) for the analysis of modes, conducting local analysis of the security of the mode at different points in its algorithm. The purpose of this is to simplify the security analysis of block cipher modes to a constraint-satisfaction problem, reducing the time needed to develop and test new modes. With this automated mode checker, they were able to discover 309 unique secure modes, many of which have never been studied before in the literature [6].

4 METHODOLOGY

4.1 General Overview

The purpose of these tests is to demonstrate the security vulnerabilities, particularly in the ECB mode of operation when a KPA is attempted.

4.1.1 Data Description. Given that this research pertains specifically to health data, I encrypted a CSV file that contains information about the diabetic status of 768 individuals. There are a variety of columns, but upon running the program, every column is removed besides the "Outcome" column, which represents a factor variable that indicates whether someone has diabetes or not. If an individual has diabetes, their 'Outcome' variable is denoted by a '1'. Alternatively, if they do not have diabetes, the variable is denoted by a '0' [5]. Out of 768 people, 268 have diabetes and 500 do not. For encryption purposes, this data is converted to a string, concatenating all the data together with a provided amount of spacing (to add more repeated values/determinism to the data).

4.1.2 Test Parameters. The following are the parameters that the user can input when conducting tests:

- Data Padding Length: The amount of space to pad data value with.
- Specific Diabetic to Conduct the KPA With: The user can choose which specific diabetic (from 268 diabetics in the dataset) they want the plaintext and ciphertext pair for.
- Mode of Operation: ECB and CBC are implemented.

The data padding length adds an amount of space to the end of each data value. This is to parameterize the amount of determinism in the data to be encrypted.

You will note that the user of the program was able to choose the specific plaintext and ciphertext combination for a specific diabetic. In KPAs, this is not realistic and is more representative of a Chosen Plaintext Attack (CPA). However, this was included as a parameter as I wanted to test (see Test Case #2) if the specific diabetic that the attacker has the plaintext, ciphertext combination of effects and the ability to properly classify diabetics using the encrypted data. In essence, this functionality would not be available to an attacker typically in a KPA, but it is included for research purposes. It is also important to mention that in the context of the simulation of the KPA, we implement an attack that assumes that the attacker only has access to the plaintext and ciphertext pair of a single diabetic, not all of the data.

4.1.3 Diabetes Classification Method. The purpose of these tests is to use the encrypted data and a small amount of information available in the KPA to correctly classify as many of the 268 diabetics as possible. Note that to prevent incorrect classifications of non-diabetics as diabetics, I have also measured how many non-diabetics were correctly *not* classified as diabetics.

A KPA against this dataset will be simulated as follows:

(1) The user will encrypt the CSV data given an implemented mode of encryption (ECB or CBC) and a padding length (amount of space) that will be appended to the end of each data value before encryption.
(2) They will choose a specific diabetic in the dataset (e.g. the second instance of a diabetic in the data) to conduct a KPA with.
(3) They will be provided with the hexadecimal representation of the plaintext and ciphertext of that diabetic.
(4) Given the ciphertext representation of that specific diabetic specified for the KPA, the program will string search for exact matches of that hexadecimal string. It will store each position of these exact matches and interpret each of them as a predicted/classified diabetic.
(5) After the classified diabetics are computed, the results are compared with the actual data. If an actual diabetic is classifed as a diabetic using the program, this is a correct classification of a diabetic. Alternatively, if a non-diabetic is classified as a diabetic accidentally, this is an incorrect classification of a non-diabetic.

4.1.4 Diabetes Classification Measurements. As implied in the previous sub-section, these are the following measurements that will be used to analyze the performance of the two modes of operation:

(1) Diabetic Classification Correctness: number of classified diabetics / number of diabetics within the data set
(2) Non-Diabetic Classification Correctness: number of non-diabetics that were *not* accidentally classified as diabetic / number of non-diabetics within the data set
(3) Run-time Performance: encryption and decryption time of the data, in seconds.

It is important to note that the first two measurements can be interpreted as proportions, and are sometimes represented by percentages in the results. The main measurement to focus on is the first measurement, the Diabetic Classification Correctness proportion. This will be referenced in most of this paper. This is because it provides an estimate of how many of the diabetics were correctly identified from the encrypted data in the KPA, which is the main topic of interest.

