International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

A Squeeze Pack and Transfer Algorithm-based Efficient Framework for Optimized Network Data Transfer in IoT Applications

Shiv Preet [1,2], Chirag Sharma [2], Rachit Garg[3]

Submitted: 01/09/2023 **Revised**: 20/10/2023 **Accepted**: 04/11/2023

Abstract*:* The wireless network is the driving force behind the new world's economy, and its invasion has spread far beyond Earth in an interplanetary network using satellites. Wireless networks' productivity must be ur-gently improved because they are prone to signal attenuation, slower transfer speeds, complete signal un-availability due to weather, and overcrowding of connected users. Due to the towers' remote position, ex-cessive latency is also an issue, as is the enormous magnitude of data transfer that comes with the big data revolution. The antiquated TCP protocol, which is ineffective for low consumption and limited storage IoT items as data buffer is insignificant in such devices, is another problem for IoT applications. A novel SPT (Squeeze Pack and Transfer) algorithm has been proposed to increase mobile network productivity and reduce storage by more than 90% for data files. Binary patterns, rather than conventional textual symbols compress data, resulting in a higher compression ratio and faster compression speed. The suggested algo-rithm will significantly improve network performance while also facilitating efficient data transfer for IoT devices with limited storage. Many researchers confront issues such as low compression ratios and re-stricted support for multiple data formats from generic compression. All of these concerns are addressed by the proposed method.

Keywords: *Lossless compression algorithms, Internet of Things, binary keys, decompression, wireless network, mo-bile network, data files, lossy compression algorithms.*

1. Introduction

and the internet are going hand in hand nowadays. Users post their Instagram reels or upload their info-tainment content on YouTube with the help of the internet only. The data boom has increased the content consumption of OTT applications like prime videos, Netflix, Sling TV, HBO Max, and even private IPTV applications [1]. The internet is the crucial means to improvise or downgrade the macro economy as well as the micro-economy of a country [2]. Economic sanctions on Russia, as well as Afghanistan, are mainly imposed on online financial transactions only [3]. Cyber warfare is also developing at a rapid pace. Coun-tries tend to have cyber-attacks on rival countries instead of conventional war. Similarly, all business transactions are now generally managed online. Paytm, Rupay, Bharat Pay, and other financial applications are now replacing traditional offline cash transactions [4].

1 Assistant Professor, Information Technology Department, ITM, Dehradun 248001, India
2 Associate Professor, Department of Computer Science and Engineering, Lovely Professional University, Phagwara 144401, India
3 Assistant Professor, Department of Computer Science and Engineering, Lovely Professional University, Phagwara 144401, India
** Corresponding Author Email: saishivam@gmail.com*

The data boom in the modern world has led to the invention of new formats for streaming content. SD (standard definition) was the norm of cable tv and DTH (direct to home) services for consuming content [5]. This trend has changed nowadays. Users want to see HD (high definition) or UHD (ul-tra-high-definition) content via these services [6]. A symbol rate of 27500 or 2.5 Mbps is required to transfer HD content via satellites.

Similarly, to transfer super HD video or 4K television streams, a symbol rate of 4.0 Mbps or more is necessary [7]. Without employing any compression method, ultra-high definition broadcasts cannot be de-livered on a 4 Mbps channel. While transferring HD or 4K broadcasts on the constrained available band-width, compression algorithms assist save satellite capacity [8]. Although improvements are being made to the satellite transponders' carrying capacity, it still requires assistance from certain technologies to effi-ciently transfer data over long distances or in an interplanetary network [9]. Elon Musk's Starlink and Greg Wyler's OneWeb are working to bring satellite internet to every country. These services require effective compression technologies in order to recoup the higher latency costs associated with satellite broadband than with optical fibre broadband [10].

Table 1. Data consumed per hour by different streming applications.

S. No.	Streaming Service	Data consumed per hour in GB
1	Netflix	7
2	YouTube	15.98
3	Hulu	7
4	Amazon Prime	1.82
5	Hotstar	1.3

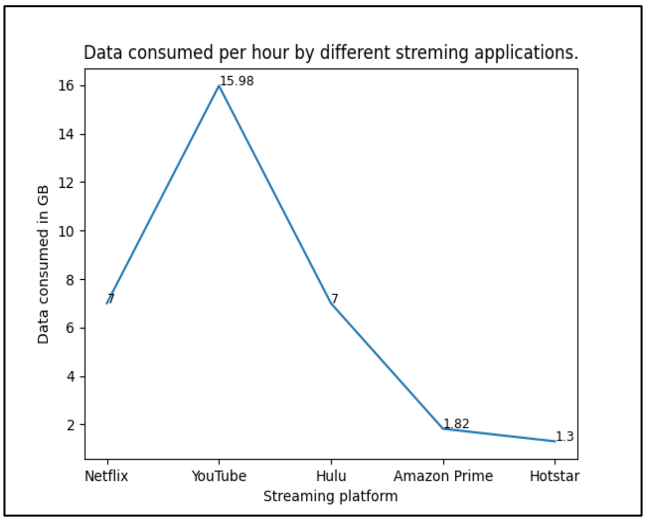


Fig 1. Per hour data consumption by different streming applications.

Table 1 displays the data usage of popular streaming services. The table clearly shows that the amount and calibre of data being consumed per hour are increasing exponentially. The mobile and ad hoc wireless networks must be utilised as effectively as possible in order to enhance connectivity and the user experience. The same has been confirmed by figure 1. It is obvious that more than one GB of data is used each hour for higher quality streaming from reputable streaming sources.

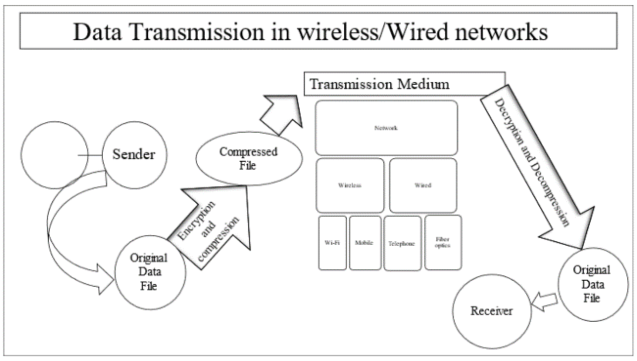


Fig 2. Transmission of data in wireless and wired networks

Data transfer in wireless and wired networks is shown in Figure 2. The procedure is initially started by a sender sending a file over the network. The data file is first compressed and encrypted before being delivered to the transmission media [11]. Once it arrives at its intended location, it is once more decrypted and decompressed before being received by the recipient. Both wired and wireless communication meth-ods are possible [12]. Additional divisions include fibre optics, telephone, and other fixed broadband services. Wireless transmission may use an ad hoc mobile or wi-fi network [13].

