Automatic Payload Signature Generation for Accurate Identification of Internet Applications and Application Services

Baraka D Sija[1], Kyu-Seok Shim[1], and Myung-Sup Kim[2]
Department of Computer and Information Science
Korea University, Sejong Campus
Sejong, 2511-Sejong-ro, Jochiwon-eup, Sejong 30019, South Korea
[e-mail: {sijabarakajia25, kusuk007, tmskim}@korea.ac.kr]
*Corresponding author: Myung-Sup Kim

Received July 15, 2017; revised October 17, 2017; accepted October 30, 2017; published April 30, 2018

Abstract

The diversity and fast growth of Internet traffic volume are highly influenced by mobile and computer applications being developed. Moreover, the developed applications are too dynamic to be identified and monitored by network administrators. Several approaches have been proposed to identify network applications, however, are still not robust enough to identify modern applications. This paper proposes both, TSA (Traffic collection, Signature generation and Applications identification) system and a derived algorithm so called CSP (Contiguous Sequential Patterns) to identify applications for management and security in IP networks. The major focus of this paper is the CSP algorithm which is automated in two modules (Signature generation and Applications identification) of the proposed system. The proposed CSP algorithm generates DNA-like unique signatures capable of identifying applications and their individual services. In this paper, we show that the algorithm is suitable for generating efficient signatures to identify applications and application services in high accuracy.

Keywords: Internet applications, signature generation, applications identification, network management, security

This research was supported by a Korea University Grant

http://doi.org/10.3837/tiis.2018.04.010 **ISSN : 1976-7277**

1. Introduction

Most of the Internet-traffic applications are still hard to identify and classify. This is due to the reason that modernly developed applications are either highly encrypted or too dynamic. Development of high-tech smart devices and high speed packets routing network devices, are one of the major causes to Internet-traffic applications dynamic and diversity. Internet-traffic applications and web services as well, are becoming unlimited in services provision. For instance, most modern applications are developed to provide limitless number of services simultaneously. One of such applications is a Korean developed app, KakaoTalk, whose platform is developed to offer more than ten different services, including games.

Cisco Visual Networking Index (CNI) Global Data Forecast [1], especially for mobile show that the Internet traffic is rapidly growing in both size and diversity. The outcomes of Internet volume growth are difficulties in controlling, monitoring and managing Internet-traffic applications in a network, hence difficulties in implementing policies to users. To ensure efficiency in a network and countermeasure network management challenges, several works have been done to generate signatures for Internet-traffic applications identification, however the approaches have limitations to generate signatures that adapt the current Internet volume growth speed and applications dynamic. Dynamic of applications, refers to how frequent their traffic (packet payload contents) change over time, due to protocols involved, network environment and the contents that the packets are packaged with. The primary and major objective of Internet-traffic applications identification and classification, is to enable network administrators to fully enforce policies to Internet users and minimize the consumption of network resources for better Quality-of-Service (QoS) and networks malfunctioning prevention. [2]

This paper and the development of the proposed CSP algorithm, is motivated by sequential patterns mining in data mining. [3-6] Moreover, the key and novelty of the algorithm we propose is highly motivated by studies that extract contiguous sequential patterns in biological data sequences. [7-9] Extraction of unique contiguous patterns from biological data sequences is essential for identification in biotechnologies. Besides, another motivation of this paper is from frequent contiguous DNA pattern mining approaches. [10-11] The two works try to combine pattern position information to obtain even longer, common frequent and identical DNA patterns. The common frequent, contiguous and longest DNA patterns generated are useful in uniquely identifying organisms' classes and genetic relations. Therefore, based on this scientific fact, CSP algorithm is derived to generate DNA like signatures for unique identification of applications and individual application services.

In CSP algorithm several flow-level data traffic of a target application is extracted based on 3-tuple information, src-IP, dst-IP and Layer-4 transport protocol. From the collected GT (Ground Truth) traffic information, N(*infinity*) number of signature candidates are generated proportional to an application packets payload complexity from which DNA-like and unique identifier signature is generated.

The remainder of this paper is organized as follows. Section 2, presents about 17 related signature generation works for different purposes, ranging from 2001 to 2017. In section 3, the proposed TSA system and CSP algorithm are presented in detail with analytic comparison. In section 4, experimental results are discussed. Discussion about the proposed system and algorithm is given in section 5. Conclusion and future work are given in section 6.

2. Related Work

Signature generation for matching up applications and identifying malicious traffic in a network and the Internet in general is yet the most reliable method. Therefore, to support this significant importance of signatures generation, this section presents about 17 signature generation related works, of which three approaches [13, 16, 19] focus on generating signatures for malware or polymorphic worm malware s as indicated in **Table 1**. **Table 1** summarizes all the related works discussed in this section. Since the proposed TSA system and CSP algorithm focus on identifying applications and their individual services, [13, 16, 19] are not discussed in the main context paragraphs.

Table 1. Chronological order of related work that focus on generating signatures for Internet applications and malicious traffic identification.

Approach or Author	Year	Algorithm or Technique	Focus or Strengths
P.K Janbandhu & M.Y Siyal [12]	2001	DSA	Biometric Signatures for Internet-based applications
Polygraph [13]	2005	BGA	Signatures for Polymorphic worms
LASER [14]	2008	LCS	Internet application identification
AutoSig [15]	2009	ASMA	Applications signatures generation
R. Perdisci et al. [16]	2010	BCA	HTTP malware signatures generation
Zhanyi Wang [17]	2012	ANN & DL	Network traffic identification
Y. Wang et al. [18]	2012	SAA	Automatic signature generation
FIRMA [19]	2013	LCS	Generation of network signatures for malware
M. Cheng et al. [20]	2013	PCA	Automatic signature extraction
FlowAntEater [21]	2013	*K*-means & SSMA	Generation and evaluation of applications signatures
J. Tharp et al. [22]	2013	Heuristics	Reconciliation of multiple matching
Hwan-Hee Kim & Min-Jung Choi [23]	2013	Heuristic	Signatures generation for HTTP-based applications
S. Yoon et al. [24]	2015	Heuristic	Internet traffic identification
FLOWR [25]	2015	SLA	Mobile apps identification
D. Mann et al. [26]	2015	DS	Biometric-based digital signatures
Hyun-Min An et al. [27]	2015	Statistical	Generates unique signatures and resolve abnormal TCP behaviors
SigBox [28]	2017	MSPA	Automatic signature generation

P.K Janbandhu and M.Y Siyal [12] introduces biometric signatures generating approach that integrates biometrics with public key infrastructures. The biometrics signatures are secure, efficacious, fast, convenient, non-invasive and correct in Internet-based applications identification. LASER [14], is a method that automatically generates an application signature, in the form of a sequence of substrings, in the payload of a packet by using a modified version of the longest common subsequence (LCS) algorithm. The inputs of this algorithm, distinct byte streams of packet payloads that belong to two flows. To improve performance of the system in terms of execution time and accuracy, this method considers only the first N number

of packets of a flow and groups these packets by their size, since large packets are not likely to carry the same kind of information as small ones. The method compares two inputs to obtain the longest comsmon subsequence between them, and then compares it with other subsequences iteratively to refine the extracted common substrings. LASER extracts the longest common subsequence from only two designated sequences, making it hard to extract efficient signatures.

In Autosig[15], applications signature candidates are generated automatically and multiple common substring of sequences from input flows are extracted as the application signature. The approach, it first divides the payload of a set of flows into short substrings called shingles. After extracting all the relevant, common shingles, Autosig merges them whether they are neighbors or overlapping. Next, a substring tree is constructed to create all the possible combinations of substrings. These combinations are considered as signatures. Although this method generates signatures automatically without determining the order of the input traffic, several minimum supports must be determined by performing a repetitive trial and error experiments. Repetitive trial and error experiments for checking the minimum support value, not clarifying clearly how are the number of sequences input taken or extracted and the form by which signatures candidates and signatures are generated, convinces that signatures generated by Autosig are less efficient.

