ICIC Express Letters
Part B: Applications
Volume **10**, Number **11**, November **2019**

ICIC International ©2019 ISSN 2185-2766
pp. **963–970**

BROTLI DATA COMPRESSION ON MOODLE-BASED E-LEARNING SERVER

Vicky Reynaldo, Arya Wicaksana and Seng Hansun

Informatics Department
Universitas Multimedia Nusantara
Jl. Scientia Boulevard, Gading Serpong, Tangerang, Banten 15811, Indonesia
vickyreynaldo@gmail.com; { arya.wicaksana; hansun }@umn.ac.id

Received June 2019; accepted September 2019

Abstract. *The limited capacity of media storage and the increased number and size of data every day could decrease the productivity level. These problems can be solved by compression methods that have been commonly used. Compression makes data storage more efficient and shortens the time of data exchange. The compression algorithm used in this study is Brotli. The algorithm was implemented in the Moodle-based UMN e-learning server as a Moodle plugin. The compression application was developed by using Java programming language, and the Moodle plugin was developed by using PHP programming language. The test results on 33 files with various compression sizes gave an average compression ratio of 0.77 and the average compression factor of 1.707, with the average compression saving percentage of 22.94%. Comparative tests with the Google Brotli apps show an average of 97.71%. The compression test was done by compressing data using the created application and decompressing data using Google's application. The compression test shows that each file was successfully decompressed using an application from Google. These results indicate that the Brotli compression algorithm has been successfully implemented in this study.*
Keywords: Brotli, Data compression, E-learning, Google, Moodle, UMN

1. **Introduction.** Nowadays, people activities are always related to data and information. Data stored in storage media grow in size every single day, which cause a problem for the storing and managing of the data [1]. This problem can be solved by a compression method. The compression method is a technique to compress data by simplifying the complexity of data to minimize the size of the data. Compression makes data more efficient and shortens the data exchange time. Many advantages can be gained from the process of compressing data, such as saving data storage, saving data exchange time, and saving bandwidth on the data transmission process [2]. There are many algorithms that can be used to do compressions, such as Huffman-Code [3], Lempel-Ziv-Welch (LZW) [4], Lempel-Ziv-Markov chain Algorithm (LZMA) [5], Dynamic Markov Compression (DMC) [6], and the recently released by Google, Brotli algorithm.

Moodle is a software package produced for Internet-based learning activities and websites. Moodle is available online and can be used freely as an open-source product under the GNU license. Every file in Moodle is stored according to the SHA1 hash of their content. It means each file with particular contents is stored once, irrespective on how many times it is included in different places, even if it is referred to by different names, and each plugin may directly access only files in its context and areas [7].

The Brotli compression algorithm uses a combination of the LZ77 algorithm, the Huffman code, and the second modeling context, which has a better compression level than the Deflate algorithm [8]. Using the Brotli algorithm is expected to produce a better compression ratio than by just using the LZ77 compression.

DOI: 10.24507/icicelb.10.11.963

This study will implement the Brotli algorithm to compress data on Moodle-based e-learning platform so that it can reduce the use of media storage in Moodle. Implementation will be made in the form of a plugin in Moodle platform since Moodle needs a plugin to be able to access files in the Moodle directory, and each plugin can only access its files. With this research, it is expected to produce a better compression ratio value than previous studies. As the experimental results show us at the end of this paper, we had successfully implemented the Brotli compression algorithm in the Moodle-based UMN e-learning server. Any data uploaded to the plugin can be compressed and decompressed correctly without any loss on the information.

In the next section, we will briefly discuss the concept of data compression before explaining the LZ77 algorithm, Huffman algorithm, and Brotli algorithm in Section 3. In Section 4, the Moodle platform being studied here will be explained together with some screenshots showing the results of the Moodle plugin that had been built. The experimental results and analysis will be given in Section 5, and we finish the organization of this paper by some concluding remarks in Section 6.

2. **Data Compression.** Data compression is the process of reducing the size of data by changing a set of data into a set of codes that can save storage space requirements and the time needed to transmit [9]. Data compression is the process of converting an input data stream (source stream or original raw data) into other data stream (output, bitstream, or compressed stream) which is smaller [10]. Data compression methods can be grouped into two major groups, [11].

1) The lossless method or class of data compression algorithms that allows the original data to be rearranged from the compressed data. Lossless data compression is used in various applications, such as ZIP and GZIP formats.

