Computers, Materials & Continua CMC, vol.64, no.3, pp.2013-2030, 2020

A Distributed Covert Channel of the Packet Ordering Enhancement Model Based on Data Compression

Lejun Zhang[1], Tianwen Huang[1], Xiaoyan Hu[1], Zhijie Zhang[1], Weizheng Wang[2], Donghai Guan[3, *], Chunhui Zhao[1, 4] and Seokhoon Kim[5]

Abstract: Covert channel of the packet ordering is a hot research topic. Encryption technology is not enough to protect the security of both sides of communication. Covert channel needs to hide the transmission data and protect content of communication. The traditional methods are usually to use proxy technology such as tor anonymous tracking technology to achieve hiding from the communicator. However, because the establishment of proxy communication needs to consume traffic, the communication capacity will be reduced, and in recent years, the tor technology often has vulnerabilities that led to the leakage of secret information. In this paper, the covert channel model of the packet ordering is applied into the distributed system, and a distributed covert channel of the packet ordering enhancement model based on data compression (DCCPOEDC) is proposed. The data compression algorithms are used to reduce the amount of data and transmission time. The distributed system and data compression algorithms can weaken the hidden statistical probability of information. Furthermore, they can enhance the unknowability of the data and weaken the time distribution characteristics of the data packets. This paper selected a compression algorithm suitable for DCCPOEDC and analyzed DCCPOEDC from anonymity, transmission efficiency, and transmission performance. According to the analysis results, it can be seen that DCCPOEDC optimizes the covert channel of the packet ordering, which saves the transmission time and improves the concealment compared with the original covert channel.

Keywords: Covert channels, information hiding, data compression, distributed system.

1 Research status

Covert channels in the network refer to the use of parts of the network that is not

[1] College of Information Engineering, Yangzhou University, Yangzhou, 225127, China.

[2] Division of Computer Science, University of Aizu, Aizu-Wakamatsu, 9658580, Japan.

[3] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 211106, China.

[4] College of Information and Communication Engineering, Harbin Engineering University, Harbin, 150001, China.

[5] Department of Computer Software Engineering, Soonchunhyang University, Asan, Korea.

[*] Corresponding Author: Donghai Guan. Email: dhguan@nuaa.edu.cn.

Received: 27 April 2020; Accepted: 12 May 2020.

transmitting data to transmit hidden information [Xue, Wang, Zhang et al. (2018)]. Lampson [Lampson (1973)] originally proposed the concept of covert channels in 1973. In 1996, Handel et al. [Handel and Sandford (1996); Wang, Yang, Fu et al. (2016)] introduced covert channels to computer networks for the first time.

Unlike the traditional secret information transmission method, the covert channel not only hides the content of the transmission but also hides the transmission method [Luo, Qin, Xiang et al. (2020)]. The existing network covert channels are covert storage channel [Rios, Onieva, and Lopez (2012); Taheri, Mahdavi and Moghim (2018)] and covert timing channel [García, Zunino and Campo (2014); Wu, Wang, Ding et al. (2012); Zhang, Liang, Zhang et al. (2018)].

This paper mainly studies the covert timing channel. The covert timing channel refers to the sender embedding information into time-related parameters. The receiver and sender receive and send hidden information through pre-set rules, that is, time parameters such as rate of change, order, and interval.

The normal network communication channel has its regularity, while the covert timing channel transmits hidden information by changing the interval of packets send. The covert timing channel may change the statistical characteristics of packets in network communication. The traditional detection method uses this statistical feature to detect the covert timing channel.

Therefore, the statistical detection method is generally used to distinguish the covert timing channel and the normal channel. The current common methods for detecting the covert timing channel are information entropy method, Compressibility-Walk method, ranking method [Wang, Ju, Zhou et al. (2009)], etc. The detection methods of covert channels in the network are essentially the statistical analysis of the time distribution of data packets.

The reason why these detection methods are successful is that the existing covert timing channel will change the time distribution of packets in the communication process. It makes the covert timing channel deviate from the statistical characteristics of normal network communication seriously [Liu, Zhai and Dai (2012)].

This model uses a lossless compression algorithm to reduce the transmission time and weaken the statistical probability of hidden information. The distributed system can hide data and deviate time distribution characteristics in this model.

2 Related works

2.1 The covert channel of the packet ordering

The principle of covert channel of the packet ordering is to obtain the corresponding secret information from the time sequence of data packets. The sender and the receiver first specify multiple transmit ports and one receive port, and they establish connections in turn. The receiver looks up the mapping table in the order in which the packets arrive to obtain the secret data.

If n bits of data are transmitted in each round, and the communication parties establish m connections. There are $m!$ different possible situations. If the port sorting can fully express the transmitted data, it must meet the requirement of $m! \geq 2^n$. Therefore, the relationship between the transmission of n bits of data per round and the number of connections m established by the communicating parties is:

$$n = \lfloor \log_2 m! \rfloor \tag{1}$$

Fig. 1 shows an example of the covert channel of the packet ordering. According to the different arrival order of port data, the receiver obtains different data by retrieving the mapping table.

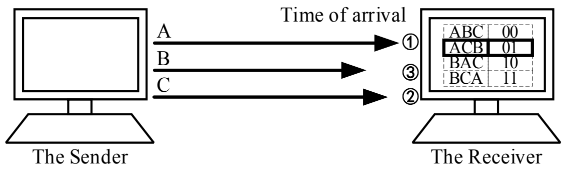


Figure 1: An example of the covert channel of the packet ordering

2.2 On\off covert channel

In a fixed time interval, the receiver can identify the bit "1" or bit "0" by judging whether there is a packet arriving, so as to obtain secret information. The sender and receiver must specify a fixed time interval, duration and synchronization method in on\off covert channel. Different from the covert channel of the packet ordering model, the on\off covert channel only takes advantage of whether packets arrive within a period of time to obtain the hidden information and can transmit the hidden information in a relatively secret way. On\off covert channel model, as shown in Fig. 2. The sender and receiver define a fixed time, the receiver begins to receive secret information after considering the transmission delay. When there is no data transmission within the specified time, it indicates that the hidden information transmitted by the sender is bit "0", and when there is data transmission within the specified time, it indicates that the hidden information transmitted by the sender is bit "1".

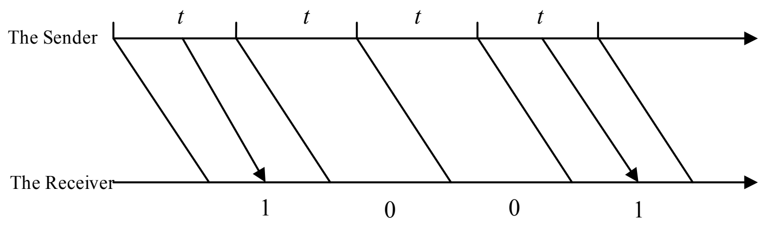


Figure 2: on\off covert channel model

2.3 Data compression

Data compression first appeared in the early 19th century. With the rise of computers, data compression gradually played an important role in the computer field. Compression algorithms are mainly divided into lossless compression and lossy compression. Lossless compression takes up more space and has lower compression than lossy compression, but it retains all the original information without any data loss. Some image formats, such as PNG, use lossless compression. Lossless compression is suitable for situations where the compressed and uncompressed data must be consistent with the original data. Lossy compression refers to discarding nonessential information and sacrificing some quality to reduce data volume and improve compression ratio. Lossy compression is mainly used in streaming media and Internet telephone.

