International Journal of Engineering & Technology, 7 (4.11) (2018) 257-261

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Securing RF Communication Using AES-256 Symmetric Encryption: A Performance Evaluation

A. Jamaluddin*, N. N. Mohamed, H. Hashim

Faculty of Electrical Engineering, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia
**Corresponding author E-mail: ardaniaahh@gmail.com*

Abstract

Radio Frequency (RF) communication plays a vital role for sensor node data transmission, which typically runs on top of lightweight protocol such as Constrained Application (CoAP) and Trivial File Transfer Protocol (TFTP). Introducing a cryptographic scheme to the process is known to be the common and most efficient method to protect RF communication. This paper presents the performance comparison of AES-encrypted data transmission over RF communication via Raspberry Pi boards, experimented on a client-server architecture. The performance analysis is measured based on throughput metric and the transmission time delay when sending three types of payload which are, a plaintext data, a ciphertext with padding and a ciphertext without padding. The result from the study indicates that there is a significant difference in data transmission time between the three types of data due to the data expansion factor. The result also showed that adding padding to the ciphertext has increased the data size slightly but not significant enough to affect transmission time of ciphertext.

Keywords: *cryptography; symmetric encryption; Advanced Encryption Standard; Raspberry Pi; RF Communication; Slice of Radio*.

1. Introduction

Today, internet is migrating from connecting people to connecting things, leading to the new concept of Internet of Things (IoT). One of the most efficient wireless communication for IoT sensor nodes is via Radio Frequency (RF), typically run on top of CoAP and TFTP protocols which provide zero security mechanism [1]. A popular method to protect the sensor node data transmission over RF in a more secure way is by implementing the cryptographic algorithms [2]. Cryptographic algorithms are ranked by their speed in encrypting/decrypting data and their robustness to withstand attacks. Hence, real-time processing of data encryption/decryption is essential in network based applications such as IoT to keep pace with the input data inhalation rate. Cryptographic algorithms are broadly generalized as symmetric and asymmetric encryption. The idea of including cryptography in IoT as one of the security methods is a transpiring arena. It plays its important character in protecting any sorts of communication and gives the well enhanced provision to offer the requisite shield against the data intruders as well as attacker.

Symmetric key cryptography is the algorithm that uses a secret or private key to lock (encrypt) and unlock (decrypt) [3] the data in a particular way between two or more parties. However, by having the condition that both parties have access to the secret key is one of the main disadvantage of this algorithm. This is because, by using the same private key making it critical to keep the key secret as the chances for the secret key to fall into the wrong hands over a massive network is hugely high. Thus, asymmetric key cryptography is introduced. If symmetric key cryptography allows only a private key between two parties, asymmetric key cryptography however allows a key pair consists of a public key and a private key. The public key is made to give access to everyone, while the private key maintains its function as confidential properties as only allows the owner to access the data.

If time is the dominant factor in encrypting and decrypting a message, then symmetric encryption is an ideal candidate as it comprises only one way factor of delivering and receiving. Reason is that, asymmetric encryption systems have high overhead, in which they are not usable to provide full-time in a practical real-world security enhancement. Symmetric encryption too, is feasible and adaptable as it can be implemented on various stages especially in small embedded devices. Advanced Encryption Standard (AES) [4] in conclusion could be seen to be faster in terms of speed and systematically efficient in terms of performance compared to other encryption algorithms. Due to that, it is said to be suitable to encrypt the actual data and commands. Therefore, AES was chosen as the standard algorithm in this experiment.

Recent advancement in embedded RF devices and sensor nodes communication has increased significantly which normally interconnect to the internet using wireless technology (e.g. radio frequency, Wi-Fi) and run on top of CoAP and TFTP protocols. In many situations, the connection is exposed to vulnerability during sensor node data transmission. The authors in [5-6] has proposed data encryption using ElGamal and RSA asymmetric schemes through RF transmission to strengthen the protocol security. The works provided alternate solution for various stakeholders to execute a rapid product research and development of any cryptographic protocol for smart devices. This previous work has become our motivation to further the research study in RF technology with three ultimate aims in order to improve the data security using encryption schemes to be implemented on IoT embedded devices. The first objective is to study the encryption method to be integrated into RF client-server communication. Next is, to provide a security solution using symmetric encryption method that will satisfy the minimal requirement for a reliable RF transmission in IoT technology. Lastly is, to analyze performance when sending