Zhanyi Wang [17], proposes a method that is based on artificial neural network and deep learning to extract well learned features and signatures for applications identification and anomalous protocol detection. Y. Wang et al. [18], focus on generating regular expression signatures with certain subset of standard syntax rules, which are of sufficient expressive power and compatible with most practical systems. The approach takes a labeled training data set as an input to produce a set of signatures for matching the application classes presented in the data. This approach involves four procedures, *pre-processing to extract application session payload, tokenization to find common substrings and incorporate position constraints, multiple sequence alignment to find common subsequences,* and *signature construction to transform the results into regular expression*. M. Cheng et al. [20], proposes an automatic signature extraction mechanism using Principal Component Analysis (PCA) technology to automatically extract signatures. The signatures extracted by this method are expressed in the form of serial consistent sequences constructed by principal components.

FlowAntEater [21], is a proposed software framework on NTASG (network traffic automatic signatures generation) which uses the *K*-means cluster algorithm to purify the traffic flow and automatically generating network traffic signatures. The approach has systematic signatures management algorithm. From a single signature multiple matches may arise whereby more than one application can be identified by that single signature. J. Tharp et al. [22], address this problem of multiple matches by developing a set of selective heuristics that can help to accurately identify the application associated with the input data stream uniquely. Hwan-Hee Kim and Min-Jung Choi [23], propose a system to exactly identify applications in HTTP traffic from extracted signatures and reduce the overhead during signatures generation. However, the approach appears to be too heuristic with no algorithm implemented. S. Yoon et al. [24], propose a new signature model called a behavior signature for Internet traffic identification that utilizes the inter-flow relation of application traffic. The behavior signature is said to be unique traffic behavior pattern appearing in the first few packets of traffic flows. However, this approach as well has limitations to accurately identify an application or it individual services since it is behavioral and considers only first few packets of a flow.

FLOWR [25], is a system that automatically identifies mobile apps by continually learning the apps features via traffic analysis. FLOWR focuses on key-value pairs in HTTP headers and

intelligently identifies the pairs suitable for app signatures. The system employs a custom supervised learning approach that leverages a very limited knowledge of app-signature seeds and autonomously grows its capacity for app identification. This approach motivation comes from a simple but effective hypothesis that unknown app-identifying features may occur with the known signatures. D. Mann et al. [26], generate biometrics digital signatures in simple and secure ways, such as using the AES which is much secure and makes the generated signature useful. However, the method is hard to implement since it requires cryptography knowledge.

Hyun-Min An et al. [27], analyze the limitations of traffic classification caused by abnormal TCP behavior, and propose a novel application-based traffic classification method using a statistical signature with resolving abnormal TCP behaviors. This method tries to resolve abnormal TCP behaviors and generate unique signatures to classify each application using the packet order, direction, and payload size of the first N packets in a flow. SigBox [28], is a system that automatically generates signatures for fine-grained traffic identification. Using a modified sequence pattern algorithm, the system extracts three types of signatures, *content*, *packet*, and *flow* signature. A flow signature, is a series of packet signatures, and a packet signature is a series of content signatures. A content signature is defined as a distinguishable and unique substring of the packet payload. A survey [29], takes different analysis as well on signature generation methods.

Although are far related to signature generation, [30-32] focus on managing networks and security which is also the ultimate goal of generating signatures. In [30], I. Butun et al propose an authentication methodology to detect the signals of IoT devices and control access to public safety networks during critical situations. Xiao Liu et al [31], mention that it is possible to build a service chain network from big data network joint SDN, cloud computing and fog computing platforms and reduce a large amount of redundant data and response time. They propose a Big Data Orchestration as a Service (BDOaaS) to dynamically orchestrate big data into services in SDN. Saeed Javanmardi et al [32], use fuzzy theory in a trust overlay network to model the network and storage of information reputation.

3. The proposed TSA System and CSP Algorithm

3.1 Architecture

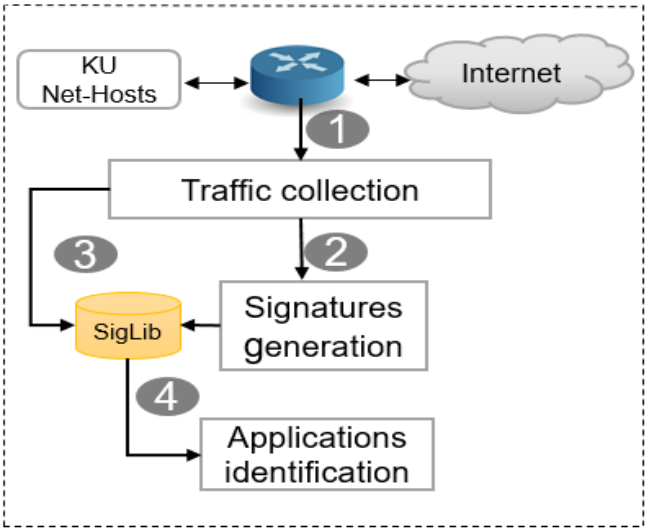

Fig. 1. The overall architecture of **TSA** system

The proposed TSA system architecture divides into three major parts, traffic collection, signature generation and applications identification, as shown in **Fig. 1**. Signatures generation part is the part where the proposed CSP algorithm applies. First, applications' traffic is collected between a target network and the Internet as **Fig. 1** (number 1) indicates. Second, from the collected target application traffic, signatures are generated and stored in the signatures library (number 2). Third, unknown applications traffic is passed to the sample signatures stored in a library for identification (number 3). Number 4 indicates a final stage where applications are identified. For clarity, the details of the overall TSA system architecture (**Fig. 1**) are later given in more elaborate and separated figures for each part.

In GT (ground-truth) traffic collection, packets payloads are captured in string sequences or hex forms through free and open source tools, Wireshark [33] and Microsoft Network Monitor [34]. When the target traffic is collected, we extract and group it to packet-level, content-level and flow-level. In every capture, when several applications traffic is collected only traffic belonging to a single application are separated for signatures generation. This separation is done under 3-tuple information, which are *src-IP, dst-IP and Layer-4 transport protocol*. We do not group captured flows based on either their source or destination port numbers, as they never endorse applications in modern networks. At this stage, due to behavioral trends of applications, a target application may roughly be identified. However, to further identify individual services belonging to that application, further separation takes place along with further generation of flows of a target application service. Since it is impossible to further separate the target application traffic online, this process is manual and done offline as **Fig. 2** shows. Of the four stages indicated in **Fig.1**, this process contributes 12.5% to the whole TSA system, thus making falls to 87.5% automated.

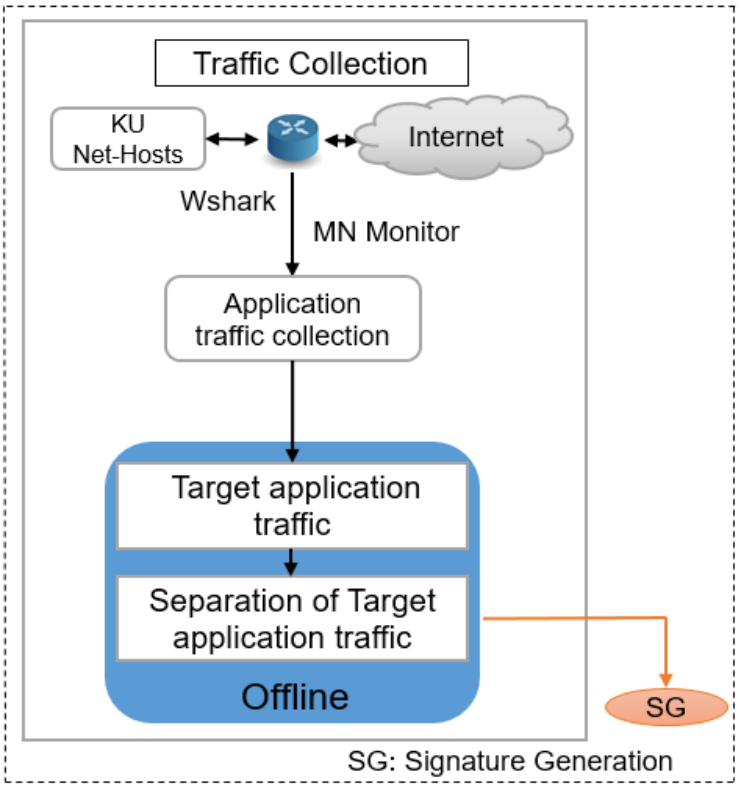

Fig. 2. Traffic collection architecture.

