Malware Detection using Computational Biology Tools

Ali Alatabbi, Moudhi Al-Jamea, and Costas S. Iliopoulos

***Abstract*—The Internet is considered to be as a rich platform of information where many people get benefit from its access but still they are being attacked by computer malwares and various other threats which distract their normal work flow to be carried out in an efficient manner. In this paper, we give an overview of the efficient read aligner software termed as REAL which is used for next generation sequencing. It reads structures as a tool to detect computer Malware. Using this tools a dynamic computer malware detection model has been presented in this paper that can detect the malwares to prevent attacks which might cause damaging or stealing sensitive information. This model is inspired by REAL which is an efficient read aligner for next generation sequencing for processing biological data. New anti-Malware technologies are introduced to the world by the clock, but at the same time new malware techniques have also emerged to misuse these technologies. Experimental results of this study shows that the proposed system is efficient and it is a novel way for detecting malware code embedded in different types of computer files, using bioinformatics tools with consistency and accuracy in detecting the malware and it was able to complete the assignment in high speed without excessive memory usages.**

***Index Terms*—Malware detection, pattern recognition, pattern matching, security.**

I. Introduction

Malware is a generic term used to describe all kinds of malicious software. Viruses, Worms, Spyware, Trojan horses are all examples of malicious software. It is created by attackers to not only cause major threat to the security and privacy of computer users and their sensitive information, but most of the time it is also responsible for a significant amount of financial loss. As the complexity of modern computing systems are growing, various bugs are unavoidable in software systems; this increases the possibility of the malware attack that usually exploits such vulnerabilities in order to damage the systems [1].

There are many approaches for malware detections which can be classified into two categories. First is the anomaly-based detection technique which uses its knowledge to monitor the program's behavior to decide the maliciousness of a program under inspection. Second technique which is considered as the most popular one is signature-based approach [2], which attempts to model the malicious behavior of malware and uses this model in the malware detection [3]. Both of the detection techniques can employ one of three different approaches: static, dynamic, or hybrid. Static approach describes the structure of the malicious code in the program that is under inspection before execution. Dynamic approach tries to detect the malicious code during or after the program execution. Hybrid approach is a combination of both previous approaches. (See Fig. 1)

Manuscript received December 30, 2012; revised March 12, 2013. This work was supported in part by the Department of informatics Kings College London.

The authors are with the Department of Informatics, King's College London, London WC2R 2LS, United Kingdom (e-mail: ali.alatabbi@kcl.ac.uk, mudhi.aljamea@kcl.ac.uk, csi@kcl.ac.uk).

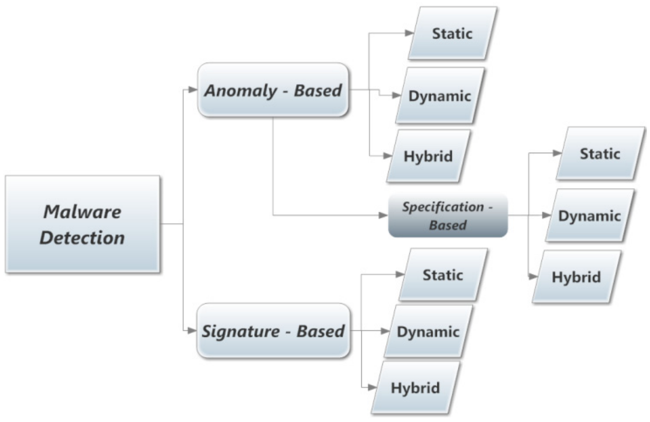


Fig. 1. A classification of malware detection techniques.

The rest of the paper is structured as follows. In Section II, the basic malware analysis and approaches are presented. In Section III, we briefly review some of the related work in the string matching malware detection approach. In Section IV, the basic definitions that are used throughout the paper are presented. We give an overview of REAL is Section V. In Section VI, we formally define the problem solved. Section VII, the experiments and results are discussed. Finally, we briefly conclude with some future proposals in Section VIII.

