FIT3164 DATA SCIENCE PROJECT 2

FINAL PROJECT REPORT

SCHOOL OF INFORMATION TECHNOLOGY
MONASH UNIVERSITY MALAYSIA

TITLE: CRYPTOML

SCOPE: MACHINE LEARNING WITH CRYPTOGRAPHY

SUPERVISOR: PROF. RAPHAEL PHAN

DUE DATE: 22ND OCTOBER 2021

WORD COUNT: 7820

TEAM: FIT3163_MA_3

1. HAW XIAO YING 29797918 (PROJECT MANAGER)
2. KHOR CHENG EE 30755980 (TECHNICAL LEAD)
3. CHAN YU ANNE 29797012 (QUALITY ASSURANCE)

Table of Contents

1. Introduction

In recent times, IoT (Internet of Things) devices have been widely used and therefore the usage of lightweight cryptography has skyrocketed. This is because the IoT devices are unable to connect to the power supply all the time, while lightweight block ciphers only perform algorithms that require less computing power (Toshihiko, 2017). The purpose of CryptoML is to research the security of these lightweight ciphers, hence we will be performing attacks on them, in particular the Speck Cipher. The Speck Cipher, released by the National Security Agency (NSA) has its defined number of encryption rounds according to its block and key size. Residual Neural Network, which is also known as ResNet, will be trained to act as an attacker on the round-reduced Speck Cipher.

The aim of this project is based on the binary classification problem where we will make modifications to improve a ResNet model to see how it affects the classification result. From the project proposal, the modifications include changes to the number of plaintext, changes to the number of relationships between plaintexts and also changes to the input parameters when training the ResNet model, where we will observe whether these modifications produce a ResNet model with better performance. To evaluate the ResNet model, plaintexts will be encrypted into cipher texts where it is then passed to the model as input and the model will predict and classify the cipher texts.

Moreover, CryptoML aims to produce visualizations for the classification results by using explainable Artificial Intelligence tools so that users are able to have a better understanding of the results. The tool that we have decided to utilize for this is a library named SHAP (SHapley Additive exPlanations), which consists of various model explainers. The SHAP explainers will take in the model predictions and the evaluation data then calculate the shapley values and produce explanations on the results predicted by the ResNet models with regards to the contribution of each feature to the predictions. With the SHAP visualizations in place, we will then be able to explain the prediction results and analyse the modified ResNet model with different input parameters, different number of plaintexts and also different number of relationships between plaintexts.

2. Background

2.1 Literature Review

2.1.1 Speck

Speck is a lightweight block cipher that uses secret key cryptography algorithms. Lightweight cipher is used more often nowadays because of the wide use of Internet of Things (IoT) devices. The number of connected devices has increased and more information is shared between devices, hence, the potential that an attacker could steal confidential information also increased. According to Buchanan, Li and Asif (2017), the main characteristic of lightweight cryptography is that the block size, key and complex rounds are usually smaller in comparison to conventional cryptography.

Speck has various block sizes and key sizes, such as Speck 32/64 and Speck 48/96. The former number indicates the block size while the latter denotes the key size. So (2020) states that the block size of Speck is always 2 words, while the key size may be 2 to 4 words. Speck uses the add–rotate–xor algorithm by encrypting the word with two rotations, which is adding the word on the right to the left and the key is xor-ed to the word on the left, and finally the right word is xor-ed with the left word (Beaulieu et. al., 2015).

One full round of encryption of the Speck 32/64 is 22 rounds which means the ciphertext is encrypted 22 times. According to So (2020), there have been no successful attacks on a full round Speck cipher. This shows the Speck Cipher is secure enough in protecting the plaintext. In Gohr's (2019) research paper, they managed to decipher the ciphertext with a deep learning neural network when the Speck 32/64 is only encrypted with 11 rounds. The neural network was also trained to classify the output of plaintexts from random ciphertexts.

2.1.2 Deep Learning

Deep learning is a type of machine learning technique that is able to learn unsupervised data without human control (Hargrave, 2021). In the field of cryptography, there have not been many studies which apply deep learning on ciphers and Gohr's research paper is one of the few that we will be reviewing.

Gohr published his research paper in 2019, where he researched on how to improve the attacks on Speck 32/64 through reducing rounds in deep learning. The design of his deep learning model that is able to produce attacks on Speck Ciphers consists of an input layer, two layers of convolutional neural networks and a prediction head that is densely connected.

Every layer of the convolutional network is made up of 32 filters and each filter layer consists of convolutions, a normalization batch and a layer for rectification. Batch normalization is used to normalize the input to the rectifier layer (Brownlee, 2019). Skip connections are then added from the layer for rectification to the input of the convolutional blocks. There are two hidden layers and one output layer in the prediction head of Gohr’s model and both of the hidden layers are densely connected. The first hidden layer also consists of a normalization batch and a layer for rectification, while the normalization batch is absent in the second hidden layer.

Gohr (2019) trained his deep learning model by inputting some randomly generated data so that it has evenly distributed keys and plaintexts. The plaintexts were then encrypted k times to produce k-rounds encrypted Speck Ciphers. These plaintexts were then used in the training process of his deep learning model. To improve the classification of key search, Gohr’s team reduced the encryption rounds of Speck to 7 rounds before applying brute force on key search to evaluate the test data which consists of the newly generated ciphertexts. Gohr (2019) also trained the fast distinguisher model by using the key search. However, the training above did not work for 8 encryption rounds of Speck Ciphers and above as the deep learning model did not learn useful functions from the training set.

Gohr also proposed another idea of basic attack where he increased the number of rounds of attack on Speck Ciphers from seven to nine rounds. This basic attack can be further improved by using generic optimization algorithms to improve on the key search (Gohr, 2019). Besides, Gohr (2019) states that the key search function is focused on the structure of the ciphertexts. With all these improvements, Gohr (2019) managed to build a much more efficient attack on the Speck Ciphers and we will be focusing on modifying other parameters for training the deep learning model to further improve it based on his findings.

2.1.3 ResNet

Residual Neural Network, also known as ResNet is a type of Artificial Neural Network (ANN) that practices deep learning, which is a type of machine learning. Artificial Neural Network is an algorithm inspired by the human brain which consists of artificial neurons and synapses that are connected to each other by nodes just like the human brain. The ANN machine learning model will go through training, layer by layer according to the neural network. Frankenfield (2020) states that ANN models learn to differentiate the pattern of the training dataset. Unlike normal ANN, ResNet often skips some layers and jumps to other layers.

The reason why ResNet performs skip connections is to solve the problem of gradient vanish (Ruiz, 2018). Ruiz states that researchers normally think that the deeper the neural network, the better the performance of trained models. However, the more complex it is, the performance might get degraded after some layers. This is because the gradient might decrease to zero from the loss function after a few layers of network which is similar to a model that has not been trained. Dwivedi (2019) states that with ResNet, the gradients can stream straightforwardly backwards from the layers ahead to the initial layers in reverse.

ResNet is implemented in our project to act as an attacker on the Speck Cipher to find out how secure the cipher is against feasible attacks by deep learning models. According to the research paper from Gohr (2019), Speck Ciphers were generated during the training of ResNet and during evaluation, ResNet will try to decipher and classify the Speck Ciphers before outputting the classification results. The research paper shows that ResNet is currently able to distinguish the plaintext of 8 rounds Speck Cipher. By using some few-shot learning techniques, the performance of the model may be further improved to distinguish the six rounds Speck ciphers.

The aim of this project is to perform modifications on the ResNet model to perform more effectively in distinguishing the plaintext of a Speck Cipher. One modification that we will perform is to increase the number of plaintexts used in training the ResNet model. In the research paper of Gohr (2019), two plaintexts were used, p1 and p2, which gave us one relationship: $p1 \oplus p2 = a$. In this project, we used four plaintexts instead of just two, where in this case, we have: $p2 = p1 \oplus a$, $p3 = p1 \oplus b$, $p4 = p3 \oplus a$. This gives the plaintext relationships as: $p1 \oplus p2 = a$, $p3 \oplus p4 = a$, $p1 \oplus p3 = b$ and automatically we also get this extra relationship: $p2 \oplus p4 = (p1 \oplus a) \oplus (p3 \oplus a) = p1 \oplus p3 = b$. By changing the number of plaintexts from two to four, we will obtain four relationships rather than just one, which decreases the randomness. Another modification that we will perform is changing the input parameters where we might increase the training depth, the number of layers, the number of training rounds and such to produce various ResNet models. We will then be able to analyze the differences in performance between the ResNet model and the modified versions and investigate whether changes to the ResNet models effectively improves the performance of distinguishing the plaintext of Speck Ciphers.