In computer science, compression refers to a technique that uses fewer bits to represent a larger-sized data file. Not only does compression reduce the amount of storage space needed, but it also aids in en-crypting data by changing its format [14]. Lossless compression and lossy compression are the two main categories of compression technologies. Each has advantages as well as drawbacks [15]. Lossless com-pression is advantageous for applications where the integrity of the data is crucial. In applications where a slight drop in quality positively impacts the information, lossy compression can be used [16]. It's not a given that quality will be compromised by using a compressed format. Usually, deleting the unnecessary part improves the quality. The JPEG (Joint Photographic Expert Group) lossy technique eliminates the pixels that cause noise to enhance the image quality [17]. This lossy technique therefore aids in improving photos [18]. These examples show that both lossy and lossless compression algorithms are effective in a variety of real-world situations, and that the appropriate technique should be employed depending on the user's needs and the work at hand. Lossless algorithms can come to the rescue occasionally, but lossy al-gorithms can occasionally produce better results [19].

Lossy compression is a type of compression where some of the original message is sacrificed in order to approximate the original message from the compressed format. This improvement in quality has a fa-vourable effect on the message [20]. To improve picture quality, noise from the raw photos is removed as an example of lossy compression. One of the technologies used for lossy compression is JPEG (Joint Pho-tographic Expert Group) [21].

Lossless compression is a type of compression where the original message's quality is preserved in the compressed form. The quality of the data has not decreased [22]. The amount of bits generated during the decompression operation is identical to those in the datafile before to compression [23]. For those re-al-time applications where the integrity of the data is crucial, lossless compression is essential. One of the lossless compression techniques is Huffman coding. Loss-less compression is typically used in the medi-cal professions to preserve image quality for prognosis and diagnosis [24].

Table 2. Comparison of Lossy and Lossless Compression

S. No.	Lossy Compression	Lossless Compression
1	Orignal form of the data file is lost permanently.	Original form of the data file remains intact.
2	Data holding capacity is more than lossless compression algorithm.	Data holding capacity is less then lossy compressin algorihtm.
3	Quality of data is altered in lossy algorithm.	Quality of data remains unchanged in lossless compresion algorithm.
4	Lossy compression algorithm affects size of data.	Lossless compresion algorithm does not affect size of data.
5	Common examples of lossy compression algorithm are Discrete Cosine Transform, JPEG and so on.	Common examples of Lossless compression algorithm are Run Length Encoding, Arithmetic encoding and so on.

Table 2 compares lossless and lossy compression. The advantages of lossless and lossy compression in various circumstances are clearly demonstrated in Table 2. In some circumstances, where a minor loss in the original file structure does not adversely affect usability, lossy compression can be used. Large-scale, important applications that require lossless data compression are preferred [25].

IoT applications are more current technologies that build an object-based network as opposed to a human one. Without human intervention, objects and people can connect with one another [26]. IoT devices cannot function properly with the current TCP (Transmission Control Protocol) protocol for three key reasons. The initial configuration of the connection is the first problem. Given the limited storage and processing capability of Internet of Things (IoT) devices, TCP does not need to carry as much data during the first connection establishment. Second, they lack larger buffers, which makes the commonly utilised TCP buffer concept problematic for the efficient operation of IoT devices. Thirdly, the TCP congestion control technique does not help data sharing in IoT devices. Instead of the standard TCP data transfer, IoT items function on small data exchanges. The purpose of TCP congestion control is merely to prevent IoT items from operating effectively [27]. Because unattended IoT devices might not be secured by the current security system, IoT devices also need a new security mechanism. Complex security measures created with desktop or laptop hardware in mind are not very effective for energy-efficient IoT devices [28].

An innovative compression method (SPT algorithm) is suggested as a solution to the problem of crowded wireless and mobile networks. The contribution of this work is given below:

• SPT is an algorithm designed for wireless and mobile networks. As a result, it looks after the energy-efficient hardware of handheld and Internet of Things devices. It can support low-power IoT and portable devices operating at their peak efficiency.

• Instead of using ordinary dictionary patterns, the SPT method uses binary patterns, which are compatible with any format that can be translated to binary.

• Binary patterns of the SPT algorithm can help to provide security to the IoT devices since its encrypted key file can be created on low power, energy-efficient devices and safeguard the data transit between two devices.

• By minimising the amount of storage needed for data files, this technique relieves pressure on data centres and helps the effort to combat global warming.

• In order to increase the productivity of mobile and wireless networks without significantly al-tering the network design, this research suggests the SPT algorithm.

• According to the study, SPT enhances data security by encrypting it with non-conventional keys rather than standard dictionary keys.

There are four sections in the research report. The many types of compression are introduced in the first section. We look at related work and other authors' contributions to compression technologies in the second section. The testing and result analysis for the SPT algorithm are covered in the third section. In the fourth section, performance of the SPT algorithm is summarised in the conclusion.

2. Related Work

Euiseok Hwang et al. [1] in his work proposes a lossless compression scheme for time-series smart meter data using bit-back asymmetric numeral systems (BB-ANS). As smart meters become more preva-lent, efficient compression methods are needed to transmit and store large amounts of data. Bit-back cod-ing, which uses Bayesian inference modeling, has been introduced as a novel compression method. When combined with asymmetric numeral systems (ANS), which are stack-like structures, significant compres-sion gains have been observed in several cases. ANS is an approach for entropy coding that combines Huffman coding and arithmetic coding, and has a first-in-last-out (FILO) form suited for bit-back coding. Compared to other compression methods, bit-back coding effectively shares probabilistic models for en-coder and decoder.

Adel Mahmoud et al. [2] proposed a variational auto-encoder scheme for compression. The proposed scheme uses a variational auto-encoder (VAE) to jointly learn approximate posterior and likelihood be-tween message and latent variables, enabling efficient compression of smart meter data within finite time intervals. The scheme is evaluated using an actual smart meter dataset, and the results show that BB-ANS outperforms other state-of-the-art lossless compression schemes. This study is the first to apply bit-back coding to time-series smart metering data, enabling efficient data compression with deep generative mod-els.

Zhaoyi Sun et al. [3] work extensively in the lossless data compression algorithms. In the field of data storage and transmission systems, data compression is a trending topic. Lossless compression is essential for binary files, telemetry data, and high-fidelity medical and scientific images, where it is necessary to retrieve the exact details. However, there is no generic compression algorithm that provides the best com-pression ratio on all data patterns. Authors propose a hybrid lossless hardware architecture that compresses most data patterns such as repeated data, Gaussian distribution data, and images. The proposed design is a highly parallelized architecture that can compress/decompress 64 bytes/cycle with minor overhead. Moreover, it provides high compression ratio on both small and large block sizes. The proposed approach involves profiling-before-compressing and then choosing the right compression hardware.