After years of development, various compression algorithms come out in an unending flow. Typical lossless compression algorithms include Huffman coding [Najmabadi, Tran, Eissa et al. (2019)], LZ77 compression coding, LZMA compression coding, etc.

The compression algorithm can reduce the storage space and transmission time of secret information and weaken the statistical probability of secret information. When part of the secret information is intercepted, the content of the secret information cannot be analyzed and predicted [Xiang, Wu, Li et al. (2018)].

The commonly used lossless compression codes can be divided into entropy coding, dictionary coding and run-length encoding. In the process of string processing, entropy coding usually needs to know the distribution of data in the whole string file in advance. In other words, entropy coding knows how often characters appear before compressing strings or files. But dictionary coding can be compressed more efficiently without knowing the files. Run-length encoding is often used to process binary images.

2.4 Distributed system

The distributed system is a loosely coupled system in which communication lines interconnect several processors. From one processor in the system, the other processors and the corresponding resources are remote, and only its resources are local. So far, the definition of distributed system has not formed a unified view. Generally speaking, the distributed system should have the following four characteristics:

(a). Distribution. Distributed systems consist of multiple computers that are geographically dispersed and can be distributed in a unit, city, country, or even globally. The function of the whole system is distributed in each node, so the distributed system has the distribution of data processing.

(b). Autonomy. Each node in the distributed system contains its processor and memory, and each node has its function of processing data.

(c). Parallelism. A large task can be divided into subtasks that are executed on different hosts.

(d). Globality. There must be a single, global process communication mechanism in a distributed system, so that any process can communicate with other processes, and it does not distinguish between local communication and remote communication.

Time synchronization in a distributed system is a critical issue. Time synchronization can ensure that all nodes in the distributed system work together [Lamport (1978)]. There are many classical time synchronization mechanisms such as receive-receive synchronization mechanism, two-way synchronization mechanism based on send-receive, and the synchronization mechanism based on send-receive. Based on these classic time synchronization mechanisms [Jiang, Chen and Hu (2017); Zhang and Zhang (2012)], many time synchronization algorithms have been born, such as RBS [Elson, Girod and

Estrin (2002)], TPSN [Ganeriwal, Kumar and Srivastava (2003)], FTSP [Maróti, Kusy, Simon et al. (2004)].

Table 1: Comparison of common time synchronization algorithms

Synchronization algorithm	**Characteristic**	**Applicable environment**
RBS	Time adjustment does not affect the calculation of time deviation. The complexity is high.	Wireless networks, wired networks
TPSN	High accuracy, can be accurate to hundreds of microseconds, not suitable for dynamic network.	Suitable for networks with high time accuracy requirements
FTSP	The time accuracy is higher than the RBS algorithm and lower than the TPSN algorithm.	Suitable for military occasions
DMTS	Time accuracy is lower than RBS, TPSN and FTSP, but the algorithm is flexible.	Wireless sensor networks with low time accuracy requirements

Sundararaman et al. [Sundararaman, Buy and Kshemkalyani (2005)] compared and analyzed multiple time synchronization algorithms. Different time synchronization algorithms have different design ideas. Some algorithms are designed to save energy, others to improve time accuracy.

3 DCCPOEDC model

This paper proposes a distributed covert channel of the packet ordering enhancement model based on data compression. The covert channel of the packet ordering is applied to the distributed system. In this type of system, the sender is divided into a master and child nodes, the master communicates with each node, and each node communicates with the receiver to transmit secret information.

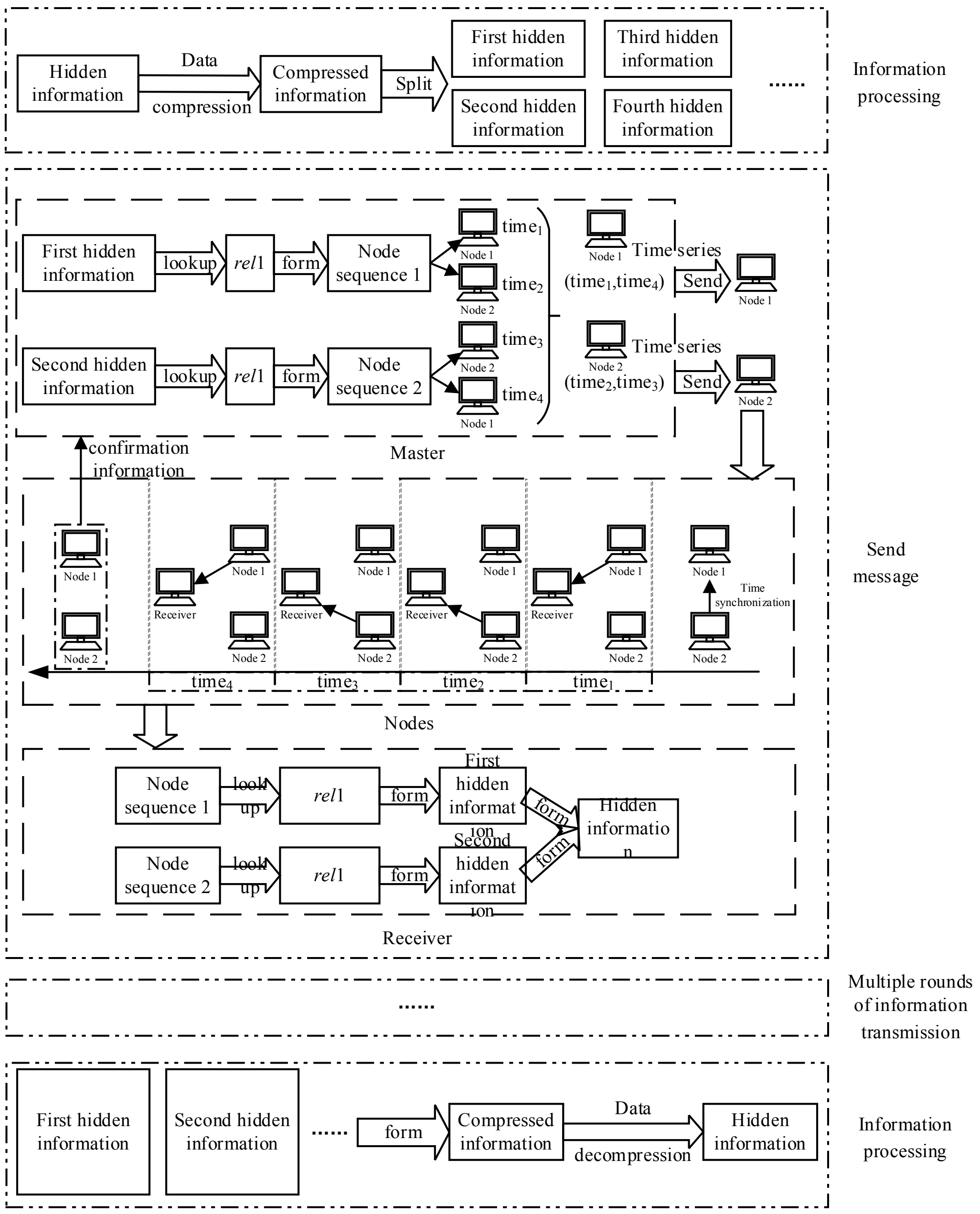


Figure 3: DCCPOEDC model

Fig. 3 is DCCPOEDC model, the specific hidden messages steps are as follows:

First, the data compression algorithm is used to compress the characters. The master needs to define the number of child nodes, determine the time interval between sending bit information and data every round according to the number of nodes, and establish a relationship 1.

