SECOMPAX: A bitmap index compression algorithm

Yuhao Wen, Zhen Chen*, Ge Ma, Junwei Cao,
Wenxun Zheng, Guodong Peng, Shiwei Li
Research Institute of Information Technology
Tsinghua University
Beijing, China
{wenyhinthu, zhenchen3419}@gmail.com

Wen-Liang Huang
China Unicom Groups Labs
China Unicom Groups
Beijing, China
huangwenliang@gmail.com

***Abstract*—Archiving of Internet traffic is essential for analyzing network events in the field of network security. Currently, bitmap indexing is used to accelerate the indexing and search queries for archival traffic data. However, the generation of bitmap index needs large storage space, which makes bitmap index compression is a must-have function. In this paper, we propose a new bitmap index encoding algorithm named SECOMPAX (Scope-Extended COMPressed Adaptive indeX), which performs better compression ratio and fast encoding speed compared with the state-of-art bitmap index compression algorithm WAH (Word-Aligned-Hybrid), PLWAH(Position list word aligned hybrid) and COMPAX (COMPressed Adaptive indeX). The comparison among WAH, PLWAH, COMPAX and SECOMPAX shows that SECOMAX accomplishes the smallest bitmap index in size and the comparable encoding time with other three methods. We also use real Internet trace from CAIDA to prove the validity of SECOMPAX. SECOMPAX has the best compression ratio in compared with other bitmap index encoding algorithms in our experiments. The encoding time is measured, and statistics of the distribution of codeword used in SECOMPAX is also investigated in experiments. It shows that SECOMPAX's extra time consumption is acceptable as the new designed codebook work effectively in encoding bit sequence which cannot be compressed in other bitmap encoding schemes.**

Keywords—Big Data; Network Forensic; Network Security; Bitmap Index; Index Encoding; Index Compression; WAH; COMPAX; SECOMPAX

I. INTRODUCTION

Analysis from Cisco predicts that the volume of Internet traffic will quadruple between 2011 and 2016 reaching 1.3 Zettabytes per year in 2016[1]. According to the internal statistics of China Unicom[15], mobile user traffic increases rapidly with CAGR (Compound Annual Growth Rate) of 135%. From the data of China Unicom in 2013, its monthly records are more than 2 trillion ($2x10^{13}$), monthly data volume is over 525TB, and has reached 5PB annually.

CNSMS[2], vCNSMS[16] and TIFAflow[3] are used for traffic acquisition and aggregation for forensic analysis. CNSMS is an architecture for traffic acquisition with TIFAflow and its UTM appliance for traffic aggregation used in forensic analysis in a cloud computing based security center. TIFAflow is a software based prober that combines TIFA [4-5] with Fastbit [6] to provide flow granularity data storage. It may be operated as an independent prober or integrated into CNSMS's UTM appliance.

Luca Deri and Francesco Fusco [7-8] propose MicroCloud-based flow aggregation for fixed and mobile networks. This architecture is used to provide real-time traffic monitoring and correlation in large distributed environments. Their system is deployed in the VIVACOM (Bulgarian Telecom) mobile network and is used for monitoring the DNS ccTLD and a large 3G mobile network.

A 10 Gbps network link can arrive at a maximum of 14.8 million packets per second. This is a big challenge to index these packets in one second. For any mobile network operator(such as China Unicom) manages several such links, even recording only network flow data, the resulting data repository could easily reach the Terabytes on a yearly basis. However, if all mobile traffic data is recorded for forensic analysis, the volume of the data could easily reach the Petabytes. This remains a major challenge to a mobile network operator that must accommodate and index such big data for further analysis.

This work introduces a new bitmap encoding method called Scope-Extended COMPressed Adaptive index (SECOMPAX) which records the origin bitmap index into a new format according to a new designed codebook.

SECOMPAX is a new design based on WAH[9], PLWAH[10], COMPAX[11], CONCISE[19], EWAH[20], VA-WAH[21], PWAH[22] etc. As we consider the occurrence of 0 and 1 in a literal word equally for encoding in a higher compression ratio than both COMPAX and PLWAH. Our compression algorithm shows a symmetry in encoding, and can be easily implemented.

This paper is organized as follows: section 2 introduces the Background of bitmap index compression. Section 3 describes the detailed design of SECOMPAX. A theoretical analysis of SECOMAX is presented in Section 4. Finally we conclude this paper in Section 5.

II. BACKGROUND

A bitmap index is a structure that can accelerates search queries over archival data, such as scientific data and network traffic. However, bitmap index needs large storage space, which makes bitmap index have plenty of room to be compressed.

WAH, PLWAH and COMPAX are the state-of-art compressed bitmap index encodings. However, all of these

methods don't consider the occurrence when the number of 1's in bitmap index is comparable with the number of 0's in literal words.

Table 1 shows a bitmap index for a sequence of integers. The cardinality is 4 in this bitmap index. As we can see, there are lots of 0's in this bitmap which can be further compressed.

TABLE I. AN EXAMPLE OF BITMAP INDEX.