Rani Nandkishor Aher and Mandaar Pande [4] explains the use of compression in remote sensing. Remote sensing is widely used in applications such as geo-exploration, topographic mapping, and weather forecasting, which produce vast amounts of multi and hyper-spectral image data that require compression. However, the data acquisition process often leads to artifacts in the form of stripes with unpredictable po-sitions and amplitudes, which deteriorate the smoothness of the original image and pose challenges for high-ratio lossless compression. To address this, a split-and-compress framework is proposed in which the image is decomposed into a smooth part and a sparse remainder, capturing the stripes and artifacts alike, and compressing the two parts separately. The decomposition is achieved using a fast, robust statis-tics-based method with linear computational complexity on the number of pixels.

Xizhe Cheng, Sian–Jheng LIN and Jie SUN [5] worked intensively on the importation of the cloud technologies. Organizations are undergoing modernization, and cloud providers are playing a crucial role in this journey. Cost efficiency and ROI are critical factors since there is a significant investment required for modernization. Additionally, there are overheads for maintaining legacy systems until they are fully migrated. Therefore, there is a lot of focus on cost reduction, influenced by storage, compression, and per-formance parameters that directly impact the cost. Authors focuses on studying data compression patterns, specifically BROTLI and ZSTD, for columnar databases. Structured and unstructured compressed data loading, storage, and query execution statistics are demonstrated to explain the impact of data compression on the massive datasets.

Ge Zhang et al. [6] has proposed a scheme for lossless compression for data matrics. A universal scheme for the lossless compression of two-dimensional tables and matrices is proposed in their research. Rather than standard row- or column-based compression, the scheme sorts each column first and records both the sorted table and the corresponding permutation table of the sorting permutations. These two tables are then separately compressed, allowing both intra- and inter-column correlations to be efficiently cap-tured, resulting in improved compression ratios, particularly when both column-wise and row-wise de-pendencies co-occur.

Kanemitsu Ootsu et. Al [7] has proposed in his work that by combining more than one lossless algo-rithms, it is possible to alter speed of compression and data transfer to keep them in sync with network transfer rate. Jinyan Hu et. Al [8] has reviewed many compression image formats. His findings explains that compression Ratio of WebP is better then PNG in lossless compression algorithms. Encoding and Decoding time of Webp compression is also better than PNG Compression. SSIN of WebP is better than PNG while it is comparable to JPEG. Webp might become one of the most popular compressed image format in the near future after JPEG. Masayuki Omote et. Al [9] has reviewed the conventional compres-sion process on the wireless network. His proposed methodology is to use different cores for differenct processes of data compression. His findings proposed that data compresion can be excuted on a different core and data transfer can be setup on a different core on a multiple core processor with the help of a con-troller which keeps track of data transfer speed. It helps in achieving better compression using generic compression algorithms on mobile devices.

D. Engel and A. Unterweger, in their paper', have stressed that lossless compressed on high-frequency data does not yield an equal compression ratio. It has differential compression on different segments of High-frequency data [10]. F. Renault, D. Nagamalai, and M. Dhanuskodi have pointed out that digital im-age processor requires some compression algorithms to enhance the quality of images. It can be a lossless compression algorithm or a lossy compression algorithm [11]. Y. Bi, D. Zhang, and J. Zhao have critically evaluated power quality data and its various parameters. It proposes an algorithm based on high-order data modulation and works on multiple differential operations of different datasets to provide a better com-pression ratio [12]. F. Xiaodong, C. Changling, L. Changling, and S. Huihe proposed in their work that with the help of adaptive recording limit and outliers-detecting rules, swinging door trending can be im-proved to achieve a better compression ratio with the elimination of the outliers [13]. F. Zhang, L. Cheng, X. Li, Y. Sun, W. Gao, and W. Zhao proposed a combination of exception compression and swinging door trending for wide-area measurement systems. This technology can achieve real-time compression for crit-ical live data [14]. H. Li, N. Sheng, and L. Zhi have proposed in their work to compress the PMUs with the waveform difference method. It achieves better compression for data generated by PMUs [15]. J. D. A. Correa, A. S. R. Pinto, C. Montez, and E. Leão proposed swinging door trending technology to identify major compression parameters. It helps in scaling compression speed alongside compression ratio for IoT devices-based data generation [16]. J. E. Tate has proposed new methods for preprocessing phasor angle to help existing techniques for better compression of the PMUs. It reduces data entropy and improves the compression ratio [17]. J. Uthayakumar, T. Vengattaraman, and P. Dhavachelvan have critically evaluated the data compression techniques to observe their impact on data. Various coding schemes have been tested to check their reliability in compressing the data [18].

Table 3. Analysis of the related work.

Reference	Main Contribution	Gaps
Euiseok Hwang et al. [1]	By using bit-back asymmetric numeral systems (BB-ANS), it is possible to achieve better compression ratios	It is suited mainly fir bit back coding only.
Adel Mahmoud et al. [2]	Variational auto-encoder scheme combined with BB-ANS can provide far better compression ratio using deep generative models.	It is bit slower and complex then normal lossless compression algorithms.
Zhaoyi Sun et al. [3]	Hybrid lossless hardware architecture can compresses most data patterns such as repeated data, Gaussian distribution data, and images.	This scheme requires a specic set of hardware to execute its functioning smoothly.
Rani Nandkishor Aher and Mandaar Pande [4]	A split-and-compress framework is proposed in which the image is decomposed into a smooth part and a sparse remainder, capturing the stripes and	Process is complex and can be used mainly for multi and hyper-spectral image data.

	artifacts alike, and compressing the two parts separately.	
5	Common examples of lossy compression algorithm are Discrete Cosine Transform, JPEG and so on.	Common examples of Lossless compression algorithm are Run Length Encoding, Arithmetic encoding and so on.
Xizhe Cheng, Sian–Jheng LIN and Jie Sun [5]	Main area of study is BROTLI and ZSTD, for columnar databases and structured and unstructured compressed data loading as well as storage.	This scheme is generally suitable for massive datasets. Its complexity hinders its use for small datasets.
Ge Zhang et al. [6]	A universal scheme for the lossless compression of two-dimensional tables and matrices is proposed which sorts each column first and records both the sorted table and the corresponding permutation table of the sorting permutations.	This scheme is suitable where there are dependiencies cooccurring between row wise and column wise data.
Kanemitsu OOTSU, Takashi YOKOTA , Takeshi OHKAWA [7]	By integrating three compression algorithms it is possible to alter speed of compression and data transfer to keem them in sync with network transfer rate	Improved Data transfer with respect to network transfer rate can be achieved.
Jinyan Hu, Shaojing Song, Yumei Gong [8]	Compression Ratio of WebP is better then PNG in lossless compression algorithms. Encoding and Decoding time of Webp compression is also better than PNG Compression. SSIN of WebP is better than PNG while it is comparable to JPEG	WebP can be thought of a better substitute for image compression other than JPEG
Masayuki Omote, Kanemitsu Ootsu, Takeshi Ohkawa and Takashi Yokota [9]	Data compresion is excuted on a different core and data transfer is setup on a different core on a multiple core processor with the help of a controller which keeps track of data transfer speed. It helps in achieving better compression using generic	Mobile devices can use generic algoroithms in the future for compression over mobile network.

	compression algorithms on mobile devices.	
Hasitha Muthumala Waidyasooriya, Daisuke Ono, Masanori Hariyama and Michitaka Kameyama [10]	With the application of BPE and BWT (Burrows - Wheeler Transform), there is an increase in 25% of compression ratio for text data files.	Data files with text having different encodings and with linguistic differentiation can be compressed in an efficient way using this compression.