2.1.4 Bit Difference – Updated

In our search for a high performance model with 4 plaintexts, where it has 4 relationships, our supervisor sent us some research papers on improving the accuracies with multiple bit differences as using 2-bit differences did not improve the performance of the model.

In Fu, Wang, Guo, Sun, and Hu (2019) research paper, they attack the round-reduced Speck cipher after finding the differential path for various block sizes of Speck. They used a technique called nested-based heuristic extended with pDDT (partial difference distribution table) to find the differential path which was time-consuming and difficult. The inspiration of nested-based heuristic came from single-player games such as Sudoku, where it also has similar problems. After researching on the differential path, they concluded that the multiple bit (0x0014, 0x0800) for Speck32 was the best differential trail for 1 round of Speck.

In Dwivedi, Morawiecki and Srivastava (2016) research paper, they used a technique called Mixed Integer Linear Programming (MILP) to find improvements of the linear approximations and differential characteristics in block ciphers such as Speck. At the end of their research, they concluded that one of the best differential characteristics was (0x0211, 0x0A04).

In Song, Huang and Yang (2016) research paper, they realised that it took too long to find the long differential characteristics by using the Mouha's framework of the block ciphers. Hence, they introduced a new method by finding the long differential characteristics with two short ones. As a result, one of the best long differential characteristics was using these two short ones, (0x2040, 0x0040).

In Biryukov, Roy, and Velichkov (2015) research paper, they used automatic search for trails and differentials to find the best differential trail for the block cipher. The algorithm will find the best probability for i + 1 rounds given that the best trail on i rounds was provided. One of the best differential trails provided was (0x8054, 0xA900).

We made use of all these multiple bit differences in our model training. However, none of the model performances improve as compared to the models that used 1-bit differences in plaintexts. Hence, we decided on using one of the 1-bit differences that was proposed by Gohr, which was (0x0040), as it provided the highest accuracy in Gohr's training result and from our model training results, it was also proven that models using this bit difference provided one of the best performances.

2.1.4 SHAP – Updated

There are plenty of explainable AI (XAI) tools which were able to help us visualize the prediction results of our model. After researching various explainable AI tools, we decided to use SHAP (SHapley Additive exPlanations) to visualize our results. The reason for this is that SHAP allows us to analyse the prediction of a machine learning model by the Shapley values. SHAP uses the concepts of Local Explanations and Cooperative Game Theory, to get the Shapley values (Rane, 2019). Shapley values explain the contribution of each feature on the differences of the actual model and the model prediction. If the feature contributes to the final prediction of the model, the Shapley value will be calculated by getting the difference of the feature and the subset without the feature (Lakshmi, 2017). Features with higher Shapley values are more important to the model. Positive Shapley values means the feature has contributed to the prediction of 1 while negative Shapley values means the feature has contributed to the prediction of 0 (Lakshmi, 2017). The results of visualization consists of 2 colours, where red indicates the positive Shapley value and blue indicates the negative Shapley values. Lakshmi (2017) states that the advantage of SHAP compared to LIME, where we planned to use in the previous semester, is that SHAP has better local accuracy and consistency as it examines all combinations from the input dataset. The disadvantage of SHAP is that SHAP takes a long period of time to run because the explainers need to calculate the Shapley values before the visualization can be completed.

2.2 Summary

After spending time on the numerous research papers on the above topics, we came to a conclusion that the performance of the ResNet model could be improved by modifying the number of plaintexts, modifying the number of relationships between plaintexts and also modifying the input parameters. This is because modifying the number of plaintexts to 4 using pairs of bit differences will enable the model to learn better as the plaintexts have an increased number of relationships. We needed to train models using the various bit differences to obtain the best bit differences so that we can achieve a refined model. Modifying the input parameters will provide additional information or techniques for the model to learn about the training data and the models will then be able to better distinguish the Speck Ciphers. Through this project, we wanted allow users to visualize the evaluation results of the models and produce a thorough analysis regarding the performance of the models. Hence, SHAP explainers will be incorporated alongside the models to develop visualizations regarding the classification results.

3. Outcomes

3.1 Implementation

<u>Speck</u>

A Speck class which contains the functions to perform the Speck algorithm, converting pairs of ciphertexts into binary arrays and generating ciphertexts as input for the ResNet model. The Speck algorithm was also programmed to take in 4 plaintexts as the input which includes adding another bit difference to the algorithm and increase the number of relationships between the plaintexts to 4.

<u>ResNet</u>

An improved version of Gohr's ResNet model where we modified the model and its layers to take in an increased number of ciphertexts as the input. The input parameters were also modified to receive an increased number of ciphertexts so that the layers will change according to the input parameters. The model will then be saved into a folder after training is completed so that they can be evaluated next.

<u>SHAP</u>

SHAP explainers were incorporated into the ResNet model evaluation where we were able to visualize the classification results predicted by the model. These explainers were to used to understand the model and calculate the shapley values for each bit position before outputting the chart showing which bit position affected the classification results for the testing data.

3.2 Results Achieved

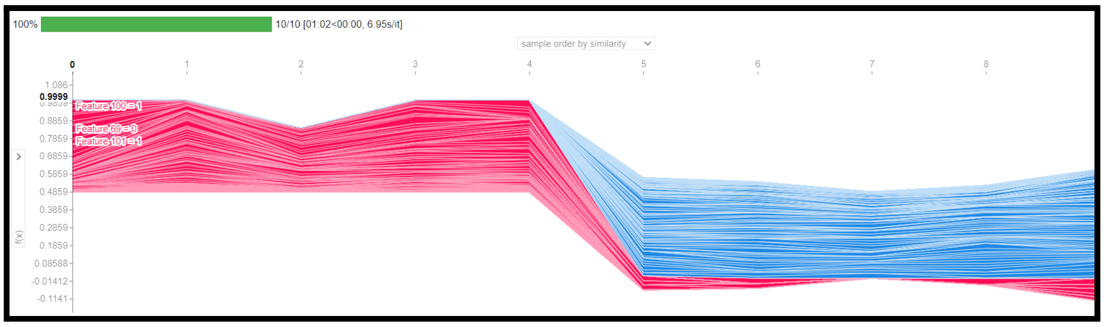
The goals set for our project CryptoML prior to the start of this project was to introduce modifications to the ResNet model created by Gohr so that we could carry out improved attacks on the Speck ciphers and incorporate explainable AI tools to produce explainable visualizations for the classification results.

Modifications such as increasing the number of plaintexts from two to four was complete as we managed to generate the input ciphertexts from four plaintexts using two bit differences instead of one and this lead to an increase in the number of relationships between the plaintexts to four. The ResNet model was also modified to take in the new input data so that it can be trained properly and learn well from the new set of data.

From the comparison table below, we can conclude that CryptoML's result for random setting has improved as a result of the increased number of relationships between the plaintexts led to the model learning better. Hence, the model could better classify the ciphertexts.

Number of Speck Rounds	Gohr's Result	CryptoML's Result
5	0.9268	0.9855
6	0.7879	0.8767
7	0.6161	0.6368
8	0.5127	0.5203

The SHAP explainers were also successfully incorporated into our model evaluation as we managed to output the classification results with its respective visualization charts to help us explain the results to users. The visualization consists of the classification results of the testing data according to the bit positions that greatly affected the results calculated by the SHAP explainers using shapley values. Figure 1 shows an example of the visualization.



```
100% 10/10 [01:02<00:00, 6.95s/it]
```

Figure 1: SHAP Visualization

Although there were changes and updates made towards the project plan, we concluded that our project CryptoML was a success as we managed to achieve all our targets before the due date and our results fared better than Gohr's results. Visualizations that we planned to include in our results were also complete and this allows users to interact with the results as well as have a better understanding of the classification results.