After separation of target application traffic, the following stage is signatures generation. The separation is essential since a single application can simultaneously provide multiple services such as messenger, audio calls, video calls, file sharing and so on. To generate efficient signatures for a unique application service, it is essential to further generate and separate several sequences from that application service traffic repeatedly, making it time-consuming and tedious process. For efficient signatures generation of packets payload, the proposed approach generates content-level signatures, packet-level signatures and finally flow-level signatures as the highest level to highest length-k signature as shown in **Fig. 3**. However, if flows of a target application or one of its service were correctly collected the CSP algorithm can straight forward generate DNA-like signatures which are unique identifier of the target application or one of its service without the need to go through the *content-packet-flow* procedure, indicated in **Fig.3**. The reason why it is essential and efficient to generate signatures through three stages, *content-level* to *packet-level* to *flow-level* is because if some packets are missing it is hard to generate a DNA-like unique signature.

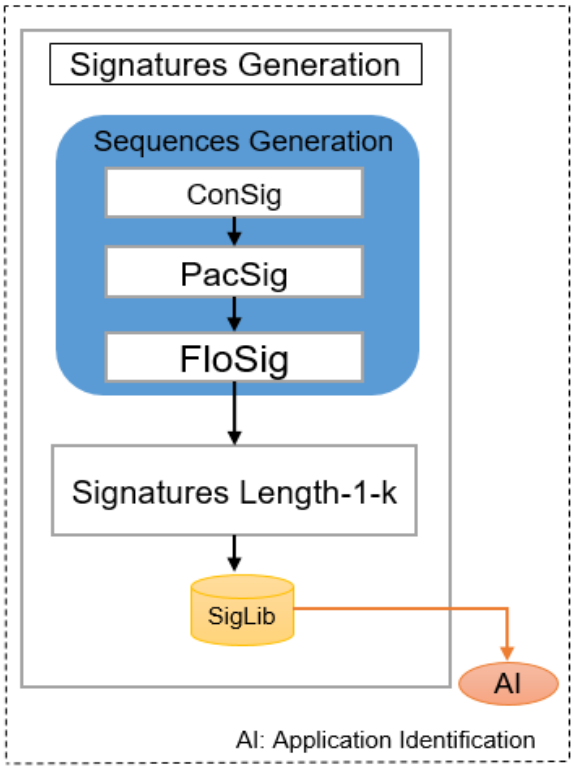

Fig. 3. Signatures generation architecture

3.2 CSP Algorithm

3.2.1 Introduction

The proposed CSP algorithm automatically generates DNA-like contiguous signatures from extracted several sequences. Contiguousness refers to repeatedly occurring of small patterns, right next to each other. Motivating works of this paper discussed in section 1 [7-9] and [10-11], biological data sequences and DNA pattern mining respectively, indicate that DNA patterns of an animal are mostly contiguously extracted. The common frequent DNA traits

extracted contiguously, are more likely to provide clues for genetic discovery and biotechnologies identification. This biological fact is the foundation of the proposed CSP Algorithm as it suggests similar behaviors to be applicable in applications identification, especially individual services identification of an application.

In CSP algorithm flow-level signature generation, a *SequenceSet* of an application or application service traffic is generated as shown in **Equation 1**. Number of ***N*** (***N*** = 1, 2, ...n) in *Sn* represents the number of *n* sequences belonging to an application or an application service. As the matter of fact, the more sequences are extracted, the higher the probability for the CSP algorithm to generate DNA-like, contiguous and unique signatures.

$$S_{equenceSet} = \{S_1, S_2, \ldots S_n\} \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \quad (1)$$

$$(S_n = Sequence, \{s.t.\ n = \boldsymbol{n}^{th}\ number\ of\ Length\text{-}1\ unique\ patterns\ for\ each\ sequence\})$$

Table 2, indicates four packets payload contents belonging to the same flow, that we randomly captured. The packets indicate how random and complicated the payload contents may look, however, the common frequent pattern "**FwMDAB0AAAAAAAAB**" can be observed indicating a DNA-like and unique relation of the four packets. Flow-level signatures are generated from packets payload contents as shown in **Table 2**, however, for easy clarification of the proposed algorithm, we created a simpler SequenceSet of two sequences to generate contiguous and DNA-like signatures as shown in **Fig. 4**. The two sequences in **Fig. 4**, represent packets payload content in strings form at flow-level.

Table 2. Four packets payload contents of the same flow

Packet_ID	Packet payload contents
5015-QUE	FwMDAB0AAAAAAAABZtvD15CnpwXiX0g2FcJeYyoenN+/2w==
5017-RES	FwMDABoAAAAAAAABmppEfFUyNx6n881LGQnyAo5KdA==
5018-QUE	FwMDAB0AAAAAAAABZ3aIyvuA8KJNTPGP0a1Zzzxwf4hlVA==
5020-RES	FwMDAB0AAAAAAAABm6Cxj14EysdqZHwtnmL5jvFVXWA0fg==

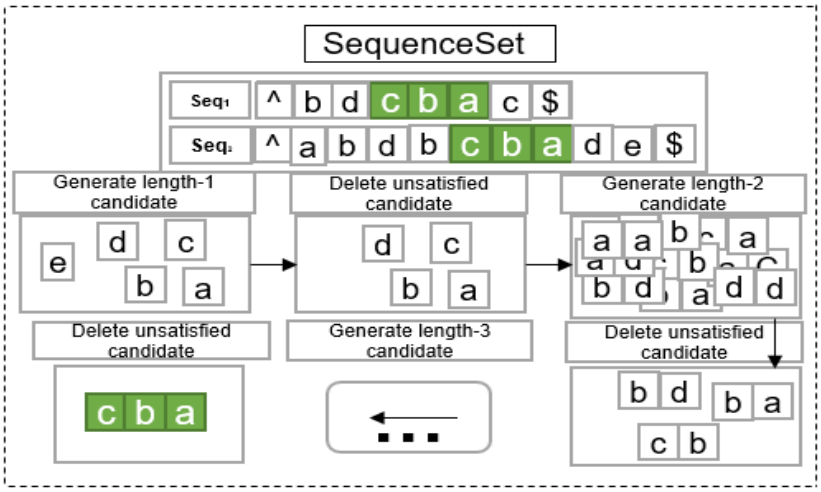

Fig. 4. An example of contiguous and DNA-like signatures generation in CSP algorithm

From two sequences in **Fig. 4**, length-1 unique candidates are extracted. In this case <a>, <b>, <c>, <d>, <e> are extracted as length-1 unique candidates. We set the support for this example as 1, (*threshold = 1*). The threshold is checked and unsatisfying Length-1 unique candidates are deleted. As shown is **Equation 2**, we define the threshold value as the number of support traffic sequences to total traffic sequences. The remaining length-1 unique candidates generate all possible combinations of Length-2 signature candidates, contiguously. i.e. all possible contiguous length-2 candidates that exist in the two sequences. Length-2 candidates that do not satisfy the threshold value are deleted. The remaining length-2 candidates generate all possible length-3 contiguous candidates. The threshold value is checked again whereby unsatisfying length-3 candidates are deleted. In **Fig. 4**, length-3 contiguous candidate "***cba***", is extracted as a DNA-like signature with threshold value of 1. Similar process iterates to highest length-k phases, according to the input sequences number and the defined threshold (∅).

$$Threshold\ (\emptyset) = \frac{\text{Number of Support traffic sequences}}{\text{Total traffic sequences}} \quad \ldots\ (2)$$

As further depicted in **Fig. 5**, CSP algorithm generates contiguous signature candidates and DNA-like, unique signatures. The green color highlighted length-3 patterns (*seq1, seq3, seq5, seq7, seq9 and seq11*), are contiguous signature candidates that CSP algorithm generates while the black color highlighted patterns (*seq2, seq4, seq6, seq8, seq10 and seq12*), are non-contiguous signature candidates not generated by the proposed CSP algorithm. Moreover, although up to 27 contiguous signature candidates can be generated from the three pattern, a, b and c, CSP algorithm considers all the patterns differently, that is; abc ≠ acb ≠ bac ≠ bca ≠ cab ≠ cba.