The contribution of numerous authors in the field of compression technologies is summarized in Table 1. It also outlines the research gaps they came up with during their investigation. Some ideas are beneficial for a particular format only. Compressions algorihms which are working flawlessly on images and hyper spectral data might not be efficient for other kinds of datasets. The crux of the research gap is the lack of general compression methods for an extensive range of formats, as well as the inconsistent compression ratios achieved by different techniques for various formats. There is need to device and algorithm which could circumvent all these issues. The SPT compression algorithm takes a leap ahead in resolving these issues by proposing the usage of binary patterns rather than standard dictionary patterns.

3. SPT (Squeeze Pack and Transfer) Algorithm

One of the more recent techniques used in lossless compression is the SPT algorithm. Instead than using conventional text or ASCII code, it searches for recurring patterns in the data file's binary form [29]. It generates key files for the compression and offers codes for common binary patterns. For file decompression, the receiver also receives the same key file. It elevates the SPT algorithm to a renowned algorithm that can virtually compress any format.

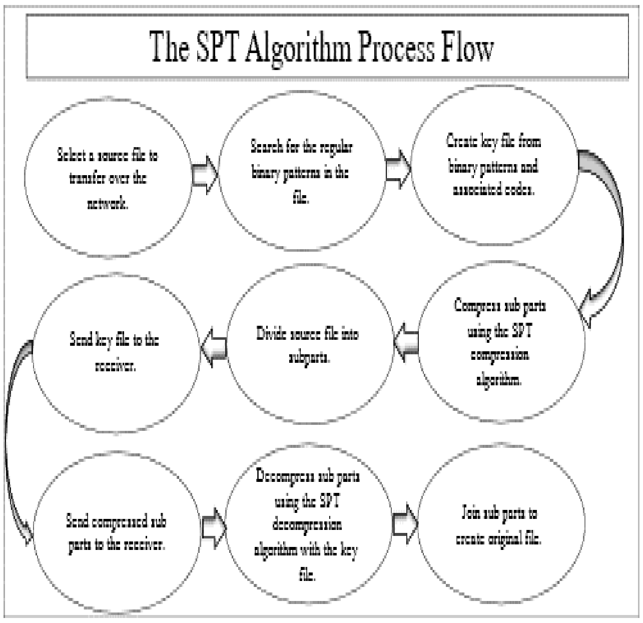

The SPT Algorithm Process Flow
Select a source file to transfer over the network.
Search for the regular binary patterns in the file.
Create key file from binary patterns and associated codes.
Compress sub parts using the SPT compression algorithm.
Divide source file into subparts.
Send key file to the receiver.
Send compressed sub parts to the receiver.
Decompress sub parts using the SPT decompression algorithm with the key file.
Join sub parts to create original file.

Fig 3. process flow of the SPT algorithm.

The SPT algorithm's flow is shown in Figure 3. It begins with the sender choosing a source file, and it is completed when the recipient receives an exact copy of the source file. There are nine steps in this method. Each stage of the algorithm serves as a foundation for the following stage. Each stage of this pro-cess is well defined and described in the pseudo-code.

SPT Algorithms

Algorithm 1: SPT Key Generation Process (first step).

```
Input: Source file from the sender.
Output: Key file for the receiver
SF: Source File
BSF: Binary Source File
Pt: Pattern Buffer
BKF: Binary Key File
bk: Binary Key
Si: Start Index
Lim: Buffer Limit
Ct: Counter
SS: Symbol Store
SSI: Symbol Store Index
FL: File Length
LC: File Length Checker
Rc: Receiver
Begin
SF= fetch_SourceFile()
BSF=Create_BSF(SF)
Pt=get_buffer(BSF,Si,Lim)
Si+=Lim
BKF=Create_KeyFile()
SSI=0
LC=0
FL=length(BSF)
While (LC<=FL)
{
  Ct=0
  While (BSF.EOF==False)
  {
    While (Pt= get_buffer(BSF,Si,Lim))
    {
```

```
    Ct++
    Si+=Lim
  }
  If(Ct>0
  {
    BKF.add(Pt,Symbol(SS,SSI))
    SSI+=1
  }
FL+=Lim
}
Rc.getKeyFile(BKF)End
End
```

The production of a binary key file and its transmission to the recipient are both shown in Algorithm 1. The algorithm first examines recurring binary patterns from the data file. By giving codes to the regular binary patterns discovered, it subsequently builds a binary key file. In the first communication procedure, it delivers the key file to the recipient as a last step.

Algorithm 2: SPT Compression Process (Second step).

```
Input: Source file from the sender.
Output: Compressed file to the receiver
BSF: Binary Source File
BCSF: Binary Compressed Source File
BKF: Binary Key File
bk: Binary Key
Kp: Key Pattern
Rc: Receiver
KSi: Key Start Index
Begin
KSi= 0
While (BSF.EOF==False)
{
BKSF.add(BSF.replace(BKF(Kp[KSi]), BKF(bk[KSi])))
KSi++
}
BSF.Send_To_Receiver(RC)End
End
```

The data file's compression is shown in Algorithm 2. It substitutes codes for patterns in the data file using the key file that algorithm 1 generated. It contributes to the compressed data file's size reduction. The dataset may be compressed more than once using the same key file in some minor variants. It facilitates better compression.

Algorithm 3: SPT Decompression Process (Third step).

```
Input: Compressed file from the receiver.
Output: Decompressed file to the receiver

BSF: Binary Source File
BCSF: Binary Compressed Source File
SF: Source File
BKF: Binary Key File
bk: Binary Key
Kp: Key Pattern
Rc: Receiver
KSi: Key Start Index
Begin
KSi= 0
While (BCSF.EOF==False)
{
    BSF.add(BCSF.replace(BKF(bk[KSi], BKF(Kp[KSi]))
    KSi++
}
SF=Create_SF(BSF)
End
```

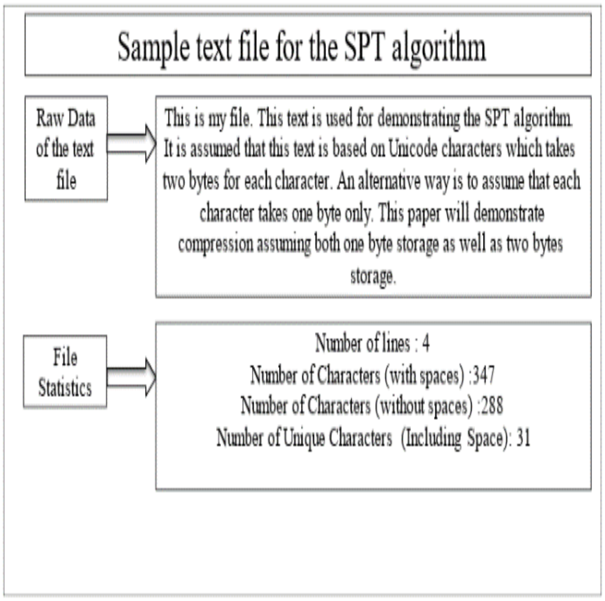
The method of data decompression is described in Algorithm 3. The receiver reconstructs a decompressed copy of the compressed file it got from the sender using the key file it produced in algorithm 1. Again, if multi-ple compression was employed, decompression follows the identical procedures but in the other direction.