Second, the master uses multiple rounds of secret information retrieval relationship 1 to get multiple groups of node sequences. Define the sending time of these multiple groups of nodes. The time series corresponding to each node can be obtained. The master sends the corresponding time series of each node to the relevant node and continues to communicate secret information when receiving the confirmation information transmitted by each node.

Third, after the nodes receive the time series sent by the master. They synchronize their time. After the synchronization is completed, the relevant nodes send the information to the receiver at the specified time. When the time series is sent, each node sends the confirmation information to the master.

Fourth, the receiver receives the information sent by the nodes. It will get multiple sets of node sequences. The receiver searches for relationship 1 to obtain multiple rounds of secret information sent by the master. The sender combines these groups of information into a complete secret information. Then the sender decompresses the information to get the original information.

3.1 DCCPOEDC sender algorithm

The model needs to determine the relationship $rel1$ between the sending order of m nodes in the distributed system and the n bits hidden information. The specific encoding method is: the m nodes are N_1, N_2,..., N_m, the hidden information sent in each round is $\underbrace{00\cdots0}_{n} \sim \underbrace{11\cdots1}_{n}$. Therefore, it is possible to establish the relationship $rel1$ between the sending order of nodes and the secret information: $N_1N_2\cdots N_{m-1}N_m = \underbrace{00\cdots0}_{n}$, $N_1N_2\cdots N_mN_{m-1} = \underbrace{00\cdots1}_{n}$,..., $N\cdots N = \underbrace{11\cdots1}_{n}$.

When the distributed master sends information, it uses the data compression algorithm to perform character compression processing on $originmsg$ to obtain $compmsg$. According to the relationship $rel1$, the master obtains the sending order of each child node in the distributed system corresponding to the n bits hidden information transmitted in each round. The master will obtain the sending order of each child node according to the hidden information content, that is, after each certain child node finishes sending information, it waits for the next child node to send information within a specified time. When the master performs multiple rounds of lookups, it will get multiple sending times for different child nodes. The master makes up a two-dimensional time series with multiple sending times corresponding to different nodes. That is, each child node will have a complete-time series to represent this hidden information. The master sends the relevant time series to the corresponding child nodes. This way of communication can avoid frequent communication between the master and the child nodes, and improve the privacy of the transmission channel.

The algorithm of the master of this model is as follows:

Algorithm 1 DCCPOEDC master hidden algorithm

Input: DCCPOEDC m nodes,n bits hidden information, relationship $rel1$

Output: Send message

```
1: function MAIN( )
2:     read $n, m, rel1$ \\Read port number, bits , relationship
3:     $socketconnet(m)$ \\$m$ nodes establish socket connections with the master
4:     $sendmsg(startframe)$ \\Start frame sending information
5:     $compmsg \leftarrow$ COMPRESS(message)\\Compression of hidden information
6:     while $*compmsg \neq ' \backslash 0'$ do \\Read the content to be sent until the end
7:         for $i \leftarrow 1$ to $h$ do
8:             $strncpy(cmsg, compmsg, n)$ \\Read $n$ bits of secret information
9:             $node[] \leftarrow findfrommap(cmsg, rel1)$ \\Read node sequence in $rel1$
10:            for $a \leftarrow 1$ to $m$ do
11:                $tseries[node[a]][i] \leftarrow time$ \\Assigning timetables
12:                $time \leftarrow time + RAND(timeinterval)$ \\Time interval is random
13:            end for
14:        end for
15:        for $b \leftarrow 1$ to $m$ do
16:            $sendInf(node[b], tseriers[b][h])$ \\Send the specified time series to all nodes
17:        end for
18:    end while
19:    $closeallconnect(m)$
```

The encoding hiding process at the sender is as follows:

(a). The master uses the data compression algorithm to convert the hidden information $originmsg$ into the compressed hidden information $compmsg$.

(b). The master divides each n bits of the hidden information $compmsg$, and obtains the transmission sequence $node[]$ according to the relationship $rel1$.

(c). After performing h operations on step (b), the master will obtain h transmission times of n child nodes in the distributed system, and combine the h transmission times to form time series $tseries[m][h]$. The time series corresponding to m child nodes are $tseries[0][]$, $tseries[1][]$,…, $tseries[m-1][]$.

(d). The master sends the time series $tseries[0][]$, $tseries[1][]$,…, $tseries[m-1][]$ to the corresponding child nodes.

(e). After receiving the time series, m child nodes establish socket connections with the designated port of the receiver, and synchronize time with node 1. Each child node then sends information according to time series $tseries[0][]$, $tseries[1][]$,…, $tseries[m-1][]$.

(f). When the child node finishes sending, it sends confirmation information to the master. After receiving the confirmation information from all the child nodes, the master transmits the next round of hidden information.

3.2 DCCPOEDC receiver algorithm

The receiver is the reverse process of sending secret information by the sender. After receiving the information of the child nodes, the node sequence is formed according to the time when the data packet arrives. The relationship *rel*1 is searched to obtain the secret information.

The specific decoding process at the receiver is as follows:

(a). The receiver composes a node sequence *node*[] according to the arrival time of the data packet, and searches for the relationship *rel*1 according to the node sequence *node*[], and converts it into n bits secret information *crecvmsg* .

(b). After receiving the secret message, the receiver uses a data compression algorithm to decompress to obtain the hidden information *originmsg* .

Algorithm 2 DCCPOEDC receiver decoding algorithm

Input: Number of m nodes,Relationaship $rel1$,Listening port $lisport$

Output: Hidden information $originmsg$

```
1:  function MAIN
2:      read $m, rel1$ \\Read port number, relationship
3:      $bind(listeningport)$
4:      while 1 do
5:          $recvfrom(clientport)$
6:          ADDNODE($node[], clientport$) \\Append node to node sequence
7:          if $strlen(node[]) = m$ then
8:              $crecvmsg \leftarrow findfrommap(node[], rel1)$ \\Read secret information in $rel1$
9:              $crecvmsg \leftarrow crecvmsg + crecvmsg$
10:             $node[] \leftarrow '\backslash 0'$
11:         end if
12:     end while
13:     $originmsg \leftarrow$ UNCOMPRESS(crecvmsg) \\ $crecvmsg$ is decompressed to $originmsg$
14: end function
```

4 Experimental results and analysis

4.1 Selection of compression algorithm

To comprehensively compare the application effects of various algorithms in this model, the main performance indexes include compression rate, compression time, and system complexity.

The original file size is FS , and the file size is FS' after using the compression algorithm. The compression ratio R_C of the compression algorithm is:

$$R_C=\frac{FS'}{FS}\bullet 100\% \tag{2}$$

When the original file size FS is unchanged, the smaller the file size FS' after compression, the lower the compression rate R_C, and the better the performance of the compression algorithm.