3.3 Project Requirements

The following tables displays whether or not the updated functional and non-functional requirements of CryptoML were met and how the requirements were satisfied.

Functional Requirement	Remark
Implementation of Speck Model	Satisfied. The Speck algorithm is able to transform the plaintexts into ciphertexts which will be used as input data for both training and evaluation.
Implementation of ResNet Model	Satisfied. The ResNet model is able to learn from the ciphertexts and classify the testing data into their respective classes. The models are also saved to be used as comparisons.
Modification of Input Data and Parameters	Satisfied. The input data can be generated according to the number of plaintexts selected and the ResNet model parameters can be tuned according to the number of plaintexts.
Implementation of Visualizations	Satisfied. SHAP visualizations will be generated alongside the classification results during evaluation to provide interactivity and better understanding of the results.

Non-Functional Requirement	Remark
Improved Model Performance	Satisfied. The modified ResNet model provides better classification results compared to Gohr's model and this can be found in the previous sections.
High Quality Visualizations	Satisfied. The generated SHAP visualizations provided are of high quality and hovering over the test samples will display the bit positions that greatly affect the classification results.
Simple and User-Friendly User Interface	Satisfied. Google Colaboratory allows Jupyter Notebooks to be executed online without the need of any installations. The user interface is simple and users can easily run our program.
Scalable Software	Not Satisfied. Google Colaboratory has usage limitations where the runtime is limited to 12 hours per session which means training for high number of epochs are off limits.

3.4 Justification of Decisions

A number of vital decisions were made throughout the implementation of our project CryptoML. First and foremost, we needed to select the best bit differences for our model as we needed a pair of bit differences compared to Gohr’s model which only needed one. The way we selected the bit differences was by first training and evaluating models with single bit difference to select the bit differences that produced the best model performance before creating different combinations of the bit differences. The pairs of different combinations were used to train and evaluate new models which we could then use to compare among the models to attain the best pair of bit differences which resulted in ((0x0040,0), (0x0020,0)).

Another major decision that we concluded was to utilize the SHAP package for our explainable AI tool instead of the LIME package. During the previous semester, we decided on using the LIME package but during the implementation of the LIME explainers we realized that it relies on the use of features and did not suit our project as we were dealing with binary numbers and its positions. After researching on other explainable AI tools, we decided to use SHAP as an alternative since it does not require the need for features. We were able to use the SHAP explainers to create the visualizations and allow users to have a better understanding of the classification results.

Lastly, our team decided to use Juypter Notebook or ipynb as our main file format. This is because every user will be able to make use of Google Colaboratory to run the program with the free GPU that it provides and it can be easily opened when the files are uploaded into Google Drive. Besides, Jupyter Notebook also allows output to be recorded right below the cells and the results could be saved along with the code. The SHAP visualizations can also be generated neatly in Jupyter Notebook alongside the results without the need of other libraries such as matplotlib.

3.5 Discussion of Results

All in all, the team members agreed to deem the project a successful one as we have managed to fulfill all of our planned objectives. From the tables in 3.3, we can see that most of the requirements have been satisfied except for one which is scalability. This is because we needed to choose between scalability and useability which we settled for the latter as we wanted every user to be able to run our program on their machines.

By using Google Colaboratory, anyone can execute our program with good performance as the application allows users to make us of their online GPUs. The downside of this choice is that there are usage limitations where users could only use up to 12 hours per session and this leads to the scalability issue as we cannot perform trainings that might take longer than that. If we were to prioritise scalability, users would need to have powerful GPUs on their machines to ensure that our program performs smoothly and not many would have the luxury of owning powerful GPUs.

Apart from this, we managed to complete all the other requirements and achieve our targets which were to improve the ResNet model which will perform better than Gohr's model and produce visualizations for the classification results using explainable AI tools. The program was also fully tested on and we have a well-documented user guide which will help users to thoroughly understand how to execute our program.

3.6 Project Outcome Limitations

As we conducted various tests on our program, we came to realize that there were some limitations to our project. First and foremost, CryptoML only supports 2 or 4 number of plaintexts as the input data for our ResNet model. This is because we need to maintain the number of relationships between the plaintexts where there is 1 relationship if there are 2 plaintexts whereas there are 4 relationships if there are 4 plaintexts.

Another limitation that we realized is that our program only works well for up to 8 rounds of Speck which was similar to Gohr's theory. The reason behind this is the ResNet model is not capable of classifying the ciphertexts which were generated after 8 rounds of Speck. This can be proven from our evaluation results where we can see that the accuracy of the prediction for 8 rounds of Speck was around 52% and if we increased the number of rounds to 9, the results might fall around 50% or lesser. This is similar to randomly classifying the ciphertexts as the probability of a binary classification is 0.5.

Furthermore, the number of samples that can be used for the SHAP visualizations are limited to around 200 because the time for the SHAP explainers to calculate the shapley values for each sample increases exponentially and for 200 samples, it took us around 10 hours to display the visualization. If users input more than 200 samples, it might exceed the usage limitation of Google Colaboratory and lead to the program crashing.

3.7 Possible Improvements

In regards to the limitations mentioned above, one improvement that could be made is to modify the data generation method so that it can take in any number of plaintexts which will then automatically alter the relationship between the plaintexts. With this said, there are a lot of things to consider such as how the relationship between plaintexts should change when the number of plaintext changes because right now we have fixed relationships between the plaintexts.

A possible improvement to tackle the limitation where our program can only work for up to 8 rounds of Speck is to enhance the ResNet model by improving the layers in the model and modifying the parameters of the ResNet model. Another improvement would be to modify the relationship between the plaintexts so that the randomness between the pair of plaintexts would decrease even more to help the model learn better and be able to easily classify the ciphertexts after 9 rounds of Speck and above.

Feasible future works include creating a Virtual Machine instance powered with GPUs on Cloud platforms to execute our program. This will allow users to perform the model trainings and model evaluations within a very short period of time and remove any worries of usage limitations which we currently face. Another add-on that we could include is to develop a front-end interface which will give users a better and much simpler experience when running our program.

3.8 Relevant Matters

Regarding the aforementioned improvements, these are some of the plans that we outlined to scale up our program in the future. Improving the data generation method will incur problems regarding the relationship between the plaintexts and this is something we will need to carefully probe on before we can perform any changes. Additional research will also need to be done to further improve the ResNet model or modify the deep learning approach to help it learn better about the Speck algorithm. The idea to create a virtual machine instance powered with GPUs might cost a fortune in our situation and this can only be possibly achieved if we are able to pitch our project and there are investors or stakeholders who might be interested to fund our project.

4. Methodology

4.1 Overall Architecture

The system architecture of this project is shown in Figure 2. We will first generate plaintexts and encrypt them into ciphertexts for training, validation and testing purposes. Then, we will train our ResNet model which is built using Tensorflow and Keras libraries by feeding it with the generated data. After the ResNet model is trained, we will evaluate the performance of the model and use the explainable AI package, SHAP (SHapley Additive exPlanations) to explain the results using visualisation. In the initial design proposed in FIT3163, we planned to use LIME (Local Interpretable Model-Agnostic Explanations) as the explainable AI tool to visualise our results but after trying to implement it in this semester, we realised that SHAP is less complex and more suitable for the explanations of our results. We also removed the implementation of the web application which was initially in our project proposal due to some factors that will be mentioned later during the critical discussion. A detailed explanation of each component will be discussed in the next section.