RowID	Column value	Bitmap Index			
		=1	*=2*	*=3*	*=4*
1	1	1	0	0	0
2	2	0	1	0	0
3	2	0	1	0	0
4	3	0	0	1	0
5	4	0	0	0	1
6	1	1	0	0	0

III. SECOMPAX

The design of SECOMPAX is introduced as follows.

1. A Literal[L] represents a 31-bit-long sequence which satisfies not all the 31 bits are 0 and not all are 1. It is encoded as a 32-bit word with 1 of its first bit.

1 0011110 10100010 00111000 0000000

2. A Fill[F] encodes a sequence of consecutive chunks of zero bits or chunks of one bits. The first 4 bits encode the codeword type (0000 means 0-Fill and 0001 means 1-Fill) and the remaining 28 bits are used as a counter, counting the number of consecutive 31-bit chunks.

Before explaining the third word type, the concept of "nearly identical" is defined here: a Literal word is NEARLY IDENTICAL (NI) to a Fill word when all the different bits are in a single byte (of the four bytes). For example, the following Literal word is nearly identical to a 0-Fill word:

0000000 00101101 00000000 00000000 (0-NI-L word)

And the following Literal word is nearly identical to a 1-Fill word:

1111111 11111111 11111100 11111111 (1-NI-L word)

3. An [FLF] word condenses a sequence of words satisfying a [F(0-Fill or 1-Fill)]-[L]-[F(0-Fill or 1-Fill)] paradigm, and it's literal word is 0-NI-L word or 1-NI-L word.

The first 3 bits encode the codeword type (011 means FLF), the 4th bit encodes the first fill word's type (0 means 0-Fill and 1 means 1-Fill), the 5th bit encodes the second fill word's type (0 means 0-Fill and 1 means 1-Fill), the 6th bit encodes the literal word's type (0 means nearly identical to a 0-fill word and 1 means nearly identical to 1-fill word), the 7th and 8th bit encode the literal word's dirty byte.

4. An [LFL] word condenses a sequence of word satisfying a [L(nearly identical to 0-fill word or to 1 fill word)-F(0-Fill or 1-Fill)-L(nearly identical to 0-Fill word or to 1-Fill word)]. And the fill word represents a sequence shorter than 128*31 bits. The first 4 bits encode the codeword type.

When first 3 bits are 001, the two Literal words are the same type. The 4th bit is 0 when two literal words are both nearly identical to 0-Fill word and 1 when two literal words are both nearly identical to 1-Fill word.

When first 3 bits are 010, the two Literal words are different types. The 4th bit is 0 when the first literal word is nearly identical to 0-Fill word and the second literal word is nearly identical to 1-Fill word, while the 4th bit is 1 when the first literal word is nearly identical to 1-Fill word and the second literal word is nearly identical to 0-Fill word. The 5th-8th bits encodes the position (00-11) of dirty bytes. The 5th and 6th bit represent the position of dirty byte of the first literal word, the 7th and 8th bit represent the position of dirty byte of the second literal word. The 9th-16th bit (the second byte) is the dirty byte of the first literal word. The 25th-32nd bit (the fourth byte) is the dirty byte of the second literal word. The 17th bit (First bit of the third byte) represents the type of fill word (0 means 0-Fill, 1 means 1-Fill). The 18th-24th bit encodes the number of 31-bit chunks form the origin sequence.

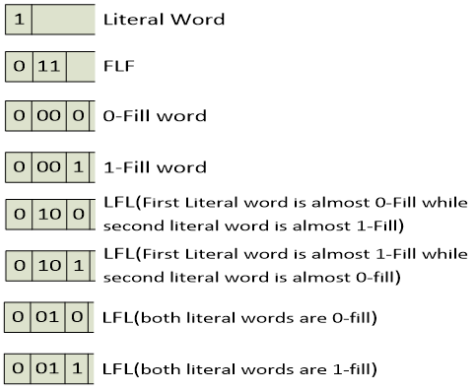
For easy understating, Fig. 1 presents four examples to indicate WAH, COMPAX and SECOMPAX schemes.

In Example 1, the origin bit sequence can be recognized by SECOMPAX as FLF, but cannot be recognized by COMPAX. So COMPAX encodes it as 3 words-an F word, an L word, and an F word, while SECOMPAX encodes it as a single FLF-word. Under this circumstance, SECOMAX performs better than COMPAX.

