Performance Evaluation of Intensity Channel Based Multi-Scale Retinex Algorithm Using CUDA

Kyi Myo Zaw [1], Dr. Aye Min Myat [2]
Dept. of Computer Engineering [1,2],
Faculty of Information and Communication Technology [1,2],
University of Technology (Yatanarpon Cyber City) [1,2],
kyimyozaw122.kmz@gmail.com [1], awba07@gmail.com [2]

Abstract

Retinex image enhancement algorithm is one of the commonly used image enhancement algorithm, that it is used to enhance the nonuniform lighting images. The colour restoration problem exists in colour channel based Retinex algorithm and the enhancing process is computationally extensive and time-consuming. Therefore, we implement the parallel Retinex algorithm that operates on intensity channel of the image using CUDA (Compute Unified Device Architecture). Our implementation can achieve 29.2-time speedup for the 4,096 × 4,096 image resolution compared with the OpenCV optimized CPU version.

Keywords- CUDA, GPGPU, Image enhancement, Parallel computing, Retinex

1. Introduction

One of the most important characteristic of the Human Visual System (HVS) is the colour constancy. It means that the human eye can see the true colour of the scene under varying illumination conditions. But most of the image devices cannot fully capture the nonuniform illumination scene. And the Retinex algorithm is commonly used to solve that kind of problem. The computation of the Retinex enhancement process is very intensive and time-consuming process because the enhancement processes is done on each pixel with complex computation process. The Retinex algorithm that operates on colour channels of the image have colour restoration problems and the computation have to compute for each colour channels and that it takes longer time to enhance. Thus, we implement a parallel version of Retinex algorithm that operates on intensity/luminance channel of the image.

With the advance in technology, the imaging devices nowadays can capture a very high-resolution images and videos. The enhancing process of the Retinex algorithm is directly proportional to the resolution of the image that it is needed to be enhanced.

The aims of our work are to solve the colour restoration problem in the Retinex enhanced images and to accelerate the computing speed of the Retinex algorithm as fast as possible by using the compute power of Graphics Processing Unit (GPU). We use CUDA which is a parallel computing platform and application programming interface (API) developed by NVIDIA to develop the parallel version of the Retinex algorithm. So, that the Retinex algorithm is fast enough to be used as an efficient pre-processing step in many image and video applications such as face recognition, object detection, vehicle tracking in traffic light, and many other related applications. To work out this, we need to analyse and divide the Retinex algorithm into two parts: serial program segment that run on the CPU and parallel program segment that run on the GPU. And also, how to solve colour restoration problems in some Retinex enhanced image.

Our research paper is constructed as follows: Section 2 expresses the previous related works that concerned with our work. Section 3 explains the detailed process of our work and how we solve the problems that are facing while developing our system and how we construct our proposed system. Section 4 gives our experimental results, performance evaluation between the CPU version and CUDA version, and the discussion. And in Section 5, we conclude our approach with suggestion on future works that can improve our work.

2. Related Works

In this Section 2, we review related works that concerned with our works. This Section 2 is composed of two subsections. Firstly, the use of CUDA in some image processing and computer vision applications and the next is the Multi-Scale Retinex (MSR) algorithm.

2.1. Some Image Processing Algorithms on GPU

CUDA is a parallel computing platform and application programming interface (API) model created by NVIDIA for its own GPU. It allows software developers and engineers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing — an approach also known as GPGPU (General-Purpose computing on Graphics Processing Units). It can be used with the high-level programming language such as C or C++. With the easy to program, CUDA has been adopted to use to accelerate many computationally intensive tasks in image processing and computer vision domains.

A parallel version of Canny edge detector using CUDA was implemented by Luo and Duraiswami [1], including all algorithm stages. They compared their GPU Canny implementation with an assembly optimized CPU implementation in the OpenCV library and achieved maximum speedup up to 3.8 times speedup on the image resolution of 3,936 × 3,936.

And the Canny edge detection algorithm using CUDA implemented on a PC server with Intel Xeon E5540 running in 2.0GHZ and 6GB and NVIDIA Tesla C1060 with 240 cores running in 1.3GHz and 4GB memory by Ogawa et al. [2] could achieve up to 50 times speedup including data transfer time testing with the image resolution of 10,240 × 10,240 compared to CPU version. If the data transfer time is not included, CUDA implementation could achieve approximately 61 times speedup compared to the CPU version.

The real-time object detection on CUDA proposed by Herout et al. [3] could outperform the reference efficient CPU/SSE implementation, by approximately 6–8 times speedup for the high-resolution videos using NVIDIA GeForce 9800GTX and Intel Core2 Duo E8200 as the experimental environment.

In the CUDA implementation of McCann99 Retinex algorithm by Seo and Kwon [4], the sequential MATLAB code took 25.429 seconds in 1,024 × 1,024 image size, and sequential C codes took 4.1 seconds, and the proposed CUDA implementation took only 0.760 seconds for the same image. So, CUDA implementation could achieve 5.3 times speedup compared to the CPU implementation written in C.

The GPURetinex algorithm proposed by Wang and Huang [5] is a data parallel algorithm devised by parallelizing the Retinex based on GPGPU/CUDA, and their experimental results showed that their GPURetinex could achieve 30 times speedup for 2,048 × 2,048 image resolution on NVIDIA Tesla C1060 and Intel Core2 Duo 3.0GHz. Because of the Gaussian kernel separable property, they separate the Gaussian convolution into row convolution and column convolution instead of traditional Gaussian convolution to implement in CUDA.

The above GPURetinex algorithm was tested up to 2,048 × 2,048 image resolution and could achieve speedup up to 30 times. So, they improved the GPURetinex [6] by optimizing the memory usage and out-of-boundary extrapolation in the convolution step. The improved GPURetinex could achieve speedup 72 times compared with the optimized single-threaded CPU implementation by OpenCV for the image resolution of 2,048 × 2,048. The speedup of the Retinex implementation based on the NPP (NVIDIA Performance Primitives) was 42 times. So, their improved version of the GPURetinex outperformed the Retinex implementation based on the NPP. They measured the speedup using the total execution time, but that didn't include the cost of memory management and the data transfer between the GPU and CPU.

The above two GPURetinex algorithms they proposed were tested up to 2,048 × 2,048 image resolution. They proposed the mathematical models that helped to optimally choose memory spaces and block sizes for the GPURetinex on [7], and gained 74 times speedup for the 4,096 × 4,096 image resolution, that was twice as fast as the GPURetinex_NPP. The acceleration was measured by the total execution time excluding the cost of memory management and data transfer between the GPU and CPU.

2.2. Multi-Scale Retinex Algorithm

The Retinex algorithm was developed by Land [8] and it is one of the most famous algorithms that attempt to explain the human colour constancy. Colour constancy is one of the most important characteristics of human visual system because it can ensure that the perceived colour to the eyes remains the same under varying illumination conditions.

The Retinex algorithm is based on HVS and there are many Retinex algorithms. Such as path-based Retinex, recursive Retinex, centre/surround Retinex and the PDE-based Retinex. Among them, the centre/surround Retinex is well suited for parallelization because of the convolution operations, log-domain processing, and normalization process in the algorithm are independent to each other and could be run in parallel.

In Retinex algorithm the input image I is formed by the product of the illumination L to the image and the reflection R from the image, i.e., $I = L \cdot R$. The aim of the Retinex algorithm is to estimate the illumination L from the original image I to discard the effect of nonuniform illumination from the image I to improve the visual quality of the image I. And the MSR is the one of the variations from the centre/surround Retinex algorithm.

The single-scale Retinex (SSR) that uses a Gaussian blur operation to compute the centre/surround information proposed by Rahman et al. [9]. The extension of SSR is multi-scale Retinex (MSR) and that combines dynamic range compression and colour/lightness rendition by using weighted three SSRs with different spatial scales. Finally, the multi-scale Retinex with colour restoration (MSRCR) [10] [11] was proposed to restore some colour loss in MSR process using colour restoration factor.

The basic form of the single-scale Retinex (SSR) is as shown in Eq. 1:

$$R_i(x,y) = \log I_i(x,y) - \log[F(x,y) * I_i(x,y)], \quad (1)$$

where x and y denote the coordinates of a pixel in image, R_i (x, y) is the Retinex output, I_i (x, y) is the image distribution in i-th spectral band, the symbol "$*$'' denotes the convolution operator, and F (x, y) is the Gaussian surround function as Eq. 2.

$$F(x,y) = Ke^{-r^2/c^2}, \quad (2)$$

where c is the Gaussian surround space constant, and K is also a constant such that,

$$\iint F(x,y)dxdy = 1. \quad (3)$$

The constant c is used for controlling the scale of $F(x, y)$. A small value of c provides a good dynamic range compression, and a large scale provides better colour rendition.

In order to combine the dynamic range compression and the tonal rendition, SSR is extended to multiscale Retinex (MSR). The result of MSR is a weighted sum of the results of SSR with different scales. The i-th spectral component of MSR output is mathematically expressed by the following equation:

$$R_{MSR_i} = \sum_{n=1}^{N} w_n R_n, \quad (4)$$

where, N : the number of scales,

R_{n_i} : the i-th component of the n-th scale,

w_n : the weight associated with the n-th scale.

R_{n_i} is defined by Eq. 5.

$$R_{n_i}(x,y) = \log I_i(x,y) - \log\left[F_n(x,y) * I_i(x,y)\right], \quad i = 1, \ldots, S, \quad (5)$$

where $F_n(x,y) = Ke^{-r^2/c_n^2}$, c_n is the constant of the n-th scale. From Eq. 4 and Eq. 5, $R_{MSR_i}(x,y)$ can be rewritten as Eq. 6.

$$R_{MSR_i}(x,y) = \sum_{n=1}^{N} w_n \left\{\log I_i(x,y) - \log\left[F_n(x,y) * I_i(x,y)\right]\right\}, \quad i = 1, 2, 3, \ldots, S, \quad (6)$$

where $R_{n_i}(x,y)$ denotes a Retinex output associated with n-th scale for an image, $I_i(x,y)$ and $F_n(x,y)$ denote a surround function. S is the number of spectral bands in the image.

A gain w_n is set to satisfy the condition $\sum_{n=1}^{N} w_n = 1$.