In this paper, several common compression algorithms are selected for testing. In order to be able to measure various compression algorithms, we use Eq. (3) to measure all compression algorithms.

$$P_C=\frac{1}{2t_C}+\frac{1}{2R_C} \tag{3}$$

The test data is from hidden information of a fixed size, and different algorithms are used to process the same data. The test algorithm uses the functions in each algorithm library to compress the data read into memory. The results are shown in Tab. 2.

Table 2: Data compression results

Data compression algorithm	**Data original size (MB)**	**Compressed size (MB)**	**Compression ratio**	**Compression time (s)**	**Performance value**
Huffman	500	126	24.2%	240	0.02
RLE	500	158	31.6%	30	0.03
Gzip	500	13	2.6%	46	0.20
LZMA	500	9	1.8%	120	0.28
Zlib	500	14	2.8%	8	0.24

It can be seen that because the algorithms of Huffman encoding and RLE encoding are relatively simple, their compression effect are not good. At the same time, Gzip, LZMA, and Zlib have higher compression effect. Combining the performance value of the compression algorithm, it can be obtained that the LZMA compression algorithm can achieve a good balance in compression time and compression rate, which meets the needs of this model.

Igor Pavlov invented the LZMA compression algorithm in 1998. LZMA use a dictionary encoding mechanism. This compression algorithm is the default compression algorithm for the 7z format in the 7-Zip program. 7-zip provides the LZMA software development kit. LZMA is a variant of LZ77 compression algorithm. LZMA combines sliding window, dictionary compression algorithm and interval coding in LZ77, it has the advantages of high compression rate, small space requirement for decompression and fast speed.

The LZMA algorithm supports a dictionary space of 4 KB to hundreds of MB. Dictionary improves the compression effect, but leads to a large search cache space.

In the implementation of the LZMA algorithm, several possible longest matches are stored in a hash list, and the data structure of a Hash chain table or a binary lookup tree is used to find the matching data [Leavline and Singh (2013)]. This method reduces the time required to match the longest string and quickly searches for matching characters.

The encoding process of LZMA is similar to the encoding process of DEFLATE. Both of the two compression codes use the sliding window and dictionary compression algorithm in LZ77 encoding, but LZMA USES the interval encoding to improve the compression performance of LZMA [Hübbe, Wegener, Kunkel et al. (2013)].

The LZMA encoding process is as follows:

(a). Write compressed data to the cache;

(b). Read the dictionary into the cache and perform sliding matching. If it is successful, go to step (e), otherwise continue to the next step;

(c). Write '0' (character match failed) into the output stream;

(d). Update the sliding cache data and go to step (f);

(e). Output flag bits and match information;

(f). Interval coding the data;

(g). If there is uncompressed data, go to step (b);

(h). End data compression.

4.2 Analysis of the information transmitted by the average node

The number of distributed nodes is m, which corresponds to m nodes, and the number of receiving ports is one. When transmitting hidden information, the m nodes are sorted according to the hidden information, and then the sender sends data packets to the receiver. The receiver listens to the receiving port, receives data packets sent by m different nodes. Secret information is obtained according to the arrival order of packets from different nodes.

In each round of information transmission, it is necessary to determine the n bits secret information corresponding to the sequence of nodes. The relationship between m nodes and n bits hidden information satisfies the Eq. (1) and is $n = \lfloor \log_2 m! \rfloor$.

Due to the introduction of a compression algorithm, the sender needs to compress the hidden information. Therefore, the compression rate CR of the compression algorithm need to be taken into account when calculating the information transmitted by the average node. It can be obtained that the information transmitted by the average node of this model is: $\frac{\lfloor \log_2 m! \rfloor}{CR \bullet m}$.

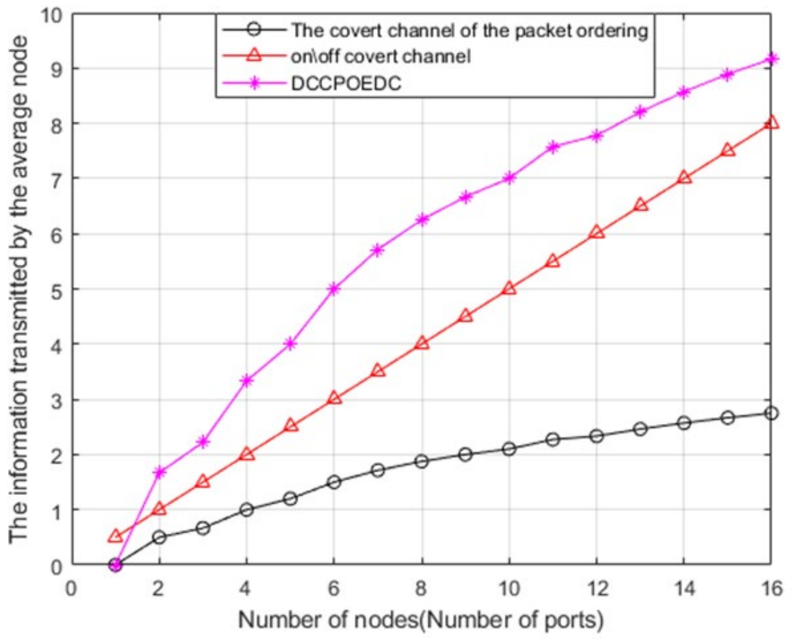
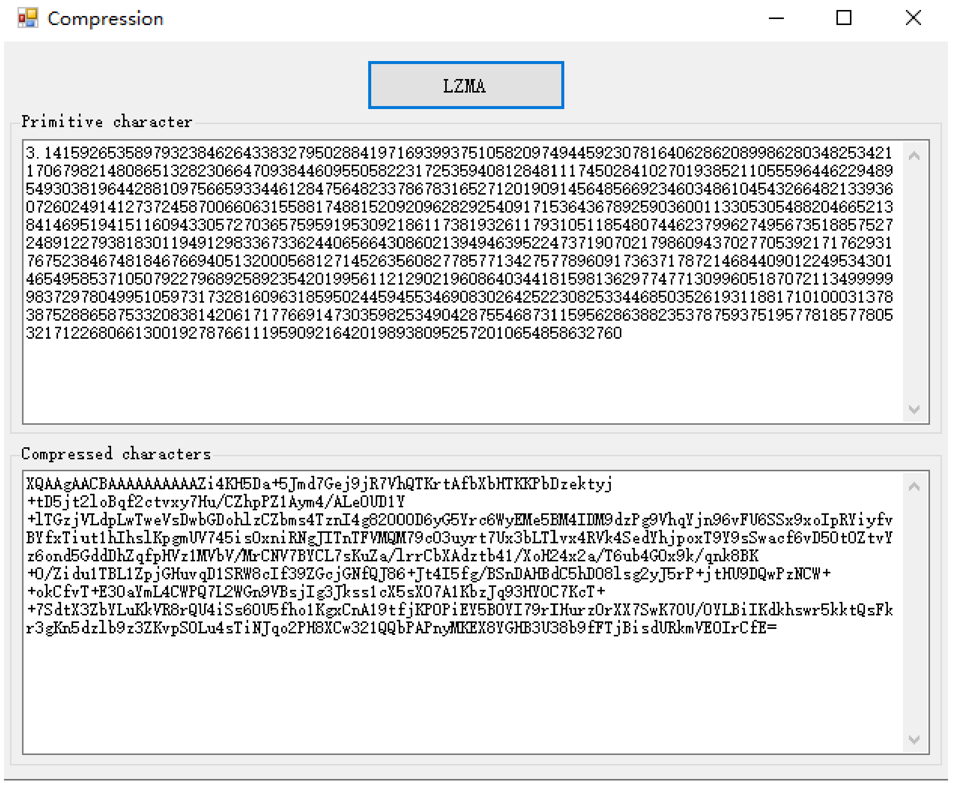


Figure 4: Relationship between information transmitted by average node and number of nodes

The compression ratio of the compression algorithm depends on the probability distribution of the file content. Therefore, the compression rate is 30% during the simulation. In Fig. 4, it can be seen that when the number of nodes (ports) is the same, the total number of hidden information transmitted by this model is much higher than the covert channel of the packet ordering and on\off covert channel. As the number of nodes (ports) increases, the average amount of hidden information that each node (port) can also transmit increases.