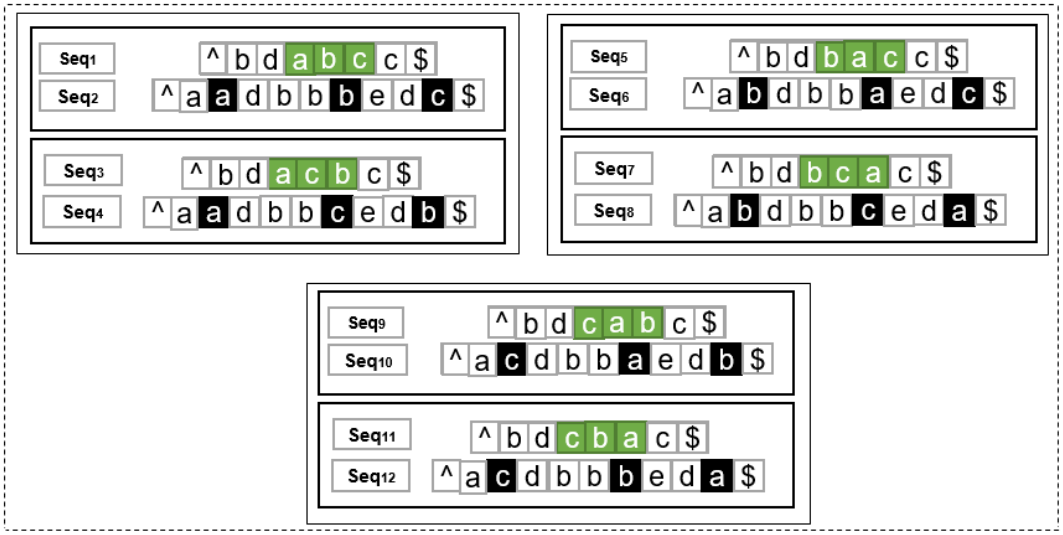

Fig. 5. Exrtraction of contiguous patterns in CSP algorithm (*seq: sequence*)

3.2.2 Pseudocodes for signature generation

The proposed CSP ***pseudocode*-algorithm part 1**, describes the general signature generation process. The most common frequent and contiguous substring extracted with highest length-k,

is the DNA -like and a unique identifier signature of an application or an application service. At first the algorithm orderly extracts unique length-1 candidates SequenceSet and store them in *Ck* set. (**Fig. 6**, lines 1-4). From the extracted length-1 unique candidates, candidates that satisfy the given threshold are extracted and stored in *Sk* set, (**Fig. 6**, lines 5-11). All possible length-2 to length-k contiguous candidates are generated and only the contiguous candidates satisfying the threshold are extracted and stored in C(*k*+1) set (**Fig. 6**, lines 12-19). Subsets inclusion relation is checked, and all the subsets are deleted (**Fig. 6,** lines 20~26).

Fig. 6. Pseudocodes for Signature generation

***Procedure**: Candidates&SignaturesGeneration*
***Input**: SequenceSet,MinSup*
***Output**: SignatureSet*

```
01: k ← 1, Ck ← Ø;
02: foreach item i in SequenceSet of seq do
03:    Ck ←Ck ∪ i;
04: end // generation of length-1 unique candidates
05: while Ck ≠Ø do
06:       Sk ← Ø;
07:       foreach element e in Ck do
08:          if (sup(e)) ≥ MinSup then
09:             Sk ← Sk ∪ e;
10:          end
11:       end //length-1 unique cand. MinSup satisfiers
12:       Ck+1 ← Ø;
13:       foreach element pair x,y in Sk do
14:          if x1,k-1 = y0,k-2 then
15:             Ck+1 ← Ck+1 ∪ {x0, k-1 + yk-1,k-1};
16:          end
17:       end
18:       k ← k+1;
19: end // generation of length-k candidates
20: foreach Si in i = 1, ... k-1 do
21:    foreach element x in Si do
22:       if ∃y in Sk, k > i s.t x ⊂ y then
23:          Si ← Si – {x};
24:       end
25:    end
26: end // subset deletion
27: return Sk-1; // Output SignatureSet = Sk-1
```

Finally, highest length-k is extracted as a unique application service signature. Set S(k-1), (**Fig. 6**, line 27). CSP algorithm has three strict conditions in generating DNA-like signatures. We summarize the conditions, consecutively as follows;

- Neither length-k nor ANY threshold (Ø) is considered during random signature candidates generation. However, since CSP algorithm considers length-3 as a minimum length for an efficient and significant signature, length-3 and above extracted signature candidates with over 0.5 threshold, are considered as potential signatures.
- In any given number of sequences and for any length-k, the threshold (Ø) of a signature candidate should be greater than 0.5 (Ø > 0.5), to be considered a DNA-like and unique identifier signature of an application or an application service
- Any signature candidate with threshold (Ø) equal or less than 0.5 is neither extracted nor considered as a DNA-like and a unique identifier signature.

In **Fig. 7**, we provide a simple case study to illustrate the proposed **Fig. 6.** In this process, A DNA-like signature of an application(App_1) is generated. First, at stage 0 we select an application for generating its respective DNA-like signature (App_1). At stage 1, six sequences (payload contents) of an App1 are extracted from which signature candidates are generated at stage 2. At stage 2, signature candidates are randomly generated from Length-1(L_1) to Length-k(L_k). Stage 1 and 2, are the core stages of DNA-like signature generation. At stage 3, the signature candidates are checked for threshold (Ø) satisfaction, where signature candidates above 0.5 threshold (Ø) are extracted. Signature candidates with threshold less than or equal to 0.5 are removed. At stage 4, the signature candidates satisfying the threshold are length checked. At this stage, signature candidates with Length-3 or higher (L >=3) are extracted while those not satisfying this condition are stored in library, $store_1$. Signature candidates with (L >=3) are processed again where the signature candidate with highest length(L_{max}) is extracted as a DNA-like signature and unique identifier of an application (App_1). The remaining signature candidates are stores in library $store_2$, for future analysis.

