# Detecting Block Cipher Encryption for Defense against Crypto Ransomware on Low-end Internet of Things

Hyunji Kim, Jaehoon Park, Hyeokdong Kwon, Kyoungbae Jang, Seung Ju Choi, and Hwajeong Seo*

IT Department, Hansung University, Seoul, South Korea,
{khj930704, p9595jh, korlethean, starj1023, bookingstore3, hwajeong84}@gmail.com

**Abstract.** A crypto ransomware usually encrypts files of victims using block cipher encryption. Afterward, the ransomware requests a ransom for encrypted files to victims. In this paper, we present a novel defense against crypto ransomware by detecting block cipher encryption for low-end Internet of Things (IoT) environment. The proposed method analyzes the binary code of low-end microcontrollers in the base-station (i.e. server) and it is classified in either ransomware virus or benign software. Block cipher implementations from Lightweight block cipher library (i.e. `FELICS`) and general software from AVR packages were trained and evaluated through the deep learning network. The proposed method successful classifies the general software and potential ransomware virus by identifying the cryptography function call, which is evaluated in terms of recall rate, precision rate and F-measure.

**Keywords:** Ransomware Virus · Deep Learning Network · Block Cipher Encryption · Static Analysis.

## 1 Introduction

In 2017, WannaCry ransomware virus spread to Microsoft Windows users through the vulnerability of file sharing protocol [1]. The Wannacry ransomware attack was one of the largest attack, which was estimated to have affected more than 200,000 computers across 150 countries.

The ransomware is largely classified into two sets, including locker ransomware and crypto ransomware. The locker ransomware locks the victim's device, which is unable to use the device anymore [2]. However, the data is not modified. For this reason, the data can be recoverable by copying the data to other devices.

The crypto ransomware encrypts files of victim's devices. Since the cryptography algorithm is designed to be secure theoretically, there is no way to recover the file without valid secret key. In order to receive the secret key, the victim

* Corresponding Author

need to pay the ransom to the hacker. With this secret key, the victim retrieve original files. The ransomware virus has became a massive threat of people with digital devices, as every user is moving towards digitization. To prevent from these cyber threats, many researchers have published the ransomware recovery and detection methods.

In [3], four most common crypto ransomwares were analyzed. They identified all ransomware viruses relied on system tools available on the target system. By generating shadow copies during execution of tools, the file recovery process is available.

The ransomware detection is simply classified in network traffic analysis and function call analysis. For the accurate detection, the machine learning method is actively studied for the ransomware detection. In [4], they analyzed the ransomware network behavior and packet selection to identify the ransomware virus. In [5], they evaluated shallow and deep networks for the detection and classification of ransomware. To characterize and distinguish ransomware over benign software and ransomware virus, they analyzed the dominance of Application Programming Interface (API). In [6], they presented a multi-level big data mining framework combining reverse engineering, natural language processing and machine learning approaches. The framework analyzed the ransomware at different levels (i.e., Dynamic link library, function call and assembly instruction level) through different supervised ML algorithms.

Due to the nature of crypto ransomware, many works focused on the cryptography function. In [7], they performed fine-grained dynamic binary analysis and used the collected information as input for several heuristics that characterize specific, unique aspects of cryptographic code. In [8], they presented a novel approach to automatically identify symmetric cryptographic algorithms and their parameters inside binary code. Their approach is static and based on Data Flow Graph (DFG) isomorphism. In [9], they targeted public key cryptographic algorithms performed by ransomware. By monitoring integer multiplication instructions of target system, the public key encryption by ransomware was detected.

However, previous works paid little attention on the architecture of block cipher and there are not many works on the defense against ransomware on low-end Internet of Things (IoT) environment. In this paper, we present a novel defense against crypto ransomware by detecting block cipher encryption on low-end IoT environment. The proposed method analyzes the binary code of low-end microcontrollers and the binary code is classified in either potential ransomware virus or benign software. Block ciphers from Lightweight block cipher library (i.e. FELICS) and general software from AVR packages for low-end microcontrollers were trained and evaluated through the deep learning network. The proposed method successful classifies the benign software and potential ransomware virus, which is evaluated in terms of recall rate, precision rate and F-measure. Detailed contributions are as follows:

### 1.1 Contribution

**Deep learning based crypto ransomware detection for low-end microcontrollers** By classifying the cryptographic function code and general code, the potential ransomware virus is efficiently detected. In order to apply the deep learning network, the binary file is transformed to the image file, where the deep learning network has a strength in image classification.

**Experiments with several options for high accuracy** In order to achieve the high accuracy for the classification, several options are evaluated. Instruction and opcode based binary extraction are compared to show the performance. Afterward, each GCC optimization option is evaluated to find the proper option.