II. Background

Malware writers keep improving their obfuscation techniques to make the programs harder to understand and to evade the malware detectors. Encryption is one of the malware approaches that are used widely to evade signature-based detectors. In this approach, an encrypted malware is typically composed of the decryptor and encryptor.

The decryptor recovers the main body whenever the infected file is run. By using a different key for each infection, the malware makes the encrypted part unique, thus hiding its signature [4]. Yet, the main problem of the encryption is that the decryptor remains constant and in such case detector will be able to detect the malware based on the descriptor's code pattern.

However, malware writers always create and develop new techniques in writing malware script or code in order to make it hard to detect. They have reached a point where the virus can modify its code and appearance after each infection in order to avoid the detective and the generic scanning. One of the approaches called “Polymorphic Malware" is capable of changing its decryptor slightly, while avoiding the problem in the previous approach.

Another more advanced approach is the "Metamorphic Malware". It is considered as one of the best approaches in using the best obfuscation techniques. It basically evolves its body into new generations, which changes the total look of the malware while keeping the same functionality. It should be able to recognize, parse and mutate its own body whenever it propagates. It is important that the metamorphic malware never reveals its constant body in memory due to not using encryption or packing, thus making it so difficult for the anti-malware scanners to detect this malware [4].

Nevertheless, there are many obfuscation techniques that are specifically used by the malware writers in the polymorphic and metamorphic malware approaches for example (Dead-Code Insertion, Register Reassignment, Subroutine Reordering, Instruction Substitution, Code Transposition and Code Integration).

However, most of the malware writers use an old version of a malware to create a new one by reordering the malware instructions. The majority of malwares that appears today is a simple repacked version of old malware [5]. Even after changing or reordering the instructions of the malware they will still share some behaviors. Different obfuscated versions of the same malware have to share (at least) the malicious intent, namely the maliciousness of their semantics, even if they might express it through different syntactic forms. Therefore, addressing the malware detection problem from a semantic point of view can lead a more robust detection system [6] which will help in detecting them since the detectors are familiar with the old malware.

Executable packing: basically is the approach of using the executable packing technique which is popular nowadays among the malware writers to obfuscate malicious code and evade detection by signature-based anti-virus software. This later technique is the most common one. In general, it is believed that nearly 80% of malware are packed and 50% of existing malware are packed versions of old malware [7] and that is due to the accessible effortless open-source and commercial executable packers that help these writers to generate an encrypted version of their malware. Since it has been packed, the signature-based anti-malware will not detect the malicious code as it will not be able to match the signature with the packed malware. As soon as the malware is executed it will be decrypted and do the harm to the computer.

On the other hand, anti-malware providers try their best to follow up with the latest developments in order to be able to detect and remove these new malwares and overcome their threats. For example, there have been universal unpackers that can help in detecting and extracting encrypted code from packed executables, but these unpackers are expensive and time consuming as it might take hours or even days to scan large collections of executables looking for malware infections.

However, (R. Perdisci et al., 2008) [5] has devised a new approach by applying pattern recognition techniques for fast detection of packed executables. The objective behind fast detection is to efficiently and accurately distinguish between packed and non-packed executables, so that only executables detected as packed will be sent to a universal unpacker, thus saving a significant amount of processing time. (See Fig. 2).

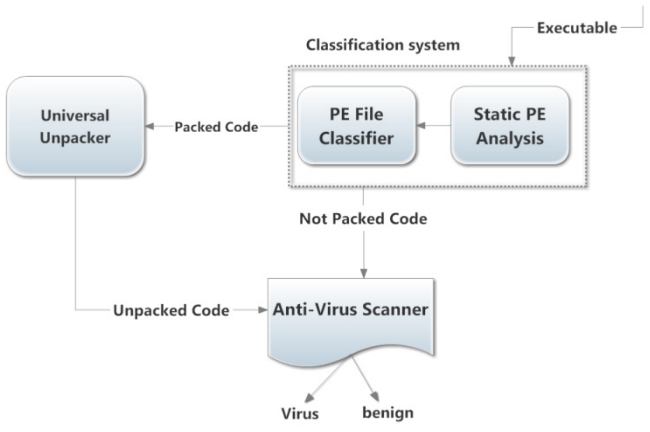


Fig. 2. Classification system produced by (R. Perdisci et al., 2008) to distinguish between packed and non-packed executables