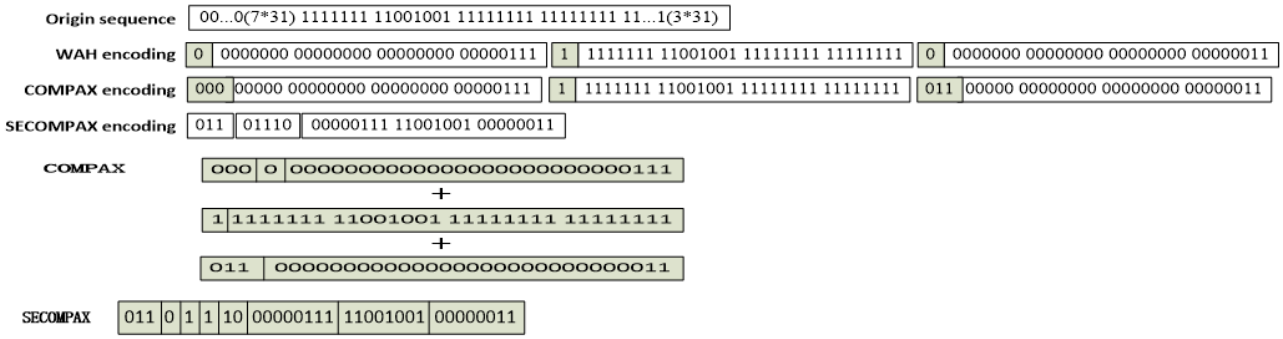
In Example 2, this origin sequence can be encoded perfectly by both COMPAX and SECOMPAX. So both of the lengths are one word (32 bits).

In Example 3, this origin bit sequence can be recognized by SECOMPAX as LFL, but cannot be recognized by COMPAX. So COMPAX encodes it as 3 words-an L word, an F word, and an L word, while SECOMPAX encodes it as a single LFL-word. Under this circumstance, SECOMAX performs better than COMPAX.

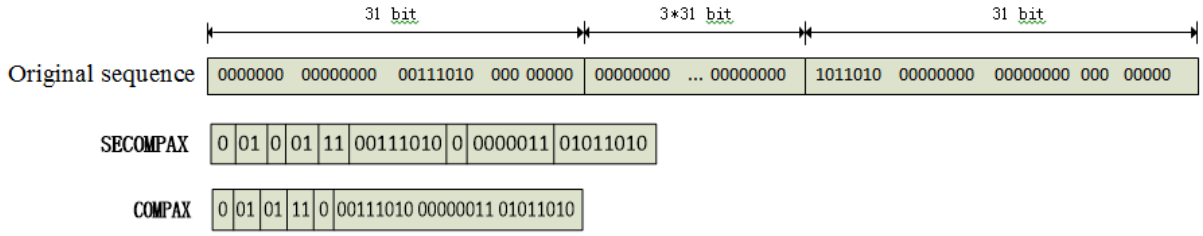
In Example 4, this origin sequence can be recognized by SECOMPAX as LFL, but cannot be recognized by COMPAX. So COMPAX encodes it as 3 words, while SECOMPAX encodes it as a single LFL-word. Under this circumstance, SECOMAX performs better than COMPAX.



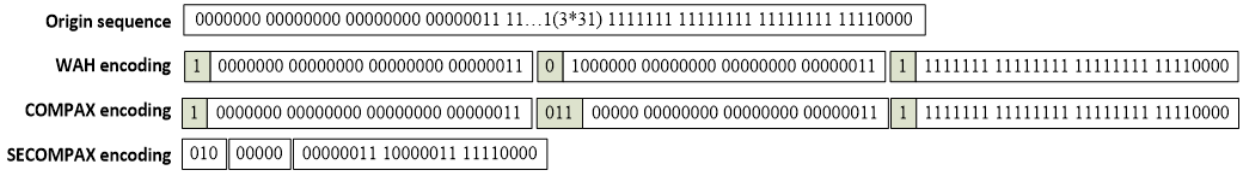
EXAMPLE 1:



EXAMPLE 2:



EXAMPLE 3:



EXAMPLE 4:

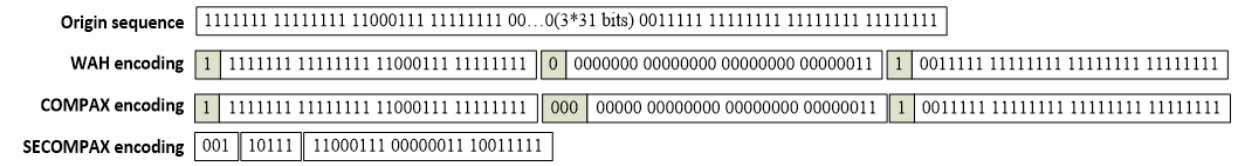


Fig. 1. Illustration of WAH, COMPAX and SECOMPAX encoding scheme.

There are some kinds of origin sequences that COMPAX cannot perfectly deal with:

1. In the form of FLF, if two Fill words are in different types, or the Literal word is nearly identical to 1-Fill word, COMPAX doesn't try to encode them as FLF. COMPAX encoded them as an F, an L, and an F, while SECOMPAX can encode them as FLF. The 2nd origin code in FLF we have mentioned before is an example. Obviously, in this circumstance, the result encoded by COMPAX is twice longer of those encoded by SECOMPAX.

2. In the form of LFL, the COMPAX can only deal with: both of the two Literal-Words are nearly identical to 0-Fill words. If one or both of the literal words are identical to 1-Fill word, COMPAX encode them as an L-Word, an F-Word, and an L-Word (L+F+L, 3 words in total), not as LFL (1 word in total). However, SECOMPAX can encode all of those circumstances as LFL, which saves a great deal of room compared with COMPAX. The 2nd and 3rd origin code in LFL which we have mentioned before can be taken as examples.