4.2 Test Case #1: Using First Diabetic for a KPA

This test runs a series of 32 individual tests (a full 'test suite') and outputs the results. Specifically for each test, it encrypts the same data and attempts to classify diabetics using the plaintext and ciphertext pair for the *first* diabetic. However, a new test is run for different padding lengths (0-15 spaces of padding), and both modes are tested. The main purpose of this test case is to evaluate how many diabetics can be correctly classified from the data in a KPA attack where only the plaintext and ciphertext pair for the very first diabetic in the data set is known by the attacker.

4.3 Test Case #2: Using the Eighty-ninth Diabetic for a KPA

This is the same as test case #1, however instead of using the plaintext and ciphertext pair for the first diabetic, it uses the 89th diabetic. The main purpose of this case is to compare how the results differ when choosing a different diabetic in the data set for the plaintext, ciphertext pair. I was particularly interested if there was a negative linear relationship in the diabetic chosen in relation to how many diabetics could be correctly classified. My hypothesis was that using the ciphertext of a diabetic later in the dataset would make it harder to properly classify other diabetics, suggesting a negative linear relationship between the ith diabetic and the amount of diabetics correctly classified. My reasoning for this is that I thought it would be very unlikely that the corresponding ciphertext for a diabetic later in the dataset would match any of the corresponding ciphertext for diabetics earlier in the ciphertext, given transformations and substitutions performed by AES to each block.

4.4 Parallel ECB Encryption Test Case

This is the same as test case #1, with one minor change: there is a flag called PARALLEL_ECB in the AES.py file. By default (in test cases #1 and #2), it is set to False, indicating that ECB does not use parallel computing to encrypt data. In this test case, I set it to True, to see if parallel computing produced performance advantages for ECB encryption with this data.

4.5 Description of Code

4.5.1 Implementation Overview. As stated, the focus of this project was to investigate some of the trade-offs between ECB and CBC. With this considered, I chose to not implement AES myself, and instead copied and refactored an existing implementation of AES-128 [7]. I implemented my entire program using Python and thus used a Python implementation of AES. For this research, I felt that a 128-bit key was sufficient to demonstrate my research findings and perform the tests, so I left that unchanged. I then implemented the ECB and CBC modes, revising the AES-128 algorithm I found online to suit the implementation of the modes that I implemented. I also utilized the concurrent library to implement parallel processing of ECB encryption, but I have turned that off by default with a flag called PARALLEL_ECB located in AES.py. For key and IV generation, I used functions from pycrytodomex. I also used this library to add padding to blocks, when needed [10].

I also had to implement the functionality to perform deterministic tests against the data set that I used. For brevity, I will explain the implementation of the tests in chronological order of how the functionality is executed. Firstly, I implemented a TestRunner class that runs the command line interface, taking user input about the varying parameters specified in this report. The user also inputs a filename **(of a file with the data folder)**, such as 'diabetes.csv' if it is within the folder. The class instance than reads in the file as a data frame using pandas [9].

After reading the data, the TestRunner class creates and stores an instance of the lstinlineDeterministic class, which is the class that runs deterministic tests. The Deterministic instance creates an instance of the TestData class, which stores the cleaned version of the data frame, that only contains the 'Outcome' column (note that this column indicates if the individual has diabetes or not). This class has the functionality to calculate the actual number of diabetics, perform the string search algorithm to predict the number of diabetics, as well as compare the predictions versus the actual results. After this, the command line interface asks if the user wants to run the full test suite, or a single test. If the full test suite is indicated via user input, runTestSuite is run, with a series of tests being run with different padding lengths and the two modes. These individual tests are run by calling runTest with the specified test parameters – recall that this function is also run when a single test is run by the user.

The general flow of tests work as follows: the data is encrypted using ECB or CBC, and the encryption time is measured. The data is also decrypted only to obtain the decryption time. Then, using the specified diabetic for the KPA, the plaintext and ciphertext pair for that diabetic is found. The encrypted data is then string-searched to create an array indicating the predicted/classified diabetics, and this is compared with the actual diabetics to compute the correctness measurements. The results are stored in an instance of TestResult and returned. Note that the plaintext and ciphertext are stored in a "plaintext.txt" and "ciphertext.txt" file for demonstration purposes. For the runTestSuite function, it takes each returned TestResult instance and uses this information to output it to a CSV file.

4.5.2 To Start the Program. The user should install Python, pycryptodomex and pandas before attempting to run the code.

(1) Ensure that there is a CSV file in the data folder corresponding to the format of the 'diabetes.csv' file that I submitted.
(2) Then, run the program from the root directory of the project by entering: python TestRunner.py
(3) Enter a CSV file name when prompted, for example: diabetes .csv
(4) Indicate whether you want to enter a single test (0), or the full test suite (1).