**In-depth analysis of instruction sets for block cipher implementation on microcontrollers** There are a number of block ciphers. We classify block cipher algorithms into two sets, including SPN structure and ARX structure, by observing distinguished features between them. This classification rule improved the performance, significantly.

The remainder of this paper is organized as follows. In Section 2, the background of crypto ransomware detection techniques for low-end microcontrollers is covered. In Section 3, we present a novel approach to defense against crypto ransomware by detecting block cipher encryption. In Section 4, the performance of proposed methods in terms of detection accuracy is evaluated. Finally, Section 5 concludes the paper.

## 2 Related Works

### 2.1 Ransomware on IoT World

With the rapid development of Internet of Things (IoT), strengthening the security and preventing ransomware virus have become a fundamental building block for success of services [10]. There are many works to be secure the IoT environment. In [11], they presented a machine learning based approach to detect ransomware attacks by monitoring power consumption of Android devices. The method monitors the energy consumption patterns of different processes to classify ransomware from benign applications. In [12], they presented a deep learning based method to detect Internet Of Battlefield Things (IoBT) malware through the device's opcode sequence. They converted opcodes into a vector space and apply a deep Eigenspace learning approach to classify ransomware and benign application. In [13], they presented a Cryptowall ransomware attack detection model based on the communication and behavioral of Cryptowall for IoT environment. The model observes incoming TCP/IP traffic through web proxy server then extracts TCP/IP header and uses command and control (C&C) server black listing to detect ransomware attacks. In [14], they presented the method to transform the sequence of executable instructions into a grayscale image. Afterward,

**Table 1.** Comparison of ransomware detection techniques based on cryptographic function call.

| Features | Gröbert et al. [7] | Lestringant et al. [8] | Kiraz et al. [9] | This Work |
|---|---|---|---|---|
| Target | Block & PKC | Block Cipher | PKC | Block Cipher |
| Analysis | Dynamic | Static | Dynamic | Static |
| Method | Heuristics | Data graph flow | System monitor | Deep learning |
| Machine | Desktop | Desktop | Desktop | Microcontroller |

the statistical method is used for separating two or more classes along with dimension reduction.

However, previous ransomware detection focused on high-end IoT devices. A number of IoT devices are equipped with low-end microcontrollers, which collect the data in distance. This is one of the most important role of IoT applications. For this reason, the ransomware detection mechanism for low-end IoT devices should be considered. In this paper, we present the novel ransomware detection mechanism for low-end microcontrollers. In order to classify the ransomware, we transform the binary code into image file. Afterward, the image is classified through deep learning network.

### 2.2 Previous Ransomware Detection Techniques based on Cryptographic Function Call

Ransomware encrypts victim's files with cryptographic function. For this reason, classifying the cryptographic function is important for ransomware detection.

In Table 1, the comparison of ransomware detection techniques based on cryptographic function call is given. In [7], block cipher and public key cryptography are detected by analyzing features of cryptographic functions. The method is heuristics depending on the implementation. In [8], the data graph flow is extracted from binary code. Sub-graph isomorphism is utilized to identity the cryptographic function call. In [9], public key cryptography, such as RSA, is detected by monitoring the multiplication instruction, which is heavily called in RSA algorithm. However, previous works do not explore the deep learning based ransomware detection. Recently, the work by [15] shows that deep-learning algorithm can improve the malware detection. However, the method is not targeting for ransomware and target platform is desktop. In this work, we firstly present the crypto ransomware detection for microcontrollers.

### 2.3 Block Cipher Implementation on Microcontrollers

A benchmarking framework of software based block cipher implementations named Fair Evaluation of Lightweight Cryptographic Systems (FELICS) was introduced by Luxembourg University in 2015 [16]. More than one hundred block cipher implementations on low-end IoT devices were submitted to FELICS by world-wide cryptographic engineers. In this paper, we utilized block cipher implementations of FELICS. The target low-end microcontroller is 8-bit AVR

ATmega128, which is the most low-end IoT platform. The microcontroller is an 8-bit single chip based on the modified Harvard architecture developed by Atmel. The unit size of its registers and instructions is 8-bit wise. The block cipher can be largely categorized in Substitution-Permutation-Network (SPN) and Addition, Rotation, and bit-wise eXclusive-or (ARX). International block cipher standard, namely AES, is based on SPN architecture, while lightweight block ciphers, such as LEA, HIGHT, SIMON, and SPECK, follow ARX architecture due to high performance and simple design [17–20]. In this paper, we utilized distinguished features of both architectures to classify the binary code.