The surround function of Retinex algorithm is given by $F_n(x,y) = K_n e^{-(x^2+y^2)/c_n^2}$, where c_n are the scales that control the extent of the surround (smaller values of c_n lead to narrower surrounds, and larger values of c_n lead to wide surrounds), and the normalization factor is $K_n = \frac{1}{\sum_x \sum_y F_n(x,y)}$.

In this system, we only used up to MSR steps in the centre/surround Retinex algorithm.

And the process of convolution operations in the MSR can be performed in the frequency domain according to the convolution theorem [12].

According to the convolution theorem, the convolution of the two functions by Fourier transform is equal to the product of the individual transforms. So, the convolution in the SSR becomes as Eq. 7.

$$R_i(x,y) = \log I_i(x,y) - \log\left[F^{-1}\{\hat{F}(v,w) \cdot \hat{I}_i(v,w)\}\right] \quad (7)$$

where $\hat{F}(v,w)$ and $\hat{I}_i(v,w)$ are the Fourier transforms of $F(x,y)$ and $I_i(x,y)$ and F^{-1} denotes the inverse Fourier transform.

We use three scales for the MSR in this proposed system, they are 15, 80 and 250. Because Jobson [11] found empirically, that a combination of three scales representing narrow, medium and wide surrounds is sufficient to provide both dynamic range compression and tonal rendition. Thus, the MSR is more computationally intensive than the SSR because it requires to compute the three large Gaussian blur convolutions on a large scale on the intensity channel of the image compares to the one Gaussian blur convolution size depends on the scale value on the intensity channel of the image. We compute the three Gaussian convolutions of the Retinex algorithm in the frequency domain as a simple multiplication as opposed to the conventional 2-D convolution in spatial domain to speedup the enhancement process of the Retinex algorithm.

3. Proposed Intensity Channel Based Retinex Algorithm Using CUDA

In this Section 3, we explain about the structure of our proposed system in details. Firstly, the image we load to be enhanced is needed to be colour (RGB) image and then convert the RGB image to HSV colour space and split the channels. “Why we choose HSV colour model for our implementation?” is explained in Section 4. After splitting the HSV channels, enhances the intensity (Value) channel by using MSR algorithm and merge the enhanced Value channel with the original Hue and Saturation channel and convert back to RGB image to get the MSR enhanced image. When performing using CUDA the Value channel is needed to be uploaded to the GPU for performing MSR operation on GPU, and downloaded back to the CPU for merging and convert back to RGB to get the MSR image.

We can get the desired image by using MSR, and there is no colour restoration step and not operates on colour channels. So, we don’t need to restore the colours for the MSR enhanced image, and the colour restoration problems in some of the Retinex algorithm enhanced images are solved.

There is one more step to get the final desired enhanced image by doing the automatic treatment of the Retinex output prior to display by using the following Eq. 8 proposed by Mooer et al. [13].

$$R'_{MSRCR_i}(x,y) = \frac{d_{max}}{r_{max} - r_{min}} * (R_{MSRCR_i}(x,y) - r_{min}), \quad (8)$$

where d_{max} = 255 for 8-bit image, r_{max} and r_{min} are the maximum and the minimum pixel value in the processed image.

The overall system block diagram is shown in Figure 1.

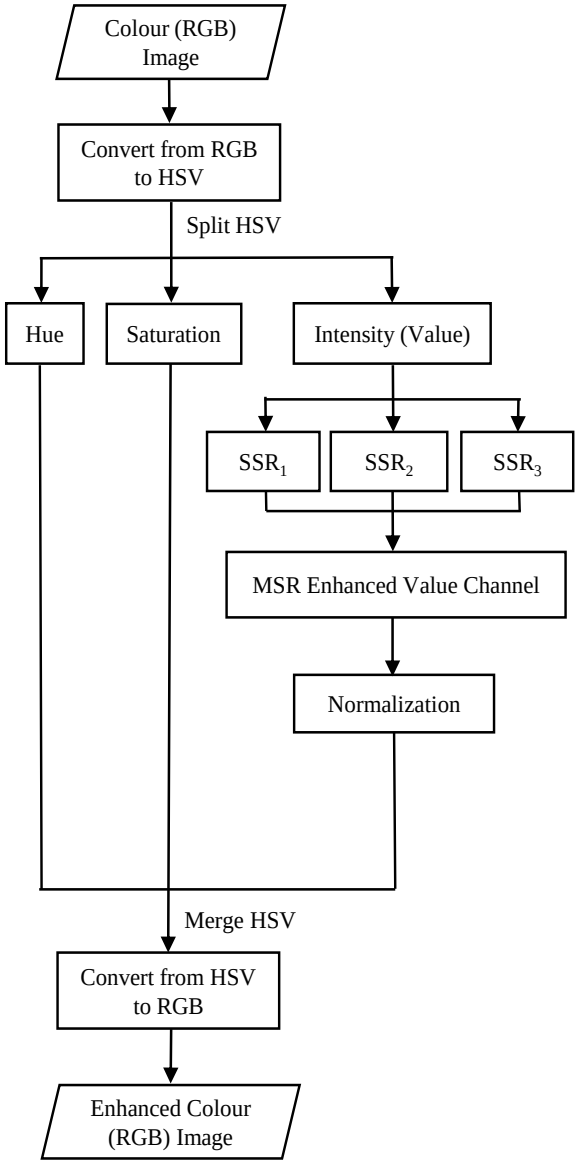

Figure 1. System Block Diagram

And the Figure 2 show the works done on the GPU using CUDA. The steps for the convolution operations that perform in frequency domain on the GPU are as follows:

1. Calculate the optimal DFT (Discrete Fourier Transform) size for padding.
2. Calculate the Fourier transform of the image with padding.
3. Generate a filter function *F*, the same size as the image.
4. Multiply the transformed image by the filter (pixel-wise multiplication).
5. Inverse Fourier transform to get the convolved image.

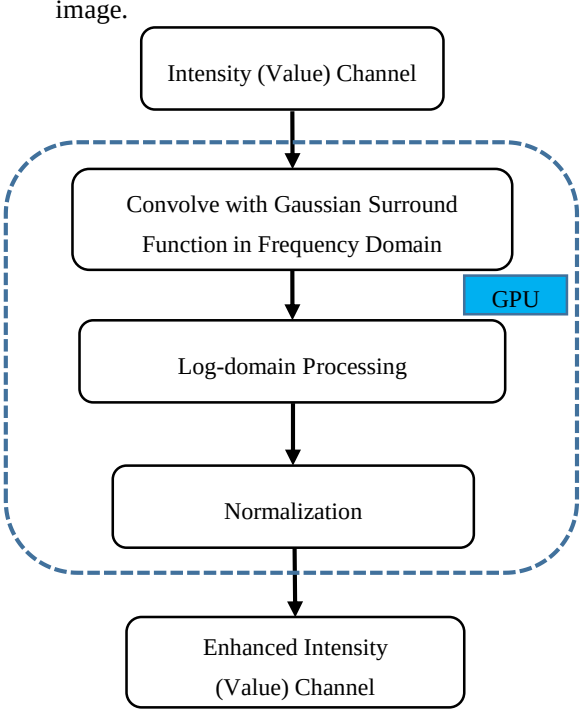

Figure 2. GPU Operations Block Diagram

4. Experimental Results and Performance Evaluation

In this Section 4, we show why we choose HSV colour model for our implementation and the experimental results from our experiments.

The computing of the convolution, log-domain processing and normalization are done on GPU to optimize the Retinex algorithm. In this section, the computational experiments are carried out to compare the performance between the CPU program and CUDA program.

The experimental photos used in this system are mostly self-collected photos. Other photos for testing are obtained from A.B. Petro CC-BY and NASA Retinex image data set.

The quality of the enhancement image is evaluate numerically using PSNR and SSIM.

4.1. Experimental Environment

Our experiments are done on the desktop PC with the following specifications:

- CPU: Intel (R) Core™ i7-6700 CPU @ 3.4 GHz, 8 GB DDR4 RAM
- GPU: NVIDIA GeForce GTX 1080
  - GPU Memory: 8 GB GDDR5X
  - Memory Interface: 256-bit
  - Memory Bandwidth: 192 GB/s
  - NVIDIA CUDA Cores: 2,560
  - System Interface: PCI Express 3.0×16
- Window 10 64-bit with Visual Studio 2017 and CUDA toolkit v10.2

The results from the experiments are shown in the following table and figures. We use six resolutions of image size 256 × 256, 1,024 × 1,024, 2,048 × 2,048, 3,072 × 3,072, and 4,096 × 4,096 in our experiments and measure the total execution time including the data transfer time between the CPU and the GPU.

4.2. Why We Choose HSV?

We making experiments with five colour models. They are Lab, Luv, YCrCb, HSL and HSV. The enhanced images of each colour model are shown in Figure 3, the top left to top right images are the original image, the Lab enhanced image and the luv enhanced image respectively, and the bottom left to bottom right images are the YCrCb enhanced image, the HSL enhanced image and the HSV enhanced image. As we can see, the Lab enhanced image and the Luv enhanced image is almost the same. The YCrCb with a little saturate colour compare to Lab and Luv, but not saturate enough to use in the proposed algorithm. The saturation of the HSL enhanced image is good but the leaves and trees in the image are undistinguishable. HSV enhanced image is the best comparing with other enhanced images, so we choose HSV colour model for our proposed algorithm.

Figure 3. Original and Experiments Colour Model Photos

4.3. Experimental Results and Discussions

We tested 50 images with fixed sizes (i.e., row resolution and column resolution are the same). More than another 50 images with various resolutions are tested. The total execution time we measure for making comparisons for our experiment are measured from 50 six fixed resolutions images for 10 iterations, and take the average for both the CPU and the GPU. For the CPU implementation, we use OpenCV optimized Gaussian blur and other optimized functions. For GPU, we implement it with the combination of OpenCV and CUDA. The following Table 1 and Figure 4 shows the comparison of total execution time between the CPU and CUDA in milliseconds, and the speedup they can gain for each image resolution. We cannot achieve speedup for the smallest image resolution 256 × 256, because the use of GPU can gain speedup when the data needed to be computed is large enough. If the data needed to compute is large enough then the CPU is much faster than the GPU.