This work of classification system extracts a number of features from executable files in PE format through static analysis, which means that they will be able to identify the packed executables without the need of running them. Yet, this technique will only improve the processing time of malware detecting since it will save time when it distinguish between packed and non-packed executables, and then rely on the unpacker and signature based anti-malware software for detecting malicious code.

Other approach of distinguishing packed from non-packed executables is based on raw binary data which was introduced by [7]. They only used the raw binary information to extract features that can effectively distinguish between packed and unpacked executables without the need of decoding the instructions of the executable. Their algorithm can quickly tell which samples are packed or encrypted [7] and according to that, these packed executables will be sent to the unpacker to unpack them.

Meanwhile, there are no proposed solutions or algorithms for detecting viruses and malwares in packed files without unpacking them first. Results prove that no algorithm can detect packed executables and computer viruses with absolute precision, detection may still be performed with high accuracy [5].

One of the most popular malware detection techniques is the pattern matching algorithm. There is a great demand for high speed and scalable pattern matching algorithms [8], specifically in the signature-based malware detection approach which we will adopt and discuss in this paper.

III. Related Work

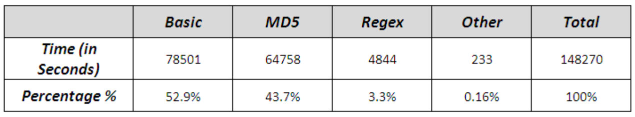
Nowadays the number of virus signatures and the network bandwidth are growing significantly and constantly, thus anti-malware vendors have to work very hard to develop solutions and algorithms that are able to deal with these growing threats. However, researchers have produced a number of solutions to deal with this problem specifically in the pattern matching technique. Thus many pattern matching algorithms have been proposed to solve the problem of intrusion detection system (IDS) [8].

The majority of these algorithms are "Shift based" which are fundamentally relying on the classic single pattern matching algorithm BM (Boyer-Moore algorithm). The core idea of BM is to utilize information from the pattern itself to quickly shift the text during searching to reduce number of compares as many as possible. BM introduces a bad character heuristic to effectively capture such information [9].

"Clam-AV" is one of the anti-virus pattern matching solutions which has been used widely in UNIX platforms lately [10]. It has been implemented in an extended version of BM (BMEXT) as a core pattern matching algorithm for scanning basic signatures along with other algorithms AC [11]. The down side of this algorithm is that its performance will decrease whenever the number of signatures increases.

Another anti-virus pattern matching solution known as (MRSI: A Fast Pattern Matching Algorithm for Anti-virus Applications) introduced in [8] is used to improve the previous solution, after analysing the different types of signatures (Basic, MD5, Regular Expression) and few other signatures types Table I.

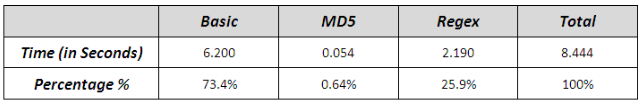
TABLE I: DIFFERENT TYPES OF SIGNATURES THAT HAS BEEN ANALYZED BY MRSI



	Basic	MD5	Regex	Other	Total
Time (in Seconds)	78501	64758	4844	233	148270
Percentage %	52.9%	43.7%	3.3%	0.16%	100%

And after studying the time processing for each signature type [8] decided to concentrate their work on matching the (basic) signatures since it is the most popular one and the most time consuming in order to improve the virus scanning speed on Clam-AV. (results in Table II) And for that they managed to achieve an 80%~100% faster virus scanning speed.