IV. Theoretical Analysis

COMPAX make an assumption that the number of 0s in the origin sequence is much more than 1s, which makes COMPAX not so efficient when dealing with origin sequence having 0s and 1s in a similar scale or more 1s than 0s. SECOMPAX make fully consideration of those circumstances and has better performance than COMPAX. Finally, the detailed codebook is shown as follows.

After the first step of encoding, we can get code stream in the form of FLFFL…FLF… etc. We set the probability of 0-Fill words on each position as a_0, probability of 1-Fill words as a_1. Since they are independent, the chance of Fill-word on each position is a_0+a_1.

In Literal Words, we set the probability of 0-NI-L words as b_0 and probability of 1-NI-L words as b_1. Thus, the chance of NI-L words is b_0+b_1. The chance of common literal word (literal excluding NI-L word) is $c = 1 - (a_0 + a1 + b_0 + b_1)$

Since the total amount of n is extremely large (almost infinity), we can take three words as a group and calculate the expectation length of WAH, COMPAX and SECOMPAX.

Obviously, we can get

$$\mathrm{E(WAH)} = 3 \tag{1}$$

Since WAH can only encode the origin sequence into the form of F/L words.

For COMPAX:

The probability of a FLF in a 3-word-group is

$$p_1 = (\mathrm{a_0^2} + \mathrm{a_1^2})\mathrm{b_0} \tag{2}$$

The probability of a LFL in a 3-word-group is

$$p_2 = (\mathrm{a_0} + \mathrm{a_1})\mathrm{b_0^2} \tag{3}$$

Then we can have

$$\mathrm{E(COMPAX)} = (\mathrm{p_1} + \mathrm{p_2}) + (1 - \mathrm{p_1} - \mathrm{p_2})\left(1 + \tfrac{2}{3}\mathrm{E(COMPAX)}\right) \tag{4}$$

From (2), (3), (4), we get

$$\mathrm{E(COMPAX)} = \frac{3}{1+2\mathrm{p_1}+2\mathrm{p_2}} \tag{5}$$

For SECOMPAX:

Similar to those in COMPAX, we have the probability of FLF:

$$p_1' = (\mathrm{a_0} + \mathrm{a_1})^2(\mathrm{b_0} + \mathrm{b_1}) \tag{6}$$

The probability of LFL:

$$p_2' = (\mathrm{a_0} + \mathrm{a_1})(\mathrm{b_0} + \mathrm{b_1})^2 \tag{7}$$

And the expectation of SECOMPAX:

$$\mathrm{E(SECOMPAX)} = \frac{3}{1+2\mathrm{p_1'}+2\mathrm{p_2'}} \tag{8}$$

The improvement of SECOMPAX (compared to COMPAX) in the standard of WAH is

$$\eta = \frac{\mathrm{E(COMPAX)} - \mathrm{E(SECOMPAX)}}{\mathrm{E(WAH)}} \text{ (then use (4) and (8))}$$

$$= \frac{(2\mathrm{p_1'}+2\mathrm{p_2'}) - (2\mathrm{p_1}+2\mathrm{p_2})}{(1+2\mathrm{p_1'}+2\mathrm{p_2'})\ (1+2\mathrm{p_1}+2\mathrm{p_2})} \tag{9}$$

From (2), (3), (6), (7),we have

$$\eta = \frac{(a_0 + a_1)^2 b_1 + 2a_0 a_1 b_0 + (a_0 + a_1){b_1}^2 + 2(a_0 + a_1)b_0 b_1}{[1 + 2(a_0 + a_1)^2(b_0 + b_1) + (a_0 + a_1)(b_0 + b_1)^2][1 + 2(a_0^2 + a_1^2)b_0 + 2(a_0 + a_1)b_0^2]}$$

In a common way, we can assume that

$$\mathrm{a_0} = \mathrm{a_1} = \mathrm{a}, \mathrm{b_0} = \mathrm{b_1} = \mathrm{b}, (1 + 2\mathrm{p_1'} + 2\mathrm{p_2'})(1 + 2\mathrm{p_1} + 2\mathrm{p_2}) \to 1$$

Then we can get

$$\eta = 6\mathrm{a^2b} + 4\mathrm{ab^2}$$

The result is shown in Fig.2. In this figure, we can find the improvement also increase when the variables *a* and *b* increase.

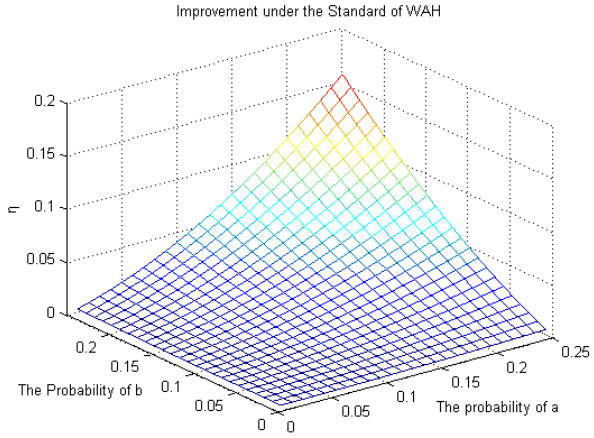


Fig. 2. Compared with standard version of WAH.