## 3 Proposed Method

This paper describes a method for detecting ransomware through binary files for low-end IoT-based embedded systems. Generally, the structure of sensor network is based on tree-structure [21]. The powerful root-node (i.e. base station) manages the structure and the leaf node (low-end IoT) collects the sensor data. The base station regularly updates the firmware of leaf nodes. When the hacker intercepts the packet between base station and firmware server and installs crypto ransomware to the firmware, the base station should detect the ransomware before deployment. In this scenario, the proposed method detects ransomware by classifying the binary file of the supplied firmware through a convolutional neural network. This approach distinguishes crypto ransomware from benign firmware depending on the existence of an encryption process. We can extend this approach to self-defense on the middle-end IoT, such as raspberry pi. The device can perform CNN on its machine through tensorflow[1].

The proposed system configuration for ransomware detection is shown in figure 1. Assembly instructions and the opcodes are extracted from the binary file and converted into image for deep learning training. In the test phase, if the encryption process is detected, it is classified as crypto ransomware. Overall, the proposed method consists of creating an image from binary code and two phases of deep learning.

### 3.1 Binary code based image generation

Instructions for images to be used for training are obtained by analyzing the binary files. Assembly instructions are a combination of opcode and operand, and operand area may specify different registers each time even though the same source code is compiled. In Figure 2, the opcode is extracted from the instruction. Afterward, the extracted opcode is converted into an image file to generate data for training.

Since the binary code varies depending on the optimization option, data sets are created by compiling for each option. If the pattern of the opcode are similar, similar characteristics will be created. These features are trained in the deep learning phase.

[1] https://www.tensorflow.org/install/source_rpi?hl=ko

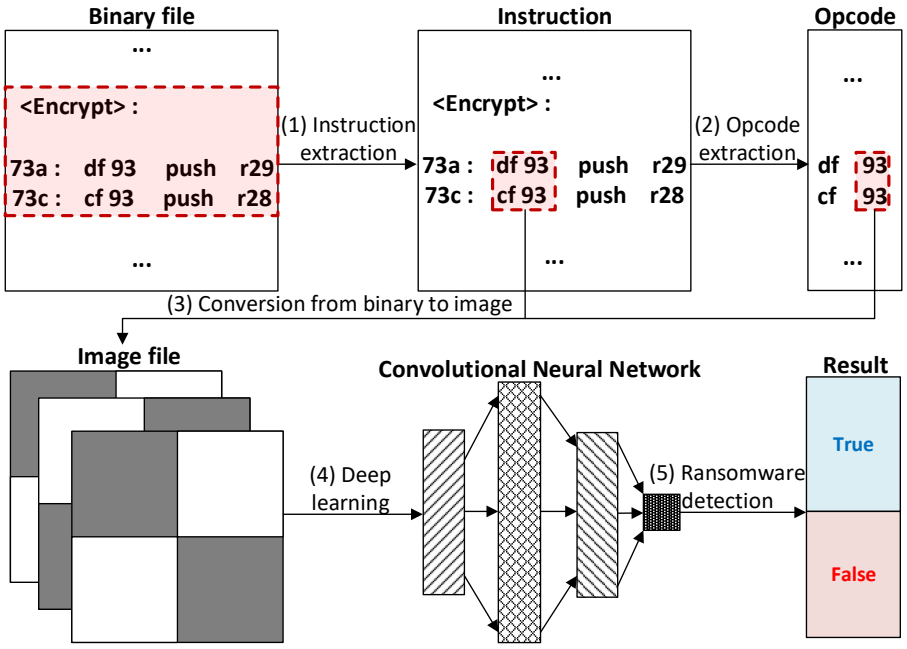


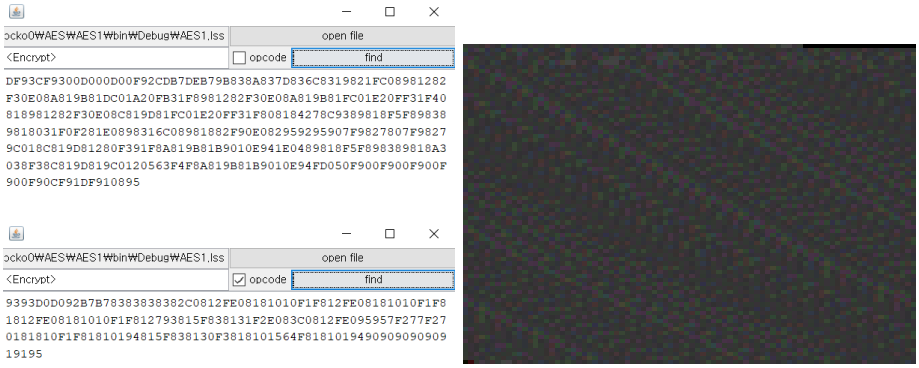

**Fig. 1.** System configuration for proposed method.

**Table 2.** Deep learning training hyperparameters.

| Hyperparameters | Descriptions | Hyperparameters | Descriptions |
|---|---|---|---|
| pretrained model | Inception-v3 | epochs | 20 |
| loss function | categorical crossentropy | steps per epoch | 10 |
| optimizer | RMSprop(lr=0.001) | batch size | 5 |
| active function | ReLu, Softmax | train, validation and test ratio | 0.7, 0.2, 0.1 |