4.3 Analysis of concealment

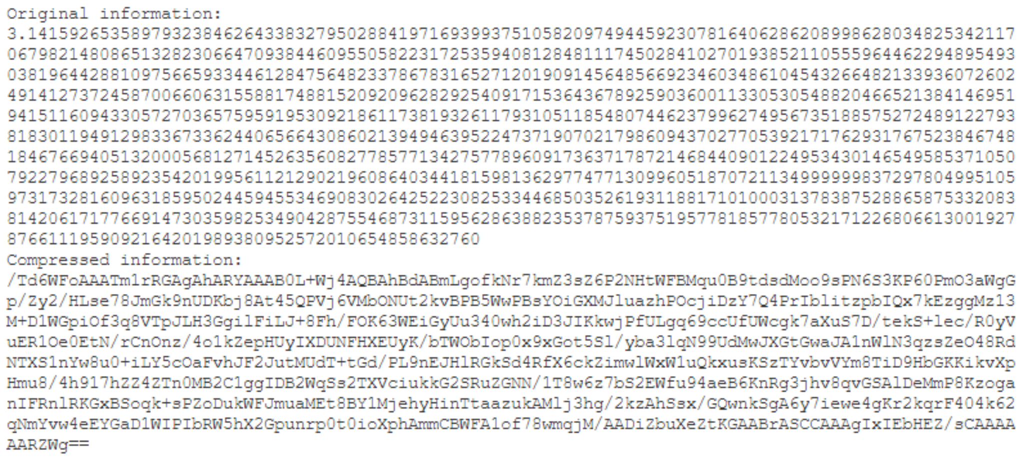
The LZMA compression algorithm uses a variable dictionary encoding in the compression process. The results of using the LZMA compression on the same information in different SDKs are different because different dictionary encodings are used in the compression. A different dictionary cannot decompress the compressed content of another dictionary. When the dictionary is not leaked, the protection of the original secret information can be realized to a certain extent. And the size of the sent hidden information can be reduced. Fig. 5 shows the results before and after compressing text in the SevenZip library. Fig. 6 shows the results before and after compressing text in the lzma.compress library. From Figs. 5 and 6, we can see that the compression results are different with different function libraries. Although the SevenZip [Chrishwang (2019)] and lzma.compress [Python Software Foundation (2020)] libraries both encapsulate the official LZMA SDK [Pavlov (2019)], these different libraries use different dictionary sizes and state bits for the default compression process. It results in a different compression result.



```
Primitive character
3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421
170679821480865132823066470938446095505822317253594081284811174502841027019385211055596446229489
549303819644288109756659334461284756482337867831652712019091456485669234603486104543266482133936
072602491412737245870066063155881748815209209628292540917153643678925903600113305305488204665213
841469519415116094330572703657595919530921861173819326117931051185480744623799627495673518857527
248912279381830119491298336733624406566430860213949463952247371907021798609437027705392171762931
767523846748184676694051320005681271452635608277857713427577896091736371787214684409012249534301
465495853710507922796892589235420199561121290219608640344181598136297747713099605187072113499999
983729780499510597317328160963185950244594553469083026425223082533446850352619311881710100031378
387528865875332083814206171776691473035982534904287554687311595628638823537875937519577818577805
321712268066130019278766111959092164201989380952572010654858632760

Compressed characters
XQAAgAACBAAAAAAAAAAZi4KH5Da+5Jmd7Gej9jR7VhQTKrtAfbXbHTKKPbDzektyj
+tD5jt2loBqf2ctvxy7Hu/CZhpPZ1Aym4/ALeOUD1Y
+lTGzjVLdpLwTweVsDwbGDohlzCZbms4TznI4g82OOOD6yG5Yrc6WyEMe5BM4IDM9dzPg9VhqYjn96vFU6SSx9xoIpRYiyfv
BYfxTiut1hIhslKpgmUV745isOxniRNgJITnTFVMQM79cO3uyrt7Ux3bLTlvx4RVk4SedYhjpoxT9Y9sSwacf6vD5OtOZtvY
z6ond5GddDhZqfpHVz1MVbV/MrCNV7BYCL7sKuZa/lrrCbXAdztb41/XoH24x2a/T6ub4GOx9k/qnk8BK
+O/Zidu1TBL1ZpjGHuvqD1SRW8cIf39ZGcjGNfQJ86+Jt4I5fg/BSnDAHBdC5hDO8lsg2yJ5rP+jtHU9DQwPzNCW+
+okCfvT+E3OaYmL4CWPQ7L2WGn9VBsjIg3Jkss1cX5sXO7A1KbzJq93HYOC7KcT+
+7SdtX3ZbYLuKkVR8rQU4iSs6OU5fho1KgxCnA19tfjKPOPiEY5BOYI79rIHurzOrXX7SwK7OU/OYLBiIKdkhswr5kktQsFk
r3gKn5dzlb9z3ZKvpSOLu4sTiNJqo2PH8XCw321QQbPAPnyMKEX8YGHB3U38b9fFTjBisdURkmVEOIrCfE=
```