TABLE II: THE PROCESSING TIME FOR EACH SIGNATURE IN CLAM-AV



	Basic	MD5	Regex	Total
Time (in Seconds)	6.200	0.054	2.190	8.444
Percentage %	73.4%	0.64%	25.9%	100%

However, the similarities between malware detection and biological molecules sequencing provide the possibilities of using short reads alignment algorithm REAL in malware detection based on signatures.

Currently, human genome sequence mapping has been completed. Typical applications of bioinformatics are: searching one or a set of gene occurrence in a gene sequence, to compare similarity relationship; or matching unknown protein sequence according to known protein sample. As the protein and gene could be represented as sets of strings, traditional pattern matching technology could be used to solve such matching problems in the malware detection area [12].

IV. PRELIMINARIES

Let Σ be a finite alphabet which consists of a set of characters (or symbols). The cardinality of an alphabet, denoted by $|\Sigma|$. The set of all non-empty strings over the alphabet Σ is denoted by Σ^{+}. The empty string is the empty sequence (of zero length) and is denoted by ε ; we write $\Sigma^{*}=\Sigma^{+}\cup\varepsilon$. A string is a sequence of zero or more characters (or symbols) in an alphabet Σ.

A string x of length n is represented by $x[1\cdots n]$, where $x[i]\in\Sigma$ for $1\leq i\leq n$. The i -th symbol of a string x is denoted by $x[i]$. We denote by $x[i\cdots j]$ the substring of x that starts at position i and ends at position j . Then a string w is a substring of x if $x=uwv$, where $u,v\in\Sigma^{*}$. Conversely, x is called a super-string of w .

Edit Distance: [13] The distance $\delta_E(x,y)$ between two strings x and y is the minimal cost of a sequence of operations that transform x into y. The cost of a sequence of operations is the sum of the costs of the individual operations. The operations are a finite set of rules of the form $\delta_E(u,v)=n$, where u and v are different strings and n is a non-negative real number. Once the operation has converted a string u into v, no further operations can be done on v. The edit distance is symmetrical and, assuming unit costs, it holds $0\leq\delta_E(x,y)\leq max(|x|,|y|)$.

- Insertion: $\delta_E(\varepsilon,a)$, i.e. inserting the letter a .
- Deletion: $\delta_E(a,\varepsilon)$, i.e. deleting the letter a .
- Substitution or Replacement: $\delta_E(a,b)$ for $a\neq b$, i.e. substituting a by b .

Hamming Distance: Given two strings of equal length, the Hamming distance between them is the number of positions for which the corresponding symbols are different. In other words, the Hamming distance between two strings of equal length is the minimum number of symbol substitutions required to change one string into the other.

Hamming distance allows only substitutions, which cost 1. The Hamming distance is symmetric, and it is finite. In this case it holds $0\leq\delta(u,v)\leq|v|$ where $|u|=|v|$.

$$H_{Dist}(X,Y)=|I|, I=\{i\,|\,s_i\neq p_i, 1\leq i\leq n\}$$

where

$$|X|=|Y|=n$$

Alignment of two strings: An alignment between two strings $x,y\in\Sigma^{*}$ whose respective lengths are n and m , is a way to visualize their similarities, Formally an alignment A between x and y is a string z such that $(\Sigma\cup\varepsilon))\times(\Sigma\cup\varepsilon)\times(\varepsilon,\varepsilon)$. Given two sequences x and y such that $x=x_1\cdots x_n$, $y=y_1\cdots y_n$.

Formally a local alignment between x and y at position q with at most k - differences is $x_q\cdots x_n$, for $|x|=n\geq q$, can be transformed in to y by performing at most k of the edit operations.

V. REAL OVERVIEW

REAL (REad ALigner) is a new read aligner, which addresses the problem of efficiently mapping $p_1,p_2,\ldots p_r$ to t with at most k -mismatches. In order for the procedure to be efficient, we make use of word-level parallelism by transforming each factor of length ℓ of t into a *signature*.