Considering the improvement percentage (compared to COMPAX), we can get the following formula.

$$\frac{E(COMPAX) - E(SECOMPAX)}{E(COMPAX)} = \frac{(2p_1' + 2p_2') - (2p_1 + 2p_2)}{(1 + 2p_1' + 2p_2')} \times 100\%$$

$$= \frac{(a_0 + a_1)^2 b_1 + 2a_0 a_1 b_0 + (a_0 + a_1){b_1}^2 + 2(a_0 + a_1)b_0 b_1}{[1 + 2(a_0 + a_1)^2(b_0 + b_1) + (a_0 + a_1)(b_0 + b_1)^2]} \times 100\%$$

For simplification, we can assume that:

$$a_0 = a_1 = a, b_0 = b_1 = b$$

Then we can get

$$\frac{E(COMPAX) - E(SECOMPAX)}{E(COMPAX)} = \frac{6a^2b + 4ab^2}{1 + 16a^2b + 8ab^2} \times 100\%$$

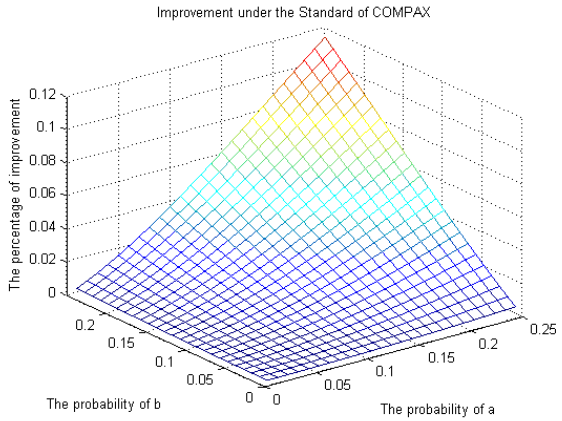


Fig. 3. Compared with standard version of COMPAX.

Fig.3 illustrates the details. As shown in figure, the improvement also increase with the variables a and b.

V. EXPERIMENTS WITH REAL DATA FROM CAIDA

We evaluate PLWAH, COMPAX2 and SECOMPAX with real Internet traffic trace from CAIDA [13-14]. This Internet traffic trace is anonymized and captured from a core router by CAIDA in 2013.

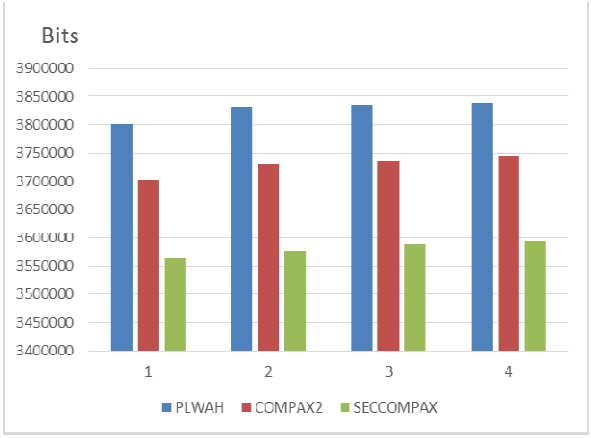


Fig. 4. The compression ratio of real Internet trace.

There are totally 13,581,181 packets in this trace. Firstly, we reorder packets with the mechanism based on the principle of locality-based hashing used in [11]. Then we convert five tuple <SIP, sport, DIP, dport, proto> to bitmaps and compress bitmaps in each column with a fixed block size of 4Kbit which is also used in [11]. In these experiments, source IP (4 bytes), destination IP (4 bytes), source port (2 bytes) and destination port (2 bytes) are compressed with PLWAH, COMPAX and SECOMPAX schemes. For SrcIP, there are four bytes and each byte expand as 256 columnar files. There are totally 1024 columnar files which contains 13,581,181 bits (the number of packets in trace). There are similar cases for DstIP and Ports fields.

A. *The compression ratio of three algorithms*

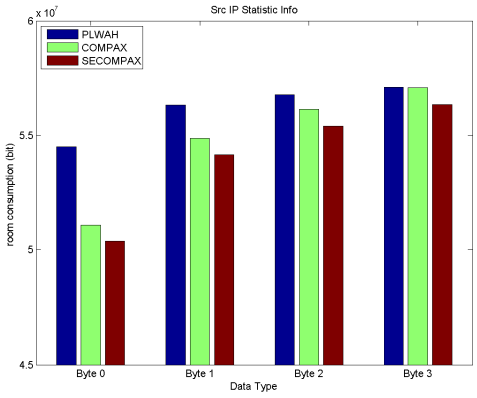


Fig. 5. The size after compression using three encoding schemes in SrcIP.