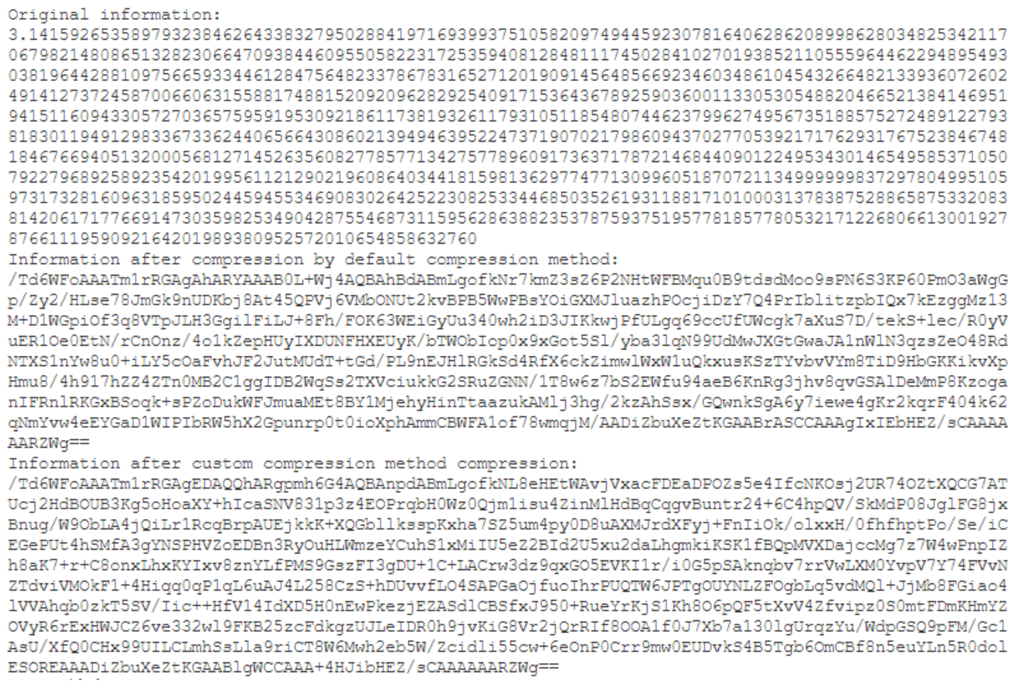
Figure 5: SevenZip library compression results



```
Original information:
3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421170
679821480865132823066470938446095505822317253594081284811174502841027019385211055596446229489549303
819644288109756659334461284756482337867831652712019091456485669234603486104543266482133936072602
491412737245870066063155881748815209209628292540917153643678925903600113305305488204665213841469519
415116094330572703657595919530921861173819326117931051185480744623799627495673518857527248912279
381830119491298336733624406566430860213949463952247371907021798609437027705392171762931767523846748
184676694051320005681271452635608277857713427577896091736371787214684409012249534301465495853710507
922796892589235420199561121290219608640344181598136297747713099605187072113499999983729780499510597
317328160963185950244594553469083026425223082533446850352619311881710100031378387528865875332083
814206171776691473035982534904287554687311595628638823537875937519577818577805321712268066130019278
766111959092164201989380952572010654858632760
Compressed information:
/Td6WFoAAATm1rRGAgAhARYAAAB0L+Wj4AQBAhBdABmLgofkNr7kmZ3sZ6P2NHtWFBMqu0B9tdsdMoo9sPN6S3KP60PmO3aWgG
p/Zy2/HLse78JmGk9nUDKbj8At45QPVj6VMbONUt2kvBPB5WwPBsYOiGXMJluazhPOcjiDzY7Q4PrIblitzpbIQx7kEzggMz13
M+D1WGpiOf3q8VTpJLH3GgilFiLJ+8Fh/FOK63WEiGyUu340wh2iD3JIKkwjPfULgq69ccUfUWcgk7aXuS7D/tekS+lec/R0yV
uER1Oe0EtN/rCnOnz/4o1kZepHUyIXDUNFHXEUyK/bTWObIop0x9xGot5Sl/yba3lqN99UdMwJXGtGwaJA1nWlN3qzsZeO48Rd
NTXS1nYw8u0+iLY5cOaFvhJF2JutMUdT+tGd/PL9nEJHlRGkSd4RfX6ckZimwlWxW1uQkxusKSzTYvbvVYm8TiD9HbGKKikvXp
Hmu8/4h917hZZ4ZTn0MB2C1ggIDB2WqSs2TXVciukkG2SRuZGNN/1T8w6z7bS2EWfu94aeB6KnRg3jhv8qvGSAlDeMmP8Kzoga
nIFRnlRKGxBSoqk+sPZoDukWFJmuaMEt8BY1MjehyHinTtaazukAMlj3hg/2kzAhSsx/GQwnkSgA6y7iewe4gKr2kqrF404k62
qNmYvw4eEYGaD1WIPIbRW5hX2Gpunrp0t0ioXphAmmCBWFA1of78wmqjM/AADiZbuXeZtKGAABrASCCAAAgIxIEbHEZ/sCAAAA
AARZWg==
```

Figure 6: Lzma.compres library compression results

Modifying the LZMA preset compression level or custom compression method can also produce different compression results [Python Software Foundation (2020)]. For example, customize the compression method in lzma.compress, modify the differences between adjacent bytes to "5", and change the compression preset to "7". It will produce a different compression result than the default compression method. Fig. 7 shows the compression results of the default compression method and the custom compression method. The results show that different compression methods have different compression results for the same information content.



```
Original information:
3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117
0679821480865132823066470938446095505822317253594081284811174502841027019385211055596446229489549З
03819644288109756659334461284756482337867831652712019091456485669234603486104543266482133936072602
49141273724587006606315588174881520920962829254091715364367892590360011330530548820466521384146951
94151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793
81830119491298336733624406566430860213949463952247371907021798609437027705392171762931767523846748
18467669405132000568127145263560827785771342757789609173637178721468440901224953430146549585371050
79227968925892354201995611212902196086403441815981362977477130996051870721134999999837297804995105
97317328160963185950244594553469083026425223082533446850352619311881710100031378387528865875332083
81420617177669147303598253490428755468731159562863882353787593751957781857780532171226806613001927
8766111959092164201989380952572010654858632760
Information after compression by default compression method:
/Td6WFoAAATm1rRGAgAhARYAAAB0L+Wj4AQBAhBdABmLgofkNr7kmZ3sZ6P2NHtWFBMqu0B9tdsdMoo9sPN6S3KP60PmO3aWgG
p/Zy2/HLse78JmGk9nUDKbj8At45QPVj6VMbONUt2kvBPB5WwPBsYOiGXMJluazhPOcjiDzY7Q4PrIblitzpbIQx7kEzggMz13
M+D1WGpiOf3q8VTpJLH3GgilFiLJ+8Fh/FOK63WEiGyUu340wh2iD3JIKkwjPfULgq69ccUfUWcgk7aXuS7D/tekS+lec/R0yV
uER1Oe0EtN/rCnOnz/4o1kZepHUyIXDUNFHXEUyK/bTWObIop0x9xGot5Sl/yba3lqN99UdMwJXGtGwaJA1nWlN3qzsZeO48Rd
NTXS1nYw8u0+iLY5cOaFvhJF2JutMUdT+tGd/PL9nEJHlRGkSd4RfX6ckZimwlWxW1uQkxusKSzTYvbvVYm8TiD9HbGKKikvXp
Hmu8/4h917hZZ4ZTn0MB2C1ggIDB2WqSs2TXVciukkG2SRuZGNN/1T8w6z7bS2EWfu94aeB6KnRg3jhv8qvGSAlDeMmP8Kzoga
nIFRnlRKGxBSoqk+sPZoDukWFJmuaMEt8BY1MjehyHinTtaazukAMlj3hg/2kzAhSsx/GQwnkSgA6y7iewe4gKr2kqrF404k62
qNmYvw4eEYGaD1WIPIbRW5hX2Gpunrp0t0ioXphAmmCBWFA1of78wmqjM/AADiZbuXeZtKGAABrASCCAAAgIxIEbHEZ/sCAAAA
AARZWg==
Information after custom compression method compression:
/Td6WFoAAATm1rRGAgEDAQQhARgpmh6G4AQBAnpdABmLgofkNL8eHEtWAvjVxacFDEaDPOZs5e4IfcNKOsj2UR74OZtXQCG7AT
Ucj2HdBOUB3Kg5oHoaXY+hIcaSNV831p3z4EOPrqbH0Wz0Qjm1isu4ZinMlHdBqCqgvBuntr24+6C4hpQV/SkMdP08JglFG8jx
Bnug/W9ObLA4jQiLr1RcqBrpAUEjkkK+XQGbllksspKxha7SZ5um4py0D8uAXMJrdXFyj+FnIiOk/olxxH/0fhfhptPo/Se/iC
EGePUt4hSMfA3gYNSPHVZoEDBn3RyOuHLWmzeYCuhS1xMiIU5eZ2BId2U5xu2daLhgmkiKSK1fBQpMVXDajccMg7z7W4wPnpIZ
h8aK7+r+C8onxLhxKYIxv8znYLfPMS9GszFI3gDU+1C+LACrw3dz9qxGO5EVKI1r/i0G5pSAknqbv7rrVwLXM0YvpV7Y74FVvN
ZTdviVMOkF1+4Hiqq0qP1qL6uAJ4L258CzS+hDUvvfLO4SAPGaOjfuoIhrPUQTW6JPTgOUYNLZFOgbLq5vdMQl+JjMb8FGiao4
lVVAhqb0zkT5SV/Iic++HfV14IdXD5H0nEwPkezjEZASdlCBSfxJ950+RueYrKjS1Kh8O6pQF5tXvV4Zfvipz0S0mtFDmKHmYZ
OVyR6rExHWJCZ6ve332wl9FKB25zcFdkgzUJLeIDR0h9jvKiG8Vr2jQrRIf8OOA1f0J7Xb7a130lgUrqzYu/WdpGSQ9pFM/Gc1
AsU/XfQ0CHx99UILCLmhSsLla9riCT8W6Mwh2eb5W/Zcidli55cw+6eOnP0Crr9mw0EUDvkS4B5Tgb6OmCBf8n5euYLn5R0dol
ESOREAAADiZbuXeZtKGAABlgWCCAAA+4HJibHEZ/sCAAAAAARZWg==
```