### 3.2 Deep learning

The deep learning phase is divided into training phase and detection phase. A convolutional neural network known for its excellent image classification performance is used as a deep learning model. Inception-v3 is used as a pre-trained model, and this model performs well even on small data sets. Three fully connected layers are added to pre-trained weights to adjust the classification problem. In addition, a suitable model that is not over-fitting to a specific data set is selected through 10-fold cross validation. Grid search is performed on the selected model to tune to the optimal hyperparameter for the model and dataset. The detailed values are given in Table 2. Since it is a multi-class classification, the Softmax activation function is used in the output layer. ReLU is used, which is mainly used in hidden layers and is faster than other activation functions.




(a) binary extraction from lss file (b) conversion from binary to image

**Fig. 2.** Binary code based image generation.

**Training phase** In the proposed method, block cipher algorithms are targeted. There are two types of classification method as shown in Figure 3. One is to detect the ransomware by classifying each cryptographic algorithm and benign firmware. The other is to classify ransomware and general firmware after training by combining algorithms of the same structure into one category. According to previous methods, a data directory is configured, and the data is reshape to $150 \times 150$ pixels through pre-processing before being used as input data to the convolutional neural network.

The input layer and the hidden layer have a structure in which the convolution layer and the pooling layer are repeated. In the two dimensional convolution layer, the filter and the input image perform a convolution operation to extract features for the input image. In the max pooling layer, the output of the previous layer is divided into a window size, and the maximum value for each area is selected. By repeating the feature extraction process, the feature of the cryptographic algorithm included in each image is learned. Then, it is transferred to a fully-connected layer. Afterward, it is transformed into a one-dimensional array. Finally it enters into a classification phase for detection.

**Ransomware Detection phase** It is a phase to classify into labels for each input image by the Softmax activation function. The result of the classification is the probability value for each class. The input image is finally classified as the top-1 class with the highest probability.

Before checking the classification results of untrained test data, the performance of the validation data should be measured. If it is verified that it is an appropriate model, test data set not used for training is inputted to a trained CNN model. Test images are generated from a binary file of a block cipher or general program.

For the block cipher program, the test data has characteristics similar to the training data by the pattern of the instruction used and the repetition of rounds.

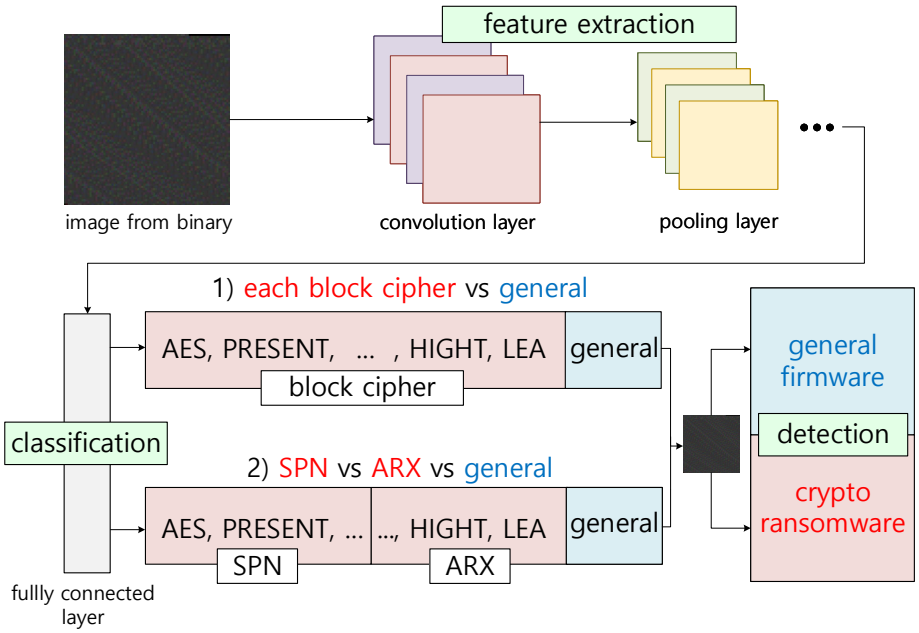



**Fig. 3.** Detailed deep learning phase for proposed method

As mentioned earlier, such characteristics can be classified by cryptographic structure (SPN and ARX) or each cryptographic algorithm. If it is classified as a cryptographic algorithm, it is judged as ransomware. It can block the installation to IoT devices. In other words, block ciphers and general programs are classified by weight values trained by image-based deep learning. Through the proposed framework, we can detect the potential ransomware virus for low-end IoT devices.