2) The lossy method or the process of data compression and then decompressed where the resulting data is not the same as the original data, but enough and can still be used as needed. Lossy compression is generally used to compress multimedia data (audio, video, and image), and especially used in applications, such as streaming media and Internet telephony. The more compression had been done, the less quality that it gets due to the loss of some data.

When testing the compression results, several indicators need to be considered in each compression technique.

- Compression ratio [9] is the ratio between the size of the original data before it is compressed to the size of the data that has been compressed. A smaller compression ratio means a more satisfying compression result.

$$Compression\ ratio = \frac{size\ after}{size\ before} \tag{1}$$

- The compression factor is the ratio between the size of the data after being compressed to the size of the original data before being compressed. The greater the value of the compression factor, the more satisfying the compression results.

$$Compression\ factor = \frac{size\ before}{size\ after} \tag{2}$$

- The percentage of savings is the ratio percentage between the difference in the size of the data from before it was compressed and after being compressed to the size of the data before being compressed. The greater the percentage of savings, the more satisfying the compression results.

$$pp = \frac{size\ before - size\ after}{size\ before} \times 100\% \tag{3}$$

3. **The Algorithms.** In this section, we explain some concepts of Brotli algorithm, starting with the LZ77 and Huffman algorithms as the two building blocks.

The LZ77 algorithm (LZ77) or Lempel Ziv 1977 is a lossless compression algorithm, developed by Abraham Lempel and Jacob Ziv in 1977. LZ77 uses the sliding windows compression method; the data structure is in the form of windows which will be divided into two parts, consisting of the text encoded (search buffer) and other parts that will be encoded (look-ahead buffer). Search buffers will have thousands of bytes, and the look-ahead buffer has tens of bytes in length [9].

The principle of this algorithm is to use the characters that have been encoded before as a dictionary. This input is similar to a window that can be shifted from left to right. This window is a dynamic dictionary that searches for input symbols with a specific pattern. When this window moves from left to right, the contents of the dictionary and the input characters that the pattern looks for will be changed [9].

Several steps must be carried out in LZ77 compression. The following are the steps in LZ77 [12].

1) Determine the size of the search buffer and look-ahead buffer.

2) Fill in the character sequence input into the available space in the look-ahead buffer.

3) If the search buffer is empty or there are no similar character patterns in the look-ahead buffer, then remove the token with an offset format of 0, the length match value is 0, and the mismatched character is followed at the beginning of the look-ahead buffer.

4) If a pattern is found, then remove the token with the offset format which is the distance from the search buffer found in the pattern, the length match value is the length of the pattern found the same, and the mismatched character is filled with one character outside of the pattern found.

5) Then, cut the character that has been converted into the token from the look-ahead buffer and load it into the search buffer, which indicates that the character has been processed in compression.

6) Fill in the look-ahead buffer until the full row of the remaining entries and do the process of matching the character pattern to find the End of File (EOF).

7) If EOF has been found and the look-ahead buffer has no characters left, it will get the LZ77 compression token.

The Huffman algorithm is made by a Massachusetts Institute of Technology (MIT) student named David Huffman and is the longest and most well-known compression method in text compression domain. The Huffman algorithm uses a coding principle that is similar to Morse code, with each character encoded only with a series of several bits, where characters that often appear are encoded using a series of short bits and characters that rarely appear are encoded with a longer bit series [13]. The essence of the Huffman algorithm is to use recursive priority queues [14] so that it forms a tree with time complexity of $O\ (n \log n)$. The steps to create a Huffman tree are as follows.

1) Sort the characters from the smallest frequency into a table.

2) Select the top two characters, made into leaf nodes from the tree. Include the character content and frequency in the node.

3) Create a new node with the parent node position of the two leaf nodes. The frequency is obtained from the sum of the frequency of the two child nodes.

4) Eliminate the letters that have been used from the table.

5) Enter the new node into the table.

6) Repeat the process from step number two until the table is finished.

Brotli is a lossless data compression algorithm that uses a combination of the LZ77 algorithm, the Huffman algorithm, and the second modeling context to compress data. The compression speed is similar to the Deflate algorithm but has a better compression rate [8]. The second modeling context used by Brotli is a feature that allows multiple Huffman trees to store the alphabet on the same block [15].