Fig. 4 shows the average size of compressed columnar file with PLWAH, COMPAX2 and SECOMPAX, where each original columnar file has 13,581,181 bits (the number of packets in trace). Compared with PLWAH, SECOMPAX reduce the size of index for source IP address by 6.74%, and destination IP by 6.05% totally. While compared with COMPAX2, SECOMPAX reduce the size of source IP by 4.01%, and destination IP by 3.97%.

Detailed comparison is illustrated for source IP (4 bytes), destination IP (4 bytes), source port (2 bytes) and destination port (2 bytes) in Fig. 5 - 7. From Fig. 5-7, it is clearly shown that SECOMPAX has a better compression ratio and smaller space consumption than COMPAX2 and PLWAH, especially in Byte0 in SrcIP and DstIP. Compared with PLWAH, SECOMPAX can reduce the size of index for SrcIP Byte 0 by 7.62% and DstIP by 8.32% respectively.

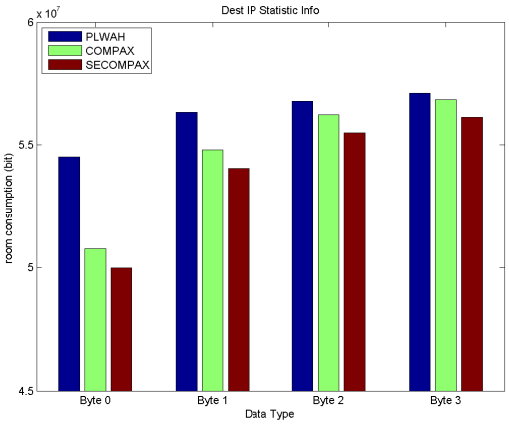


Fig. 6. The size after compression using three encoding schemes in DstIP.

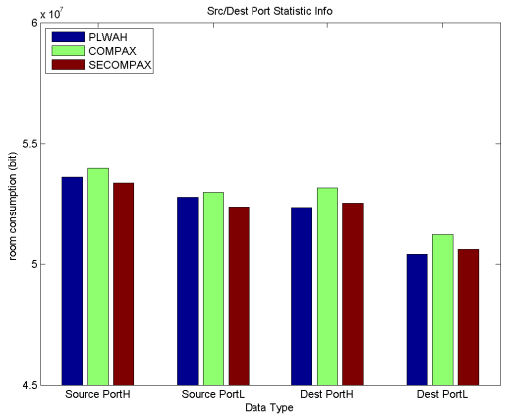


Fig. 7. The size after compression using three encoding schemes in Ports.

B. *The Encoding Time of three algorithms*

The experiments to measure the encoding time are also conducted with four different encoding algorithms. The results of compression time are shown in Fig. 8-11. In these experiments, each columnar file is encoded as a whole without segmentation in 4Kbit to neglect the I/O operations for comparison.

It is obvious that the more "1" occurrences (query hits), the longer the compression time will be. Hence, the X-axis is the

query hits (the number of "1" occurrences), and the Y-axis the encoding time in compression.

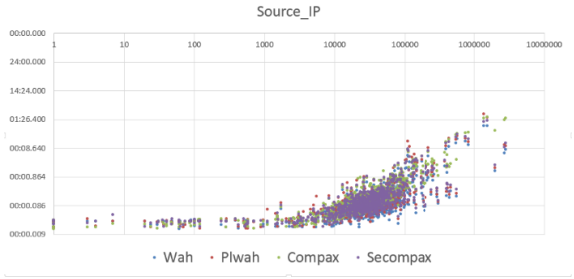


Fig. 8. The encoding time of four schemes in SrcIP.

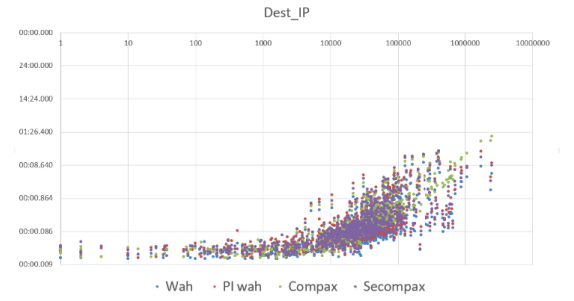


Fig. 9. The encoding time of four schemes in DstIP.

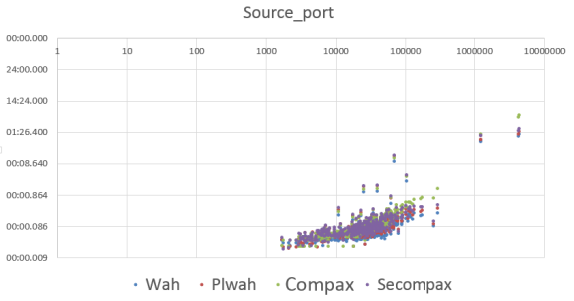


Fig. 10. The encoding time of four schemes in SrcPort

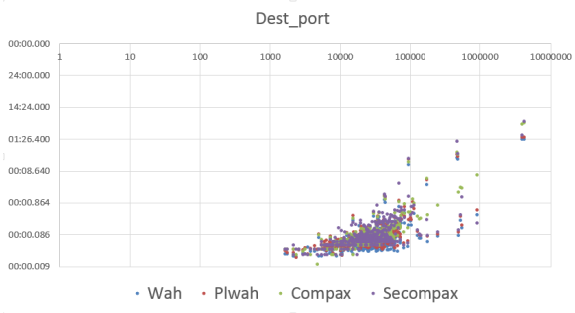


Fig. 11. The encoding time of four schemes in DstPort

As shown in Fig. 8-Fig.11, there are no big gaps in the encoding time used in four algorithms. The reason is that current CPU is powerful enough to handle the bit-wise operations in word aligned way used in encoding procedure in modern processor architecture. While the compression time is very close, SECOPMAX has the smallest space consumption among all encoding schemes.