The basic process of the SPT algorithm is described in below steps:

- Select source file SF which the sender wants to send to the receiver. It can be of any format.
- Convert source file SF into binary file BSF.
- Find similar/regular binary patterns in the binary file BSF.
- Create a new key file, KF.
- Record regular binary patterns from the binary file BSF to the key file KF.
- Assign codes to the binary patterns in the key file KF. KF now contains binary patterns and their corresponding assigned code. KF= {Pattern1:Code1, Pattern2:Code2, Pattern3:Code3, …. Patternn: Coden}.
- Send key file KF to the receiver.
- Create a compressed source file CSF from binary file BSF using key codes from the file key file KF.
- Split compressed source file CSF into subparts.
- Transfer subparts to the receiver.
- Rejoin the subparts at the receiver end.
- Decompress the rejoined CSF file using the key file sent earlier in step 7.

Implementation

The effectiveness of the SPT algorithm has been described in a notepad file with 347 characters. Figure 2 illustrates its content. Notepad is used to create this file. It has also been demonstrated that the character length of the compressed file can explain how the SPT algorithm functions. This procedure is enclosed in the actual world [30].


Sample text file for the SPT algorithm

Raw Data of the text file

This is my file. This text is used for demonstrating the SPT algorithm. It is assumed that this text is based on Unicode characters which takes two bytes for each character. An alternative way is to assume that each character takes one byte only. This paper will demonstrate compression assuming both one byte storage as well as two bytes storage.

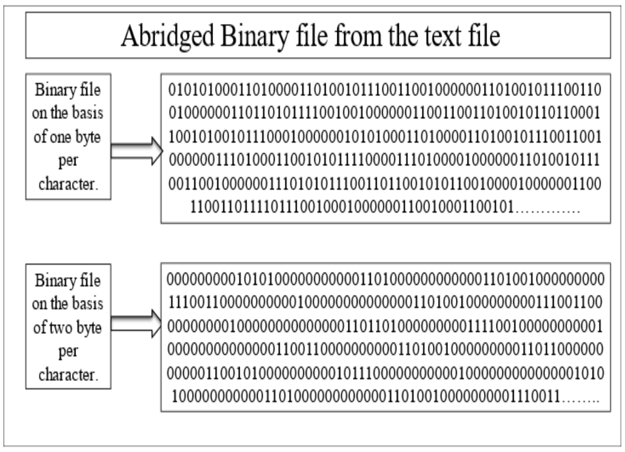
File Statistics

Number of lines : 4
Number of Characters (with spaces) :347
Number of Characters (without spaces) :288
Number of Unique Characters (Including Space): 31

Fig 4. Text file (Sample) for the SPT algorithm.

The SPT algorithm's material is shown in Figure 4. Those 347 characters are in this notepad docu-ment. One byte of RAM is required for each character (two bytes are required for Unicode characters). The 31 unique characters in the file are also shown in Figure 3. Utilising the Python programming language, it has been estimated. To find binary patterns, this example file will be momentarily turned into a binary file. In order to discover recurring binary patterns in larger files, they can be separated for parallel analysis. It facilitates effective load balancing and expedites the dataset's processing so that binary patterns can be found there [31]. A condensed form of the binary file is shown in Figure 3, from which regular binary pat-terns can be extracted to produce a binary key file that will be transmitted to the recipient at the start of the data transmission.

Fig. 5. Binary file (abridged) from the sample text file.



An abbreviated version of the binary file created from the sample text file is shown in Figure 5. This file was created solely for the time being. It is merely described here to show how the SPT algorithm functions. This file will be created in the temporary files of a real-world programme, and once a key file is made from it, memory will be promptly freed up. Figure 3 illustrates binary files developed using both two-byte storage for each character (UNICODE) and only byte storage for each character (ANSI). Because the SPT algorithm also compresses complicated characters, it can be used with both ANSI and UNICODE characters. each character (UNICODE) and only byte storage for each character (ANSI). Because the SPT algorithm also compresses complicated characters, it can be used with both ANSI and UNICODE characters

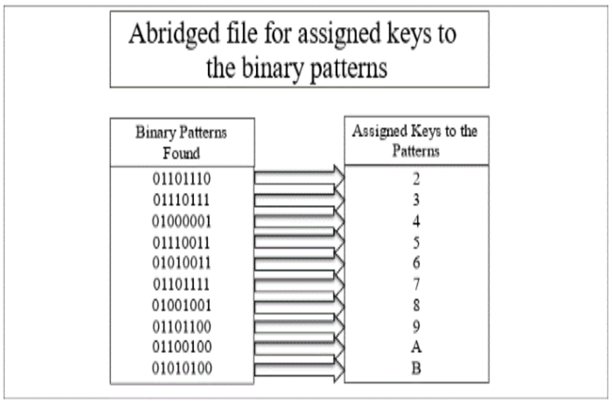


Fig. 6. key file (abridged) for assigned keys to the binary patterns.

The key file's partial content is depicted in Figure 6. Codes are allocated to binary patterns so that they can be substituted in place of entire words. The SPT method can be configured to perform multiple passes in order to generate keys for the specified codes. It also aids in reducing the size of the compressed file and improving compression. Figures 6 and Figure 7 after final compression both depict this procedure. By showing the final compressed version of the data file, both figures demonstrate the effectiveness of the SPT method. It should be emphasised that the final compressed version is merely displayed to illustrate how the SPT algorithm compresses data. This will not be displayed in real-world scenarios as the codes will be encrypted only.