the unencrypted (plaintext) and encrypted (ciphertext) data using AES algorithm through RF transmission in terms of transmission time and data transmission throughput.
To rapidly build and test the AES algorithm without major changes in the RF transmission software code, a flexible RF transceiver is needed. Slice of Radio (SRF B023) wireless RF transceiver has the capability to be the most feasible device that could reconfigure AES algorithm in RF data transmission. The Slice of Radio brings secure wireless data transmission to embedded device such as Raspberry Pi in a simple and low cost package. The transceiver is attached together on two Raspberry Pi Arm processor boards, implementing python source code to execute the program. The proposed work comprises of a persuasive argument that it could make a noteworthy contribution on a topic that is important to the development of IoT embedded devices, particularly in enhancing the security of the data uploaded using a low budget devices.

2. Related Works

In the present era, not only business but almost all the aspects of human life are driven by information. Securing the data to be uploaded to the Internet has been a prerequisite thing to do before transmitting it, especially in a wireless environment. However, current practices show that there are preponderances of data that remained unsecured, mostly due to the fact that security features are not built into products or users are disregarding them. This leads to the unauthorized access to the information by many false hands. For instance, the implementation of large sized protocol such as Secure Socket Layer (SSL) or Transport Layer Security (TLS) deteriorate the efficiency and performance of the security on the data that it may be compromised by brute force. Based on review findings, in order to achieve confidentiality and integrity of security properties, it is crucial to decide the suitable security mechanism for this work. Therefore, the study of encryption method [7] is done to support the practicality of the protocol on a low cost module. The Raspberry Pi microcomputer board [8] and its Slice of Radio RF transceiver is chosen as a perfect module due to its low power consumption and promotes a rapid result in the experimentation.
In the aspect of speed and as well as level of security, symmetric encryption scheme were given due importance [9] and thus was selected as a general protocol to be used in this work. In particular, the AES encryption scheme was selected. There are two factors identified which contribute to the selection of encryption algorithm. Within a limited CPU and memory storage, the first factor greatly depends on the speed. It has to be fast and facile to be implemented on both software and hardware. The second one is that it must be powerful with no collision found in the algorithm. Compared to another symmetric block cipher such as DES and 3DES, AES works faster even in small devices such as on mobile phone and smart card [9-10]. Besides, data size expansion analysis is also included to analyze the effect of the transmission time on the variable size of the file. Thus, the transmission of plaintext is first tested followed by ciphertext and measured afterwards.
In the cryptography family, AES belongs to the category of block ciphers. A block cipher is an algorithm that encrypts data per-block basis. This is in contrast to stream ciphers which are based on generating a "perpetual" cryptographic key stream and using that to encrypt one bit or byte at a time. As such, block ciphers work on a larger data at a time. In terms of speed, stream ciphers is basically faster than block ciphers as the blocks need to work on a larger data and require more CPU cycles [11]. Even so, the scrutinization of this experiment is to ascertain whether it is possible to provide a security solution that will satisfy the minimal requirement for a reliable RF transmission to be applied in IoT technology. In this case, we have chosen to perform the encryption process in block-oriented encryption mode because it provides the integrity protection and authentication as well as its capability of bulk encryption of known size data which we are assuming in this experiment.
The size of each block is measured in bits. If it is 256 bits long, then AES will operates on 256 bits of plaintext to produce 256 bits of ciphertext [4]. The keys supported by AES comprises of three different lengths which is 128 bits, 192 bits and 256 bits key. The strength of the security is measured by the length of the keys which wins 256 bits key all the way round. However, to provide a full strength of security in this experiment, 128 bits of key is not strong to prove the security of the data in the developed system. This leads to the decision of using AES 256 bits key (32 bytes) to provide the best stronghold of security for the data encrypted.

3. Methodology

To find a suitable device to be used in energy constricted environment could be challenging and time costing. In this work, the Raspberry Pi Arm Boards Version 3 have been selected to be deployed for reason that it is highly supported by researchers and commonly applied in similar applications. It has also been selected due to its compatibility in terms of size and performance. It is small in size, but it has the capability equivalent to a computer. Therefore, it is manageable to handle and execute the Python Language on it. The microprocessor has a number of input and output (I/O) ports including USB port, HDMI, Micro USB power and SD card interface (to install Raspbian, the standard Raspberry Pi operating system).
Under the requirement of energy constricted environment, a low power consumption device is needed to carry out the transmission. To form the secure transmission for Raspberry Pi over RF, the Slice of Radio device is connected to the device. It is very easy to use because it sends and receives via the standard onboard PI serial port, which means special software or drivers would not be needed. The experiment can be executed when the serial port is configured accordingly. Figure 1 shows the closed-up look of Slice of Radio (SRF B023).