Table 1. Total Execution Time in Milliseconds

Image Size (M x N)	Total Execution Time (milliseconds)		Speed-up (%)
	CPU	CUDA	
256 × 256	285.6	588.9	(None)0.48
512 × 512	788.2	612.0	1.29
1,024 ×1,024	2,933.6	693.3	4.23
2,048 × 2,048	15,756.5	992.8	15.87
3,072 × 3,072	31,566.4	1,412.5	22.35
4,096 × 4,096	73,122.6	2,504.3	29.20

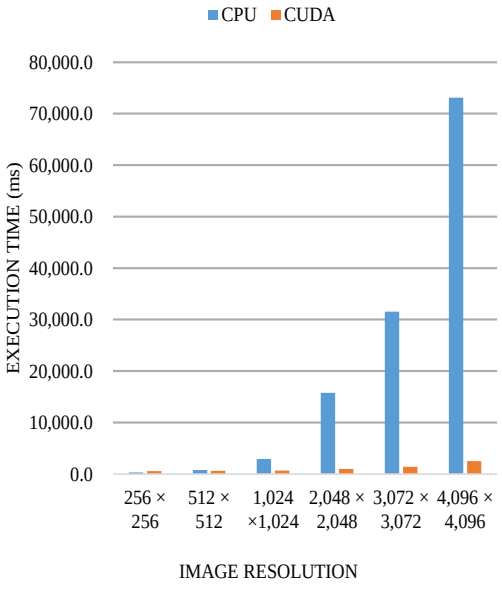

Figure 4. Total Execution Time Bar Chart

Figure 5 shows the resulted enhanced images from our proposed system in which from left column to right column images are the original images, colour channels based Retinex enhanced images, and the intensity channels based enhanced images. As we can see in this figure, the colour channels based Retinex algorithm images are blur and invert of colour compare to the intensity channel based enhanced images. We

solved this colour restoration problem by enhancing the intensity channel of the image with the MSR algorithm.

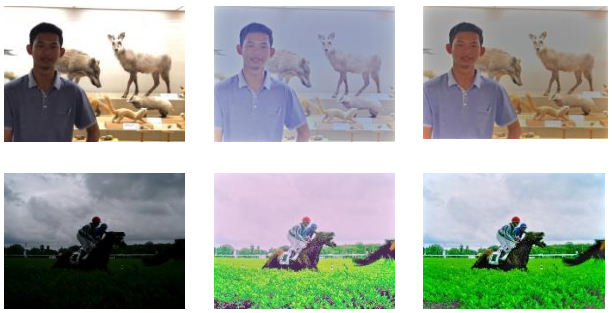

Figure 5. Some Result Photos from the System

The PSNR and SSIM values of the above photos is described on the following Table 2. The SSIM returns the MSSIM of the images.

Table 2. PSNR and MSSIM for Image Quality Evaluation

Image	PSNR	MSSIM [R, G, B]
Person	12.0383	[0.866376, 0.870432, 0.87286]
Cavalls	9.91055	[0.482524,0.490889, 0.495475]

The PSNR value less than 15 here means that the enhanced image is very different from the original image. As we can see from the Figure 5, the darker region in the image is alleviate by performing Retinex enhancement operation. The MSSIM value is a float number between 0 and 1 (the higher the better). The PSNR value and SSIM values of the cavalls image are not good but the image is much more visually pleasing compared to the original image.

5. Conclusion

In this paper, we implemented the intensity channel based MSR algorithm using CUDA. The system cannot work well when the image is already well contrast or very good dynamic range compression and the output enhanced images form this system.

In the Figure 5, the bottom right image is more visual pleasing than the top right image. In the future, we will find the better solution to get more visual pleasing for the image like the Figure 5 top right image.

As we can see from our experiments, the choice of right colour model for intensity based MSR algorithm is important for further implementation on the GPU and the enhanced image quality is not dependent on the PSNR and SSIM values. But it depends on the user needs. It could achieve more speedup compare to the CPU version and also could get the more visual pleasing image. The maximum speedup it can get for the 4,096 × 4,096 image resolution in 29.2 times over the CPU version.

In the future, we will find a way to make an enhanced image to be more visual pleasing and better quality.

6. References

[1] Y. Luo and R. Duraiswami, "Canny edge detection on NVIDIA CUDA," *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2008.

[2] K. Ogawa, Y. Ito, and K. Nakano, "Efficient Canny Edge Detection Using a GPU," *2010 First International Conference on Networking and Computing*, 2010.

[3] A. Herout, R. Jošth, R. Juránek, J. Havel, M. Hradiš, and P. Zemčík, "Real-time object detection on CUDA," *Journal of Real-Time Image Processing*, vol. 6, no. 3, pp. 159–170, Aug. 2010.

[4] H. Seo and O. Kwon, "CUDA implementation of McCann99 retinex algorithm," *5th International Conference on Computer Sciences and Convergence Information Technology*, 2010.

[5] Y.-K. Wang and W.-B. Huang, "Acceleration of the Retinex algorithm for image restoration by GPGPU/CUDA," *Parallel Processing for Imaging Applications*, 2011.

[6] Y.-K. Wang and W.-B. Huang, "Acceleration of an improved Retinex algorithm," *Cvpr 2011 Workshops*, 2011.

[7] Y.-K. Wang and W.-B. Huang, "A CUDA-enabled parallel algorithm for accelerating retinex," *Journal of Real-Time Image Processing*, vol. 9, no. 3, pp. 407–425, 2012.

[8] E. H. Land, "An alternative technique for the computation of the designator in the retinex theory of color vision.," *Proceedings of the National Academy of Sciences*, vol. 83, no. 10, pp. 3078–3080, Jan. 1986.

[9] D. Jobson, Z. Rahman, and G. Woodell, "Properties and performance of a center/surround retinex," *IEEE Transactions on Image Processing*, vol. 6, no. 3, pp. 451–462, 1997.

[10] D. Jobson, Z. Rahman, and G. Woodell, "A multiscale retinex for bridging the gap between color images and the human observation of scenes," *IEEE Transactions on Image Processing*, vol. 6, no. 7, pp. 965–976, 1997.

[11] D. J. Jobson, "Retinex processing for automatic image enhancement," *Journal of Electronic Imaging*, vol. 13, no. 1, p. 100, Jan. 2004.

[12] C. Solomon and T. Breckon, *Fundamentals of digital image processing: a practical approach with examples in Matlab*. Chichester: Wiley-Blackwell, 2012.

[13] A. Moore, J. Allman, and R. Goodman, "A real-time neural system for color constancy," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 237–247, 1991.

A New Method To Hide Encrypted Data In QR Code Using Modified Blowfish Algorithm

Kyaw Thant Hla [1], Myo Min Hein [2], Htin Kyaw [3]
Department of Computer Science [1,2,3],
Defence Services Academy Pyin Oo Lwin [1,2,3],
The Republic of The Union Of Myanmar [1,2,3]
kyawthanhla@gmail.com [1], myominhein@gmail.com [2], mgnyeinaye@gmail.com [3]

Abstract

Nowaday almost all the areas have had the QR codes. Therefore QR code needs to improve security. The transferring of data can be done using various technique and numerous type of algorithms. Blowfish algorithm is some problem in the existing algorithm weak key, generate bad S-boxes and high resources consumption. So, Blowfish algorithm is made up to prone plain text or brute force attack. Therefore to evaluate the enhancement of Blowfish by modifying the key generation and Feistel structure because strong key can be controlled by good S-boxes. Modified blowfish algorithm is useful in sharing the ciphertext form one user to another user with security feature. In this paper we present a technique that use QR Codes in the field of Cryptography. We focus to get strong messages in encrypted format and send it to the required destination hiding in a QR Code.

Keywords- Quick Response codes, AES algorithm, DES algorithm, Blowfish algorithm, Cryptography

1. Introduction

Nowadays, paper, voice and digital messages are still frequently used and exchanged in our daily life. To safely manage confidential information has increasingly become a challenge. The system integrates digital security system with QR Code technology, which brings in a developing idea for processing sensitive message in modern world. In the information age, although digital messages are frequently used, paper works are still widely used in multiple areas such as government, education, and business. In addition it has high accessibility, people can put it in the bag or pocket and print on the shirt or products, and take it everywhere without worrying about losing power or file corruption. While people pay great attention to the security of their digital messages, physical message security is missed [4].

Somdip Dey [1] proposed the SD-EQR method, where only text message encryption is shown. Unicode format is used for encrypting, so this method can be used to encrypt any type of the message or a file (picture, video, audio, etc.) and send it to the receiver safely. The QR code brings security to encrypted messages and receiver can access the original message much quickly, just by scanning QR code and decrypting it using software, which uses the above mentioned SDEQR algorithm. There are many techniques which gives the security to these codes as like by providing tracing codes, by authorization methods, by affixing the information, by providing benefits in digital education system, by using AES and Blowfish Algorithm, security against malwares and by embedding encrypting techniques etc [1].

2. Design of Proposed System

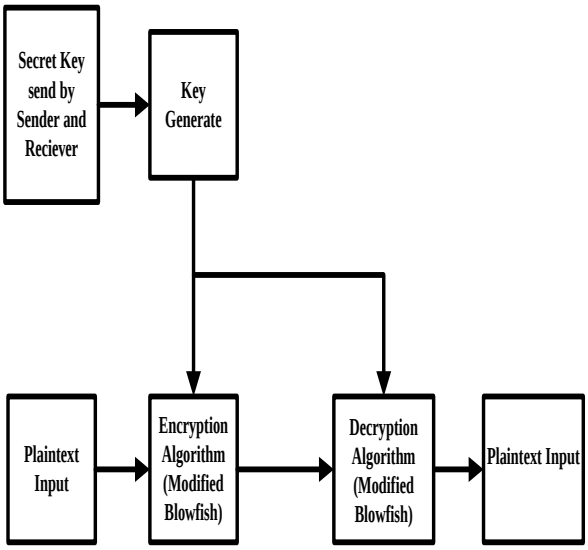

Figure 1. Overall Encryption and Decryption Process

In our proposed system, It takes text message as input and generates key of modified blowfish key generation process. After that getting the cipher text which is hidden in QR code and send to the receiver. In the receiver side, the process is reversed only by getting the clear text.