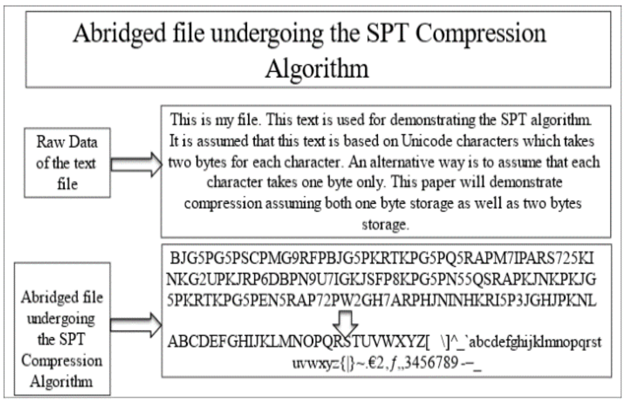

Abridged file undergoing the SPT Compression Algorithm
Raw Data of the text file
This is my file. This text is used for demonstrating the SPT algorithm. It is assumed that this text is based on Unicode characters which takes two bytes for each character. An alternative way is to assume that each character takes one byte only. This paper will demonstrate compression assuming both one byte storage as well as two bytes storage.
Abridged file undergoing the SPT Compression Algorithm
BJG5PG5PSCPMG9RFPBJG5PKRTKPG5PQ5RAPM7IPARS725KI NKG2UPKJRP6DBPN9U7IGKJSFP8KPG5PN55QSRAPKJNKPKJG 5PKRTKPG5PEN5RAP72PW2GH7ARPHJNINHKRI5P3JGHJPKNL
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrst uvwxyz{|}~.€2,ƒ,,3456789 -–_

Fig. **7**. File (abridged) undergoing the SPT Compression Process.

The condensed method for generating the key file for the SPT algorithm is shown in Figure 7. The key file can be constructed in n passes, where n is the number of times the codes will be handled as a string in order to discover the regular patterns in it and build various new keys based on fresh codes. As a result, less memory is required to store the final compressed version.

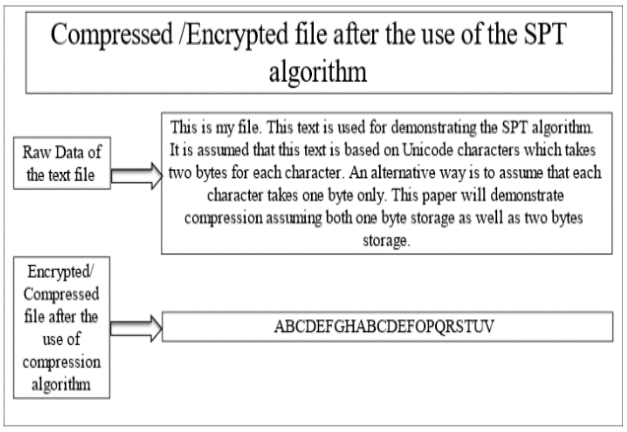

Compressed /Encrypted file after the use of the SPT algorithm
Raw Data of the text file
This is my file. This text is used for demonstrating the SPT algorithm. It is assumed that this text is based on Unicode characters which takes two bytes for each character. An alternative way is to assume that each character takes one byte only. This paper will demonstrate compression assuming both one byte storage as well as two bytes storage.
Encrypted/ Compressed file after the use of compression algorithm
ABCDEFGHABCDEFOPQRSTUV

Fig. 8. Final compressed file undergoing the SPT Compression Process.

This compressed file's final state is shown in Figure 8. It appears at first glance that these are the English language's alpha-bets arranged in alphabetical order. However, if one looks closely, it is possible to spot certain alterations in this pattern. However, it is clear that a file with 347 characters was only compressed to 22 characters. With the aid of the SPT algorithm, storage space is increased by 93.66%.

Analysis of the result

In addition to being able to compress a wide variety of data file types, the SPT algorithm offers a su-perior compression ratio. Any form that can be converted to binary digits (0, 1) can be compressed using the SPT algorithm [28]. The SPT technique attempts to convert the original file into a binary format before compressing it using key patterns derived from regular patterns discovered in the binary file.

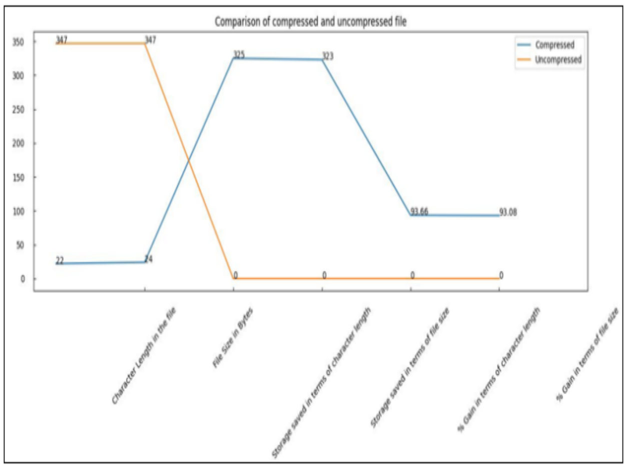


Fig. 9. Statistics of compressed and uncompressed versions of the data file.

Figure 9 displays the statistics for the file's compressed and uncompressed versions. While com-pressing a file of 347 characters, it only uses 22 characters. The uncompressed version's initial file size was 347 bytes, however the SPT Algorithm's compression procedure reduced it to to 24 bytes.

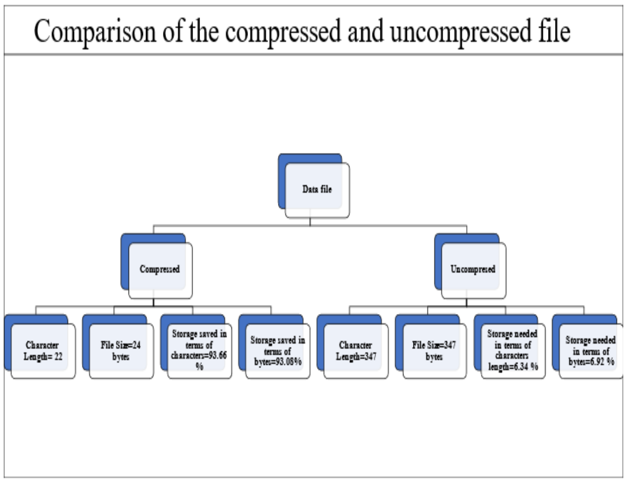

Comparison of the compressed and uncompressed file
Data file
Compressed
Uncompresed
Character Length= 22
File Size=24 bytes
Storage saved in terms of characters=93.66 %
Storage saved in terms of bytes=93.08%
Character Length=347
File Size=347 bytes
Storage needed in terms of characters length=6.34 %
Storage needed in terms of bytes=6.92 %

Fig. 10. In-depth comparison of the uncompressed and compressed files.

The data file's compressed and uncompressed versions are contrasted in Figure 10. It is clear that the SPT algorithm allows data to be compressed by an average of more than 90%. Figure 8 demonstrates a rise in character length of 93.66%. Only 22 characters remain from the original 347 characters in the com-pressed version. Similar to how the memory size of 347 bytes is reduced to 24 bytes, only 93.08% more memory storage is gained. Figure 9 shows that only 6.34% of the actual storage is required for character length, and the remaining characters can be truncated to reduce the file size.

Similar to that, saving the file only uses 6.92% of the available memory. The remaining memory can be retrieved and put to use for other purposes. It is crucial for online storage services like Google Drive, OneDrive, and other similar services to compress files in order to ease the burden on data centres and combat global warming.