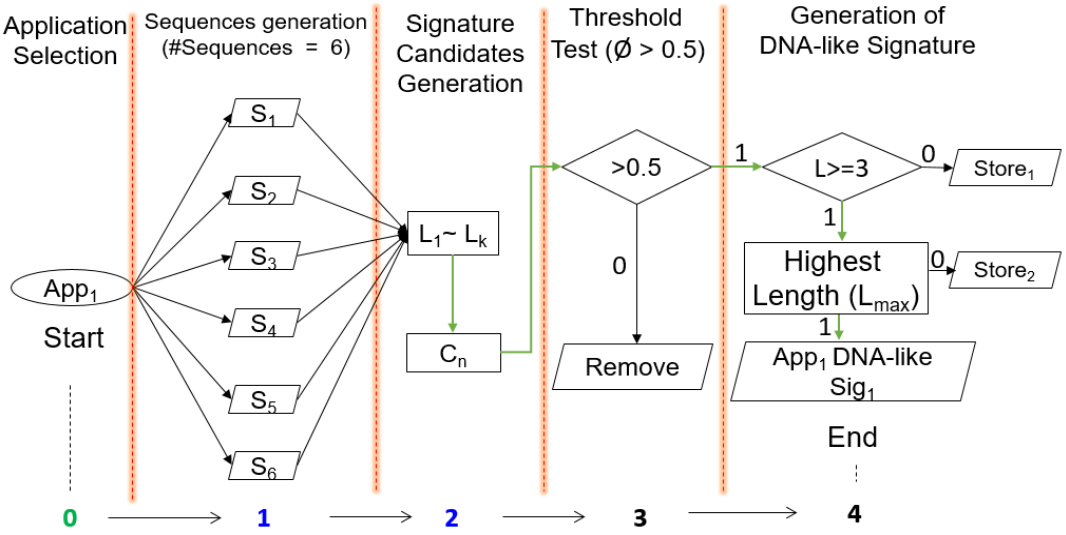

Fig. 7. A case study example for the proposed CSP algorithm part 1

3.2.3 The architecture and pseudocodes for identification

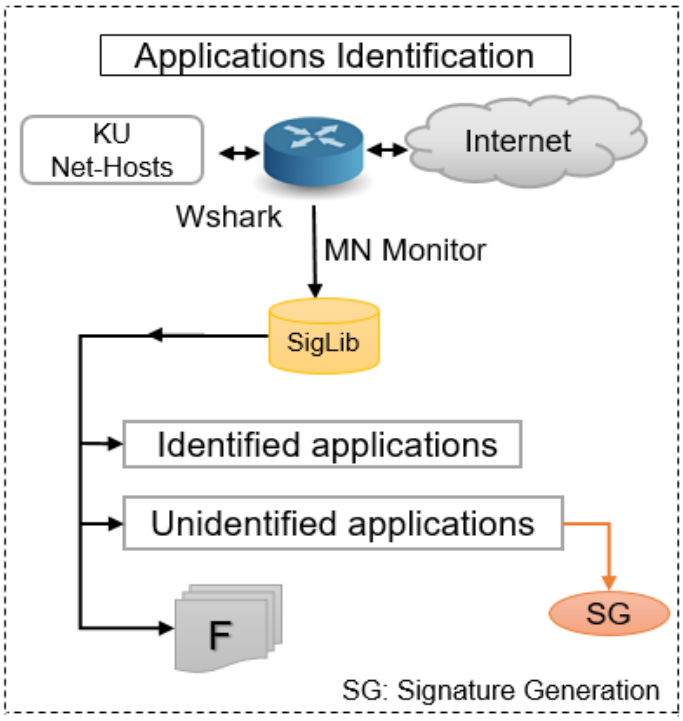

Fig. 8. Applications identification architecture

Once the signatures are generated are stored in a signature library. However, due to application traffic dynamic over time caused by network environment such as protocols variations and contents being transmitted, the signatures easily lose consistency as false positives rise. To encounter this challenge and ensure efficiency in signatures at any time, the applications identification architecture is designed as shown in **Fig. 8.** When the unknown application traffic is passed to the signatures library, an identified and unidentified application or application service is outputted. The traffic of unidentified application services is collected, studied and its signatures are newly generated. From this mechanism, CSP algorithm can discover new DNA-like signatures, previously unknown. Old signatures in the library are eliminated from the library and stored in a library *F* for future analysis. Elimination of old signatures from the library is necessary to keep the signatures up to date and efficient.

Fig. 9, is the pseudocodes for Identification process. The unknown application traffic is the input to the signatures library. The unknown application traffic is passed to a bunch of signatures from which the identified traffic is labelled as *App(t)* and grouped to identified application traffic Tid-Set. Application traffic that is not identified is grouped to unidentified traffic Tunid-Set. The identification process iterates until no more unknown application traffic to be passed (**Fig. 9**, lines 1-9). Both, accurately identified traffic and those unidentified are returned as Tid and Tunid, respectively (**Fig. 9**, line 10).

```
Procedure: Apllication Identification
Input: SignatureSet, Uknown trafficSet
Output: Identification Results
01: Tid ← Ø, Tunid ← Ø ;
02: foreach traffic in unknown trafficSet do
03:   if ( ∃ s in SignatureSet s.t  s⊆ t) then
04:       App(t) ← App(s);
05:       Tid ← t;
06: else
07:       Tunid ← t;
08:   end
09: end // traffic identification
10: return Tid, Tunid  //Identification Results
```

Fig. 9. Pseudocode for application identification

A case study example for the proposed **Fig. 9**, solely relies on the DNA-like signature generated in **Fig. 7**. However, due to space limitation we couldn't include its diagram. In this case, an application(App_1) is uniquely identified. We take three applications (App_1, App_2 *and* App_3) traffic sequences to a library where a DNA-like signature for App_1 is stored. Our expectation here is to identify only App_1 from the signature. Based on the received sequences, a perfect matching should be 100% true positive for App_1. However, such a perfect matching and identification of App_1 is never possible in real network environment due to noise sequences. Thus, App_2 and App_3 may be wrongly identified by noise sequences of application (App_1), making its true positive always less than 100%.

3.2.4 Analytic comparison of the proposed approach

Table 3. Evaluation factors for an efficient signatures generating method

Work	Automation %	Traffic level	# Sequences input	Signatures form and length-k	Threshold (Ø)
Y. Wang et al. [18]	Unanalyzed	Session	Multiple	Regular expression	Used but untold value
M. Cheng et al. [20]	Unanalyzed	Flow	N/A	N/A	Variant
FlowAntEater [21]	Unanalyzed	Flow	Multiple	Random	Used but untold value
S. Yoon et al. [24]	Unanalyzed	Packet	N/A	N/A	Used but untold value
Hyun-Min An et al [27]	Unanalyzed	Packet	N/A	N/A	Used but unclear
SigBox [28]	Unanalyzed	Flow	Multiple	Longest length-k	N/A
Proposed method	87.5%	Flow	*N* (limitless)	Contiguous & DNA-like, Minimum Length-3	> 0.5

In Section 3.2.1 and 3.2.2, we described how contiguous, DNA-like and unique signatures are generated based on the proposed TSA system and CSP algorithm for applications and

individual application services identification. To justify novelty of the proposed system and algorithm, this section, presents two tables, **Table 3** and **Table 4** that compare the proposed TSA system and CSP algorithm to other six works in various aspects.

Table 3, shows five factors that are significant in evaluating how good a method or a system is on packet-payload signatures generation, however, are not standards. Although, all the approaches do not analyze to what extent they are automated, the general proposed approach system is analyzed to about 87.5% automated, as also explained in section 3.1. If we were dealing with generating signatures for identifying a protocol format, then an approach [18] utilizes the right traffic level, however in signature generation, flow-level traffic is significant. In packet-payload signature generation, the number of sequences collected highly affects the final signature to be extracted. Thus, the proposed CSP algorithm is designed to receive limitless number of sequences as input. Since CSP algorithm is motivated by the fact that frequent and common contiguous packet-payload pattern, are most likely to provide uniqueness in applications identification, we generate contiguous, DNA-like signatures of length-3, a minimum length. In CSP algorithm, when given N number of sequences, a signature candidate must exceed 0.5 threshold ($\varnothing$) to be extracted a DNA-like and unique signature. Threshold value setting is significant in generating efficient signatures, however the rest of the approaches do not clearly clarify.

Besides, **Table 4** in general, summarizes the superiority of the proposed method, as an approach capable of identifying both, an application and its individual services.