## 4 Evaluation

For the experiment, Google Colaboratory, a cloud-based service, was utilized. It runs on Ubuntu 18.04.3 LTS and consists of an Intel Xeon CPU with 13GB RAM and an Nvidia GPU (Tesla T4, Tesla P100, or Tesla K80) with 12GB RAM. In terms of programming environment, Python 3.6.9, tensorflow 2.2.0-rc and Keras 2.3.1 version are used. In order to create the dataset required for the experiment, we created a program that extracts instructions and opcodes of specific functions from the lss file. And instruction and opcode were converted to BMP image files using open source on github[2].

In Table 3, the experiment targets binary files of general firmware and block ciphers (e.g., SPN and ARX) in low-end embedded environment. Cryptographic

[2] https://github.com/Hamz-a/txt2bmp

**Table 3.** Dataset (block ciphers and general firmware).

| Architecture | Descriptions of programs |
|---|---|
| SPN | AES, RECTANGLE, PRESENT, PRIDE, PRINCE |
| ARX | HIGHT, LEA, RC5, SIMON, SPECK, SPARX |
| General | Bluetooth, GPS, WiFi, RFID, XBee, etc. |

modules written in C language among implementations of FELICS (Fair Evaluation of Lightweight Cryptographic System) are selected. In the case of general firmware, programs, such as WiFi, Bluetooth, xBee, and RFID, are mainly obtained.

Since the crypto ransomware performs an encryption process unlike ordinary firmware, we extract the instructions and opcodes of the encryption function from binary file. As shown in Figure 4, instructions and opcodes are converted into image files. SPN using S-box has a more complicated structure than ARX. In addition, it is possible to visually confirm that there is a common characteristic for each architecture. We trained these features using the CNN and progressed experiments to classify cryptographic algorithms and general firmware.

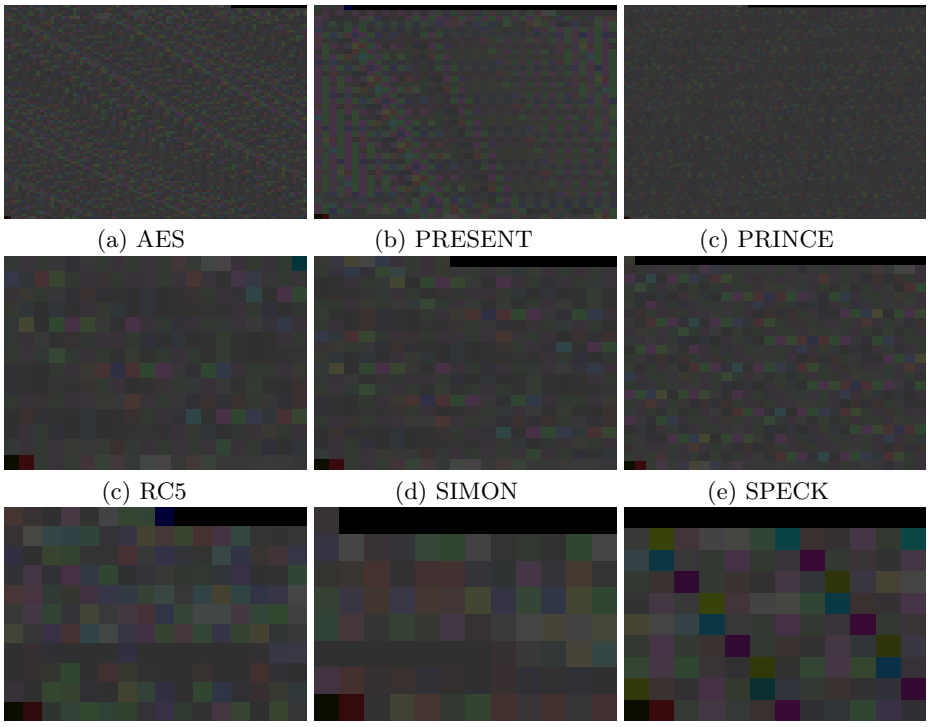
Since this experiment uses an unbalanced data set, it is commonly evaluated through the F-measure, which is a harmonic mean of precision and recall rather than accuracy. There are micro averaged and macro averaged in the F-measure. The micro averaged method considers the number of data belonging to each class. The macro averaged method considers all classes with the same weight. Therefore, results of macro method are a slightly lower measurement in the case of an unbalanced dataset.