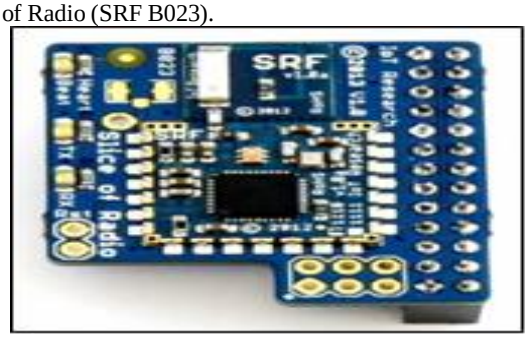

Fig. 1: Slice of Radio (SRF B023)

The workflow to test the communication is shown in Figure 2. The SRF must be attached on both Raspberry Pi ARM boards and the Raspbian Linux Operating System was installed on both devices to execute the program. Then, the SRF USB Radio is plugged in GPIO header and new serial port is identified. Before powering up, the devices should be connected with a mouse, keyboard and a display. It is important to make sure that RF modules are configured with 115200 bps as baud rate.

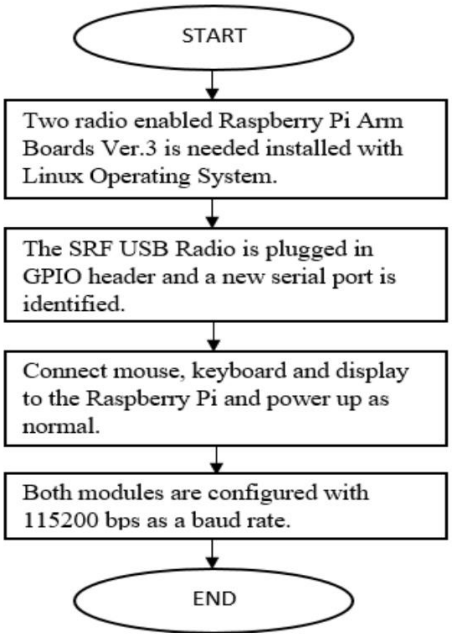

Fig. 2: Flowchart of the communication test using Slice of Radio

Figure 3 shows a client-server test bed set up using two Raspberry Pi Arm boards V3 devices. The experiment was conducted in the laboratory where the two Raspberry Pi Arm boards were placed at a distance of approximately one meter apart. Both devices were associated with the keyboard and mouse to execute the program and a monitor connected to HDMI port respectively to display the program. For power adapter, the micro USB Power Supply which provides at least 700 mA at 5V is important. In this work, the PC power supply was used to power up the devices. The workstation comprised of a client-server infrastructure communicates via RF using the Slices of Radio on each Raspberry Pi respectively.

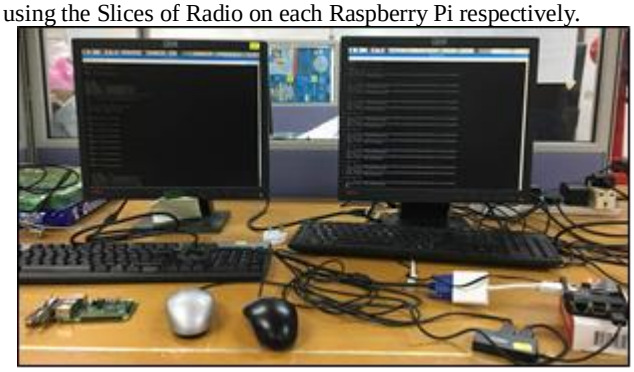

Fig. 3: Client-Server test bed set up

RF (Radio Frequency) is able to transmit data on low power consumption. Bound by the protocol of RF Communication Protocol standard, the Slice of Radio held its performance based on the theory of operation in three ways for this project.