Table 4. General comparison of the proposed approach over related works

Work	Year	Proposed System	Developed Algorithm	Identification of applications	Identification of individual app. services
Y. Wang et al [18]	2012	✓ (Unnamed)	X	✓	X
M. Cheng et al [20]	2013	✓ (Unnamed)	X	✓	X
FlowAntEater [21]	2013	✓ (FlowAntEater)	✓ (SSMA)	✓	X
S. Yoon et al [24]	2014	✓ (Unnamed)	X	✓	X
H. Min An et al. [27]	2015	✓ (Unnamed)	X	✓	X
SigBox [28]	2017	✓ (SigBox)	✓ (MSPA)	✓	X
Proposed method	2017	✓ (TSA)	✓ (CSP)	✓	✓

4. Experimental Results and Analysis

This section, presents experimental analysis of five applications, Afreeca, BitTorrent, Skype, KakaoTalk and Diablo III. Based on the five HTTP protocol based applications, several experiments were conducted to test the feasibility of the proposed TSA system and CSP algorithm. Experimental results are presented in three divisions; signature generation, applications identification and DNA-like unique signature generation for an application individual services identification.

4.1 Time-Space Complexity of the proposed Method.

In this section, we summarize both time and space complexity of the proposed approach in two parts, signature generation and application identification part. Generally, when finding a DNA-like signature for long and complicated payload contents of an application or an application service, the generation of signature candidates and the DNA-like signatures is

done in ***n*** nested loops where in each nested loop an independent memory for storage is utilized. Therefore, the general running time of CSP algorithm for signature generation at worst case is $O(n^n)$ time complexity and O(n) in space complexity. However, based on signature generation simulations done, **4** nested loops are computed, making the worst-case $O(n^4)$ and O(n) in time and space complexity, respectively. Once the DNA-like signatures are generated, applications identification process follows in which a single loop function is involved. Applications identification phase has O(n) and O(1) at its worst-case time and space complexity, respectively.

4.2 Flow-level Signature generation.

We define a flow-level signature as a signature extracted from a set of 3-tuple (*source IP, destination IP and layer-4 transport protocol, TCP or UDP*) packets payloads. Thanks to tools such as Wireshark [33] and Microsoft Network Monitor [34], since the identification of applications in general is no longer a challenging task. The tools are powerful to extract network traffic at any time and identify their origin hosts. However, it is very hard for normal network administrators to identify such applications without having sample signatures and it is even harder to identify individual services or hosts of an application without the sample signatures. The solution of this problem is introduced in section 4.3, where contiguous and DNA-like, unique signatures for Afreeca and Diablo III are generated. In **Table 5**, we present HTTP protocol based, sample request lines with their total number extracted for the five applications, at flow-level. It should be noted that the sample HTTP request lines in **Table 5**, do not exactly reflect all the number of HTTP protocol based requests lines indicated in the last column of **Table 5**. HTTP request lines vary in almost each web server or web application request.

Table 5. HTTP protocol based request lines for the five applicatiuons

Application	HTTP protocol based, sample request line	# HTTP request lines extracted
Afreeca	GET /css/global/afmain/afmain.css HTTP/1.1	22
Bit Torrent	GET / HTTP/1.1 Host: www.bittorrent.com\r\n	9
Skype	GET / HTTP/1.1 Host: www.skype.co.kr / Host: skype.daesung.com	17
KakaoTalk	GET /th/talkp/wkCYS8Ri0V/7cFGY0gvxBtc8YLaeRrzm0/ rhop9y_110x110_c.jpg HTTP/1.1	10
Diablo III	GET /d3/ko/ HTTP/1.1	15

4.3 Applications identification

Based on **Table 5**, several flow-level sequences are selected, and several signature candidates are extracted. From the extracted signature candidates, those whose threshold values are highest, are what depicted in **Table 6**. As described in section 3, CSP algorithm generates signatures from signature candidates whose threshold values exceed 0.5. Since the threshold values for all generated signatures are far higher than the 0.5, the applications as well are identified with high accuracy. However, a high threshold for a generated signature does not always guarantee a high accuracy, because both accuracy and threshold have two different mechanisms and the threshold computations are done prior to accuracy computations.

$$Accuracy = \left\{ \frac{YES}{YES+NO} \right\} x\ 100\% \quad \cdots\cdots (3)$$

YES = *#Target application traffic identified by a signature*, **NO** = *#Target application traffic unidentified by a signature*

Table 6. Number of sequences extracted, threshold and accuracy

Application	#traffic Sequences	#support sequences	Threshold of a signature	YES	NO	Accuracy (%)
Afreeca	7	5	0.71	9	2	82.80
Bit Torrent	6	6	1.00	10	0	100.00
Skype	8	7	0.86	7	1	87.50
KakaoTalk	5	4	0.80	8	1	88.90
Diablo III	11	8	0.73	10	1	90.91

Accuracy is defined as the number of target application traffic identified by a signature to a total number of target application traffic identified by a signature plus traffic unidentified by a signature as shown in **Equation 3**. An accuracy value ranges from 0% to 100%. If a signature matches none of the target application traffic passed to it, then the signature accuracy scores 0%, whereby if the signature matches all the target application traffic, the signature accuracy scores 100%.

4.4 Individual services identification of an application

In this section, we analyze and generate contiguous, DNA-like and unique signatures to identify individual application services (hosts) of two applications, *Afreeca* and *Diablo III*. Individual application services of an application can be, a messenger service, voice calls services, video calls service, file sharing service, shopping, gaming and so on. However, in this experiment the individual services, are identified by their DNA-like signatures of particular hosts. **Table 7** presents 6 packets payload contents belonging to one flow, extracted from a host "admin.img.afreecatv.com". Its respective contiguous and DNA-like signature is shown in **Table 8.**

Table 7. HTTP protocol based request lines for a host "admin.img.afreecatv.com"

Application service Host	Payload contents
admin.img.afreecatv.com	GET /hotissue_vod/2017/07/07/2496595f18a43ec0a.jpg HTTP/1.1
	GET /hotissue_vod/2017/07/07/7274595f18717d956.jpg HTTP/1.1
	GET /hotissue_vod/2017/07/07/6626595ef95f2f555.jpg HTTP/1.1
	GET /hotissue_vod/2017/07/07/2496595f18a43ec0a.jpg HTTP/1.1
	GET /hotissue_vod/2017/07/07/7274595f18717d956.jpg HTTP/1.1
	GET /hotissue_vod/2017/07/07/6626595ef95f2f555.jpg HTTP/1.1

Table 8. DNA-like signature generation for the host "admin.img.afreecatv.com"

Application	# of Sequences	Signature Candidates(L-3)	Threshold	DNA-like Signature
AfreecaTV	6	595	1.0	595
		859	0.6	
		95f	0.6	

Table 9, presents 6 HTTP protocol based, sample request lines belonging to one flow, extracted from a host "liveimg.afreecatv.co.kr". Its respective contiguous and DNA-like signature is shown in **Table 10.** From this experiment we could discover that, there is a clear DNA-like and unique signature that can identify a specific service of an application. These signatures appear contiguously and constant regardless of the number of HTTP requests and regardless of a browser that a client may have applied. DNA-like signatures that are solely undocumented to average network administrators appear in all forms. The DNA-like signatures are composed of either only strings, only numerals, only special characters or a mixture of any ASCII codes.

Table 9. HTTP protocol based request lines for a host "liveimg.afreecatv.co.kr"

Application service Host	Payload contents
liveimg.afreecatv.co.kr	GET /193887189_*240x135*.gif?4 HTTP/1.1
	GET /193863873_*240x135*.gif?3 HTTP/1.1
	GET /193886771_*240x135*.gif?9 HTTP/1.1
	GET /193884599_*240x135*.gif?4 HTTP/1.1
	GET /193802223_*240x135*.gif?7 HTTP/1.1
	GET /193885374_*480x270*.gif?6 HTTP/1.1

Table 10. DNA-like signature generation for the host "liveimg.afreecatv.co.kr"

Application	# of Sequences	Signature Candidates (L-4)	Threshold	DNA-like Signature
AfreecaTV	6	0x13	0.83	1938
		1938	1.00	
		240x	0.83	
		40x1	0.83	
		9388	0.66	
		_240	0.83	
		x135	0.83	

Similarly, for Diablo III, **Table 11** presents 3 HTTP protocol based, sample request lines belonging to one flow, extracted from a host "bnetcmsus-a.akamaihd.net". Its respective contiguous and DNA-like signature is shown in **Table 12**. As the payload contents show in **Table 12**, a signature '*149*' is a DNA-like and unique identifier signature of Diablo III, application host "*bnetcmsus-a.akamaihd.net*". A single host provides a specific service of an application and has a DNA-like signature which is a unique identifier signature of that host and the service provided. In **Table 11**, We couldn't include many payload content sequences due to space limitations.