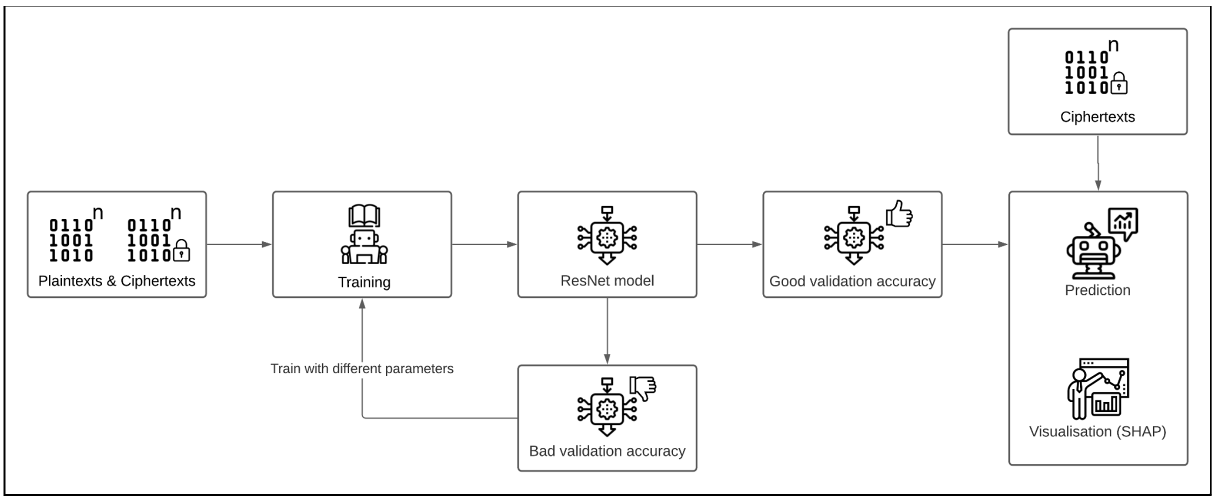


Figure 2: Overall Architecture

4.2 Generating Data

A dataset with 10^7 plaintexts will be generated as training data while a dataset with 10^6 plaintexts will be generated as testing data. Data with random relationship is labelled as 0 whereas data with real difference relationship is labelled as 1. The relationship of the plaintexts will be decided randomly in the function. Figure 3 is a high-level pseudocode of the algorithm to generate data.

```
Algorithm: Generating Data

Inputs: num_of_plaintexts, diffa, diffb, num_rounds
Outputs: A binary array of ciphertexts and label

Relationship ← generate 0 (random) or 1 (real difference) randomly

if num_of_plaintexts is 2 then
        set p0 by random
        if random relationship then
                set p1 by random
                set label to 0
        else
                p1 = p0 XOR diffa
                set label to 1
        end
        ciphertexts = encrypt(p0, p1) by n rounds (num_rounds)
        convert ciphertexts to binary array
else
        set p0 by random
        if random relationship then
                set p1 by random
                set p2 by random
                set p3 by random
                set label to 0
        else
                p1 = p0 XOR diffa
                p2 = p0 XOR diffa
                p3 = p2 XOR diffb
                set label to 1
        end
        ciphertexts = encrypt(p0, p1, p2, p3) by n rounds (num_rounds)
        convert ciphertexts to binary array
end
```

Figure 3: Algorithm for generating data

4.2.1 Random Data

If two ciphertexts are used as input to ResNet, we only need to generate plaintexts: p0 and p1. Both of them are generated randomly. Then p0 and p1 will undergo encryption for n rounds, where n is the number of rounds of Speck we initiated in the parameter. There will also be n keys so that every round of encryption will use a different key. The final step is to convert the ciphertexts into a binary array.

On the other hand, if four ciphertexts are used, the process of data generation will be the same as the above. The only thing that is different is that we will need to randomly generate p0, p1, p2 and p3, four plaintexts rather than two plaintexts. These four plaintexts will then be encrypted into ciphertexts to be converted into a binary array.

4.2.2 Real Difference Data

If we are only using two ciphertexts as the input to ResNet, we will need two plaintexts. First, a plaintext (p0) will be generated randomly. Another plaintext (p1) will be generated where the relationship between p0 and p1 is: $p0 \oplus p1 = a$, where a is the real difference we stated in the parameter. Therefore, to generate a p1 which has real difference relationship with p0, $p1 = p0 \oplus a$. After obtaining p0 and p1, they will undergo encryption for n rounds, where n is the number of rounds of Speck we initiated in the parameter. There will also be n keys so that every round of encryption will use a different key. The final step is to convert the ciphertexts into a binary array.

If we are using four ciphertexts as the input to ResNet, we need to generate four plaintexts: p0, p1, p2 and p3. First of all, p0 is generated randomly. Unlike two plaintexts with only a relationship between p0 and p1, four plaintexts will need two real differences which are a and b to ensure there are real differences between these four plaintexts. The relationships between four plaintexts are: $p1 = p0 \oplus a$, $p2 = p0 \oplus b$, $p3 = p2 \oplus a$. After obtaining p0, p1, p2 and p3 using the relationships mentioned above, these four plaintexts will be encrypted into four ciphertexts for n rounds, where n is the number of rounds of Speck we initiated in the parameter. There will also be n keys so that every round of encryption will use a different key. The final step is to convert the ciphertexts into a binary array.

4.3 Implementation of ResNet

After generating the data, we will need to find the optimal model to solve the binary classification problem by differentiating if the ciphertexts have random relationships or real difference relationships. The libraries we used are Tensorflow and Keras. Now, we will need to find the optimal values for each parameter of the function to build and train ResNet. For example, these are the parameters we can modify before training the ResNet model: *num_epochs*, *num_rounds*, *depth*, *num_blocks*, *diffa* and *diffb*. To determine the performance of the model, we will need to look into the best validation accuracy of the trained model. The best performing models that we obtain for 5, 6, 7 and 8-rounds of Speck will be saved for future evaluation. The high-level pseudocode for the algorithm to build the ResNet model is shown in Figure 4 whereas the architecture of the ResNet model is shown in Figure 5. The reason we are using ResNet as our model is because it has the concept of skip connection which mitigates the gradient vanishing problem we might face. In the ResNet model, L2 regularizer and batch normalization is applied to reduce the overfitting problem. We have ReLU as the activation functions and Sigmoid as the final activation function since we are dealing with a binary classification problem here.

```
Algorithm: Implement ResNet model

Inputs: num_epochs, num_rounds, depth, num_blocks and diffa and diffb
Outputs: A trained ResNet model

Add input and pre-processing layers
Reshape the shape of layers to (num_blocks × word_size=16 × 2,)
Add Conv1D layer
Add BatchNormalisation layer
Add 'relu' as activation function layer

for i := 0 to depth do
    Add Conv1D layer
    Add BatchNormalisation layer
    Add 'relu' as activation function layer
    Add Conv1D layer
    Add BatchNormalisation layer
    Add 'relu' as activation function layer
    Add shortcut
end for

Add Flatten layer
Add Dense layer
Add BatchNormalisation layer
Add 'relu' as activation function layer
Add Dense layer
Add BatchNormalisation layer
Add 'relu' as activation function layer
Add Dense layer

Compile model using 'adam' as optimizer

Train model of num_rounds of Speck with num_epochs of epochs using dataset with real
differences of diffa and/or diffb

If best validation accuracy better than previous model, save model
```