Firstly, when the serial data is received on SRF, it goes into the memory. Once the complete packets (payload) of data were produced, it would be sent out over the radio transmitter. By default, the buffer size is 12 characters per payload. If longer data are said to be send out, it will simply split it up in to smaller packets. For example, as can be seen in Figure 4, the data is split in to 4 packets and each one is transmitted in turn.

```
See the quick brown fox jump over the lazy dogs
-----1--------|--------2------|-------3-------|-------4-----|
```

Fig. 4. Packet structure

Secondly, the data will wait in the transmit buffer before being sent. This process is called as timeout, which happens only when the data sent were less than a packet. By default, the timeout is 100ms for SRF. Finally, to improve the performance and efficiency, padding is the efficient option to fill the buffer for a better performance, especially when the data comprises variable lengths. For example as shown in Figure 5, by adding the padding option (++), both packets will be sent immediately without timeout because of the completed characters per packet.

```
Mary had a little lamb++
--------1------|-------2------|
```

Fig. 5: Packet with padding

4. Performance Evaluation

In this work we consider two types of transmitted data, which are plaintext and ciphertext. Here, we take plaintext as to mean the original unencrypted data, while ciphertext is the text produced after the plaintext has been encrypted by AES algorithm. The objective is to analyze the effect on data transmission time when sending encrypted (ciphertext) data as opposed to the original data. This section provides information on the performance metrics to evaluate these two situations when sending the plaintext and ciphertext data transmitted via RF Communication. The metrics are explained below.

4.1. Data Transmission Throughput

The throughput of data transmission defines how much data can be transmitted in a given amount of time. From the result of execution time, the throughput of every data is calculated to indicate the performance and transmission speed as in (1):

$$\text{Data transmission throughput} = \frac{\text{Total data size (KB)}}{\text{Total transmission time (s)}} \quad (1)$$

4.2. Transmission Time Difference when Sending Plaintext and Ciphertext

The transmission time when sending both plaintext and ciphertext via RF Communication are the method that has consumed majority of the time spent on this experiment. There obviously a difference in the transmission of both types of data respectively. The difference is calculated using the equation below.

$$T_d = T_c - T_p \quad (2)$$

For plaintext and ciphertext:
Td is Time Difference
Tc is Transmission Time for Ciphertext
Tp is Transmission Time for Plaintext

For plaintext and ciphertext with padding:
Td is Time Difference
Tc is Transmission Time for Ciphertext with padding
Tp is Transmission Time for Plaintext

4.3 Data Expansion Rate

Due to the addition of padding to complete the 12 characters per payload, the size of the data of ciphertext was actually predicted to be expanded significantly per addition. The data expansion rate could be measured using the two equations below:

a) For plaintext and ciphertext:

$$\text{Data expansion factor} = \frac{\text{Data size after encryption (ciphertext)}}{\text{Data size before encryption (plaintext)}} \quad (3)$$

b) For plaintext and ciphertext with padding:

$$\text{Data expansion factor} = \frac{\text{Data size after encryption with padding (ciphertext)}}{\text{Data size before encryption (plaintext)}} \quad (4)$$

5. Results and Discussion

The output of the experiment can be summarized in Table 1 until Table 5-6. In this experiment, the main aspect to be examined is the increase in the time of transmission due to data expansion factor from server to client. In methodology section, the quality is measured based on the metrics of throughput of data transmission and delay on transmission time between plaintext (T), ciphertext (C) as well as ciphertext with padding (P). In this section, the plaintext used are numbered from T1 to T10 where Cn is the ciphertext of Tn and Pn is the ciphertext with padding of Tn, for n = 1 to 10.

Table 1: Types of data and its respective sizes

Types of Data	Plaintext (T)		Ciphertext (C)		Ciphertext with Padding (P)	
Data Size (KB)	T1	0.014	C1	0.031	P1	0.036
	T2	0.026	C2	0.064	P2	0.072
	T3	0.040	C3	0.095	P3	0.096
	T4	0.050	C4	0.128	P4	0.132
	T5	0.062	C5	0.128	P5	0.132
	T6	0.074	C6	0.160	P6	0.170
	T7	0.086	C7	0.192	P7	0.192
	T8	0.098	C8	0.224	P8	0.228
	T9	0.112	C9	0.226	P9	0.266
	T10	0.126	C10	0.256	P10	0.264
Total	0.688KB		1.504 KB		1.588 KB	

Table 2: Transmission time for plaintext and ciphertext and the time difference

Data (T/C)	Plaintext Transmission Time (ms)	Ciphertext Transmission Time (ms)	Time Difference (ms)
1	0.990	3.296	2.306
2	2.816	5.975	3.159
3	3.190	7.552	4.362
4	3.595	8.279	4.684
5	5.638	11.295	5.657
6	6.665	13.647	6.982
7	7.583	14.407	6.824
8	8.459	18.571	10.112
9	9.422	19.463	10.041
10	10.550	21.334	10.784

The time difference is theoretically expected to be present in the transmission of data especially when the data sizes of two types of data are different to begin with. The assumption has indeed been proven when the transmission time in Table 2-3 increased gradually for plaintext, ciphertext and ciphertext with padding labelled as 1 to 10 (T/C/P) proportional to the each data size.