4.5.3 To Run a Single Test.

(1) Enter a mode of operation to encrypt with (ECB or CBC).
(2) Enter a padding length. Note that the data length for one single data value (a 1 or 0) will be equal to 1 byte + the provided padding length (amount of appended spaces).
(3) Enter a specific diabetic to conduct the KPA with (to obtain the plaintext, ciphertext pair of).
(4) Review the results.

4.5.4 To Run the Test Suite.

(1) Enter a specific diabetic to conduct the KPA with (to obtain the plaintext, ciphertext pair of).
(2) After the test suite has run, enter a CSV to save the results to.
(3) Review the results. The output CSV file will be in the 'results' folder.

4.6 Environment Details

- Lenovo IdeaPad
- AMD Ryzen 7 5825U with Radeon Graphics, 2.00 GHz
- 16.0 GB (13.8 GB usable)
- x64-based processor
- 64-bit operating system (Windows 11)
- Additional Sofware Installed: Python, PyCryptodomex, pandas, Python time module, concurrent.futures

5 RESULTS

I conducted all of my test cases by running the program's automated test suite. I have created a variety of graphs to illustrate the results of each test case. Note that the correctness proportions can be represented as percentages.

5.1 Test Case #1: Using First Diabetic for a KPA

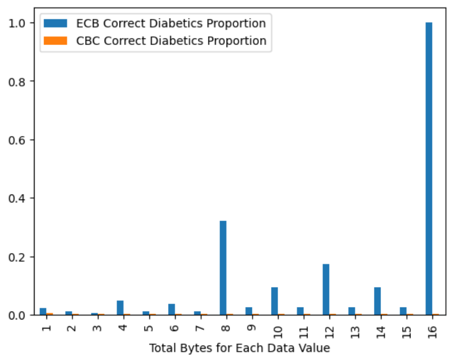

Figure 2: Comparing the proportion of diabetics correctly classified in KPA at Using the first diabetic when using ECB vs CBC

In Figure 2, the proportion of correctly classified diabetics are compared for different levels of determinism (the higher the amount of bytes for a signle dat value, the more padding) and for the two modes. Note that for CBC (orange), the KPA attacker typically correctly identified 0.4% of the diabetics. This means that the attacker was unable to use the given ciphertext of a single diabetic to identify other diabetics. Thus, the design of the algorithm of CBC is generally resistant to attacks that seek to exploit the determinism of the resulting ciphertext, given plaintext that has many repeated attributes. However, with ECB (blue), there is a general upward trend that implies that the higher the level of padding (determinism in the data), the more diabetics the attacker can classify. For example, when there are 8 bytes for each data value (1 byte of data, 7 bytes of padding), 32.1% of the diabetics were correctly classified, while the remaining were not.

Most notably, when a single data value takes 16 bytes, 100% of the diabetics can be correctly classified. Since the AES algorithm encrypts data in fixed 16-byte (128-bit) blocks, each block of data corresponds to exactly 1 data value (with padding) in the CSV file. Since AES encrypts data the same way given the same input, this means that ECB will encrypt each diabetic and non-diabetic to the same ciphertext. Thus, if a plaintext and ciphertext pair of a single diabetic is provided to an attacker, the attacker can classify all of the diabetics in the dataset with only a single diabetic they know the ciphertext of.