A compressed data consists of a header and a meta-blocks sequence. Each meta-block decompresses from 0 to 16,777,216 (16 MiB) uncompressed bytes. The header has the size of the sliding window used during the compression process. This size is needed by the system so that it can decompress the compressed files. The size of the sliding window is obtained from entering the level given to the system. Each meta-block is compressed using a combination of LZ77 and Huffman coding algorithms. Huffman coding is used as a prefix code. The prefix codes for each meta-block are independent of the meta-blocks before and after. In the Brotli compression format, a reference can be used or fixed to a fixed dictionary (static dictionary entry) [16,17].

The compressed data consists of command lines. Each command consists of two parts, a sequence of bit literals (a string that is not detected as a duplicate in the sliding window) and a pointer to a duplicate string, which is represented as a pair <length, backward distance> [16]. The Brotli algorithm has three types of compression modes, namely BROTLI_GENERIC (default), BROTLI_TEXT (for UTF-8 format), and BROTLI_FONT (for WOFF 2.0) [15].

4. **The Moodle Platform and Results.** Moodle is a software package produced for Internet-based learning activities and websites. Moodle is available online and can be used freely as an open-source product under the GNU license [18]. Moodle itself stands for Modular Object-Oriented Dynamic Learning Environment, which means a place for dynamic learning using object-oriented models. In its provision, Moodle provides a complete software package that is ready for use [19].

Every file in Moodle is stored according to the SHA1 hash of their content. It means each file with particular contents is stored once, irrespective of how many times it is included in different places, even if it is referred to by different names and each plugin may directly access only files in its context and areas [7].

The Brotli compression algorithm will be implemented by using the Java programming language, and the Moodle plugin will be created by using the PHP programming language. Figure 1 shows the model of the application that was built.

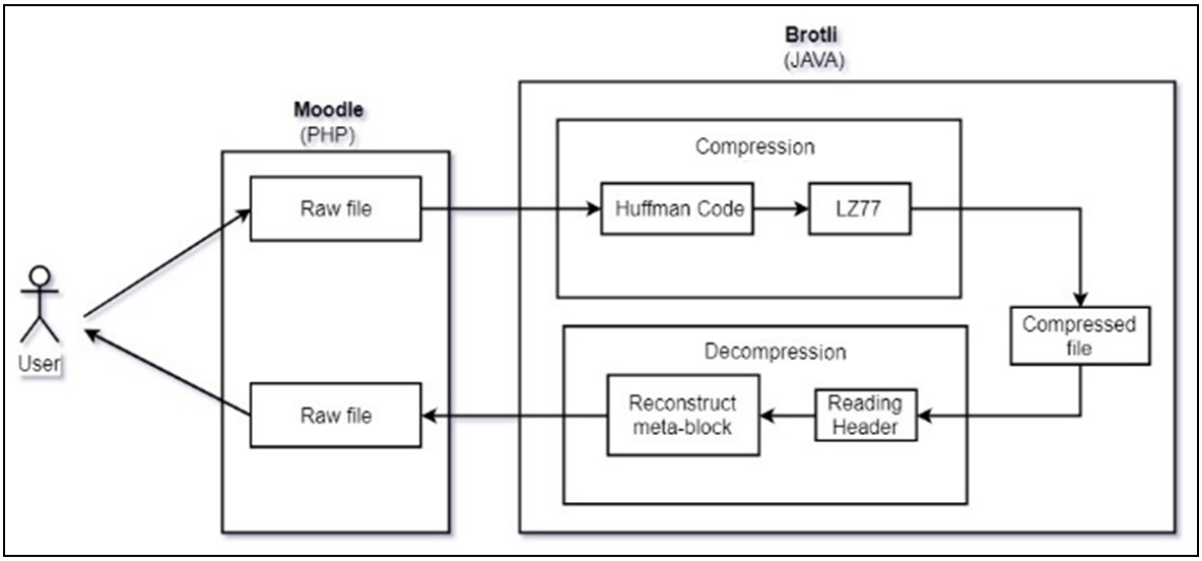

FIGURE 1. Application model

The application in Moodle is made in the form of a plugin. The plugin application is designed to be able to do the compression and decompression of each uploaded file. The Brotli compression application will be designed using UTF-8 mode. Each file uploaded into the Moodle plugin will be sent to the Brotli compression system, and the compression

process will be carried out. The results of the compression process will be saved into Moodle's data folder. When the file wants to be accessed again, the plugin will call Brotli to be decompressed and send it back to the plugin.