2.1. Proposed Architecture of the System

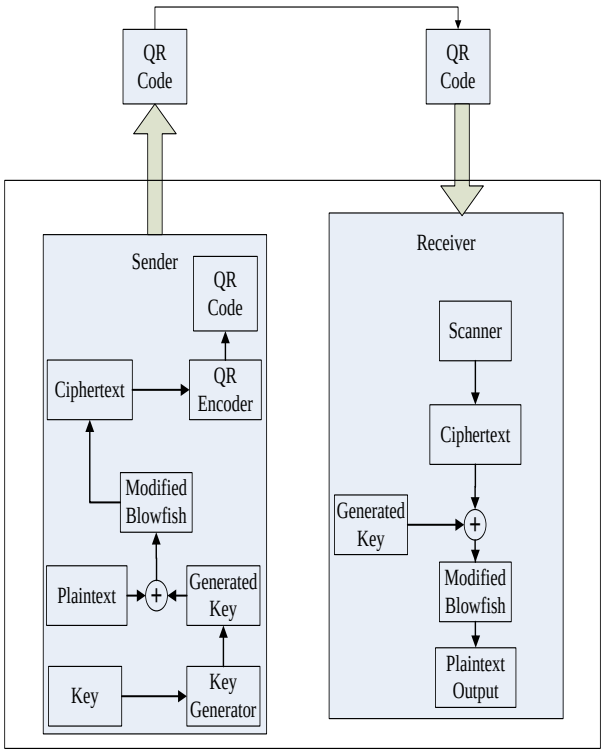

Figure 2. Proposed Architecture of the System

There are two main sites in the proposed system. They are sender site and receiver site. There are five main modules in the sender site such as input a plaintext, generate the key from modified key generation, converting above message into modified blowfish algorithm, getting the ciphertext and encoding into QR Code. After encoding QR code in which message was hidden is needed to send to the receiver. There are some processes in receiver sites. They are decoding QR Code, generate the key from modified key generation, getting ciphertext, decrypting by using modified blowfish algorithm and receiving original message. The main purpose of the system is to get integrate security into normal QR Code. Modified blowfish algorithm can also help message be secure.

2.2. Quick Response Code

QR code was invented by a Japanese company named Denso Wave in 1994. The QR code consists of an array of nominally square module arranged in an overall square pattern, and the black and white square module represent the digits one and zero, respectively. Fig. 3 shows the basic structure of the QR code, such as the version information, format information, data codewords, error correction codewords, position detecting patterns, alignment patterns, timing patterns and the quiet zone.

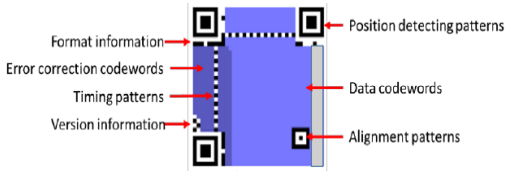

Figure 3. Quick Response Code Symbol

2.3. Modified Blowfish Algorithm

Modified Blowfish is included in 64-bit encryption block cipher with key length which varies between 32 bit and 448 bits. The feistel structure of Modified Blowfish algorithm with 16 rounds of encryption is shown in the following figure 3.3.

2.3.1. Modified Key-Generation. Functioning to change the lock (Minimum 32 bits, Maximum 448 bits) into multiple sub key arrays (sub key) totaling 4168 bytes.

(a) User inputs a random number ranging from numbers of 1 to n.

(b) This integer is converted into ACSII code values. The ACSII code values are merging individual. Finally, getting total sum is one integer value which values is changed from binary digit.

(c) If a changing integer values is not enough to 4 bits of our binary array then we need to be added to our array. After that we need to transform to our binary array to integer values.

(d) Getting the binary data then is converted back to integer, N.

(e) Finally, the key at the N^{th} Position in our keys generation is used in the algorithm for encryption and decryption of the message.

The modified algorithm is more secure and also efficient. The only fact known to the receiver and sender is the number entered by the user.

In this case can be calculated as:

Initial Stage

$$N(l) = \sum_{n=l-1}^{n=0} ASCII(K(n))$$

{ if l > 1 k =N go to initial

Else

[5] = Binary (N)

Temp = B [4];

B [0] = Temp;

Fk = Digit (B);

Where;

k = key array

l = length of key array

B = binary array

Fk = final key (index)

N = number of sum

2.3.2. Modified Feistel Network. In the modified feistel network, blowfish has 16 iterations, the input is a 64-bit data element, X. To perform the encryption process: There is a permutation box (P-box) consisting of 18 pieces of 32 bits sub key: P1, P2, P3, ..., P18.

This P-box has been set from the beginning, the first 4 P-boxes are as follows:

P1 = 0x243f6a88

P2 = 0x85a308d3

P3 = 0x13198a2e

P4 = 0x03707344 are respectively.

S-box forms of 4 pieces each worth 32 bits that have 256 inputs. Four 32-bit S-boxes each have 256 entries:

S1,0,S1,1,.....,S1,255

S2,0,S2,1,.....,S2,255

S3,0,S3,1,.....,S3,255

S4,0,S4,1,.....,S4,255 are respectively.

The Blowfish function F can be described as follows:-

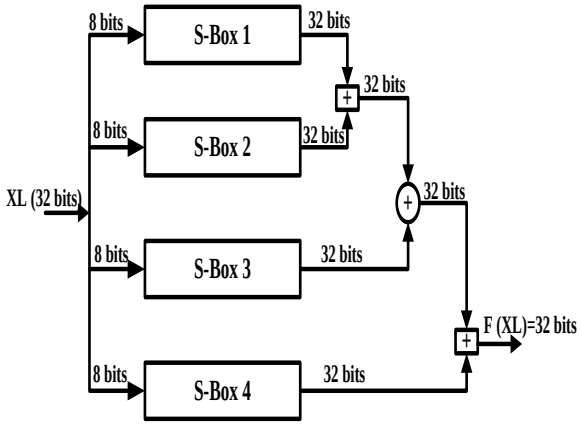

Figure 4. Blowfish Algorithm S-Box Process

The Modified network finds very efficient since the first to eight round use normal F-function and the nine to the sixteen rounds use Modified F-function. It splits the 64 bits block into two equal block having 32 bits size each, left block is XORed with first substitution array P1 and thus obtain result is fed into a function called F function. Inside the F function substitution operations are carried out which in turn convert 32 bit blocks into another 32 bit blocks. Thus resulted 32 bit entries are XORed with the right half and the result obtained is swapped as the left half for the next round. This change will further complicate encryption and decryption process for messages. The feistel structure of Modified Blowfish algorithm with 16 rounds of encryption is shown in the following figure 5.

Divide X into two parts, each consisting of 32bit:

XL and XR

For i = 1 to 8:

XL = XL XOR Pi

XR = F (XL) XOR XR

Switch XL and XR

For i = 9 to 16:

XL = XL XOR P9

XR = NMF (XL) XOR XR

Switch XL and XR

After the sixteenth iteration, exchange XL and XR again to undo the last account.

Finally, reassemble XL and XR to get the cipher texts.

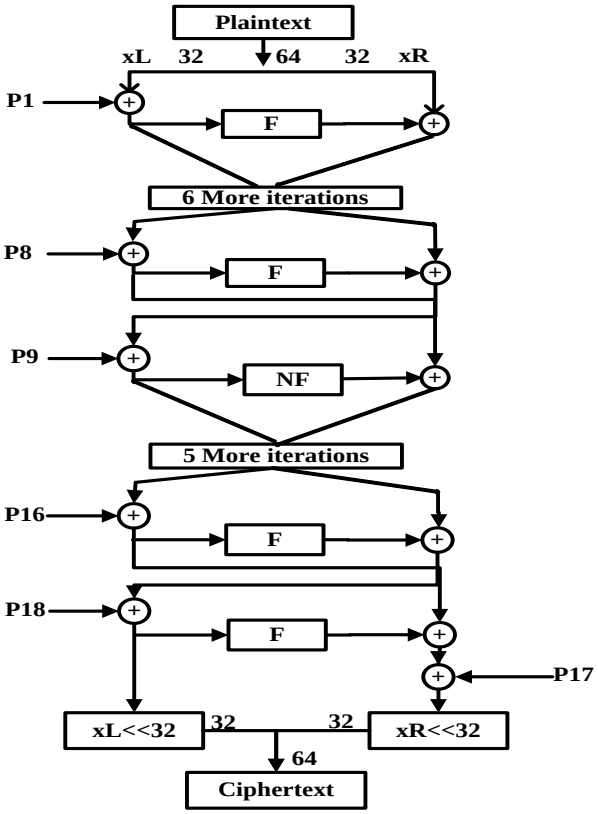

Figure 5. Blowfish Encryption Process

Decryption is exactly the same as encryption, except that P1, P2…P18 are use in the reverse order.

2.4 Research Procedure

Experimentation was done using different file sizes ranging from 1kb to 5kb. The average time is computed using 5 trials of each file size. During the experimentation, the researcher used Intel® Core ™ i5 CPU 6200-U @2.30 GHz with 4G RAM. The file and key used for all testing done were the same.

3. Implementation of Encoding and Decoding Process

The test includes two processes encryption process and decryption process. This experiment is based on C#, Excel and VBA.net platform that can change to plain text and ciphertext.

3.1. Encoding Process

Key pair generation is based on Modified blowfish algorithm. QR code (Quick response) is a two dimensional information storage tool. The Modified Blowfish Algorithm information is encrypted using the Private Key which is then encoded to QR Code as shown in the Fig.6.

Figure 6. Encoding Form

3.2. Decoding Process

The Modified Blowfish Algorithm information is decrypted using the Private key, which is then decoded to QR Code as shown in the Fig.7.

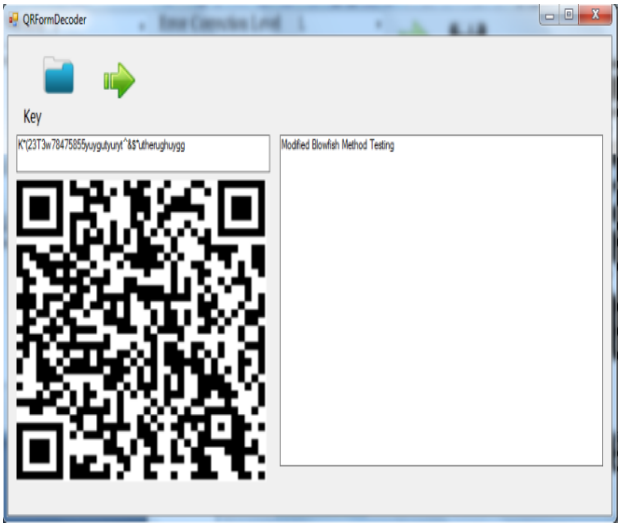

Figure 7. Decoding Form

4. Experimental Result

The experimental result of execution time is the summation of encryption time. The execution time and an average of it have been calculated.