Table 3: Transmission time for plaintext and ciphertext with padding and time difference

Data (T/P)	Plaintext Transmission Time (ms)	Ciphertext with Padding Transmission Time (ms)	Time Difference (ms)
1	0.990	3.340	2.350
2	2.816	6.590	3.774
3	3.190	8.396	5.206
4	3.595	9.889	6.294
5	5.638	11.418	5.780
6	6.665	10.638	3.973
7	7.583	12.255	4.672
8	8.459	14.459	6.000
9	9.422	18.765	9.343
10	10.550	18.983	8.433

Table 4. Throughput of data transmission

Types of Data	Plaintext		Ciphertext		Ciphertext with Padding	
Throughput (KB/s)	T1	14.141	C1	9.422	P1	10.778
	T2	9.233	C2	10.711	P2	10.926
	T3	12.539	C3	12.579	P3	11.434
	T4	13.908	C4	15.461	P4	13.348
	T5	10.997	C5	11.337	P5	11.561
	T6	11.103	C6	11.730	P6	15.980
	T7	11.341	C7	13.327	P7	15.667
	T8	11.585	C8	12.062	P8	15.769
	T9	11.887	C9	11.612	P9	14.175
	T10	11.943	C10	12.000	P10	13.907
Average (KB/s)	11.868		12.024		13.3545	

The throughputs for these types of data in Table 4 are calculated by using the equation provided in methodology section. According to rhetorical assumption made, the higher the throughput of data transmission, the better the performance. For being directly sent from server to client, ciphertext with padding has indeed proven to have the highest throughput. Meanwhile, the lowest would be ciphertext due to the incomplete characters per payload, creating timeouts which causes delay that slowed the process of transmission. The problem then solved by adding padding to align the data size and completed the characters per payload, so that delay would be prevented.

Table 5: Data expansion factor

Data (T/C/P)	Data Expansion Factor between Ciphertext and Plaintext	Data Expansion Factor between Ciphertext with Padding and Plaintext
1	2.214	2.571
2	2.462	2.769
3	2.375	2.400
4	2.56	2.640
5	2.065	2.129
6	2.162	2.297
7	2.233	2.233
8	2.286	2.327
9	2.018	2.375
10	2.032	2.095

Instead of using rate in expansion data size, the unit is changed to factor to act as an indicator for the increment or decrement on the data size. In Table 5, there is a slight expansion of data size between ciphertext and plaintext. This is due to the encryption process that used CBC mode of operation with 32 bits of key that translated the plaintext into series of hex strings configuration, which load the payload longer than the plaintext. The hex strings however did not alter any data on plaintext.

In theory, AES does not expand the data, except for a few bytes of padding at the end of the last block. This is to align the data to the size of a block thus explaining the expansion in Table 5. Most symmetric ciphers work on blocks of data considerably larger than a single byte (AES-128, for example, works 16 bytes at a time). Padding at the end of the data is important to fill and align the incomplete block if the data is not a multiple of the block size. The resulting data inherently are not compressible at any rate because they are basically random; no dictionary-based algorithm is able to effectively compress them.