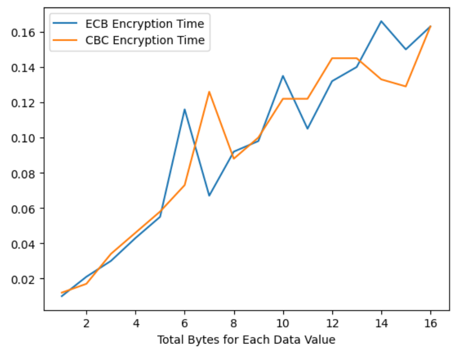

Figure 3: Encryption Time for ECB vs CBC at Different Data Lengths

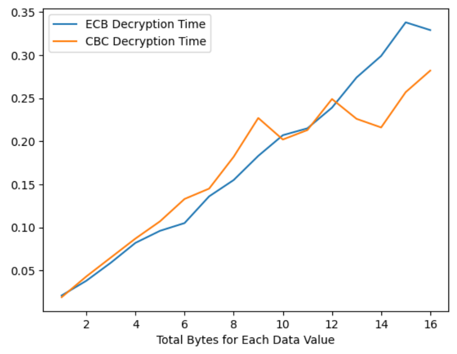

Figure 4: Decryption Time for ECB vs CBC at Different Data Lengths

5.1.1 *Encryption/Decryption Time Comparison.*

The encryption and decryption times in test #1 are compared between ECB and CBC at different data lengths in Figures 3 and 4. While the encryption time takes longer for both modes as the data length increases, it is not clear which mode performs consistently faster, even after repeated tests. It is important to note that in this test, the ECB implementation used is not parallelized, which could be the reason why ECB is not faster as research suggests. The two modes are both $O(n)$ and $O(1)$ in terms of space and time complexity, so I compared the same test with parallel encryption in Figure 7 to see if the performance advantages of ECB could be realized through parallelization. Given the small amount of data to encrypt, it also the runtime complexity disadvantages of CBC compared to ECB are harder to demonstrate.

5.2 Test Case #2: Using the Eighty-ninth Diabetic for a KPA

In the following test, instead of using the first diabetic for the KPA, the eighty-ninth diabetic was used. The purpose of this was to see

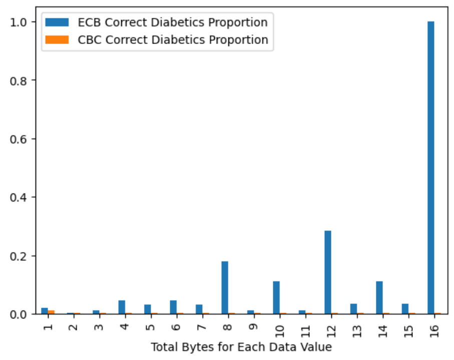

Figure 5: Comparing the proportion of diabetics correctly classified in KPA at using the eighty-ninth diabetic when using ECB vs CBC

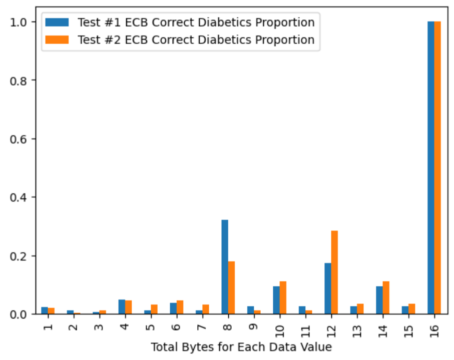

Figure 6: Comparing the proportion of diabetics classfied in KPA with the first diabetic vs the eighty-ninth diabetic when using ECB only

if there is some sort of negative linear relationship between the position of the diabetic used for the KPA and the amount of diabetic sthan can be classified correctly. However, as shown in Figures 5 and 6 this hypothesis was proven to be false by these results. Using the 89th diabetic for the KPA (orange) versus the 1st diabetic (blue) sometimes produces a higher accurate classification rate compared to the first in some cases. To further analyze this relationship, I picked a variety of different diabetics in the dataset to simulate a KPA with to see how well the attacker could correctly classify diabetics, and there was no apparent linear relationship. For example, conducting a KPA attack using the last diabetic (the 268th one) in the dataset produced almost zero accurate classifications. Given that certain diabetics in the dataset will resemble the same ciphertext as previous ones, this relationship is spurious.

5.3 Parallel ECB Encryption Test Case

As can be seen in Figure 6 in comparison to Figure 3, the encryption time unexpectedly increased after adding parallel processing to ECB's encryption algorithm. This could be explained to the overhead costs associated with parallel computation, paired with the small amount of data to encrypt [8]. As encryption time was not

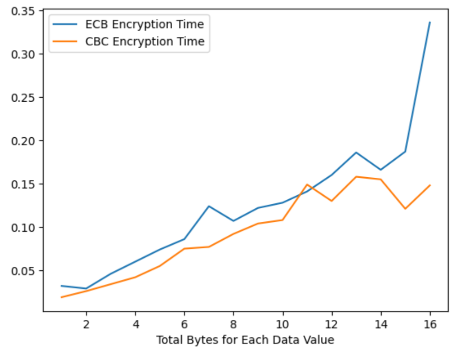

Figure 7: Encryption time of ECB (parallelized) vs CBC

the primary focus of this research, this was not investigated further. However, from other empirical work it has been shown that parallelization provides a run-time performance advantage for ECB over modes such as CBC [2].