Table 1. Encryption Execution Time

Data Type	Key Size	Text Size	One Round Encryption Time (Miliseconds)	
			Original Blowfish	Modified Blowfish
Char	32 bit	64 bit	0.9375	0.86
Number	32 bit	64 bit	1.148	1.07
Special Char	32 bit	64 bit	1.203	0.945
Combination	32 bit	64 bit	1.351	1.273
AVG			1.159875	1.037

The table 1 represents the five different type of data and corresponding encryption execution time taken by blowfish and Modified Blowfish algorithm in milliseconds. The results shows that the Proposed Blowfish is better depend on the average of execution time (Average of execution time is 1.037 milliseconds for Proposed Blowfish and 1.160 milliseconds for Original Blowfish).

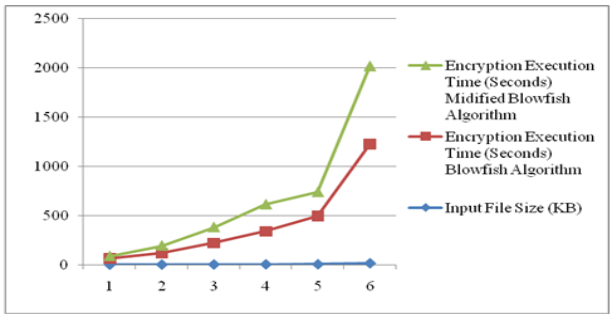

Figure 8. Encryption Throughput Time

The above figure displays the results of comparison between Original Blowfish and Modified Blowfish in term of performance throughput time. High throughput means less time for encryption process.

The execution time is the summation of decryption time. The execution time and an average of it have been calculated.

Table 2. Decryption Execution Time

Data Type	Key Size	Text Size	One Round Decryption Time (Miliseconds)	
			Original Blowfish	Modified Blowfish
Char	32 bit	64 bit	0.867	0.812
Number	32 bit	64 bit	1.453	1.171
Special Char	32 bit	64 bit	1.304	0.937
Combination	32 bit	64 bit	1.304	1.14
AVG			1.232	1.015

The results shows that the Proposed Blowfish is better depend on the average of execution time (Average of execution time is 1.015 milliseconds for Proposed Blowfish and 1.232 milliseconds for Original Blowfish).

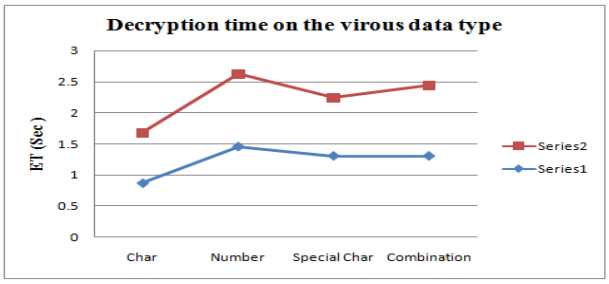

Figure 9. Decryption Throughput Time

The avalanche effect of the modified Blowfish is compared to Original Blowfish. The plaintext message used was "Mg Mg go to school" and varying 1 bit of the modified key. The following keys were used: DCBABCDEF10, DCBABCDEF11, DCBABCDEF12, DCBABCDEF13 and DCBABCDEF 14. Figure 4 shows the avalanche percentage. Blowfish has 40% avalanche, and the modified Blowfish is at 55 %. The higher the avalanche percentage, the higher will be the security, this means that the modified algorithm even had a better avalanche, thus better security.

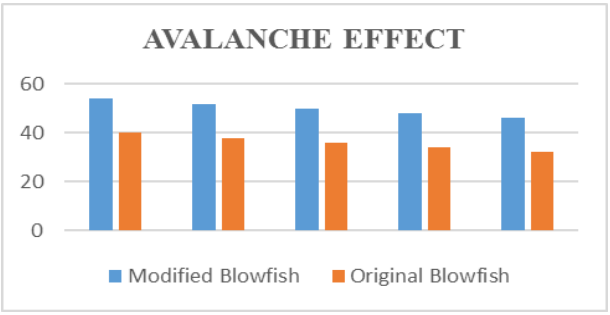

Figure 10. Avalanche Effect of the Modified Blowfish and Original Blowfish

There is no security in QR code because it was standardized. Security is added to the system. QR code security is based on modified blowfish algorithm (cipher method), encoding and decoding process. By changing the ciphertext, every user can make their own security. It can be organization use in messaging system. No one can decode hidden message without getting all these two parts (ciphertext and QR image). So, it can be said that security is integrated into QR Code.

5. Conclusion

In this paper, we proposed a scheme for information prevention using modified blowfish algorithm and QR code techniques. Since cipher text is used for encryption, this method can be used to encrypt any type of message or file (picture, video, audio, etc.) and send it to the receiver safely or the method can also be used to store important data or information safely. The inclusion of QR Code adds an extra level of security to the encrypted message and the receiver can access the original message very quickly, just by scanning the QR Code and decrypting it using a software, which uses the above mentioned modified blowfish algorithm. It provided better security for QR code. At the same time, the scheme ensured the safe storage of the information of the issuing authority.

6. References

[1] Anupam Baruah1, Prof.(Dr.)Lakshmi Prasad Saikia "Information security using Blowfish Algorithm in E-Banking", International Journal of Computer Science and Mobile Computing, Vol.6 Issue.3, March- 2017.

[2] B Ajay Ram "Distributed Secret Sharing Approach with Cheater Prevention based on QR Code" International Journal of Research in Advanced Computer Science Engineering, Vol.3, Issue 8, January2018.

[3] Basheer N. Ameen "A Novel Method for Ciphering a Message Based on QR Codes" International Journal of Scientific & Engineering Research, Volume 8, Issue 4, April-2017.

[4] Dini Davis " A Survey on Secret Data Hiding in Quick Response Barcodes", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 7, Issue 1, January 2017

[5] Harpreet Sandhu, Kamesh Dubey "Implementing Security in QR Codes by using Blowfish Algorithm" International Journal of Latest Engineering and Management Research (IJLEMR) ISSN: 2455-4847.

Roles of Intrusion Detection System and Machine Learning Techniques for Internet of Things (IOT)

Phyu Thi Htun[1], Dr. Kyaw Thet Khaing [2]
Lecturer [1], Deputy Director [2]
Department of Information Technology Engineering [1], e- Government and ICT Coordination, Moe Datacenter [2]
Technological University (Thanlyin) [1], Department of Technology Promotion and Coordination [2]
ms.phyuthihtun@gmail.com [1], drkyawthetkhaing@moe.edu.mm [2]

Abstract

Todays, there are many people who using internet with IoT devices in different purposes. According to the researcher's reports, the number of connected devices with network is expected to grow 50 billion by the year 2020. Every person would use device at least 6 to maximum 24 devices. Emerging IoT Technologies are changing the ICT with more speedy development, nowadays. As long as the progressive of internet usages on IoT, the attacks using with those IoT devices also increased. IoT infrastructure's security will be a critical point and it will be deployed in. And there is also important that to detect those network attacks in real time become more and more difficult whether it can be secured with encryption or authentication. The Network Intrusion Detection System can be used Machine Learning Theory with anomaly based intrusions and attacks detection. Using those algorithms, the system will try to detect the anomaly IoT Network attack. Experimental results examine that those algorithms can be classified IoT Network attacks significantly with the HCRL, IoT Network Intrusion Detection Datasets.

Keywords- Internet of Things, Intrusion Detection System, Security, Machine Learning, IoT Network Intrusion Detection Datasets.

1. Introduction

With the rapid development of Internet technology and communications technology, our lives are gradually led into an imaginary space of virtual world. People can chat, work, shopping, keeps pets and plants in the virtual world provided by the network. However, human beings live in a real world; human activities cannot be fully implemented through the services in the imaginary space. It is the limitation of imaginary space that restricts the development of Internet to provide better services. To remove these constraints, a new technology is required to integrate imaginary space and real-world on a same platform which is called as Internet of Things (IoT).

What can I do with IoT? [3] IoT lets you solve your business problems using your own data. The Internet of Things isn't just about connected devices-it's about information and how connecting devices to the cloud provides powerful, immediate insights that can transform your business by lowering costs by reducing loss or materials, improving operational and mechanical processes, or building new lines of business that are possible with reliable real-time data. Create a real competitive advantage by using IoT to turn your data into insights, and those insights into action.

Internets of Things (IoT) facilitate integration between the physical world and computer communication networks, and applications (apps) such as infrastructure management and environ mental monitoring make privacy and security techniques critical for future IoT systems. Consisting of radio frequency identifications (RFIDs), wireless sensor networks (WSNs), and cloud computing [4], IoT systems have to protect data privacy and address security issues such as spoofing attacks, intrusions, denial of service (DoS) attacks, distributed denial of service (DDoS) attacks, jamming, eavesdropping, and malwares [6].

The Internet of Things is increasingly becoming a ubiquitous computing service, requiring huge volumes of data storage and processing. Unfortunately, due to the unique characteristics of resource constraints, self-organization, and short range communication in IoT, it always resorts to the cloud for outsourced storage and computation, which has brought about a series of new challenging security and privacy threats. In this article, we introduce the architecture and unique security and privacy requirements for the next generation mobile technologies on cloud-based IoT, identify the inappropriateness of most existing work, and address the challenging issues of secure packet forwarding and efficient privacy preserving authentication by proposing new efficient privacy preserving data aggregation without public key holomorphic encryption. Finally, several interesting open problems are suggested with promising ideas to trigger more research efforts in this emerging area .It's generally prohibitive for IoT devices with restricted computation, memory, radio band width, and battery resource to

execute computational-intensive and latency-sensitive security tasks especially under heavy data streams [7].