Compared with encoding time, the decoding time can be neglected as decoding is much simpler than encoding.

C. The statistics of codewords used in SECOMPAX

As SECOMPAX achieves best compression ratio with comparable encoding speed. We wonders which codeword contributes the most reduced size, and how much each codeword contribute the compression in general. For this purpose, we check the distribution of codeword in the compressed files.

Fig.12 shows the state diagram of SECOMPAX encoding automaton. The states of encoder represent the different word types have been stored. The symbols are explained as follows: {A: Fill, B: NI-Fill, C: Literal}. Symbols labeling the edges are introduced as follows: (x, y) means the next character is x, and the output is y except that if 'y=null', output nothing, and if y is underlined, the output is a codeword as a whole.

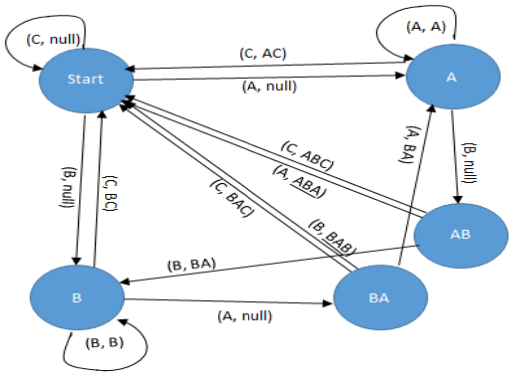


Fig. 12. Encoding Automaton in SECOMPAX

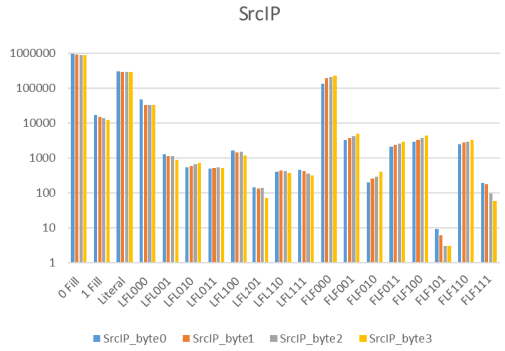


Fig. 13. The codeword distribution in SrcIP.

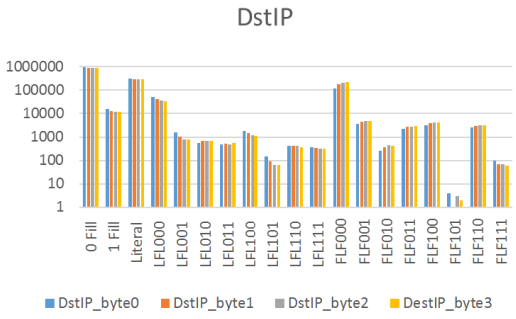


Fig. 14. The codeword distribution in DstIP.

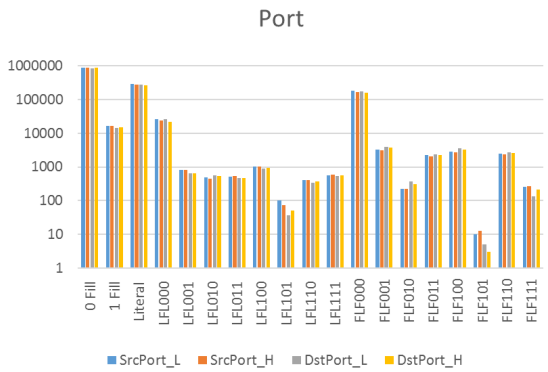


Fig. 15. The codeword distribution in Port.

The statistics of codewords used in SECOMPAX are shown in Fig.13-15. The X-axis shows the different codeword used in the compressed bit sequence, and their counts are shown as Y-axis. The statistic of four bytes in SrcIP and DstIP are shown in Fig.13 and Fig.14. The statistics of 2 Bytes in SrcPort and DstPort are shown in Fig.15 together.

From the statistics of codewords, it is obvious that 0-fill word accounts for the most size of compressed sequences. 1-fill word accounts for the second largest word type. Literal words account for the third largest. LFL000 and FLF000 also contributes the fourth largest in compression.