6 CONCLUSION

In this project, we reviewed two traditional block cipher modes of operation, ECB and CBC. We analyze the security differences between the two modes with theoretical analysis from existing literature, and empirically investigate the differences by simulating KPA attacks. These KPA attacks were conducted to classify as many diabetics as possible from the encrypted data, given the ciphertext of only 1 diabetic. We found that CBC was by far the superior mode, classifying nearly 0% of the diabetics appropriately. Alternatively, the simulated attacks generally saw an increase in the percentage of diabetics that were able to be accurately classified with ECB data became more repetitive (padding increased). When the data size (in bytes) equalled the size of the blocks used in this implementation of the AES encryption (16 bytes), accurate classifications reached 100%.

In addition to the empirical security analysis of the modes, this project also timed the encryption time of encryption and decryption in both modes. We did not find significant evidence of a faster run-time for either of the modes for data of this size, however this is presumably due to the small sample size of the data.

As a result of this analysis, we find that due to the clearly demonstrated security vulnerabilities, CBC is the recommended block cipher mode for health data similar to the diabetes data this project uses. Despite ECB's run-time performance and error propagation advantages that have been documented in the literature [2], the security vulnerabilities far outweigh the benefits and thus make this an unsuitable mode to use if one seeks to have a high level of security.

7 FUTURE WORK

In the future, I believe that more security vulnerabilities should be studied in block cipher modes, particularly with regard to CBC, given its prevalence in the field of security. Particularly, I believe that a practical demonstration of block-wise adapter attacks on modes such as CBC should be performed and analyzed, particularly about how they could be exploited in regular applications. I also think that a deeper analysis of the error propagation issues of CBC in comparison to other modes of operation should be investigated further. Lastly, I believe that this practical demonstration could be extended by including other modes of operation, such as Cipher Feedback (CFB) or Counter (CTR) mode.

REFERENCES

[1] Auday H. AL-Wattar. 2023. A New Approach For the Image Encryption Using AES Cipher in ECB Mode. *Turkish Journal of Computer and Mathematics Education* 14, 1 (2023), 1061–1074.

[2] Diedon Bujari and Erke Aribas. 2017. Comparative analysis of block cipher modes of operation. In *International Advanced Researches & Engineering Congress*. 1–4.

[3] Computerphile. 2020. Modes of Operation - Computerphile. (2020). https://www.youtube.com/watch?v=Rk0NIQfEXBA Last accessed 14 April 2024.

[4] Antoine Joux, Gwenaëlle Martinet, and Frédéric Valette. 2002. Blockwise-Adaptive Attackers Revisiting the (In)Security of Some Provably Secure Encryption Modes: CBC, GEM, IACBC. In *Advances in Cryptology — CRYPTO 2002*, Moti Yung (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 17–30.

[5] Kaggle. 2020. Diabetes Dataset. (2020). https://www.kaggle.com/datasets/mathchi/diabetes-data-set/data Last accessed 14 April 2024.

[6] Alex J Malozemoff, Jonathan Katz, and Matthew D Green. 2014. Automated analysis and synthesis of block-cipher modes of operation. In *2014 IEEE 27th Computer Security Foundations Symposium*. IEEE, 140–152.

[7] Marcomini. 2020. Building AES-128 from the ground up with python. (2020). https://medium.com/wearesinch/building-aes-128-from-the-ground-up-with-python-8122af44ebf9 Last accessed 14 April 2024.

[8] R. Nelson, D. Towsley, and A.N. Tantawi. 1988. Performance analysis of parallel processing systems. *IEEE Transactions on Software Engineering* 14, 4 (1988), 532–540. https://doi.org/10.1109/32.4676

[9] pandas. 2024. pandas. (2024). https://pandas.pydata.org/ Last accessed 15 April 2024.

[10] PyPI. 2024. pycryptodomex. (2024). https://pypi.org/project/pycryptodomex/ Last accessed 14 April 2024.

[11] Chadi Riman and Pierre E Abi-Char. 2015. Comparative analysis of block cipher-based encryption algorithms: a survey. *Information Security and Computer Fraud* 3, 1 (2015), 1–7.

[12] M Vaidehi and B Justus Rabi. 2014. Design and analysis of AES-CBC mode for high security applications. In *Second International Conference on Current Trends In Engineering and Technology-ICCTET 2014*. IEEE, 499–502.

[13] Filippo Valsorda. 2013. The ECB penguin. (2013). https://words.filippo.io/the-ecb-penguin Last accessed 14 April 2024.