Figure 4: Algorithm for implementing a ResNet model

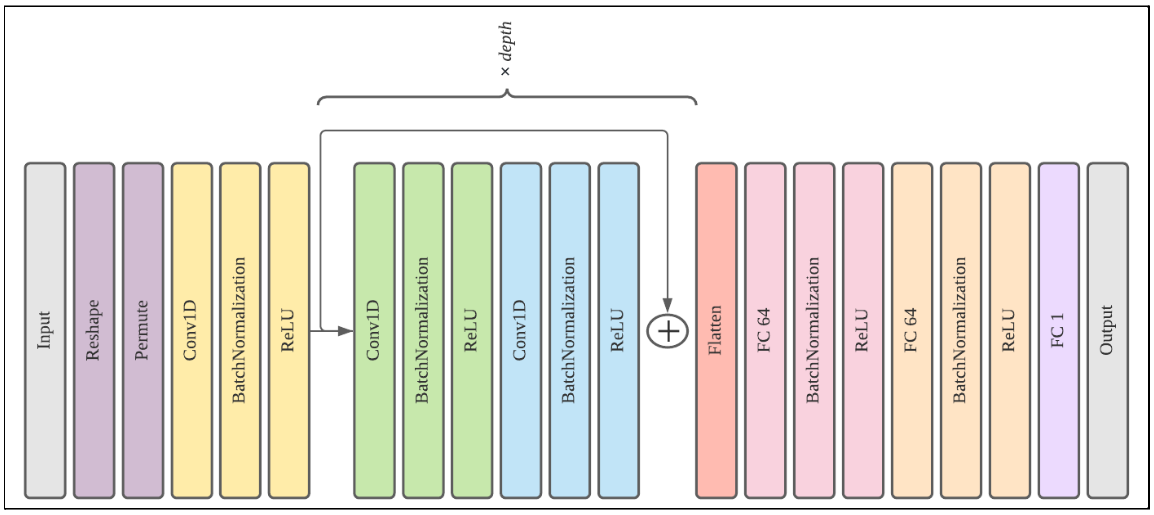


Figure 5: Simple Architecture of the model

4.4 Implementation of SHAP

After the model has been trained, the next step is to evaluate the model using the testing data and visualize it using SHAP (SHapley Additive exPlanations). To use this explainable AI library, we need to first install the SHAP package. After that, by inputting the model and testing data to the evaluation algorithm, the accuracy, true positive rate, true negative rate and mean squared error of the model will be reported. Then, using the testing data and the model as inputs, we create visualizations of the results given the number of samples. The evaluation algorithm and implementation of SHAP is shown in Figure 6.

```
Algorithm: Implement SHAP

Inputs: h5file, test_data, num_of_samples
Outputs: Accuracy, True Positive Rate, True Negative Rate, Mean Square Error, Visualisation

model ← load model from .h5 file
model.predict(test_data)
tp ← true positive
tn ← true negative
fp ← false positive
fn ← false negative
acc ← (tp+tn)/(tp+tn+fp+fn)
tpr ← tp/(tp+fn)
tnr ← tn/(tn+fp)
mse ← sum of squared differences of predicted and actual values

shap.initjs()
explainer = shap.KernelExplainer(model.predict, test_data[:num_of_samples])
shap_values = explainer.shap_values(test_data[:num_of_samples])
shap.force_plot(explainer.expected_value[0], shap_values[0], test_data)
```

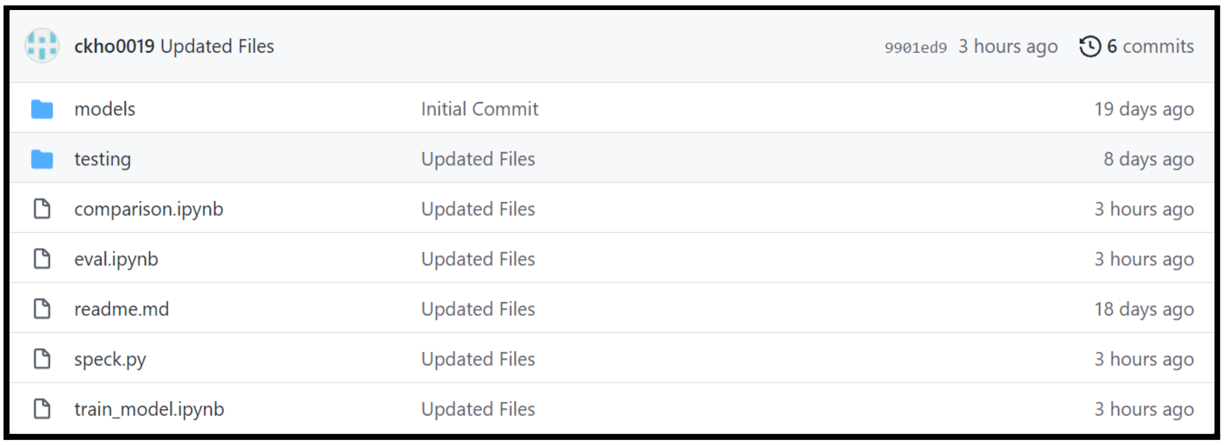
Figure 6: Evaluation & Implementation of SHAP

5. Software Deliverables

5.1 Summary of software deliverable

5.1.1 What is delivered

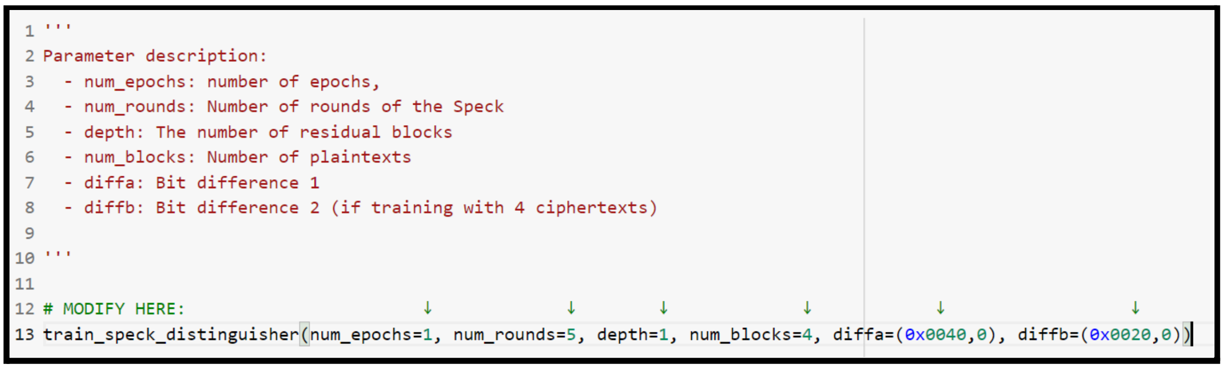
The source code for our system is stored in a Github repository. The full source code can be found at https://github.com/ckho0019/CryptoML. In the Github repository, there is a Speck file containing the Speck Algorithm with its functions and 3 jupyter notebooks that will allow users to train models, evaluate model predictions with SHAP visualizations and view the comparison results between Gohr's model and our model. Besides, the models folder stores all the trained models whereas the testing folder stores all testing files. The trained model with 2 plaintexts starts with filename "net" while the trained model with 4 plaintexts starts with "best". The models inside models folder are the models trained by our team that have the highest accuracy for each round of Speck.



ckho0019 Updated Files	9901ed9 3 hours ago	6 commits
models	Initial Commit	19 days ago
testing	Updated Files	8 days ago
comparison.ipynb	Updated Files	3 hours ago
eval.ipynb	Updated Files	3 hours ago
readme.md	Updated Files	18 days ago
speck.py	Updated Files	3 hours ago
train_model.ipynb	Updated Files	3 hours ago

Figure 7: Github Repository

5.1.2 Sample screenshots and description of usage



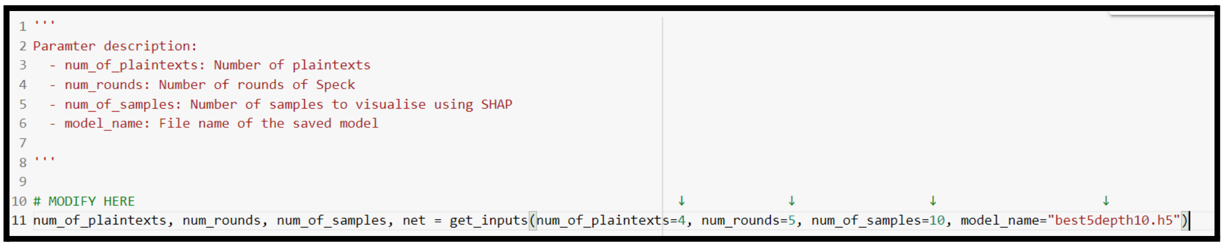
```
'''
Parameter description:
  - num_epochs: number of epochs,
  - num_rounds: Number of rounds of the Speck
  - depth: The number of residual blocks
  - num_blocks: Number of plaintexts
  - diffa: Bit difference 1
  - diffb: Bit difference 2 (if training with 4 ciphertexts)

'''

# MODIFY HERE:                   ↓              ↓         ↓              ↓             ↓                   ↓
train_speck_distinguisher(num_epochs=1, num_rounds=5, depth=1, num_blocks=4, diffa=(0x0040,0), diffb=(0x0020,0))
```

Figure 8: train_speck_distinguisher in train_model.ipynb

In the **train_model.ipynb** file, users can modify the number of epochs, number of rounds, depth, number of plaintexts and the bit differences of the plaintexts. Running this cell will train the model with the respective parameters. Then, the trained model will be saved into the models folder.