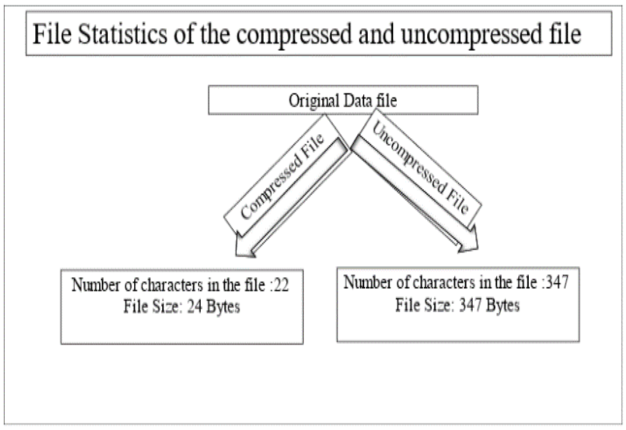

File Statistics of the compressed and uncompressed file
Original Data file
Compressed File
Uncompressed File
Number of characters in the file :22
File Size: 24 Bytes
Number of characters in the file :347
File Size: 347 Bytes

Fig. 11. Line chart for compression parameters.

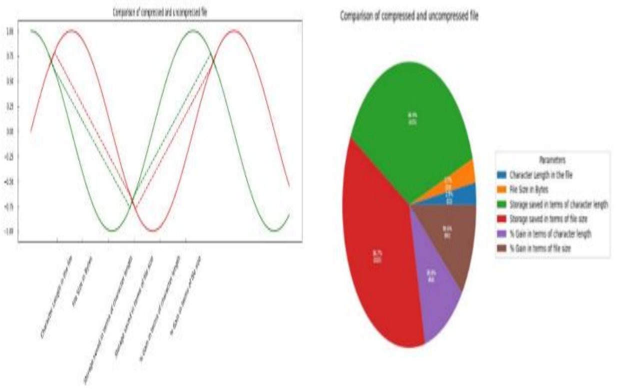


Fig. 12. compression parameters revisited.

The relationship between the causes and consequences of compression settings and their impacts is shown in Figure 12. Input settings for the file's uncompressed version are located on the far right side. Their compressed results are described on the left [35]. This graph makes it clear that during the compression process, the SPT algorithm drastically altered the value of the data file. The figure indicates that the SPT algorithm focuses on character length. It is simple for the SPT algorithm to offer the best compression ratio as a lossless data compression solution once the character length is reduced.

Table 4. Comparison of space and time complexity.

S.No.	Software/ algorithm	Space Complex ity	Time Complexity
1	Deflate	O(n)	O(1)
2	LZSS	O(n)	O(1)
3	LZMA	O(n)	O(1)
4	SPT Algorithm	O(n)	O(1)

Table 4 compares the SPT algorithm's time and space complexity to that of other methods. O(n) space complexity and O(1) time complexity are required for the widely used lossless compression. The RAM required to compress the input file grows linearly in proportion to the input size. The space com-plexity of the Deflate algorithm is O(n), and its time complexity is O(1). Similar to this, the LZSS algo-rithm's time and space complexity are O(n) and O(1), respectively. In terms of space and temporal com-plexity, LZMA (Lempel-Ziv-Markov chain) algorithm also achieves O(n) and O(1). Similar time and space complexity is also achieved by the SPT algorithm implementation, which was developed in JAVA, as shown in Table 7. Table 7 clearly shows that the time and space complexity of the SPT algorithm are comparable to those of other well-known compression methods.

4. Conclusion

This paper discusses in detail the SPT compression algorithm. Textual data was compressed by more than 92% in a test implementation. In the process, it employed three sub-algorithms. (1) Instead of using regular text, binary patterns were used to produce keys. (2) Data is compressed using a key file at the sender's end. (3) At the receiver's end, data is decompressed with the same key file created in step one (1). The SPT algorithm's high compression ratio guarantees it will minimize storage space requirements while enhancing network bandwidth productivity during data transfer. In that regard, it can benefit emerging Internet of Things devices as they have less storage than traditional telecommunications equipment. The SPT algorithm contributes to data security by using non-conventional binary patterns instead of typical dictionary definitions in its key file and enhances the security of handheld and IoT devices. This algorithm is still in its early stages of development, and it will be refined and further optimized soon to provide more excellent service to technology.

References

[1] Euiseok Hwang, "Lossless Data Compression with Bit-back Coding on Massive Smart Meter Data", IEEE, 2022

[2] Adel Mahmoud; Samuel Farid; Mark Maged; Othman Mohamed; Reham Karam; Khaled Salah; M. Watheq El-Kharashi, "An Efficient Hardware Accelerator For Lossless Data Compression", IEEE, 2022

[3] Zhaoyi Sun; Yuliang Huang; Roberto Leonarduzzi; Jie Sun, "A low-complexity destriping method for lossless compression of remote-sensing data", IEEE, 2022

[4] Rani Nandkishor Aher; Mandaar Pande, "Analysis of Lossless Data Compression Algorithm in Columnar Data Warehouse", IEEE, 2022

[5] Xizhe CHENG; Sian–Jheng LIN; Jie SUN, "SortComp (Sort-and-Compress) - Towards a Universal Lossless Compression Scheme for Matrix and Tabular Data", IEEE, 2022

[6] Ge Zhang; Huanyu He; Haiyang Wang; Weiyao Lin, "Integer Network for Cross Platform Graph Data Lossless Compression", IEEE, 2022

[7] Kanemitsu OOTSU, Takashi YOKOTA, Takeshi OHKAWA,"A Consideration on Compres-sion Level Control for Dynamic Compressed Data Transfer Method",IEEE"2016

[8] Jinyan Hu, Shaojing Song, Yumei Gong,"Comparative Performance Analysis of Web Image Compression",IEEE"2017

[9] Masayuki Omote, Kanemitsu Ootsu, Takeshi Ohkawa and Takashi Yokota,"Efficient Data Communication using Dynamic Switching of Compression Method",IEEE"2013

[10]]Hasitha Muthumala Waidyasooriya, Daisuke Ono, Masanori Hariyama and Michitaka Kameyama,"Efficient Data Transfer Scheme Using Word-Pair-Encoding-Based Compres-sion for Large-Scale Text-Data Processing",IEEE"2014

[11] D. Engel and A. Unterweger, "Lossless compression of high-frequency voltage and

[12] current data in smart grids," Proc. IEEE Int. Conf. Big Data, pp. 3131-3139, 2016.

[13] F. Renault, D. Nagamalai and M. Dhanuskodi, "Advances in digital image processing and information technology," Proc. 1st Int. Conf. Digit. Image Process. Pattern Recognit., pp. 23-25, 2011.