The plugin application will accept the user uploaded files and do the compression before it is transferred to the Moodle system file. If the upload has finished, the plugin will display a message that the file was uploaded successfully. From there the user can move the page to see a list of files that have been uploaded and the user can download the file. On the download page, the file will be decompressed before displaying the download link on the page. Figures 2 to 4 show several views of the application.

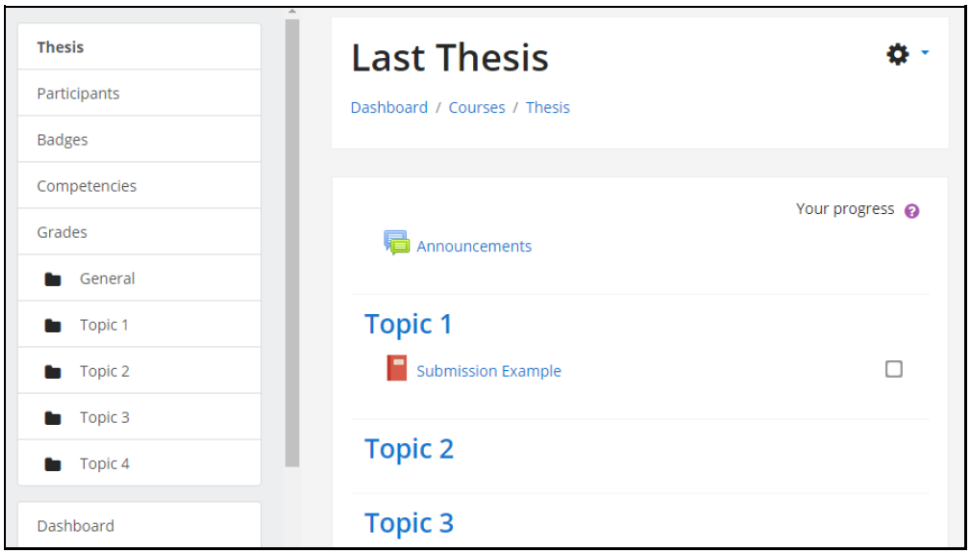

FIGURE 2. Course n web page

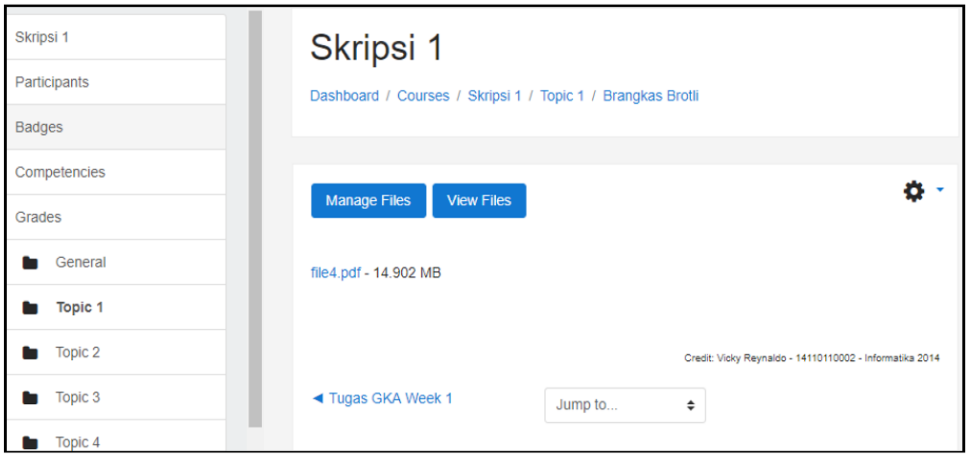

FIGURE 3. Plugin page

Figure 3 shows the page for downloading files. Every file in the plugin will be displayed on this page. There are two navigation buttons, first is the Manage Files button that functions to move to the file upload page and the View Files button to move to this page. Figure 4 shows the page for uploading files. Moodle form is used to upload the files.