IoT is an emerging technology that has attracted a considerable number of researchers from all around the world. There have been major contributions making this technology adapted into our daily life. It covers many fields including healthcare, automobiles, entertainments, industrial appliances, sports, homes, etc. The pervasiveness of IoT eases some everyday activities, enriches the way people interact with the environment and surroundings, and augments our social interactions with other people and objects [1].

Based on a large number of low-cost sensors and wireless communication, the sensor network technology puts forward new demands to the Internet technology. It will bring huge changes to the future society, change our way of life and business models.[2] Internet of Things (IoT) facilitate integration between the physical world and computer communication networks, and applications (apps) such as infrastructure management and environ mental monitoring make privacy and security techniques critical for future IoT systems. Consisting of radio frequency identifications (RFIDs), wireless sensor networks (WSNs), and cloud computing, IoT systems have to protect data privacy and address security issues such as spoofing attacks, intrusions, denial of service (DoS) attacks, distributed denial of service (DDoS) attacks, jamming, eavesdropping, and malwares [6].

This Intrusion Detection Systems (IDSs) have become a major focus of computer scientists and practitioners as computer attacks have become an increasing threat to commercial business as well as our daily lives. Researchers have developed two main approaches for intrusion detection: misuse and anomaly intrusion detection. Misuse consists of representing the specific patterns of intrusions that exploit known system vulnerabilities or violate system security policies.

On the other side, anomaly detection assumes that all intrusive activities are necessarily anomalous. This means that if we could establish a normal activity profile for a system, we could, in theory, flag all system states varying from the established profile as intrusion attempts. These two kinds of systems have their own strengths and weaknesses.

The former can detect known attacks with a very high accuracy via pattern matching on known signatures, but cannot detect novel attacks because their signatures are not yet available for pattern matching. The latter can detect novel attacks but in general for most such existing systems, have a high false alarm rate because it is difficult to generate practical normal behaviour profiles for protected systems. IoT devices can apply supervised learning techniques to evaluate the runtime behaviours of the apps in the malware detection. In the malware detection scheme as developed in [5], an IoT device uses K-NN and random forest classifiers to build the malware detection model. As illustrated in Fig. 5, the IoT device filters the TCP packets and selects the features among various network features including the frame number and length, labels them and stores these features in the database. The K-NN based malware detection assigns the network traffic to the class with the largest number of objects among its K nearest neighbors. The random forest classifier builds the decision trees with the labelled network traffic to distinguish malwares. According to the experiments in [22], the true positive rate of the K-NN based malware detection and random forest based scheme with dataset are 99.7% and 99.9%, respectively.

IoT devices can offload app traces to the security servers at the cloud or edge devices to detect malwares with larger malware database, faster computation speed, larger memories, and more powerful security services. The optimal proportion of the apps traces to offload depends on the radio channel state to each edge device and the amount of the generated app traces. RL techniques can be applied for an IoT device to achieve the optimal offloading policy in a dynamic malware detection game without being aware of the malware model and the app generation model [11].

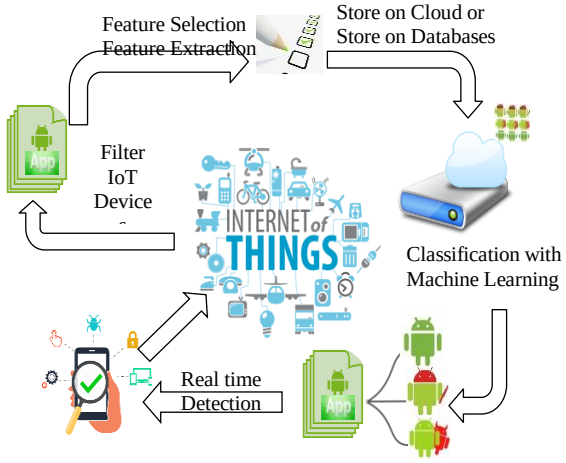

Figure 1. Illustration of the ML-based Malware Detection.

In this paper, we only consider to analyse anomaly detection systems for IoT devices, extend the definition of anomaly detection to not only take into account normal profiles but also handle known attacks and explore supervised machine learning techniques. These techniques have proven their efficiency in predicting the different classes of the unlabelled data in the test data set for the HCRL IoT Network intrusion detection contest for IoT devices.

The rest of the paper is organized as follows. Section 2 presents the related works corresponding with the analysis of Internet of Things (IoT) and the machine learning Algorithms for AIDS. Section 3 described

methodology with system overview and description on HCRL IoT dataset. Section 4 introduces machine-learning techniques, Random Forests and k-Nearest Neighbor, and presented the proposed method RFKNN. Using those machine learning algorithms in our proposed system, which also presented in Section 4, Section 5 describes the experimental results obtained by using the machine-learning algorithms with WEKA tool [13]. Section 6 concluded for our research by using the out coming results using with those machine learning algorithms.

2. Related Work

Samir Fenanir, F. Semchedine and A. Baadache [8] suggest that according to the results of their study, it is observed that DT and KNN performed better than the other algorithms; however, the KNN takes much time to classify compared to the DT algorithm. Furthermore, with the three correlation methods used to reduce datasets dimension such as PCC, SCC and KTC, the classifiers produce good performance when the threshold of the correlation coefficient is greater than 0.9; below this threshold, performances are poor and sometimes unacceptable.

Raza et al. [9] implemented a real-time IDS for the IoT called SVELTE. This system consists of a 6LoWPAN Mapper (6Mapper), intrusion detection module, and a mini firewall. It analyses the mapped data to identify any intrusions in the network. Its performance in detecting various attacks seems promising. However, it has only been tested to detect spoofed or altered information, sinkhole, and selective-forwarding attacks.

Summerville, Zach, and Chen [10] developed an IDS for IoT based on a deep packet analysis approach which employs a bit-pattern technique. The network payloads are treated as a sequence of bytes called bit-pattern, and the feature selection operates as an overlapping tuple of bytes called n-grams. When the corresponding bits match all positions, a match between the bit-pattern and n-grams occurs. The system is evaluated by deploying four attacks and demonstrates a very low false-positive rate.

Amouri, Alaparthy, and Morgera [12] developed an IDS for IoT networks by applying supervised machine learning. The IDS attempts to profile the benign behaviour of the nodes and identify any anomalies on the network traffic. The results demonstrate that the system is able to successfully distinguish benign and malicious nodes.

Shukla [14] proposed an IDS that uses a combination of machine learning algorithms such as K-means and decision tree, to detect wormhole attacks on 6LoWPAN IoT networks. Nonetheless, results of this work are promising, the evaluation of the proposed IDS was based on a simulation and the effectiveness of the IDS has not been tested against other attacks.

Meidan et al.[20] and McDermott et al. [21] both focus on the detection of botnets in the IoT ecosystem and employ deep learning techniques to achieve this. The results in both cases are promising as they can successfully detect the botnets; however, these methods have not been deployed to detect a range of attacks and have been evaluated in a simulated environment.

Sheikh Tahir Bakhsh , Saleh Alghamdi, Rayan A Alsemmeari1 and Syed Raheel Hassan [25] suggest to implement and install intrusion detection and prevention system to keep IoT safe. IDSs can be categories into three types: signature-based, anomaly-based, and hybrid. In addition, IDS and IPS can be deployed as network-based, host-based, or hybrid-based. The proposed system provides a solution for intrusion detection to cover IoT security aspects. The proposed IDPIoT receives packets from the network interface and decodes the packets for processing to deliver to the detector agent. The detector agent checks each packet header for a certain type of behaviour to detect any anomalies in the packet header. The system analyser compares packet against pre-defined detection rules, such as matching the logging and alerting system is activated. It sounds alarms, log messages, and sends them to the output module. The system saves the output data and alert system to a pre-configured destination such as a log file or a database. Moreover, prevention agents drop the suspicious packet and block the source by providing real-time mitigation of attacks and isolation of the servers.

Table 1. Summary of Current Work on Intrusion Detection Systems for Internet of Things

Work	Security Threat	Detection Method
R.Stephen [32]	Hello Flood/ Sybil	Packet Metrics
S. Raza [33]	Sinkhole & Selective forwarding	Hybrid
D. Shreenivas[34]	Routing attacks against RPL protocol	Hybrid
P. Pongle [35]	Wormhole	Anomaly -based
D. H. Summerville [36]	Worm propagation, SQL code injection, and directory traversal	Anomaly -based
Midi [37]	ICMP flood, Replication, Smurf	Hybrid
McDermott [21]	Botnets	Machine Learning
Meidan [20]	Botnets	Machine Learning
Brun [38]	UDP Flood, TCP SYN, Sleep Deprivation Attack, Barrage Attack, and Broadcast Attack	Deep Learning

Zeeshan AliKhan and PeterHerrmann [26] introduced work for the three similar network types Mobile Ad hoc Networks (MANET), Wireless Sensor Networks (WSN), and Cyber-Physical Systems (CPS) and discuss if they are also suitable for IoT systems.

Several approaches have used data from network simulations or have evaluated the system on a small array of IoT devices, which may significantly decline from a realistic environment. Additionally, such approaches focus on detecting whether specific cyber-attacks have occurred, i.e. whether packets are malicious or benign, and not classify the type of attack. This is an important feature of IDS, as specific countermeasures can be employed for specific attack type [27].

3. Methodology

3.1. System Overview

Hacking and Countermeasure Research Lab (HCRL) created various types of network attacks in Internet of Things (IoT) environment for academic purpose. Two typical smart home devices -- SKT NUGU (NU 100) and EZVIZ Wi-Fi Camera (C2C Mini O Plus 1080P) -- were used. All devices, including some laptops or smart phones, were connected to the same wireless network. The dataset consists of 42 raw network packet files (pcap) at different time points.

* The packet files are captured by using monitor mode of wireless network adapter. The wireless headers are removed by Aircrack-ng.

* All attacks except Mirai Botnet category are the packets captured while simulating attacks using tools such as Nmap. The case of Mirai Botnet category, the attack packets were generated on a laptop and then manipulated to make it appear as if it originated from the IoT device.