[14] Y. Bi, D. Zhang and J. Zhao, "A new data compression algorithm for power quality online monitoring," Proc. Int. Conf. Sustain. Power Gener. Supply, pp. 1-4, 2009.

[15] F. Xiaodong, C. Changling, L. Changling, and S. Huihe, "An improved process data com-pression algorithm," Proc. 4th World Congr. Intell. Control Autom., vol. 3, pp. 2190-2193, 2002.

[16] F. Zhang, L. Cheng, X. Li, Y. Sun, W. Gao and W.

Zhao, "Application of a real-time data compression and adapted protocol technique for WAMS," IEEE Trans. Power Syst., vol. 30, no. 2, pp. 653-662, Mar. 2015.

[17] H. Li, N. Sheng, and L. Zhi, "WAMS/PMU data preprocessing and compression," Adv. Mater. Res., vol. 986/987, pp. 1700-1703, Jul. 2014.

[18] J. D. A. Correa, A. S. R. Pinto, C. Montez, and E. Leão, "Swinging door trending compres-sion algorithm for IoT environments," Proc. Companion Proc. 9th SBESC, pp. 143-148, 2019.

[19] J. E. Tate, "Preprocessing and Golomb–Rice encoding for lossless compression of phasor angle data," IEEE Trans. Smart Grid, vol. 7, no. 2, pp. 718-729, Mar. 2016.

[20] J. Uthayakumar, T. Vengattaraman and P. Dhavachelvan, "A survey on data compression techniques: From the perspective of data quality coding schemes datatype and applications," J. King Saud Univ. - Comput. Inf. Sci., vol. 33, pp. 119-140, 2021.

[21] K. Gibson, D. Lee, J. Choi, and A. Sim, "Dynamic online performance optimization in streaming data compression," Proc. IEEE Int. Conf. Big Data, pp. 534-541, 2018.

[22] K. Zhiwu, X. Rui, L. Xianling and Y. Rui, "Research on lossless compression technique based on running-data of the nuclear power plant," Proc. Int. Conf. Comput. Intell. Commun. Netw., pp. 956-959, 2015.

[23] M. A. Khan, J. W. Pierre, J. I. Wold, D. J. Trudnowski, and M. K. Donnelly, "Impacts of swinging door lossy compression of synchrophasor data," Int. J. Elect. Power Energy Syst., vol. 123, 2020.

[24] M. Cui, J. Wang, J. Tan, A. R. Florita and Y. Zhang, "A novel event detection method using PMU data with high precision," IEEE Trans. Power Syst., vol. 34, no. 1, pp. 454-466, Jan. 2019.

[25] M. H. F. Wen and V. O. K. Li, "Optimal phasor data compression unit installation for wide-area measurement systems—An integer linear programming approach," IEEE Trans. Smart Grid, vol. 7, no. 6, pp. 2644-2653, Nov. 2016.

[26] P. H. Gadde, M. Biswal, S. Brahma and H. Cao, "Efficient compression of PMU data in WAMS," IEEE Trans. Smart Grid, vol. 7, no. 5, pp. 2406-2413, Sep. 2016.

[27] R. Jumar, H. Maaß, and V. Hagemeyer, "Comparison of lossless compression schemes for high rate electrical grid time series for smart grid monitoring and analysis," Comput. Elect. Eng., vol. 71, pp. 465-476, 2018.

[28] R. Klump, P. Agarwal, J. E. Tate, and H. Khurana, "Lossless compression of synchronized phasor measurements," Proc. IEEE PES General Meeting, pp. 1-7, 2010.

[29] R. Wenyu, Y. Timothy and N. Klara, "ISAAC: Intelligent synchrophasor data real-time compression framework for WAMS," Proc. IEEE Int. Conf. Smart Grid Commun., pp. 430-436, 2017.

[30] Shiv Preet, Ashish Kr. Luhach, "Comparison of Various Routing and Compression Algo-rithms: A Comparative Study of Various Algorithms in Wireless Networking," Springer, 2016.

[31] Shiv Preet, Ashish Kr. Luhach, Ravindra, "An overview of the Internet of Things and its Research Issues," ``In International Journal of Computer Technology and Applications," IJCTA, 2016.

[32] Shiv Preet, Dr. Amandeep Bagga " Predefined SPT (Squeeze, Pack And Transfer) Key File Update: A Mapreduce Way Of Automatic Key Updates For SPT Algorithm," ICICCT, 2021.

[33] Shiv Preet, Dr. Amandeep Bagga "Satellite Internet Communication: A Race With Con-temporary Optical Fiber Network with the Help of SPT Algorithm," ICTSGS, 2021.

[34] Shiv Preet, Dr. Amandeep Bagga "Squeeze Pack, and Transfer Algorithm: A new over-the-top compression application for Seamless data transfer over the wireless network," IJITEE, 2019.

[35] Shiv Preet, Dr. Amandeep Bagga," Lempel–Ziv–Oberhumer: A critical evaluation of loss-less algorithm and its applications," ICCS, 2018.

[36] W. Wang et al., "Frequency disturbance event detection based on synchrophasors and deep learning," IEEE Trans. Smart Grid, vol. 11, no. 4, pp. 3593-3605, Jul. 2020.

[37] W. Wang, C. Chen, W. Yao, K. Sun, W. Qiu and Y. Liu, "Synchrophasor data compression under disturbance conditions via cross-entropy-based singular value decomposition," IEEE Trans. Ind. Informat., vol. 17, no. 4, pp. 2716-2726, Apr. 2021.

[38] W. Wang, W. Yao, C. Chen, X. Deng, and Y. Liu, "Fast and accurate frequency response estimation for large power system disturbances using the second derivative of frequency da-ta," IEEE Trans. Power Syst., vol. 35, no. 3, pp. 2483-2486, May 2020.

[39] W. Yao et al., "A fast load control system based on mobile distribution-level phasor meas-urement unit," IEEE Trans. Smart Grid, vol. 11, no. 1, pp. 895-904, Jan. 2020.

[40] X. Wang, Y. Liu, and L. Tong, "Adaptive Subband Compression for Streaming of Continu-ous Point-on-Wave and PMU Data," 2021.

[41] Z. Jellali, L. Najjar Atallah and S. Cherif, "Linear prediction for data compression and re-covery

enhancement in wireless sensor networks," Proc. Int. Wireless Commun. Mobile Comput. Conf., pp. 779-783, 2016

[42] Vijayalakshmi, V., & Sharmila, K. (2023). Secure Data Transactions based on Hash Coded Starvation Blockchain Security using Padded Ring Signature-ECC for Network of Things. International Journal on Recent and Innovation Trends in Computing and Communication, 11(1), 53–61. https://doi.org/10.17762/ijritcc.v11i1.5986

[43] Mark White, Thomas Wood, Maria Hernandez, María González , María Fernández. Enhancing Learning Analytics with Machine Learning Techniques. Kuwait Journal of Machine Learning, 2(2). Retrieved from http://kuwaitjournals.com/index.php/kjml/article/view/184