In addition, the idea of using the pigeonhole principle to split each read into ν fragments is adopted. The general idea for the k -mismatches problem is that inside any match of the pattern of length m , with at most k errors, there must be at least $m-k$ letters belonging to the pattern [14]. By requiring $v-k$ of the fragments (instead of all of them) to be perfectly matched on t, the non-candidates can be filtered out very quickly.

REAL algorithm has been presented recently to the gene and DNA sequencing area and so far it performed very well with positive results. It is worthy of trying to use this pattern matching algorithm in signature pattern matching of anti-malware and malware detection area. However the performance could be evaluated and we might need to do some changes on the algorithm. Full details on how the algorithm work can be found in [15].

VI. Problem Definition

This work considers two areas, signature-based malware and bioinformatics sequencing pattern matching. Real algorithm could solve the problem described as:

We formally define the problem of mapping tens of millions of short sequences to a reference genome as follows:

Find whether the pattern $\rho_i = \rho_i[1...\ell]$, for all $1 \le i \le r$, with $\rho_i \in \sum$, $\sum = \{A,C,G,T\}$, occurs with at most k -mismatches in $t = t[1..n]$, with $t \in \Sigma^*$.

In particular, we are interested in reporting a pattern, for all $1 \le i \le r$, in a case that occurs with the least possible number of allowed mismatches, exactly once in t [15], [16].

Problem 1. Given a set of patterns $\{\rho_1, \rho_2, ..., \rho_r\}$ of length ℓ , with $\rho_i \in \Sigma^*$, Σ is a bounded alphabet, and an integer threshold $h > 0$, find whether ρ_i, for all $1 \le i \le r$, occurs in text t of length n and/or in text $\hat{t}$, where $t, \hat{t} \in \Sigma^*$ and $\delta_E(t, \hat{t}) \le h$.

VII. The Experiment

TABLE III: The Detection Process Results in The Experiment

File Type	Signature format	Dataset			
		Dataset Size	Number of mismatches	infected files	detected files
Portable Executable	MD5	50	0	30	30
	Body-based	100	1	65	65
	MD5	50	0	30	30
	Body-based	100	2	60	60
Mail File	MD5	50	0	30	30
	Body-based	100	1	65	65
	MD5	50	0	30	30
	Body-based	100	2	60	60
Graphics	MD5	50	0	30	30
	Body-based	100	1	65	65
	MD5	50	0	30	30
	Body-based	100	2	60	60
OLE2 component	MD5	50	0	30	30
	Body-based	100	1	65	65
	MD5	50	0	30	30
	Body-based	100	2	60	60
HTML	MD5	50	0	30	30
	Body-based	100	1	65	65
	MD5	50	0	30	30
	Body-based	100	2	60	60

We experimented with different types of files including Portable Executables (".exe" and ".dll"), email files, Graphics (".jpg" and ".gif"), OLE2 component (eg: VBA script), normalized Web files (HTML, PHP, Java Script) and normalized ASCII text file.

We generated two different types of signatures MD5 hash and body-based signature, either by using “Sigtool”, a tool for generating MD5 hash or body-based signature, also we developed a Signature generator component in on Microsoft C# programming language.

The following steps are for generating the signatures given an infected file:

Step 1: For body-based signature we started by loading the infected file content as byte array to the memory (for larger file we only read small segments of the file (2KB, 2048 Bytes), the selected segment could be taken arbitrary from any part of the file. For MD5 hash we added an extra step to the process by passing the byte array extracted from the file to the MD5 hash generator function. Note that different signatures can be created from different parts of the infected file body by selecting different offsets, (the beginning, middle or the end of the file) finding informative areas in the file body will improve the detection process.

Step 2: Convert the selected byte array to Hexadecimal signature and write the output to the virus signature library file.