Figure 7: The compression results of the default compression method and the custom compression method

The existing detection algorithms of covert timing channels usually use rules and statistics. The regular data of the receiver over a period of time is used as the basis for detecting the covert timing channel. In this model, data traffic is distributed among multiple connections in a distributed system. To detect whether the data traffic existing in multiple connections is related to each other, large performance and time overhead are required.

Use Wireshark to perform packet capture analysis on the receiver, and use Wireshark's IO Graphs tool, which can display the overall traffic situation in the packet capture file. It is useful for viewing peaks/troughs in traffic. Use relevant filters to display specific information on the chart. Fig. 8 shows the distribution diagram of UDP traffic in daily use at the specified network adapter. Fig. 9 shows a distribution diagram of UDP traffic at the runtime of DCCPOEDC at the specified network adapter.

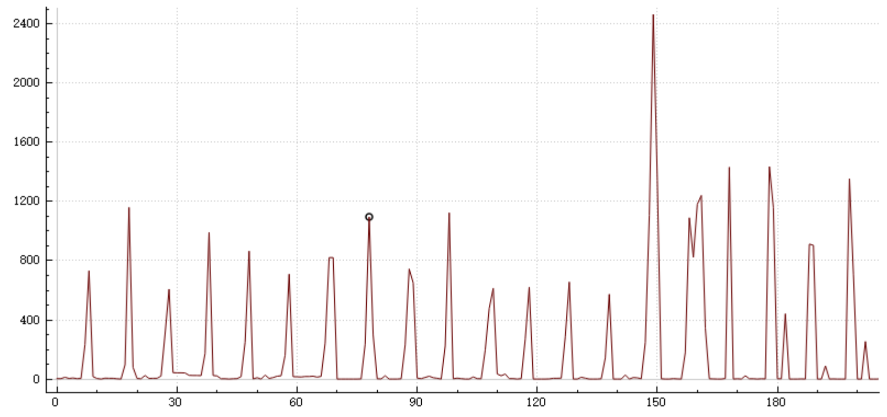


Figure 8: Distribution diagram of UDP traffic during daily use

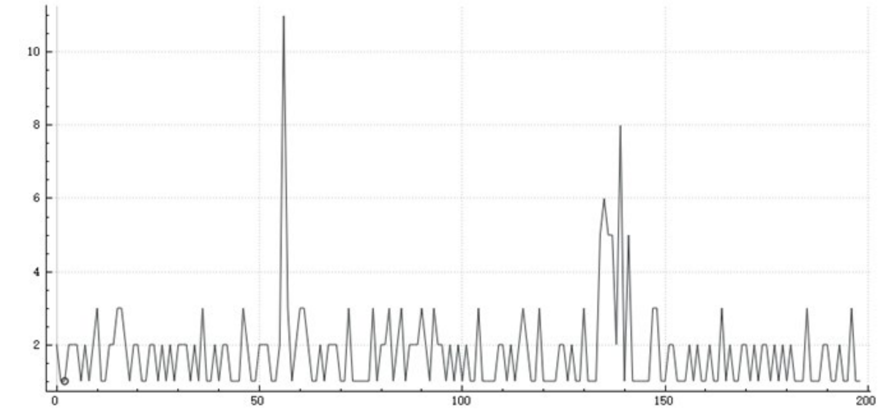


Figure 9: Distribution diagram of UDP traffic during the running of DCCPOEDC

In the Figs. 8 and 9, the abscissa is time, and the ordinate is the total number of data packets transmitted at the current time. It can be seen that the traffic distribution during the running of this model is similar to the traffic distribution during daily use.

In summary, through analysis of the LZMA compression algorithm, it can be seen that different function libraries and different compression methods can have different effects on the LZMA compression result, which can reduce the data size while protecting hidden information. From the analysis of DCCPOEDC, each node in the distributed system receives time series instead of secret information. Therefore, it can ensure that the secret information is not leaked in the transmission process. By analyzing the principle of the covert channel used in this article, it can be found that because the content of the data packet has not been modified, the packet capture software is used to analyze the data packet in DCCPOEDC, which is similar to the normal data packet transmitted in the network. In the process of covert communication, the content of the data packet can withstand the scrutiny of the security device. By comparing the distribution diagram of the transmission traffic of this model, it can be seen that the transmission diagram of this model is similar to the traffic distribution diagram of the computer during daily use, and it has some concealment to the transmission channel.

4.4 Transmission rate

Because the compression algorithm is used, the sender needs to compress the hidden information. Therefore, when calculating the transmission rate, the compression ratio CR of the compression algorithm needs to be considered. The transmission rate of this model is: $\frac{\lfloor \log_2 m! \rfloor}{CR \bullet t}$.