5. **Discussion.** Application testing phase was divided into two parts; i.e., testing the implementation of the algorithms and compression testing by comparing with Google's Brotli. Application testing is done to find out whether the implementation was successful

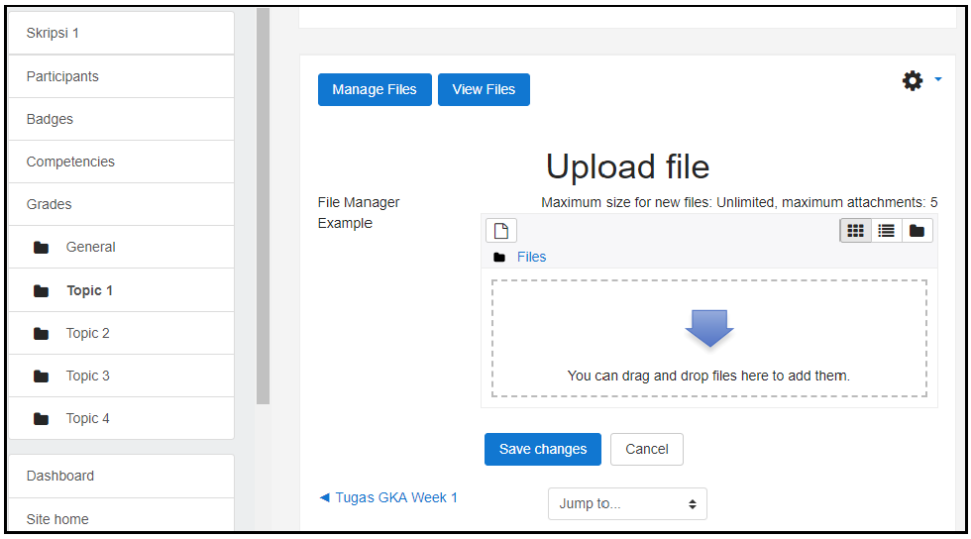

FIGURE 4. Plugin upload page

TABLE 1. Compression experiment 1

No.	File	C. Ratio	C. Factor	Percentage saving	Level
1	file1.doc	0.956	1.046	4.44%	1
2		0.955	1.047	4.46%	2
3		0.956	1.046	4.43%	3
4		0.953	1.049	4.67%	4
5	file2.doc	0.25	4.004	75.03%	1
6		0.249	4.023	75.15%	2
7		0.247	4.041	75.25%	3
8		0.238	4.203	76.21%	4
9	file3.doc	0.956	1.045	4.35%	1
10		0.956	1.046	4.38%	2
11		0.956	1.046	4.36%	3
12		0.951	1.052	4.91%	4

in the Moodle platform. In the compression experiment by using three files with a .doc extension, we obtained the following results as shown in Table 1.

In the experiment by comparing the results of compression in this study with Google's Brotli, we obtained the following results, as shown in Table 2.

From the results of 33 sample files, the average compression ratio is 0.77, with 1.00 as the largest ratio and 0.152 as the smallest ratio. The average compression factor obtained is 1.707, with 6.581 as the biggest compression factor and 1.00 as the smallest compression factor. The average saving percentage obtained was 22.94%, with 84.81% as the biggest savings percentage and 0% as the smallest percentage savings. From the experimental results, several files get a relatively small ratio of compression ratio, compression factor, and saving percentage. It could happen to files with PNG and GIF extensions because the file types have a lossless compression system with the LZW algorithm, so many bits are passed during the compression process. The second testing was done by comparing the compressed file results from the plugin and the compressed file results that belong to Google's Brotli. The average similarity is 97.71%, with the greatest similarity being 99.99% in files with PNG extension, and the smallest similarity being 61.83% in files with XLS extension.

Table 2. Compression experiment 2

No.	File	Brotli (kb)	Brotli Google (kb)	Level	Similarity
1	file1.doc	3,167.89	3,188.746	1	99.80%
2		3,170.054	3,174.101	2	99.87%
3		3,169.084	3,169.51	3	99.98%
4		3,157.455	3,141.525	4	99.94%
5	file2.doc	21.076	21.092	1	93.58%
6		21.075	21.096	2	97.46%
7		21.073	21.02	3	98.14%
8		20.952	20.97	4	99.40%
9	file3.doc	2,036.575	2,036.692	1	98.96%
10		2,035.816	2,036.152	2	99.80%
11		2,035.819	2,036.047	3	99.90%
12		2,034.26	2,034.227	4	99.76%

6. **Conclusions.** The implementation of Brotli compression algorithm on Moodle-based e-learning was successful. Any data uploaded to the plugin can be compressed and decompressed correctly without any loss on the information. The use of the plugin is also compatible with Google Brotli app. In the compression test, the average compression ratio is 0.77, the average value of the compression factor is 1.707, and the average value of the percentage of savings is 22.94%. PNG and GIF extension files also give relatively small test values. It is due to the many bit literals that cannot be read. The average similarity between the built plugin and the Google Brotli app is 97.71%. Differences occurred due to the implementation of the algorithm in Java programming language, which is different from the original implementation in the C programming language. As for the future research, the implementation of other general purposes compressors, such as Zstd, LZ4, Sprintz [20], and Deflate [21] can be implemented in the Moodle-based e-learning to get the comparison results with the Google's Brotli compression technique.