For example to create a body-based signature for the file "program.exe" using the CALMAV signature tool.

```
root@localhost :/tmp/$sigtool  program .exe > test .hdb
```

To create MD5 hash signature use the "–md5" option of sigtool as follow:

```
root@localhost:/tmp/$sigtool −−hex−dump

dumpprogram.exe > test.hdb
```

The virus library contains list of the signatures stored one signature per line, as shown in Table IV

TABLE IV: The Structure of The Virus Library File

	Malware-name	Offset	Seg-length	Sig-type	Hex-Sig
0	90				
1	Trojan-428	0	2048	BBS	de 76 2f f0 ...
2	Worm.Mydoom.I	0	2048	BBS	de 76 cd 51 ...
3	HTML.Phish.B	0	2048	MD5	9 77 9d ef ...
	Win.spam.ts	0	2048	BBS	40 00 68 c4 ...
	...				
	...				
90	Heuristics.Safe	0	2048	BBS	40 00 68 c4 ...

The first line in the file contains the total number of signature in the library.

VIII. Conclusion

We have introduced a novel way for detecting malware code embedded in different types of computer files, using bioinformatics tools, namely REAL (short read aligner for next generation sequencing), which uses approximate string matching. One of the benefits of this approach is that REAL is implemented in such a way that it does not necessarily load the whole file in memory. Instead, it loads blocks of the file depending on the the physical memory of the individual machine. Concerning the storage used for indexing, no additional hard disk space is necessary for REAL, as it does not store an index of the file data. The presented experimental results are very promising, in terms of efficiency and sensitivity on the detection process.

As it is shown by the results in Table III , REAL showed a consistency and accuracy in the the detection process and it was able to complete the assignment much faster, despite not using a stored preprocessed index of the scanned files. REAL outperform classic pattern matching such as Knuth-Morris-Pratt and Boyer-Moore [15].

Our future work will focus on two parts. First of all, using some heuristic algorithm for optimizing segmentation and selection of the signature region, utilized very well (e.g. level 8 memories). Hence, additional algorithms will be designed to further optimize the memory cost.

Second, provide support for signature based container meta-data by allowing matching for signatures in files stored inside different container types such as compressed and encrypted files.

In Addition, Future work will focus on studying the capability of perm-term analysis instead of segmentation, experiments with different and larger malware collections, and a combination of this technique with machine learning analysis of malicious code.

REFERENCES

[1] Y. Zeng, F. Liu, X. Luo, and C. Yang, "Formal description and analysis of malware detection algorithm mom a," in *Proc. International Symposium on Computer Science and Computational Technology*, pp. 139-142, 2009.

[2] P. Szor, *The art of computer virus research and defense*, Addison-Wesley Professional, 2005.

[3] N. Idika and A. P. Mathur, "A survey of malware detection techniques," Purdue University, pp. 48, 2007.

[4] W. D. Doyle, "Magnetization reversal in films with biaxial anisotropy," in *Proc. 1987 INTERMAG Conf.*, 1987, pp. 1-6.

[5] I. You and K. Yim, "Malware obfuscation techniques: A brief survey," in *Proc. Int. Conf. on Broadband, Wireless Computing, Communication and Applications*, pp. 297-300, 2010.

[6] R. Perdisci, A. Lanzi, and W. Lee, "Classification of packed executables for accurate computer virus detection," *Pattern Recognition Letters*, vol. 29, no. 14, pp. 1941-1946, 2008.

[7] M. D. Preda, M. Christodorescu, S. Jha, and S. Debray, "A semantics-based approach to malware detection," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 30, no. 5, pp. 25, 2008.

[8] L. Nataraj, G. Jacob, and B. S. Manjunath, "Detecting packed executables based on raw binary data," Technical report, Jun 2010.

[9] X. Zhou, B. Xi, Y. Qi, and J. Li. Mrsi, "A fast pattern matching algorithm for anti-virus applications," in *Proc. Networking, Seventh International Conference of IEEE*, pp. 256-261, 2008.