Table 11. HTTP protocol based request lines for a host "bnetcmsus-a.akamaihd.net"

Application service Host	Payload contents
bnetcmsus-a.akamaihd.net	GET /cms/blog_thumbnail/7r/7R958LUWZVRO1498773751026.jpg HTTP/1.1
	GET /cms/blog_thumbnail/if/IFQK2ZAUD4Y11497921286440.jpg HTTP/1.1
	GET /cms/blog_thumbnail/vf/VFC4Y21FN8XK1495055257876.jpg HTTP/1.1

Table 12. DNA-like signature generation for the host "bnetcmsus-a.akamaihd.net"

Application	# of Sequences	Signature Candidates(L-3)	Threshold	DNA-like Signature
Diablo III	3	149	1.0	149

In this section, we have presented experimental results in three divisions; signature generation, applications identification and DNA-like unique signature generation for an application individual services identification, however, based on experimental analysis focus and presentation, comparisons with related works is inapplicable.

5. Discussion

Signature generation is the following stage after network application traffic had been collected and separated. Indeed, collection of GT (Ground Truth) traffic is done randomly. Random collection of traffic which is unavoidable due to the current developed tools, hence results to hard manual work on separating the target applications traffic. Moreover, encrypted applications such as skype, have limitations on packets payloads signature extraction, however, generating signatures for encrypted applications is beyond the contribution of this paper.

Table 13. Massive signature candidates' generation in CSP algorithm *(T.S = Total Signatures)*

L-1 (*U.P*)	L-2	L-3	L-4	L-k
	'aa'	'aaa'	'aaaa'	'a^k'
'a'	'ab'	''	''	''
	'ac'	''	''	''
	'ba'	''	''	''
'b'	'bc'	"-----"	"----"	'----'
	'bc'	''	''	''
	'ca'	''	''	''
'c'	'cb'	''	''	''
	'cc'	'ccc'	'cccc'	'c^k'
T. S = 3	T. S = 9	T. S = 27	T. S = 81	T. S = 3^K

The proposed CSP algorithm generates substantial number of signature candidates as shown in **Table 13** and depicted in **Fig. 10**. **Table 13**, shows signature candidates generation from length-1(L-1) to length-k(L-k), of length-1(L-1) U.P (Unique Pattern), '*a*', '*b*' and '*c*'. TSA system is designed to identify both, applications and individual application services instantly under CSP algorithm in high accuracy. From the generated flow-level signatures, applications with their individual services are identified. However, since the signatures generating module is conducted offline, it affects the efficiency of the system as false positives arise. To solve the problem and maintain the efficient of the proposed system in general, better techniques are under consideration.

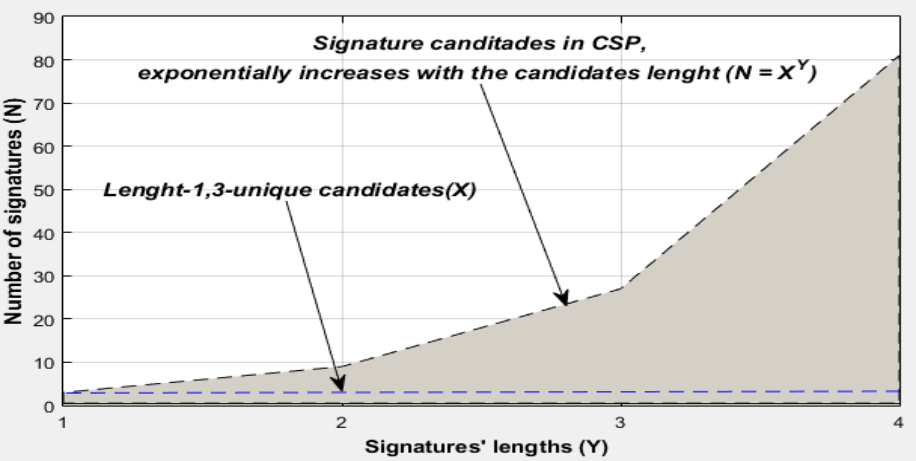

Fig. 10. Massive signature candidates' generation in CSP algorithm

6. Conclusion and Future Work

This paper, proposes a novel, CSP algorithm for signatures generation in the novel TSA system, in which signatures generation and identification of five HTTP protocol based applications is conducted. CSP algorithm and TSA system proposals go beyond, as they provide an architecture to identify not only an application but also the individual services that an application can offer. Although, there are few challenges and limitations such as random collection of mixed traffic, manual separation of target traffic, offline signature generation and so on, the proposed CSP algorithm shows promising outcomes and stimulates development of algorithms on Internet applications signature generations.

Several tasks remain as our future work on the novel CSP algorithm and TSA system. The proposed TSA system still collects GT (Ground Truth) traffic randomly meanwhile the signatures generation stage operates offline. This is our major challenge and future work, to have the whole TSA system fully automated. We also plan in the future, to provide more details, evidences and extended results of the proposed approach, especially on application individual services identification.

Appendix

TSA: Traffic collection, Signature generation, Application identification
CSP: Contiguous Sequential Patterns
DSA: Digital Signature Analysis
BGA: Bayes Generation Algorithm
LCS: Longest Common String
BCA: Behavioral Clustering Algorithm
ASMA: Adaptive Shingle Merging Algorithm
ANN: Artificial Neural Network
DL: Deep Learning
SAA: Sequence Alignment Algorithm
PCA: Principal Component Analysis
SSMA: Systematic Signatures Management Algorithm
SLA: Supervised Learning Approach
DS: Digital Signature
MSPA: Modified Sequence Pattern Algorithm

References

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021, Article (CrossRef Link), Document ID; 1454457600805266, Mar 2017

[2] Young-Joon Won, Seong-Chul Hong, Byung-Chul Park, and James Won-Ki Hong "Automated Application Signature Generation for Traffic Identification," *POSTECH*, Korea, Aug. 16, 2008.

[3] Ramakrishnan Srikant and Rakesh Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," in *Proc. of* Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology, p.3-17, March 25-29, 1996. Article (CrossRef Link)

[4] Florent Masseglia, Fabienne Cathala and Pascal Poncelet, "The PSP Approach for Mining Sequential Patterns," in *Proc. of* Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery, p.176-184, September 23-26, 1998. Article (CrossRef Link)

[5] Carl Mooney and John Roddick, "Sequential pattern mining approaches and algorithms," *ACM Computing Surveys (CSUR)*, v.45 n.2, p.1-39, February 2013, Article (CrossRef Link)

[6] Pham, Thi-Thiet, "Efficiently Mining Sequential Generator Patterns Using Prefix Trees," *Fundamenta Informaticae*, vol. 138, no.3 pp. 373-386, 2015. Article (CrossRef Link)

[7] Tae-Ho Kang, Jae-Soo Yoo and Hak-Yong Kim 2008. "Mining frequent contiguous sequence patterns in biological sequences," in *Proc. of IEEE International Conference on Bioinformatics and Bioengineering*, 2007, pages 723-8. Article (CrossRef Link)

[8] Syeda Farzana Zerin, Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer and Byeong-Soo Jeong, "A fast-indexed based contiguous sequential pattern mining technique in biological data sequences," in *Proc. of Proceedings of 2nd International Conference on Emerging Databases (EBD'10)*, Jeju.