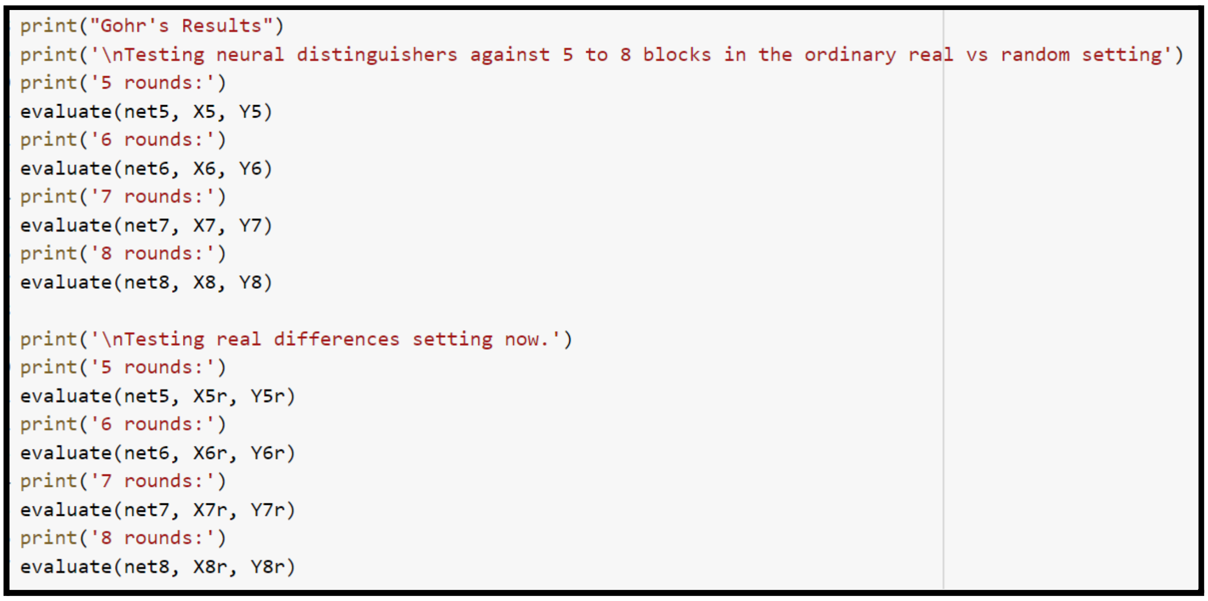
```
'''
Paramter description:
  - num_of_plaintexts: Number of plaintexts
  - num_rounds: Number of rounds of Speck
  - num_of_samples: Number of samples to visualise using SHAP
  - model_name: File name of the saved model

'''

# MODIFY HERE
num_of_plaintexts, num_rounds, num_of_samples, net = get_inputs(num_of_plaintexts=4, num_rounds=5, num_of_samples=10, model_name="best5depth10.h5")
```

Figure 9: get_inputs function in eval.ipynb

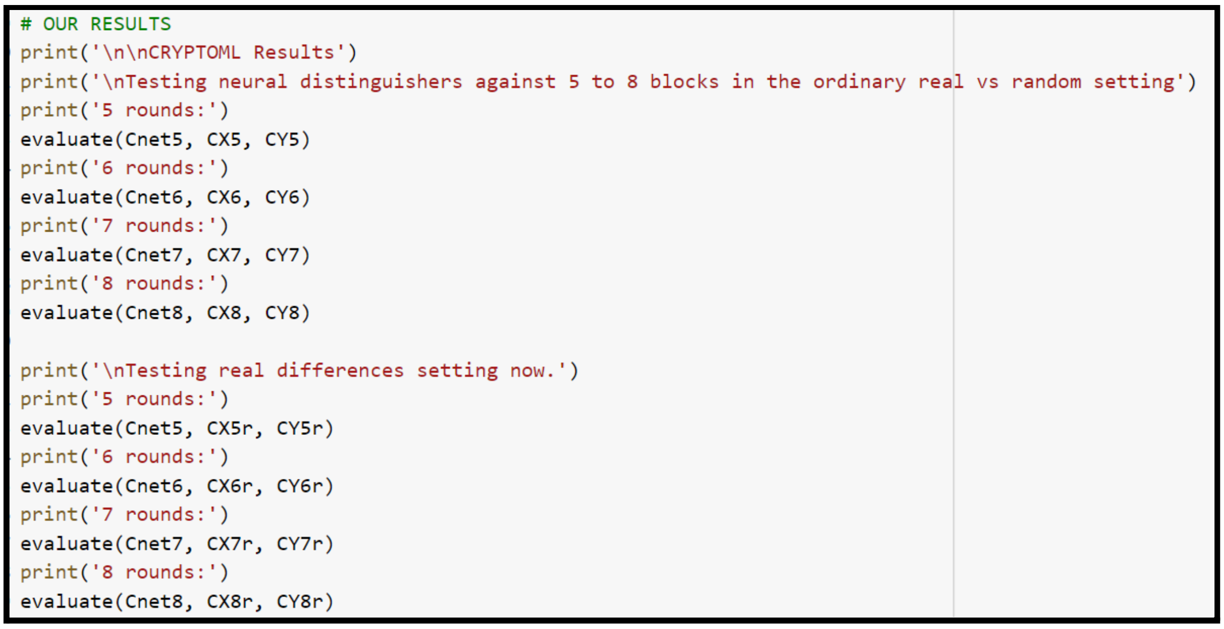
In the **eval.ipynb** file, users can modify the number of plaintexts, number of Speck rounds, number of samples to be visualized and the file name of the trained model so that new testing data is created to evaluate the classification results of the model.



```
print("Gohr's Results")
print('\nTesting neural distinguishers against 5 to 8 blocks in the ordinary real vs random setting')
print('5 rounds:')
evaluate(net5, X5, Y5)
print('6 rounds:')
evaluate(net6, X6, Y6)
print('7 rounds:')
evaluate(net7, X7, Y7)
print('8 rounds:')
evaluate(net8, X8, Y8)

print('\nTesting real differences setting now.')
print('5 rounds:')
evaluate(net5, X5r, Y5r)
print('6 rounds:')
evaluate(net6, X6r, Y6r)
print('7 rounds:')
evaluate(net7, X7r, Y7r)
print('8 rounds:')
evaluate(net8, X8r, Y8r)
```

Figure 10.1: comparison.ipynb



```
# OUR RESULTS
print('\n\nCRYPTOML Results')
print('\nTesting neural distinguishers against 5 to 8 blocks in the ordinary real vs random setting')
print('5 rounds:')
evaluate(Cnet5, CX5, CY5)
print('6 rounds:')
evaluate(Cnet6, CX6, CY6)
print('7 rounds:')
evaluate(Cnet7, CX7, CY7)
print('8 rounds:')
evaluate(Cnet8, CX8, CY8)

print('\nTesting real differences setting now.')
print('5 rounds:')
evaluate(Cnet5, CX5r, CY5r)
print('6 rounds:')
evaluate(Cnet6, CX6r, CY6r)
print('7 rounds:')
evaluate(Cnet7, CX7r, CY7r)
print('8 rounds:')
evaluate(Cnet8, CX8r, CY8r)
```

Figure 10.2: comparison.ipynb

Running the above cells will then output the classification results such as the accuracy, true positive rate and mean squared error as well as the SHAP visualizations.

5.2 Software Qualities

5.2.1 Robustness

Robustness refers to the ability of a computer system to deal with errors and wrong inputs during execution. For CryptoML, we added some error checking for the user inputs. For example, the num_blocks can only be 2 or 4, as our current model only allows 2 or 4 plaintexts. Besides, the number of epochs, number of rounds and depth cannot be less than 1. This is to make sure users do not input invalid values to our model. We also checked on the file input when loading the model so that users do not upload different file types into the program. Unit testing is done to test on every line of our code so that the risk of breaking our code is minimized.