[10] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM*, vol. 20, no. 10, pp. 762–772, 1977.

[11] T. Kojm. Clamav. (2004). URL. [Online]. Available: http://www. clamav. net

[12] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[13] M. Elloumi, P. Hayati, C. S. Iliopoulos, S. P. Pissis, and A. Shah, "Detection of fixed length web spambot using real (read aligner)," in *Proceedings of the CUBE International Information Technology Conference*, *ACM*, pp. 20-825, 2012.

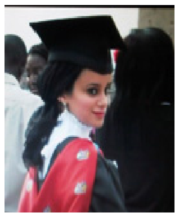
[14] D. Gusfield, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997, ch. 11.

[15] Appliedbiosystems URL. [Online]. Available: http://www.appliedbiosystems.com.

[16] K. Frousios, C. S. Iliopoulos, L. Mouchard, S. P. Pissis, and G. Tischler, "Real: An efficient read aligner for next generation sequencing reads," in *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, BCB '10, New York, NY, USA, pp. 154-159, 2010.

[17] P. Antoniou, J. W. Daykin, C. S. Iliopoulos, D. Kourie, L. Mouchard, and S. P. Pissis, "Mapping uniquely occurring short sequences derived from high throughput technologies to a reference genome," in *Proc. Information Technology and Applications in Biomedicine, 2009. ITAB 2009. 9th International Conference of* IEEE, pp. 1-4, 2009.

Ali Alatabbi got his B.Sc. in Electrical Engineering (Computer and Control), Department of Electrical Engineering, College of Engineering, University of Basra, Basra, Iraq-1994, M.Sc. E-Commerce Engineering, Westminster University, Harrow Computer School, London, 2003. He got his MCPD certificate(Microsoft Certified Professional Developer): Enterprise Applications Developer, London, 2007- 2008. He is currently a Ph.D. student in the Department of Informatics, King's College London, as a member of the Bioinformatics and Algorithm Design Group. His research focuses on Design and Analysis of String Algorithms, Algorithms for Molecular Sequences, Approximate string-matching algorithms, Computational Linguistics and Arabic language Morphology Analysis) and he have publications in these area.



Moudhi M. Al-Jamea was born on 5/5/1983 in Saudi Arabia. Got her BS in Management Information Systems from King Fisal University in Saudi Arabia (2006), M.Sc. in Information Technology with E-business from University Of Greenwich London (2010) , Ph.D., Candidate in the Department of informatics at Kings College, University of London (2015). Moudhi is a lecturer at al Dammam University sponsored by them to pursue her Ph.D. at Kings College. Also Moudhi is the President/CEO of Superior for Information Technology Services Inc., www.superior-sa.com. Moudi's research intrest is in the (Design and Analysis of String Algorithms, Algorithms for Molecular Sequences, Approximate string-matching algorithms and Information Security .

Costas S. Iliopoulos got his B.Sc. in Mathematics, University of Athens, Greece (1980), his M.Sc. in Computer Science by research, University of Warwick, Coventry, England (1981) and in 1983 he got his Ph.D. in Computer Science, University of Warwick, Coventry, England. He is a Professor of Algorithm Design at the Department of Computer Science at King's College London and Marie-Curie Research Fellow at the Computer Technology Institute of Universityof Patras. Also he is the Head of the Algorithm Design Group at King's college London and the Director of the M.Sc. Program. His research interest is in the Design and Analysis of String Algorithms, Algorithms for Molecular Sequences and Algorithm Engineering for Music and Biological sequences and he have so many publications in theses area. Prof. Iliopoulos is an Editor in Chief of the Journal of Discrete Algorithms , Editor of the Journal of Computer Mathematics and Combinatorial Computing and he is a Member of the Programme Committee of the (AICCSA '10, ISAAC '08 ,AWOCA '07, SPIRE '07.. etc) more information available at Prof. Iliopoulos page http://www.dcs.kcl.ac.uk/staff/csi/