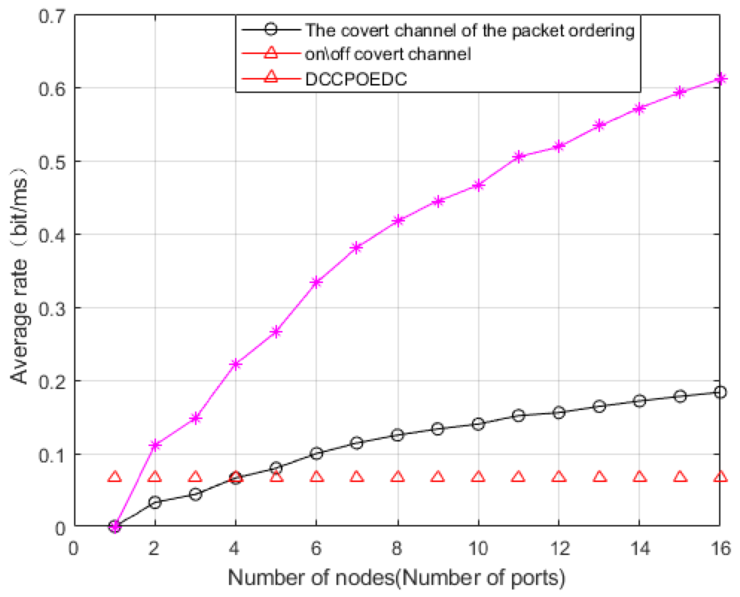


Figure 10: Transmission rate

Fig. 10 shows the relationship between the number of nodes and the average rate when the average interval time is 15 ms.

The on\off covert channel model determines whether the data packet arrives at the receiver to express the secret information. It needs to express information in the same time interval, so its transmission rate is low. The covert channel of the packet ordering model and DCCPOEDC are better than the on\off covert channel model because they do not use the interval between data packets to represent hidden information. And because DCCPOEDC model uses a data compression algorithm, the transmission time of the same information is less than the covert channel of the packet ordering model.

5 Conclusions

This paper focuses on the problems of inadequate concealment, low channel capacity and obvious data distribution in the covert channel of the packet ordering model. A distributed covert channel of the packet ordering enhancement model based on data compression is proposed. Through the model analysis, the compression algorithm suitable for this model is selected, and the anonymity, transmission efficiency and transmission performance are analyzed. It can be seen that the model optimizes the covert channel of the packet ordering model, increases the total amount of data transmitted and improves the concealment.

Acknowledgement: The authors would like to thank the reviewers for their detailed reviews and constructive comments, which have helped improve the quality of this paper.

Funding Statement: This work is sponsored by the National Natural Science Foundation of China Grant No. 61100008, Natural Science Foundation of Heilongjiang Province of China under Grant No. LC2016024, Natural Science Foundation of the Jiangsu Higher Education Institutions Grant No. 17KJB520044 and Six Talent Peaks Project in Jiangsu Province No. XYDXX-108.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Chrishwang. (2019): SevenZip. *https://www.nuget.org/packages/SevenZip/19.0.0.*

Elson, J.; Girod, L.; Estrin, D. (2002): Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review*, vol. 36, pp. 147-163.

Ganeriwal, S.; Kumar, R.; Srivastava, M. B. (2003): Timing-sync protocol for sensor networks. *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, pp. 138-149.

García, S.; Zunino, A.; Campo, M. (2014): Survey on network-based botnet detection methods. *Security and Communication Networks*, vol. 7, no. 5, pp. 878-903.

Handel, T. G.; Sandford, M. T. (1996): Hiding data in the OSI network model. *International Workshop on Information Hiding*, vol. 1174, pp. 23-38.

Hübbe, N.; Wegener, A.; Kunkel, J. M.; Ling, Y.; Ludwig, T. (2013): Evaluating lossy compression on climate data. *International Supercomputing Conference*, vol. 7905, pp. 343-356.

Jiang, Z. Y.; Chen, Y.; Hu, B. (2017): Research on time synchronization algorithm for wireless sensor networks. *Computer Engineering and Applications*, vol. 53, no.1, pp. 1-8.

Lamport, L. (1978): Time, clocks, and the ordering of events in a distributed system. *Communications of the Association for Computing Machinery*, vol. 21, no. 7, pp. 558-565.

Lampson, B. W. (1973): A note on the confinement problem. *Communications of the ACM*, vol. 16, no. 10, pp. 613-615.

Leavline, E. J.; Singh, D. A. A. G. (2013): Hardware implementation of LZMA data compression algorithm. *International Journal of Applied Information Systems*, vol. 5, no. 4, pp. 51-56.

Liu, G.; Zhai, J.; Dai, Y. (2012): Network covert timing channel with distribution matching. *Telecommunication Systems*, vol. 49, no. 2, pp. 199-205.

Luo, Y.; Qin, J.; Xiang, X.; Tan, Y.; Liu, Q. et al. (2020): Coverless real-time image information hiding based on image block matching and dense convolutional network. *Journal of Real-Time Image Processing*, vol. 17, no. 1, pp. 125-135.

Maróti, M.; Kusy, B.; Simon, G.; Lédeczi, Á. (2004): The flooding time synchronization protocol. *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pp. 39-49.

Najmabadi, S. M.; Tran, T. H.; Eissa, S.; Tungal, H. S.; Simon, S. (2019): An architecture for asymmetric numeral systems entropy decoder-a comparison with a canonical Huffman decoder. *Journal of Signal Processing Systems*, vol. 91, no. 7, pp. 805-817.

Pavlov, I. (2019): LZMA SDK. *https://www.7-zip.org/sdk.html.*

Python Software Foundation. (2020): LZMA-compression using the LZMA algorithm. *https://docs.python.org/dev/library/lzma.html#filter-chain-specs.*

Rios, R.; Onieva, J. A.; Lopez, J. (2012): HIDE_DHCP: Covert communications through network configuration messages. *IFIP International Information Security Conference*, vol. 376, pp. 162-173.

Sundararaman, B.; Buy, U.; Kshemkalyani, A. D. (2005): Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, vol. 3, no. 3, pp. 281-323.

Taheri, S.; Mahdavi, M.; Moghim, N. (2018): A dynamic timing-storage covert channel in vehicular ad hoc networks. *Telecommunication Systems*, vol. 69, no. 4, pp. 415-429.

Wang, C. D.; Ju, S. G.; Zhou, C. H.; Song, X. M. (2009): A measurement of covert channels threat. *Chinese Journal of Computers*, vol. 32, no. 4, pp. 751-762.

Wang, Z.; Yang, R.; Fu, X.; Du, X.; Luo, B. (2016): A shared memory based cross-VM side channel attacks in IaaS cloud. *IEEE Conference on Computer Communications Workshops*, pp. 181-186.

Wu, J.; Wang, Y.; Ding, L.; Liao, X. (2012): Improving performance of network covert timing channel through Huffman coding. *Mathematical and Computer Modelling*, vol. 55, no. 1, pp. 69-79.

Xiang, L.; Wu, W.; Li, X.; Yang, C. (2018): A linguistic steganography based on word indexing compression and candidate selection. *Multimedia Tools and Applications*, vol. 77, no. 21, pp. 28969-28989.

Xue, X.; Wang, S.; Zhang, L.; Feng, Z.; Guo, Y. (2018): Social learning evolution (sle): computational experiment-based modeling framework of social manufacturing. *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3343-3355.

Zhang, S.; Zhang, H. (2012): A review of wireless sensor networks and its applications. *IEEE International Conference on Automation and Logistics*, pp. 386-389.

Zhang, X.; Liang, C.; Zhang, Q.; Li, Y.; Zheng, J. et al. (2018): Building covert timing channels by packet rearrangement over mobile networks. *Information Sciences*, vol. 445, pp. 66-78.