3.2. Rules for Filters

In the dataset, the developer used Wireshark Rule to Filter Only Attack Packets for:

Man-In-the_Middle as:

eth.addr == f0:18:98:5e:ff:9f and
((((ip.src == 192.168.0.16 and ip.dst == 192.168.0.13)
or (ip.src == 192.168.0.13 and ip.dst == 192.168.0.16))
and !icmp and tcp)
or (arp.src.hw_mac == f0:18:98:5e:ff:9f
and (arp.dst.hw_mac == bc:1c:81:4b:ae:ba or arp.dst.hw_mac == 48:4b:aa:2c:d8:f9)))

Scanning as :

ip.src == 192.168.0.15 and ip.dst == 192.168.0.24 and ((tcp.flags.syn == 1 and tcp.window_size == 1024) or tcp.flags.reset == 1)

Mirai Botnet as: ip.dst == 210.89.164.90 and

Denial of Service (DoS) as :

ip.src == 222.0.0.0/8 and tcp.flags.syn == 1 and ip.dst == 192.168.0.13 and tcp.dstport == 554 and tcp

The description for the data in each files are can be learned at [27].

Table 2. Summary of Datasets

Category	Sub-category	# of Packets
Normal	Normal	1,756,276
Scanning	Host Discovery	2,454
Scanning	Port Scanning	20,939
Scanning	OS/Version Detection	1,817
Man in the Middle (MITM)	ARP Spoofing	101,885
Denial of Service (DoS)	SYN Flooding	64,646
Mirai Botnet	Host Discovery	673
Mirai Botnet	Telnet Bruteforce	1,924
Mirai Botnet	UDP Flooding	949,284
Mirai Botnet	ACK Flooding	75,632
Mirai Botnet	HTTP Flooding	10,464

3.3. Feature Selection

The main requirements to consider when developing a machine learning based IDS for IoT are:

- Lightweight - not require considerable computational processing power.
- Stand-alone-not dependent on any other software or alert based system.
- Fast - malicious activity must be detected in almost real time to reduce impact.
- To work over encrypted traffic - most commercial IoT devices employ transport encryption.

Given the above requirements, it was decided to initially investigate whether it is possible to detect malicious behavior from single packets. The reasoning behind this approach is that, as single packets are the smallest piece of network information, they are quicker to processes, and subsequently improve the speed of identifying malicious activity. The raw PCAP files containing the network packets were initially converted and represented in a Packet Description Markup Language (PDML) [29] format. PDML conforms to the XML standard and contains details about the packet dissection/layers. As a result, it allows access to all the packet attributes that can be used as features. A network packet consists of a series of layers (Physical, Data Link, Network, Transport, and Application), each layer being a child of the previous layer, built from the lowest layer up [30]. Each layer has its own header composed of a range of different fields providing

information, and a payload. For the classification experiments discussed in this work, all the fields that compose each of the aforementioned layers were extracted, in order to investigate which ones are most relevant in detecting benign and malicious behaviour on IoT. In addition to these attributes, few more fields were also included such as: frame information [31] and packet type - which specifies whether the data packet was inbound or outbound to an IoT device on the tested. Additionally, features that represented identifying properties were removed (e.g. source IP address, time, packet ID) to ensure the model was not dependent on specific network configurations and that the features of the network behaviour were captured, rather than the network actors and devices. Finally, because the network traffic is encrypted, the payload information from the Application Layer was not considered as a feature. In total, 121 features were extracted from each packet and represented as a feature vector.

4. Machine Learning Algorithms

To overcome the limitations of the rule-based systems, a number of IDSs employ data mining techniques. Data mining is the analysis of (often large) observational data sets to find patterns or models that are both understandable and useful to the data owner [15][19]. Data mining can efficiently extract patterns of intrusions for misuse detection, establish profiles of normal network activities for anomaly detection, and build classifiers to detect attacks, especially for the vast amount of audit data. Data mining-based systems are more flexible and deployable. Over the past several years, a growing number of research projects have applied data mining to intrusion detection with different algorithms. We propose an approach to use random forests and k-Nearest Neighbor in intrusion detection. For instance, those had been applied to prediction, probability estimation, and pattern analysis in multimedia information retrieval and bioinformatics. Unfortunately, to the best of our knowledge, Random Forests algorithm has not been completely applied to detect novel attacks (unknown attacks) in automatic intrusion detection. Fortunately, we can take advantages from k-NN that can classify in more precisely and an important pattern recognizing method based on representative points [22].

4.1. Random Forests (RDF)

The Random Forests [11] is an ensemble of unpruned classification or regression trees. Random forest generates many classification trees. Each tree is constructed by a different bootstrap sample from the original data using a tree classification algorithm. After the forest is formed, a new object that needs to be classified is put down each of the tree in the forest for classification. Each tree gives a vote that indicates the tree's decision about the class of the object. The forest chooses the class with the most votes for the object.

The main features of the random forests algorithm are listed as follows:

• It runs efficiently on large data sets with many features.
• It can give the estimates of what features are important.
• It has no nominal data problem and does not over-fit.
• It can handle unbalanced data sets.

4.2. k-NN: k-Nearest Neighbor

k-NN classification is an easy to understand and easy to implement classification technique[18]. Despite its simplicity, it can perform well in many situations. k-NN is particularly well suited for multi-modal classes as well as applications in which an object can have many class labels. For example, for the assignment of functions to genes based on expression profiles, some researchers found that k-NN outperformed SVM, which is a much more sophisticated classification scheme. The 1-Nearest Neighbor (1NN) classifier is an important pattern recognizing method based on representative points [19].

In the 1NN algorithm, whole train samples are taken as representative points and the distances from the test samples to each representative point are computed. The test samples have the same class label as the representative point nearest to them. The k-NN is an extension of 1NN, which determines the test samples through finding the k nearest neighbors.

4.3. RFKNN

Although the processing time in detection was long, the k-Nearest Neighbor is also the good modeling algorithm in our experiments. The reason that the Random Forest cannot consider on pattern recognition, and also k-NN is a good pattern recognition method which used in many researches [17] [18].

Thus, we can extend this experiment by combining those two algorithms as RFKNN; the system can get the more accurate and detection rate to detected intrusion. Random Forest will process in the filtering stage and the k-NN will use as a classifier. On the other hand, using virtualized environment prevents us from launching in-depth attacks against an IoTs firmware and hardware, thus somewhat limiting the depicted ways through which an attack against such machines can be launched. Nevertheless, the collected dataset is adequate for our purposes.

4.4. Intrusion Detection System (IDS)

The system methodology is depicted in Figure 2. We have generated data by real-life equivalent simulations using the open source Contiki/ Cooja simulator16, because of lack of availability of public IoT attack datasets. The Cooja simulation generates raw packet capture (PCAP) files, which are first converted in to Comma Separated Values (CSV) files for text-based processing. The CSV files are then fed into the feature pre-processing module of our system. The features are calculated based on the traffic flow information in the CSV files. First, a feature extraction process takes place. Thereafter, feature normalization is applied to all datasets to reduce the negative effects of marginal values. In the pre-feature selection step, as a result of feature importance analysis, some of the features are dropped. After feature pre-processing, the datasets corresponding to each scenario are labeled and mixed. As a result, preprocessed dataset is produced, consisting of a mixture of attack and benign data. These datasets are fed into deep learning algorithm. Deep layers are trained with regularization and dropout mechanisms, their weights are adjusted and the IoT Attack Detection Models are created [39].

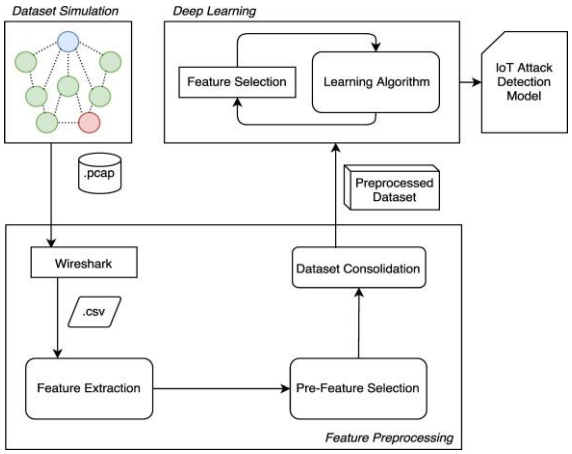

Figure 2. System Flow of the IDS

5. Experimental Results

To explore how well classification algorithms can learn to profile IoT devices on the network, detect wireless attacks, and classify the type of such attacks, the performance of supervised machine learning when the corresponding network activity data was used to train and evaluate the classification model. In the case of identifying whether a packet is malicious or benign, classification is evaluated relative to the training dataset, producing four outputs:

- true positives (TP) - packets are predicted as being malicious, when they are indeed malicious.
- true negatives (TN) - packets are predicted as being benign, when they are indeed benign.
- false positives (FP) - packets are predicted as being malicious, when in fact, they are benign.
- false negatives (FN) - packets are predicted as being benign, when in fact, they are malicious.

There are several measures which can be used to evaluate the performance of a classifier. The goal is to maximize all measures, which range from 0 to 1. Therefore, higher values correspond to better classification performance. The most common measures are precision, recall, F-measure, and accuracy. Precision (P) measures the proportion of malicious packet identifications was correct, whereas recall (R) measures what proportion of malicious packets were identified correctly. The two measures are often used together in F-measure (F), which calculates the harmonic mean of precision and recall, and provides a single weighted metric to evaluate the overall classification performance. Such measures are calculated using equations in the following equations:

$$P=(TP/TP+FP) \quad (1)$$

$$R=(TP/TP+FN) \quad (2)$$

$$F=(2.(P.R/P+R)) \quad (3)$$

Summarize our experimental results to detect the category of the attacks for intrusion detection with over the IoT Network Intrusion datasets [28] whether normal, scanning, Man in the Middle (MITM), Denial of Services (DoS) and Mirai Botnet.

Firstly, the system will extracted the features from the packets and also reduced some features in dataset by using Random Forest algorithm at each connection according to their ranking.

So, system will try to detect various attacks using corrected dataset. The system will reduced in training time and will increase the accuracy of the system's classification. The experimental results will come out by using WEKA tool [13].

In the experiments process, the system used 10 trees and the reduced features (default 6 in WEKA) to classify. The accuracy of the system will be increased other systems. The datasets have with different statistical distributions, the accuracy decrease rather than Cross Validation results with those train files. But as to detect the unknown attack, the results in test file that contains more unknown attack types (novel attacks) than the other datasets get more detection rate of Random Forest can compare with other methods as shown in figure 3.