### 4.1 Instruction-based vs Opcode-based

Since our proposed system classifies each cryptographic module according to the instruction (e.g. pattern of operation type and number of times) performed by encryption, the method of extracting the encryption part from the binary file is largely based on instruction and opcode. In the case of instruction-based, the opcode and operand are extracted together. In order to compare the performance, only opcode part, which is the actual operation, is extracted.

As shown in Figure 5, training loss and validation loss decreases smoothly without significant difference. The training data has not been over-fitted. In the case of opcode based, the overall loss value was calculated less than instruction based approach.

After verifying the model with the data used in the training, the test data, which was not used in the training through the model is predicted. In Table 4, detailed results are given. By classifying with the test set, the instruction-based performance was slightly better. However, in the case of opcode-based, the standard deviation is 0.12, which has more stable performance. It indicates that the method of extracting only the opcodes specifying the actual operation is more effective in finding the instruction pattern of the binary file.



(a) AES (b) PRESENT (c) PRINCE

(c) RC5 (d) SIMON (e) SPECK

(f) xBee (g) GPS (h) WiFi

**Fig. 4.** Images of binary files.

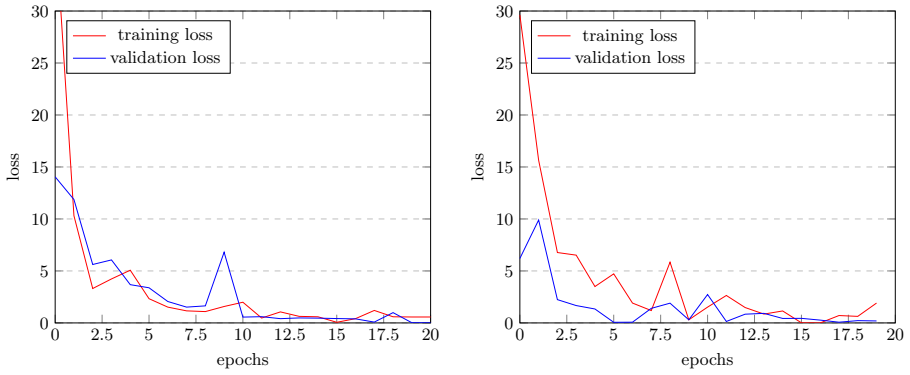



**Fig. 5.** Training and validation loss depending on instruction and opcode; left: instruction, right: opcode.

**Table 4.** Validation and test on instruction and opcode.

| Target | Validation | | | | | | Test | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | | precision | | recall | | F-measure | | precision | | recall | |
| | micro | macro | micro | macro | micro | macro | micro | macro | micro | macro | micro | macro |
| Instruction | 0.91 | 0.84 | 0.91 | 0.87 | 0.91 | 0.95 | 0.80 | 0.60 | 0.80 | 0.60 | 0.80 | 0.60 |
| Opcode | 0.96 | 0.93 | 0.96 | 0.92 | 0.96 | 0.94 | 0.77 | 0.58 | 0.77 | 0.59 | 0.77 | 0.60 |

**Table 5.** Validation and test on GCC optimization options.

| Op. | validation | | | | | | test | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | | precision | | recall | | F-measure | | precision | | recall | |
| | micro | macro | micro | macro | micro | macro | micro | macro | micro | macro | micro | macro |
| `O0` | 0.96 | 0.93 | 0.96 | 0.92 | 0.96 | 0.94 | 0.77 | 0.58 | 0.77 | 0.59 | 0.77 | 0.60 |
| `O1` | 0.90 | 0.81 | 0.90 | 0.80 | 0.90 | 0.85 | 0.85 | 0.79 | 0.85 | 0.82 | 0.85 | 0.81 |
| `O2` | 0.92 | 0.89 | 0.92 | 0.90 | 0.92 | 0.9 | 0.81 | 0.67 | 0.81 | 0.69 | 0.81 | 0.68 |

### 4.2 GCC optimization option

After compiling with optimization option for each encryption algorithm, an opcode-based classification experiment was performed. For the experiment, the GNU AVR-GCC compiler and `O0`, `O1` and `O2` options were used.

Binary files used in the experiment were not changed significantly depending on the optimization option. However, the classification performance changed slightly depending on the optimization level. As shown in Figure 6, the loss value was reduced without over-fitting as a whole. As a result of classifying the untrained test data, the optimization option `O1` showed the best performance.