5.2.2 Security

The definition of security is using secured coding techniques to prevent any sort of software vulnerabilities that could be capitalised by hackers and cause loss of data privacy. For CryptoML, security does not relate too much as our project does not deal with data or information about users. The only measure we put in place to prevent any sort of exploitation is input handling. For instance, we limit the number of plaintexts to only 2 and 4 as currently we only allow these two numbers, prevent users from entering negative values and lastly, checking file input for loading the models to prevent users from uploading different file types into the program and breaking the code. With these measures set, we make sure that extra layers of security are there to protect our program.

5.2.3 Useability

Usability refers to how usable the software is in relation to its intended purpose. We have compiled all the codes into a few jupyter notebooks and Google Colab acts as our user interface. Users have to run each cell of the code to train, evaluate and visualize the model. There are two ipynb file users can modify and each file has comments to guide users on which part of the code can be modified. We have designed and properly documented the jupyter notebooks to make them user friendly so that users without any programming background might be able to look out for the parts to modify and execute the program without any trouble.

5.2.4 Scalability

Scalability is defined by the capability of the project's ability to adapt to the changes made and not lose its performance. For CryptoML, there are opportunities for improvement in terms of scalability as our program now runs on Google Colaboratory, which there are usage limitations per session and this might deter users from training models with large number of epochs or creating visualizations with huge number of samples as these might exceed the usage limits of Google Colaboratory. Some of our future plans for scalability include buying our own GPUs to host the program so limitations that we are facing now won't be an issue when we manage to host CryptoML on our own machines.

5.2.5 Documentation and Maintainability

In all our notebooks, there are comments to explain on each block of code so that users can easily understand how our program works. Besides, there are comments to guide users on which part of the code can be modified so that users can train and evaluate the model as they wish. In the README file on Github, there is also guidance on how to use each file. These comments also help us to ensure the maintainability of our source code as in the event an error or bug occurs, we can quickly identify which part of the code went wrong.

5.3 Sample Source Code

Screenshots of sample source code is shown in the Appendix.

6. Critical Discussion

6.1 How well the project was executed

Ultimately, CryptoML was deemed a successful project as we managed to complete the project within the scope, time and cost. Most of the project executions went according to what we have planned from the previous semester and although there were unforeseen circumstances such as the ongoing pandemic and various limitations that deterred some of our plans, we were still able to make use of what we have and accomplish all our goals and objectives.

6.2 How and why we deviated

In our opinion, we deviated slightly from the initial project proposal produced in FIT3163. First and foremost, the explainable AI tool we planned to use was LIME (Local Interpretable Model-Agnostic Explanations) but we ended up with SHAP (SHapley Additive exPlanations) as during the execution of the project, we realised that the implementation of LIME relies on the use of features and did not suit our project as we were dealing with binary numbers and its positions. After researching other explainable AI tools, we decided to use SHAP as an alternative. For SHAP, we just needed to input the model and testing data for the explainer to perform. Since our testing data is quite large and the explainer takes time to calculate the shapley values, we will only choose a portion of the testing data as input. Hence, we still managed to implement the visualisation using explainable AI which is one of the project requirements.

Besides that, we decided not to implement a web application during the execution of this project. In the project proposal, the web application is designed for the users to train a model and then visualise it using explainable AI. However, during the implementation of this project, we became aware of the training time where it took us 7 hours to train a model with 200 epochs using online GPUs. If the user does not have a GPU on their machine, it might take days for the user to complete the training. Another reason why we decided not to implement the web application is that the modifications of the parameters do not require complex steps. The intention of implementing a web application was to provide a simple user interface where users can easily understand. Now that our current design using the Jupyter Notebook fulfilled this intention, there was no need to implement a web application anymore. Besides the above mentioned deviations, all functionalities and methodologies remained the same from what we have proposed in our project proposal from the previous semester.

In terms of project management, we adhered to the proposed project management plan with some minor changes. Since we were unable to have face-to-face meetings due to the pandemic, our communication channel was substituted with virtual meetings using Zoom and daily updates using WhatsApp. In addition, we decided to replace the main tasks of Sprint 4 with completing the project reports since we concluded that the implementation of the web application was unnecessary. This is because we did not expect the large amount of reports to be submitted during the planning of this project.

6.3 How our thinking evolved during the project execution

During the start of the project execution, we followed the plan according to the project proposal from the previous semester. At this stage, we only had a vague idea about the goals and objectives for this project. It was only after we started implementing this project under the supervision of Prof. Raphael that our understanding of the project gradually increased and we were able to have discussions with him rather than just blindly following his instructions. Initially, we will ask our supervisor for his opinion on how to overcome the obstacles we faced but as time went by, we looked to other team members for help first where we tried our best to solve the problems by going through online materials and then exchange our views. For example, the parts which were deviated from our initial proposal mentioned above were modified after a long discussion between the project members. After the decision to modify the initial proposal was confirmed, we informed our supervisor during a meeting and explained to him why such a change was needed and he gave us the green light for the modification. From there on, we started relying on our own members instead of looking for our supervisor whenever we hit a roadblock.

6.4 Limitations

One of the major limitations we encountered while completing CryptoML was the usage limit of online GPUs such as Google Colaboratory and Kaggle. We decided to use online GPUs as our machines were not powerful enough to train the models as it took us about an hour to complete one epoch compared to three minutes for one epoch on the online GPUs. The concern we faced while running our program on these platforms was that it has usage limits where Google Colab allows up to 12 hours per session while Kaggle allows up to 9 hours per session. Due to these usage limits, we were not able to train models with a high number of epochs or create SHAP visualizations with large number of samples as the time taken for SHAP explainers to calculate the shapley values increased exponentially. The session will be interrupted as soon as it exceeds the usage limit. Hence, scalability is currently not viable and we will be looking into other ways to execute our program as part of our future plans.

7. Conclusion

All things considered, CryptoML has been successfully developed to generate better attacks on the round-reduced Speck Cipher. Every objective and goal that we set has been accomplished and most importantly our model managed to achieve higher classification accuracies compared to Gohr's model as we managed to prove our hypothesis that modifying the number of plaintexts and the number of relationship between the plaintexts will increase the performance of the ResNet model. Most of the functional and non-functional requirements that were planned throughout the previous semester were satisfied except one which was to produce a scalable product. This was because we chose to prioritize useability where we believe everyone who knows about CryptoML should be able to have the chance to give our program a try. Hence, we decided to choose Google Colaboratory as our user interface despite the usage limitations as everyone will be able to access it and the results together with the visualizations will be displayed alongside our program. Our plans for the future includes creating VM instances powered with GPUs to host our program and also developing a web interface to provide a much simpler and user friendly user experience.

The successful implementation of SHAP explainers which we used to create visualizations for our classification results indicates a major breakthrough in Cryptanalysis as this was the first time visualizations were introduced to explain the evaluation results of the ResNet model on the study of Cryptanalysis. With this visualizations in place, we allow users to interact with our results as users will be able to choose different sample orderings and view how the various bit positions affect the prediction of each sample. There are definitely room for improvements regarding the visualizations to explain the evaluation results with more clarity and detailed explanations.

As a result of completing this project, we got to better understand why NSA decided that a minimum of 22 rounds of Speck was the standard of the Speck32/64 lightweight ciphers. It is the minimal number of rounds needed for Speck to be strongly secured against all sorts of attacks and the results have proven that so far none of the deep learning methods have been successful in attacking the Speck Cipher. Most Cryptanalysis studies such as CryptoML have only been able to attack the round-reduced Speck Cipher and we trust that NSA will continue to strengthen the security of these lightweight ciphers to ensure that IoT devices are firmly fortified from any sort of cyberattacks.