VI. CONCLUSION

In this paper, we propose a new bitmap index compression algorithm named SECOMPAX (Scope-Extended COMPressed Adaptive index), which considers the possibility where the number of 1's are comparable to 0's or even much more than 0's in the original bit sequences in bitmap index. Thus, when the occurrence of consecutive 0's is not extremely much, our experiments results show that SECOMPAX performs better compression ratio and faster querying speed compared with the state-of-art algorithm, i.e., WAH, PLWAH and COMPAX. What's more, the decoding time still remains comparable to the state-of-art algorithm.

To prove the validity of SECOMPAX, we use real Internet trace from CAIDA. With experiments' results, SECOMPAX has the best compression ratio in compared with other bitmap encoding algorithms. The encoding time is also measured in experiments, it shows that SECOMPAX's extra time consumption is acceptable because of the new designed codebook. The statistics of codewords' types used in SECOMPAX in real encoding procedure is also investigated and the new codebook work effectively in encoding the bit sequence which cannot compressed in other bitmap encoding schemes.

In the future, we'll try to combine the usage of SECOMPAX and COMPAX, which makes it possible to compress origin sequence into a more compact way in order to fit more complex conditions.

ACKNOWLEDGMENT

This work is supported in part by Ministry of Science and Technology of China under National 973 Basic Research Program (No. 2013CB228206 and No.2012CB315801), National Natural Science Foundation of China (grant No. 61233016), and China NSFC A3 Program (No.61140320).

REFERENCES

[1] Cisco Visual Networking Index Forecast (2011 - 2016). http://www.cisco.com/web/solutions/sp/vni/vni_forecast_highlights/index.html.

[2] Z. Chen et al., Cloud Computing-Based Forensic Analysis for Collaborative Network Security Management System. Tsinghua Science and Technology, pp.40-50, vol.18, No.1, 2013.

[3] Z. Chen et al., TIFAflow: enhancing traffic archiving system with flow granularity for forensic analysis in network security. Tsinghua Science and Technology, vol.18, No.4, 2013.

[4] G. Maier et al., Enriching Network Security Analysis with Time Travel. Proceedings of the ACM SIGCOMM 2008 conference on Data communication, New York, USA, pp.183-194.

[5] J. Li et al., TIFA: Enabling Real-Time Querying and Storage of Massive Stream Data. IEEE ICNDC' 2011.

[6] FastBit. An Efficient Compressed Bitmap Index Technology. https://sdm.lbl.gov/fastbit/

[7] L. Deri et al, MicroCloud-based Network Traffic Monitoring, IFIP/IEEE International Symposium on Integrated Network Management (IM), 2013.

[8] Deri, Luca, and Francesco Fusco, Real-time MicroCloud based flow aggregation for fixed and mobile networks, 9th IEEE International Wireless Communications and Mobile Computing Conference (IWCMC), pp.96-101, 2013.

[9] K. Wu et al. Optimizing bitmap indices with efficient compression. ACM Transactions of Database Systems, 31:1–38, March 2006.

[10] F. Deli`ege and T. B. Pedersen. Position list word aligned hybrid: optimizing space and performance for compressed bitmaps. Proc. of the 13th Int. Conf. on Extending Database Technology, EDBT '10, 2010.

[11] Fusco, F., Stoecklin, M., Vlachos, M.: NET-FLi: On-the-fly Compression, Archiving and Indexing of Streaming Network Traffic. Proceedings of the International Conference on Very Large DataBases (VLDB), pp. 1382–1393, 2010.

[12] Fusco, Francesco, Michail Vlachos, and Marc Ph Stoecklin. "Real-time creation of bitmap indexes on streaming network data." The VLDB Journal-The International Journal on Very Large Data Bases 21, no. 3 (2012): 287-307.

[13] CAIDA, www.caida.org.

[14] Shannon, Colleen et al. "The internet measurement data catalog." ACM SIGCOMM Computer Communication Review 35, no. 5 (2005): 97-100.

[15] Wenliang Huang, Zhen Chen, Wenyu Dong, Hang Li, Mobile Internet big data platform in China Unicom, Tsinghua Science and Technology, Volume 19, Issue 1, pp. 95-101, Feb. 2014.

[16] Z. Chen, W. Dong, H. Li, P. Zhang, X. Chen, J. Cao, Collaborative network security in multi-tenant data center for cloud computing, Tsinghua Science and Technology, 19 (1), pp.82-94, Feb. 2014.

[17] McEliece, Robert. The theory of information and coding. Cambridge University Press, 2002.

[18] Bose, Ranjan. Information theory, coding and cryptography. Tata McGraw-Hill Education, 2002.

[19] Colantonio, Alessandro, et al. "Concise: Compressed 'n' composable integer set." Information Processing Letters 110, no. 16 (2010): 644-650.

[20] Lemire, Daniel, et al. "Sorting improves word-aligned bitmap indexes." Data & Knowledge Engineering 69, no. 1 (2010): 3-28.

[21] Guzun, Gheorghi, Guadalupe Canahuate, David Chiu, and Jason Sawin. "A Tunable Compression Framework for Bitmap Indices.ICDE 014.

[22] van Schaik, Sebastiaan et al. "A memory efficient reachability data structure through bit vector compression." In Proceedings of the 2011 ACM SIGMOD, pp.913-924. ACM, 2011.