Table 6 shows the result of sorting the command used for each structure in the order in which they are used frequently. In the case of the SPN structure, the pattern of `LD-XOR-ST` is common in the part where S-box operation is performed, and operations to access the memory frequently occurred. In the ARX structure based on addition, rotation and exclusive-or operation, the ratio of arithmetic and logical operations, such as `ADD, XOR, SUB` were frequently performed. In the case of general firmware, there are many instructions to activate the interrupt function and to access the I/O register, not found in the encryption code, and the ratio of branch statements was largest among them. When applying the higher optimization option, it seems that the instruction pattern, count, and order that are effective for classifying each algorithm through the CNN model have changed slightly, affecting classification performance.

Cryptography algorithms of the same structure share similar instruction patterns. Since block cipher algorithms repeat rounds, certain patterns are repeated. Therefore, the BMP image generated based on the opcode also has a pattern. For this reason, cryptographic algorithm is successfully classified through the proposed approach.

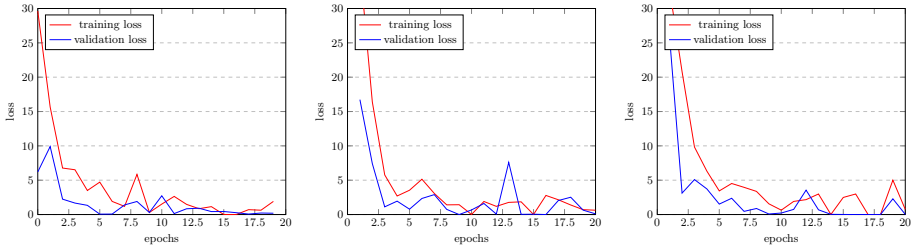



**Fig. 6.** Training and validation loss depending on optimization option; left: `O0`, middle: `O1`, right: `O2`.

**Table 6.** Frequently used instructions for each architecture.

| Architecture | Ordered by frequency in program | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| SPN | LD | ST | MOV | XOR | ADD | SUB | AND | SWAP |
| ARX | LD | ADD | XOR | MOV | ST | SUB | ROR | RJMP |
| General | LD | RJMP | BNE | CP | OUT | NOP | MOV | SEI |

### 4.3 Block cipher vs General

In this experiment, block cipher algorithms are divided in two design groups (i.e. SPN and ARX) to classify ransomware or general firmware.

In Table 7, F-measure for each option is given. The CNN model is properly fitted at high speed. The loss value decreases. When the epoch exceeds 10, the loss value converges to almost 0.00 (See Figure 7).

The previous experiment does not properly classify, because the instructions used in cryptographic algorithms of the same architecture have similar patterns. By dividing the block cipher into two structures (i.e. SPN and ARX), the proposed method achieved better performance than classifying each block cipher module. Among the test data, the recognition of cryptographic algorithm as general firmware occurred once in the case of optimization options `O0` and `O2`, respectively. The recognition of general firmware as cryptographic algorithm occurred once in `O0`. It accurately predicted test data in optimization option `O1`. The ransomware can be detected with a high overall probability for all optimization options.

## 5 Conclusion

In this paper, we presented a novel approach to detect ransomware virus through classification of block cipher module for low-end microcontrollers. The binary code is converted to image file and deep learning network is applied. By observing the specific features of SPN and ARX architectures, block cipher classification is divided into two categories. This approach significantly improved the accuracy.

**Table 7.** Validation and test depending on architectures (SPN, ARX, or general).

| Op. | validation | | | | | | test | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | F-measure | | precision | | recall | | F-measure | | precision | | recall | |
| | micro | macro | micro | macro | micro | macro | micro | macro | micro | macro | micro | macro |
| `O0` | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.94 | 0.91 | 0.94 | 0.91 | 0.94 | 0.95 |
| `O1` | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| `O2` | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.98 | 0.99 | 0.98 | 0.99 | 0.98 | 0.99 |

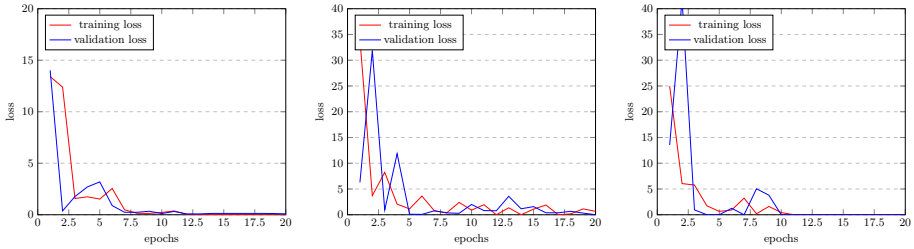



**Fig. 7.** Training and validation loss depending on classification rule and (i.e. SPN, ARX, and general program) optimization option; left: `O0`, middle: `O1`, right: `O2`.