[9] Subramanian Rajasekaran, Lawrence Arockiam, "Frequent Contiguous Pattern Mining Algorithms for Biological Data Sequences," *International Journal of Computer Applications*, vol. 951, no. 14, 15-20 June 2014. Article (CrossRef Link)

[10] Md. Rezaul Karim, Md. Mamunur Rashid, Byeong-Soo Jeong and Ho-Jin Choi, "An Efficient Approach to Mining Maximal Contiguous Frequent Patterns from Large DNA Sequence Databases," *Genomics & Informatics*, vol. 10, no. 1, pp.51-57, March 2012. Article (CrossRef Link)

[11] Syeda Farzana Zerin and Byeong-Soo Jeong, "A Fast-Contiguous Sequential Pattern Mining Technique in DNA Sequences Using Position Information," *IETE Technical Review*, vol.28, Issue. 6, Sep 2014. Article (CrossRef Link)

[12] P.K. Janbandhu, M.Y. Siyal, "Novel biometric digital signatures for Internet-based applications," *Information Management & Computer Security*, vol. 9, Issue. 5, pp.205-212, 2001. Article (CrossRef Link)

[13] James Newsome, Brad Karp, Dawn Song, Polygraph: "Automatically generating signatures for polymorphic worms," in *Proc. of Security and Privacy 2005 IEEE Symposium on, IEEE*, pp. 226-241, 2005. Article (CrossRef Link)

[14] Byung-Chul Park, Young-Joon Won, Myung-Sup Kim, James Won-Ki Hong, "Towards automated application signature generation for traffic identification," in *Proc. of Network Operations and Management Symposium, 2008. NOMS 2008. IEEE*, pp. 160-167, 2008. Article (CrossRef Link)

[15] Mingjiang Ye, Ke Xu, Jianping Wu and Hu Po, "Autosig-automatically generating signatures for applications," in *Proc. of Computer and Information Technology, 2009. CIT'09. Ninth IEEE International Conference on, IEEE*, pp. 104-109, 2009. Article (CrossRef Link)

[16] Roberto Perdisci, Wenke Lee and Nick Feamster, "Behavioral clustering of HTTP-based malware and signature generation using malicious network traces," in *Proc. of Proceedings of the 7th USENIX conference on Networked systems design and implementation*, p.26-26, April 28-30, 2010, San Jose, California.

[17] Zhanyi Wang, "The Applications of Deep Learning on Traffic Identification," *networks[C]//Advances in neural information processing systems*, pp.1097-1105, 2012.
[18] Yu Wang, Yang Xiang, Wanlei Zhou, Shunzheng Yu, "Generating regular expression signatures for network traffic classification," *Trusted network management Journal of Network and Computer Applications*, vol. 35, pp. 992-1000, 2012. Article (CrossRef Link)
[19] Rafique M.Z. and Caballero J. "FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors," in *Proc. of International Workshop on Recent Advances in Intrusion Detection, Research in Attacks, Intrusions, and Defenses. RAID 2013. Lecture Notes in Computer Science*, vol. 8145. Springer, Berlin, Heidelberg, 2013. Article (CrossRef Link)
[20] Mu Cheng, Huang Xiaohong, Wu Jun and Ma Yan, "Network traffic signature generation mechanism using principal component analysis," *China Communications*, Nov 2013, Vol 10, Issue 11, Page(s). 95 – 106. Article (CrossRef Link)
[21] Cheng Mu, Xu TIAN, Xiao-hong HUANG and Yan Ma, "FlowAntEater: network traffic automatic signature generator," *The Journal of China Universities of Posts and Telecommunications*, vol. 20, August 2013, Pages 69-74. Article (CrossRef Link)
[22] Justin Tharp, Jin-Oh Kim, Sang C. Suh and Hyeon-Koo Cho, "Reconciling Multiple Matches for the Signature-Based Application Identification," *Journal of Communications*, vol. 8, no. 12, December 2013. Article (CrossRef Link)
[23] Hwan-Hee Kim, Mi-Jung Choi, "Towards Automatic Signature Generation for Identification of HTTP-based Applications," in *Proc. of the Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Hiroshima, Japan, Sep. 25. 2013.
[24] Sung-Ho Yoon, Jun-Sang Park and Myung-Sup Kim. "Behavior Signature for Fine-grained Traffic Identification," *Applied Mathematics & Information Sciences*, vol. 9, no. 2L, pp. 523-534, 2015. Article (CrossRef Link)
[25] Q. Xu, Y. Liao, S. Miskovic, Z. M. Mao, M. Baldi, A. Nucci, T. Andrews, "Automatic generation of mobile app signatures from traffic observations," in *Proc. of Computer Communications (INFOCOM) 2015 IEEE Conference on.*, pp. 1481-1489, 2015. Article (CrossRef Link)
[26] Deep Mann, Shashank Gupta, Ankit Sharma and Shakil Akhtar, "Digital Signature using Biometrics," in *Proc. of Proceedings of the World Congress on Engineering and Computer Science 2015 Vol I WCECS 2015*, San Francisco, USA, October 21-23, 2015.
[27] Hyun-Min An, Su-Kang Lee, Jae-Hyun Ham and Myung-Sup Kim, "Traffic Identification Based on Applications using Statistical Signature Free from Abnormal TCP Behavior," J. Inf. Sci. Eng. Vol. 31, no. 5, pp.1669-1692, 2015. Article (CrossRef Link)
[28] Kyu-Seok Shim, Sung-Ho Yoon, Su-Kang Lee, Young-Joo Won and Myung-Sup Kim, "SigBox: Automatic Signature Generation Method for Fine-grained Traffic Identification," *Journal of Information Science and Engineering*, vol. 33, no. 2, 2017. Article (CrossRef Link)
[29] Vinoth George C and Vinodh Ewards, "A Survey on Signature Generation Methods for Network Traffic Classification," *International Journal of Advanced Research in Computer Science*, Vol 4, No.4, Mar-Apr 2013. Article (CrossRef Link)
[30] I. Butun, M. Erol-Kantarci, B. Kantarci, H. Song, "Cloud-centric multi-level authentication as a service for secure public safety device networks," *IEEE Commun. Mag.*, vol. 54, no. 4, pp. 47-53, Apr. 2016. Article (CrossRef Link)
[31] Xiao Liu, Yuxin Liu, Houbing Song, and Anfeng Liu, "Big Data Orchestration as a Service Networking," *IEEE Commun. Mag*., vol. 55, Issue 9, pp. 94 – 101, Sep. 2017. Article (CrossRef Link)
[32] Saeed Javanmardi, Mohammad Shojafar, Shahdad Shariatmadari and Sima S. Ahrabi, "FR trust: a fuzzy reputation-based model for trust management in semantic P2P grids", *International Journal of Grid and Utility Computing*, vol. 6 no. 1, p.57-66, December 2015. Article (CrossRef Link)
[33] G. Combs. Wireshark [Online], accessed on Jun. 2014. Available: Article (CrossRef Link),
[34] Microsoft. Microsoft Network Monitor, Jun. 24th, 2010. Article (CrossRef Link)

Baraka D. Sija (sijabarakajia25@korea.ac.kr) is an MS degree student in the Department of Computer and Information Science, Korea University, Korea. He received his BS degree in Information and Communication System from Semyung University, Korea, in 2016. His research interests include Internet traffic monitoring and analysis, network management, network services and Internet security.

Kyu-Seok Shim (kusuk007@korea.ac.kr) is a PhD degree student in the Department of Computer and Information Science, Korea University, Korea. He received his BS and MS degree in the Department of Computer Science and the Department of Computer and Information Science, Korea University, Korea, in 2014 and 2016, respectively. His research interests include Internet traffic classification and network management.

Myung-Sup Kim (tmskim@korea.ac.kr) is a professor in the Department of Computer and Information Science, Korea University, Korea. He received his BS, MS, and PhD degree in Computer Science and Engineering from POSTECH, Korea, in 1998, 2000, and 2004, respectively. From September 2004 to August 2006, he was a postdoctoral fellow in the Department of Electrical and Computer Engineering, University of Toronto, Canada. He joined Korea University in September 2006. His research interests include Internet traffic monitoring, Internet traffic analysis, network management, network services and Internet security.