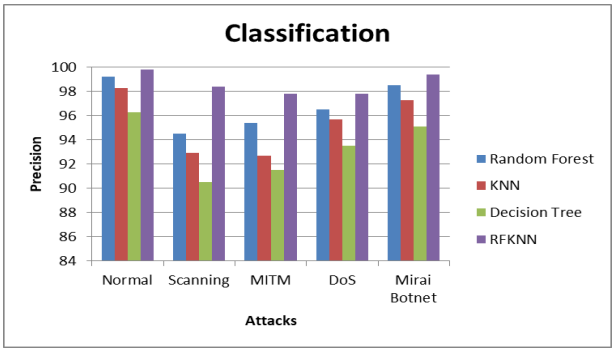

Figure 3. The Classification Comparisons Between Machine Learning Algorithm Random Forest, k-NN, Decision Tree and RFKNN

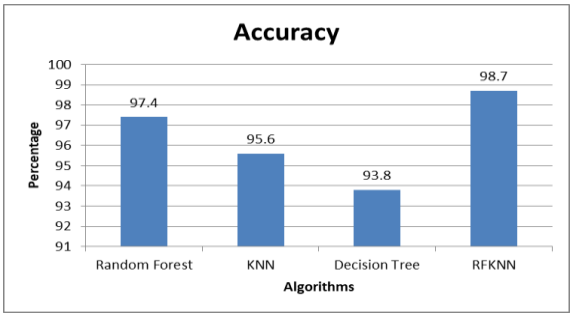

Figure 4. The Comparison Results Between Machine Learning Algorithm Random Forest, k-NN, Decision Tree and RFKNN.

6. Conclusion and Further Extension

There are many recent researchers' report concerning detection and false alarm rates in decision trees, artificial neural networks and a probabilistic classifier. But it was still high false positives and irrelevant alerts in the field of detection attacks. This paper has presented a survey of the various data mining techniques that have been proposed towards the enhancement of anomaly intrusion detection systems. And, we applied the classification methods on HCRL IoT Network Intrusion dataset for classifying the attacks (intrusions). The results showing the performance of the Random Forest is better than other classifiers. But the time taken is more for Random Forest than other classifiers.

In addition, as an expansion, an even more diverse group of attacks could be performed, such as application layer DoS/DDoS, Layer 1,2 attacks on physical IoT devices, something which requires the use of real IoT devices, access to which we did not have at the time of the experiments[24].

As the future extension, there is proposing for a new dataset, Bot-IoT, which incorporates legitimate and simulated IoT network, along with various types of attacks. We also hope to present a realistic tested environment for addressing the new dataset in capturing of complete network information, and to label accurate, as well as complex attack diversity

7. References

[1] A. Kliarsky, "Detecting Attacks Against The Internet of Things", SANS Institute 2017.
[2] A Servey On IOT_Security&PrivacyIssues.pdf
[3] What is IOT?
https://azure.microsoft.com/en-us/overview/internet-of-things-iot/what-is-the-internet-of-things/
[4] S.L.Thomas, "Backdoor Detection Systems for Embedded Devices" Thesis, University of Birmingham April 2018
[5] T.Sherasiya, H. Upadhyay and H.B.Patel,"A Survey: Intrusion Detection System For Internet Of Things", International Journal of Computer Science and Engineering (IJCSE),ISSN(P): 2278-9960; ISSN(E): 2278-9979 Vol. 5, Issue 2, Feb - Mar 2016, 91-98
[6] L. Xiao, X. Wan, X.Lu,Y. Zhang, and D. Wu,"IoT Security Techniques Based on Machine Learning", 19 Jan 2018.https://arxiv.org/pdf/1801.06275.pdf
[7] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," Soft Computing, vol. 20, no. 1, pp. 343–357, Jan. 2016
[8] Samir Fenanir, Fouzi Semchedine and Abderrahmane Baadache ,"A Machine Learning-Based Lightweight Intrusion Detection System for the Internet of Things", Revue, d'Intelligence Artificielle, Vol 33, No 3, June 2019. pp. 203-211
http://iieta.org/journals/ria
[9] Shahid Raza, Linus Wallgren, and Thiemo Voigt. Svelte: Real-time intrusion detection in the internet of things. Ad hoc networks, 11(8):2661– 2674, 2013.
[10] L. Santos, C. Rabadao, and R. Goncalves. "Intrusion detection systems in internet of things: A literature review.", In 2018 13th Iberian Conference on Information Systems and Technologies (CISTI). IEEE, 2018.
[11] L. Breiman, "Random Forests", Machine Learning 45(1):5–32, 2001.
[12] A. Amouri, V. T. Alaparthy, and S. D. Morgera. Cross layer-based intrusion detection based on network behavior for iot. In Wireless and Microwave Technology Conference (WAMICON), 2018 IEEE 19th, pages 1–4. IEEE, 2018.
[13] WEKA software, data mining with open source machine learning software in java.
https://www.cs.waikato.ac.nz/ml/weka/.
[14] P. Shukla. "Ml-ids: A machine learning approach to detect wormhole attacks in internet of things." In Intelligent Systems Conference (IntelliSys), 2017, pages 234–240. IEEE, 2017.
[15] J. H. David and, H. Mannila, and P. Smyth, "Principles of Data Mining", The MIT Press, August, 2001.
[16] J.Zhange and M. Zulkerline, "Network Intrusion Detection using Random Forests", 2011.

[17] J. Zhang and M. Zulkernine, ”Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection”, Symposium on Network Security and Information Assurance Proc. of the IEEE International Conference on Communications (ICC), 6 pages, Istanbul, Turkey, June 2006.
[18] S. Thirumuruganathan , “A Detailed Introduction to K-Nearest Neighbor (KNN) Algorithm”, World Press, May 17, 2010.
[19] X Wu, V Kumar, J Ross Quinlan, J Ghosh, “Top 10 Data mining Algorithm”, Knowledge and Information Systems, Volume 14, Issue 1, pp 1-37 ,2008 – Springer.
[20] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici. N-baiot network based detection of iot botnet attacks using deep auto encoders. IEEE Pervasive Computing, 17(3):12–22, 2018.
[21] C. D. McDermott, F. Majdani, and A. V. Petrovski. “Botnet detection in the internet of things using deep learning approaches.” In 2018 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2018.
[22] K.T.Khaing and T.T.Naing, “Enhanced Feature Ranking and Selection using Recurisive Featue Elemination and k-Nearest Neighbor Algorithms in SVM for IDS”, Internaiton Journal of Network and Mobile Technology(IJNMT), No.1, Vol 1. 2010.
[23] P.T.Htun and K.T.Khaing,” Enhanced Attack Classification Model For Network Intrusion Detection System”, Thesis, 2014
[24] N. Koroniotisa, N. Moustafaa, E. Sitnikovaa, and B. Turnbulla , “Towards the Development of Realistic Botnet Dataset in the Internet of Things for Network Forensic Analytics: Bot-IoT Dataset.” arXiv:1811.00701v1, 2 Nov 2018 https://arxiv.org/pdf/1811.00701.pdf
[25] S. T. Bakhsh , S. Alghamdi, R. A. Alsemmeari1 and S. R. Hassan, “An adaptive intrusion detection and prevention system for Internet of Thing”, International Journal of Distributed Sensor Networks 2019, Vol. 15(11) The Author(s) 2019 DOI: 10.1177/1550147719888109
http://journals.sagepub.com/home/dsn
[26] Z. AliKhan and P. Herrmann,” Recent Advancements in Intrusion Detection Systems for the Internet of Things”, Hindawi Security and Communication Networks Volume 2019, Article ID 4301409, 19 pages
https://doi.org/10.1155/2019/4301409
[27] E. Anthi, L. Williams, M. Słowi´nska, G. Theodorakopoulos and P. Burnap,“A Supervised Intrusion Detection System for Smart Home IoT Devices”.
https://ieeexplore.ieee.org/document/8753563
[28] H. Kang, D. H. Ahn, G. M. Lee, J. D. Yoo, K. H. Park, and H. K. Kim, "IoT network intrusion dataset", IEEE Dataport, 2019. [Online]. Available: http://dx.doi.org/10.21227/q70p-q449.
[29] Pdml - the wireshark wiki. https://wiki.wireshark.org/PDML.
[30] Scapy p.04 looking at packets — the packet geek. https://thepacketgeek. com/scapy-p-04-looking-at-packets/.
[31] Wireshark go deep. https://www.wireshark.org/.
[32] R. Stephen and L. Arockiam,”Intrusion detection system to detect sinkhole attack on rpl protocol in internet of things. International Journal of Electrical Electronics and Computer Science, 4(4):16–20, 2017.
[33] S. Raza, L. Wallgren, and T. Voigt. Svelte: Real-time intrusion detection in the internet of things. Ad hoc networks, 11(8):2661– 2674, 2013.
[34] D. Shreenivas, S. Raza, and T. Voigt. Intrusion detection in the rpl-connected 6lowpan networks. In Proceedings of the 3rd ACM International Workshop on IoT Privacy, Trust, and Security, pages 31–38. ACM, 2017.
[35] P. Pongle and G. Chavan, “Real time intrusion and wormhole attack detection in internet of things.” International Journal of Computer Applications, 121(9), 2015.
[36] D. H. Summerville, K. M Zach, and Y. Chen., “Ultra lightweight deep packet anomaly detection for internet of things devices.” In Computing and Communications Conference (IPCCC), 2015 IEEE 34th International Performance, pages 1–8. IEEE, 2015.
[37] D. Midi, A. Rullo, An and Mudgerikar, and E. Bertino, “Kalisa system for knowledge-driven adaptable intrusion detection for the internet of things.” In Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on, pages 656–666., 2017.
[38] F. Restuccia, S. DOro, and T. Melodia. “Securing the internet of things in the age of machine learning and software-defined networking.” IEEE Internet of Things Journal, 5(6):4829–4842, 2018.
[39] F. Y. YAVUZ, D. ÜNAL E. GÜL,” Deep Learning for Detection of Routing Attacks in the Internet of Things,”, International Journal of Computational Intelligence Systems, Vol. 12 (2018) 39-58