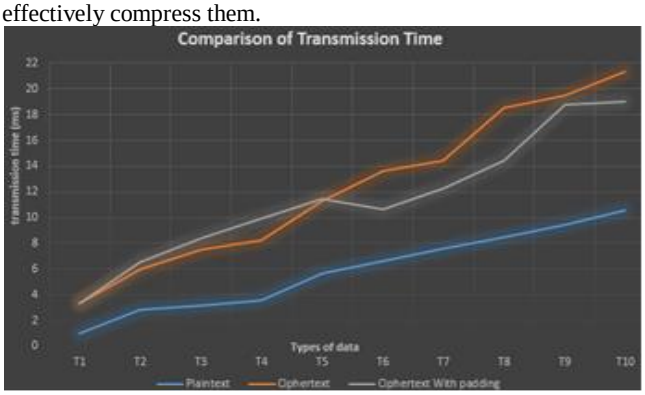

Fig. 6: Comparison of Transmission Time between Plaintext (T), Ciphertext (C) and Ciphertext with Padding (P)

As seen in Figure 6, it can be concluded that the transmission time increased gradually for all types of data due to increased payload. As the characters increased per payload, the transmission time for the three types of data too, was increased. From the result, we can see how the two types of encrypted data affects transmission time by comparing their performances to that of the plaintext transmission. This is because for plaintext, 12 characters per payload were directly being sent from server to client. The transmission time for ciphertext with padding was expected to be faster than the transmission time of ciphertext without padding. This is due to the ciphertext payload that has been sent, which was less than a default 12 characters per packet length incurred some delay as the packet needed to be stored at the buffer temporarily for 100 ms or more. This action is called as timeout as explain in methodology section. The data was finally received at the client, but with a redundant time delay. Hence, ciphertext with padding which allow stuffing at the end of the data to fill the last block to complete the payload, transmission time is comparatively faster than ciphertext alone.

6. Conclusion

This paper presents the performance evaluation of data transmission when sending three types of data: the plaintext, the cipher text without padding and ciphertext with padding over RF. Based on the result presented in Table 2-3, ciphertext with padding transmission performed best in term of transmission time which can reduce 7.92% time over the ciphertext without padding transmission. This may come from the fact that padding technique can prevent the delay and timeout in the transmission [8]. Besides, the transmission time when sending ciphertext with padding has produced average delay of 5.5825ms which is approximately two times slower than the plaintext transmission. There are several points can be concluded from the experiment results. In conclusion, encryption scheme has increased the transmission time over RF. By adding padding on the ciphertext can reduce the delay slightly.

In future work, it would be interesting to study a broader metrics other than the transmission time and throughput performance metrics in a higher level of devices, so that its practicality could be applied to a higher level, more so to industries.

Acknowledgement

This project was made possible by funds from Malaysian National Research Grant Scheme (NRGS) 600-RMI/NRGS 5/3 (5/2013) from the Ministry of Education, Malaysia.

References

[1] Paul B, Chiriyath A. R and Bliss D. W. Survey of RF communications and sensing convergence research. IEEE Access, 5, 252–270.
[2] Saleh M. E, Aly A. A and Omara F. A. Data security using cryptography and steganography. Int. J. Adv. Comput. Sci. Appl., 2016, 7(6), 391–397.
[3] Zakir M and Sarker H. A cost effective symmetric key cryptographic algorithm for small amount of data. Proceedings of the IEEE Int. Multitopic Conf., 2005, pp. 1–6.
[4] F. Information. Announcing the Advanced Encryption Standard (AES). 2001.
[5] Adnan S. F. S, Isa M. A. M, Rahman K. S. A, Muhamad M. H and Hashim H. Simulation of RSA and ElGamal encryption schemes using RF simulator. Proceedings of the IEEE Symp. Comput. Appl. Ind. Electron., 2015, pp. 124–128.
[6] Isa M. A. M, Hashim H, Manan J. L. A, Adnan S. F. S and Mahmod R. RF simulator for cryptographic protocol. Proceedings of the IEEE Int. Conf. Control Syst. Comput. Eng., 2014, pp. 518–523.
[7] Joshi A, Wazid M and Goudar R. H. An efficient cryptographic scheme for text message protection against brute force and cryptanalytic attacks. Procedia Comput. Sci., 48, 360–366.
[8] Thota P and Kim Y. Implementation and Comparison of M2M Protocols for Internet of Things. Proceedings of the Int. Conf. Appl. Comput. Inf. Technol., 2017, pp. 43–48.
[9] Alanazi H. O, Zaidan B. B, Zaidan A. A, Jalab H. A, and Shabbir M. New comparative study between DES, 3DES and AES within nine factors. J. Comput., 2010, 2(3), 152–157.
[10] Singh G and Supriya S. A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. Int. J. Comput. Appl., 2013, 67(19), 33–38.
[11] Singhal N and Raina J. P. S. Comparative analysis of AES and RC4 algorithms for better utilization. Int. J. Comput. Trends Technol., 2011, 2(6), 177–181.