## 6 Acknowledgement

This work was partly supported as part of Military Crypto Research Center(UD170109ED) funded by Defense Acquisition Program Administration(DAPA) and Agency for Defense Development(ADD) and this work was partly supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-00264, Research on Blockchain Security Technology for IoT Services) and this work was partly supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. NRF-2020R1F1A1048478).

## References

1. S. Mohurle and M. Patil, "A brief study of Wannacry threat: Ransomware attack 2017," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 5, 2017.
2. A. Kharaz, S. Arshad, C. Mulliner, W. Robertson, and E. Kirda, "`UNVEIL`: A large-scale, automated approach to detecting ransomware," in *25th USENIX Security Symposium (USENIX Security 16)*, pp. 757–772, 2016.
3. M. Weckstén, J. Frick, A. Sjöström, and E. Järpe, "A novel method for recovery from Crypto Ransomware infections," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, pp. 1354–1358, IEEE, 2016.
4. A. Tseng, Y. Chen, Y. Kao, and T. Lin, "Deep learning for ransomware detection," *IEICE Tech. Rep.*, vol. 116, no. 282, pp. 87–92, 2016.

5. R. Vinayakumar, K. Soman, K. S. Velan, and S. Ganorkar, "Evaluating shallow and deep networks for ransomware detection and classification," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 259–265, IEEE, 2017.
6. S. Poudyal, D. Dasgupta, Z. Akhtar, and K. Gupta, "A multi-level ransomware detection framework using natural language processing and machine learning," in *14th International Conference on Malicious and Unwanted Software" MALCON*, 2019.
7. F. Gröbert, C. Willems, and T. Holz, "Automated identification of cryptographic primitives in binary programs," in *International Workshop on Recent Advances in Intrusion Detection*, pp. 41–60, Springer, 2011.
8. P. Lestringant, F. Guihéry, and P.-A. Fouque, "Automated identification of cryptographic primitives in binary code with data flow graph isomorphism," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pp. 203–214, 2015.
9. M. S. Kiraz, Z. A. Genç, and E. Öztürk, "Detecting large integer arithmetic for defense against crypto ransomware," *Tech. Rep.*, 2017.
10. I. Yaqoob, E. Ahmed, M. H. ur Rehman, A. I. A. Ahmed, M. A. Al-garadi, M. Imran, and M. Guizani, "The rise of ransomware and emerging security challenges in the Internet of Things," *Computer Networks*, vol. 129, pp. 444–458, 2017.
11. A. Azmoodeh, A. Dehghantanha, M. Conti, and K.-K. R. Choo, "Detecting crypto-ransomware in IoT networks based on energy consumption footprint," *Journal of Ambient Intelligence and Humanized Computing*, vol. 9, no. 4, pp. 1141–1152, 2018.
12. A. Azmoodeh, A. Dehghantanha, and K.-K. R. Choo, "Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 1, pp. 88–95, 2018.
13. A. Zahra and M. A. Shah, "IoT based ransomware growth rate evaluation and detection using command and control blacklisting," in *2017 23rd International Conference on Automation and Computing (ICAC)*, pp. 1–6, IEEE, 2017.
14. A. Karimi and M. H. Moattar, "Android ransomware detection using reduced opcode sequence and image similarity," in *2017 7th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 229–234, IEEE, 2017.
15. R. Kumar, Z. Xiaosong, R. U. Khan, I. Ahad, and J. Kumar, "Malicious code detection based on image processing using deep learning," in *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence*, pp. 81–85, 2018.
16. D. Dinu, A. Biryukov, J. Großschädl, D. Khovratovich, Y. Le Corre, and L. Perrin, "FELICS–fair evaluation of lightweight cryptographic systems," in *NIST Workshop on Lightweight Cryptography*, vol. 128, 2015.
17. J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1999.
18. D. Hong, J.-K. Lee, D.-C. Kim, D. Kwon, K. H. Ryu, and D.-G. Lee, "LEA: A 128-bit block cipher for fast encryption on common processors," in *International Workshop on Information Security Applications*, pp. 3–27, Springer, 2013.
19. D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, *et al.*, "HIGHT: A new block cipher suitable for low-resource device," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 46–59, Springer, 2006.
20. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "SIMON and SPECK: Block ciphers for the internet of things.," *IACR Cryptology ePrint Archive*, vol. 2015, p. 585, 2015.

21. J. L. Williams, J. W. Fisher, and A. S. Willsky, "Approximate dynamic programming for communication-constrained sensor network management," *IEEE Transactions on signal Processing*, vol. 55, no. 8, pp. 4300–4311, 2007.