8. Appendix

```
def make_resnet(num_blocks=2, num_filters=32, num_outputs=1, d1=64, d2=64,
      word_size=16, ks=3, depth=5, reg_param=0.0001, final_activation='sigmoid'):
  '''
  This function makes residual tower of convolutional blocks.

  Inputs:
    - num_blocks: Number of plaintexts
    - num_filters: Number of filters in the Conv1D layers
    - num_outputs: Number of outputs for the final prediction (0 or 1 in this case)
    - d1: Units in the first Dense layer
    - d2: Units in the second Dense layer
    - word_size: Word size which will affect the shape of Input layer
    - ks: Kernel size for the Conv1D layers in the residual blocks
    - depth: The number of residual blocks
    - reg_param: Parameter for regularizers
    - final_activation: The activation function in the final Dense layer

  Output:
  A Keras model
  '''
  assert num_outputs == 1
  assert num_blocks in [2,4]

  # Input and preprocessing layers
  inp = Input(shape=(num_blocks * word_size * 2))
  rs = Reshape((2 * num_blocks, word_size))(inp)
  perm = Permute((2,1))(rs)

  # Add a single residual layer that will expand the data to num_filters channels
  # This is a bit-sliced layer
  conv0 = Conv1D(num_filters, kernel_size=1, padding='same',
             kernel_regularizer=l2(reg_param))(perm)
  conv0 = BatchNormalization()(conv0)
  conv0 = Activation('relu')(conv0)

  # Add residual blocks
  shortcut = conv0
  for _ in range(depth):
    conv1 = Conv1D(num_filters, kernel_size=ks, padding='same',
             kernel_regularizer=l2(reg_param))(shortcut)
    conv1 = BatchNormalization()(conv1)
    conv1 = Activation('relu')(conv1)
    conv2 = Conv1D(num_filters, kernel_size=ks,
             padding='same',kernel_regularizer=l2(reg_param))(conv1)
    conv2 = BatchNormalization()(conv2)
    conv2 = Activation('relu')(conv2)
    shortcut = Add()([shortcut, conv2])
  # Add prediction head
  flat1 = Flatten()(shortcut)
  dense1 = Dense(d1,kernel_regularizer=l2(reg_param))(flat1)
  dense1 = BatchNormalization()(dense1)
  dense1 = Activation('relu')(dense1)
  dense2 = Dense(d2, kernel_regularizer=l2(reg_param))(dense1)
  dense2 = BatchNormalization()(dense2)
  dense2 = Activation('relu')(dense2)
  out = Dense(num_outputs, activation=final_activation,
              kernel_regularizer=l2(reg_param))(dense2)
  model = Model(inputs=inp, outputs=out)

  return model

def train_speck_distinguisher(num_epochs, num_rounds=7, depth=1, num_blocks=2,
      diffa=(0x0040,0), diffb=(0x0020,0)):
  '''
  This function trains the model and saves it to the models folder.

  Inputs:
    - num_epochs: Number of epochs you want to train
    - num_rounds: Number of rounds of the Speck
    - depth: The number of residual blocks
    - num_blocks: Number of plaintexts
    - diffa: Bit difference 1
    - diffb: Bit difference 2 (if training with 4 ciphertexts)

  Outputs:
    - net: Network
    - h: The trained model
  '''
  assert num_epochs > 0
  assert num_rounds > 0
  assert depth > 0
  assert num_blocks in [2, 4]
  assert type(diffa) is tuple and type(diffb) is tuple

  # Create the network
  net = make_resnet(num_blocks=num_blocks, depth=depth, reg_param=10**-5)
  net.compile(optimizer='adam',loss='mse',metrics=['acc'])

  # Generate training and validation data
  if num_blocks == 2:
    X, Y = sp.make_train_data_2pt(10**7, num_rounds, diff=diffa)
    X_eval, Y_eval = sp.make_train_data_2pt(10**6, num_rounds, diff=diffa)

  else:
    X, Y = sp.make_train_data_4pt(10**7, num_rounds, diffa=diffa, diffb=diffb)
    X_eval, Y_eval = sp.make_train_data_4pt(10**6, num_rounds, diffa=diffa,
                          diffb=diffb)
  # Set up model checkpoint
  check = make_checkpoint(dir+'best'+str(num_rounds)+'depth'+str(depth)+'.h5')

  # Create learnrate schedule
  lr = LearningRateScheduler(cyclic_lr(10,0.002, 0.0001))

  # Train and evaluate
  h = net.fit(X,Y,epochs=num_epochs,batch_size=bs,validation_data=(X_eval, Y_eval),
       callbacks=[lr,check])
  np.save(dir+'h'+str(num_rounds)+'r_depth'+str(depth)+'.npy',
       h.history['val_acc'])
  np.save(dir+'h'+str(num_rounds)+'r_depth'+str(depth)+'.npy',
       h.history['val_loss'])
  dump(h.history,open(dir+'hist'+str(num_rounds)+'r_depth'+str(depth)+'.p','wb'))
  print("Best validation accuracy: ", np.max(h.history['val_acc']))

  return net, h



'''
Parameter description:
  - num_epochs: number of epochs,
  - num_rounds: Number of rounds of the Speck
  - depth: The number of residual blocks
  - num_blocks: Number of plaintexts
  - diffa: Bit difference 1
  - diffb: Bit difference 2 (if training with 4 ciphertexts)

'''

# MODIFY HERE:
train_speck_distinguisher(num_epochs=1, num_rounds=5, depth=1, num_blocks=4,
       diffa=(0x0040,0), diffb=(0x0020,0))
```

9. References

Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., & Wingers, L. (2015). The simon and Speck lightweight block ciphers. *Proceedings of the 52nd Annual Design Automation Conference*, 1-6. https://doi.org/10.1145/2744769.2747946

Biryukov, A., Roy, A., & Velichkov, V. (2015). Differential analysis of block ciphers simon and Speck. *Fast Software Encryption*, 546–570. https://doi.org/10.1007/978-3-662-46706-0_28

Brownlee, J. (2019). *A Gentle Introduction to Batch Normalization for Deep Neural Networks*. Retrieved from https://machinelearningmastery.com/batch-normalization-for-trainingof-deep-neural-networks/

Buchanan, W. J., Li, S., & Asif, R. (2017). Lightweight cryptography methods. *Journal of Cyber Security Technology*, *1*(3-4), 187–201. https://doi.org/10.1080/23742917.2017.1384917

Dwivedi, A. D., Morawiecki, P., & Srivastava, G. (2019). Differential cryptanalysis of round-reduced speck suitable for internet of things devices. *IEEE Access*, *7*, 16476–16486. https://doi.org/10.1109/access.2019.2894337

Frankenfield, J. (2020). *Artificial Neural Network (ANN)*. Retrieved from https://www.investopedia.com/financial-technology-and-automated-investing-4689759#:~:text=An%20artificial%20neural%20network%20(ANN)%20is%20the%20piece%20of%20a,by%20human%20or%20statistical%20standards

Fu, K., Wang, M., Guo, Y., Sun, S., & Hu, L. (2016). MILP-based automatic search algorithms for differential and linear trails for Speck. *Fast Software Encryption*, 268–288. https://doi.org/10.1007/978-3-662-52993-5_14

Gohr, A. (2019). Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning. *Advances in Cryptology – CRYPTO 2019*, 150–179. https://doi.org/10.1007/978-3-030-26951-7_6

Hargrave, M. (2021). *Deep Learning*. Retrieved from https://www.investopedia.com/tech-stocks-4689742#:~:text=Deep%20learning%20is%20a%20subset,learning%20or%20deep%20neural%20network

Ruiz, P. (2018). *Understanding and visualizing ResNets.* Retrieved from https://towardsdatascience.com/understanding-and-visualizing-resnets442284831be8

So, J. (2020). Deep learning-based cryptanalysis of lightweight block ciphers. *Security and Communication Networks*, *2020*, 1–11. https://doi.org/10.1155/2020/3701067

Song, L., Huang, Z., & Yang, Q. (2016). Automatic differential analysis of ARX block ciphers with application to Speck and Lea. *Information Security and Privacy*, 379–394. https://doi.org/10.1007/978-3-319-40367-0_24

Toshihiko, O. (2017). *Lightweight cryptography applicable to various IoT devices.* Retreived from https://www.nec.com/en/global/techrep/journal/g17/n01/170114.html

10. Annex

Chan Yu Anne

Background, Software Deliverables

Haw Xiao Ying

Methodology, Critical Discussion

Khor Cheng Ee

Introduction, Outcomes, Conclusion