REFERENCES

[1] J. Uthayakumar, T. Vengattaraman and P. Dhavachelvan, A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications, *Journal of King Saud University – Computer and Information Sciences*, https://doi.org/10.1016/j.jksuci.2018.05.006, 2018.

[2] P. Lancett, The advantages of file compression | techwalla.com, *Techwalla*, https://www.techwalla.com/articles/the-advantages-of-file-compression, 2018.

[3] H. Liang, X. Zhang and H. Cheng, Huffman-code based retrieval for encrypted JPEG images, *Journal of Visual Communication and Image Representation*, vol.61, pp.149-156, 2019.

[4] A. P. U. Siahaan, R. Rahim and I. B. A. I. Iswara, Application of Data Encryption Standard and Lempel-Ziv-Welch algorithm for file security, *International Journal of Engineering & Technology*, vol.7, pp.783-785, 2018.

[5] A. Rosete, K. Baker and Y. Ma, Using LZMA compression for spectrum sensing with SDR samples, *Proc. of the 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference*, New York, 2018.

[6] J. de-la-Pena-Sordo, I. Pastor-Lopez, I. Santos and P. G. Bringas, Using compression models for filtering troll comments, *Proc. of the 10th Conference on Industrial Electronics and Applications (ICIEA)*, Auckland, New Zealand, 2015.

[7] Moodle, *Moodle Documentation*, https://docs.moodle, 2017.

[8] Google, *Brotli*, https://github.com/google/brotli, 2015.

[9] H. Dheemanth, LZW data compression, *American Journal of Engineering Research (AJER)*, vol.3, no.2, pp.22-26, 2018.

[10] D. Salomon, *Data Compression the Complete Reference*, 4th Edition, 2007.

[11] GIS Geography, *Lossless Compression vs Lossy Compression – GIS Geography*, https://gisgeography.com/lossless-compression-vs-lossy-compression/, 2018.

[12] Msdn.microsoft.com, *[MS-WUSP]: LZ77 Compression Algorithm*, https://msdn.microsoft.com/en-us/library/ee916854.aspx, 2018.
[13] Techie Delight, *Huffman Coding Compression Algorithm – Techie Delight*, http://www.techiedelight.com/huffman-coding/, 2018.
[14] J. Morris, *Data Structures and Algorithms*, https://www.cs.auckland.ac.nz/software/AlgAnim/huffman.html, 2018.
[15] V. Krasnov, *Result of Experimenting with Brotli for Dynamic Web Content*, https://blog.cloudflare.com/results-experimenting-brotli/amp/, 2015.
[16] J. Alakuijala and Z. Szabadka, *Brotli Compressed Data Format*, Technical Report, 2016.
[17] J. Alakuijala, A. Farruggia, P. Ferragina, E. Kliuchnikov, R. Obryk, Z. Szabadka and L. Vandevenne, Brotli: A general-purpose data compressor, *ACM Trans. Information Systems (TOIS)*, vol.37, no.1, pp.1-30, 2019.
[18] E. G. Hasan, A. Wicaksana and S. Hansun, The implementation of winnowing algorithm for plagiarism detection in Moodle-based e-learning, *Proc. of the 17th IEEE/ACIS International Conference on Computer and Information Science*, Singapore, 2018.
[19] Docs.moodle.org, *Complete Install Packages for Windows – MoodleDocs*, https://docs.moodle.org/35/en/Complete_install_packages_for_Windows, 2018.
[20] D. Blalock, S. Madden and J. Guttag, Sprintz: Time series compression for the Internet of Things, *Proc. of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol.2, no.3, 2018.
[21] C. Fransisca and M. I. Prasetiyowati, Implementation of deflate compression algorithm in a PHP based and MySQL based website, *KNS&I STIKOM Bali*, vol.8, no.1, 2014.