Distributed negative selection algorithms for mobile cloud computing

Dinh Thi Thanh Uyen, Trinh Van Ha, Duong Thuy Huong, Vu Van Dien

***Abstract*- The rapid advances in mobile computing (MC) will become a strong trend in the development of information technology as well as business and industry. However, devices are facing many challenges in terms of hardware resources, communication and security. Significant limited resources impede the improvement of service quality. This paper proposes a technique for balancing the performance and security of MCs using the immunology algorithm.**

***Index Terms*- Mobile computing, security, immune algrithms.**

I. INTRODUCTION

Cloud computing was first introduced in 2007. It was noted as a profitable business option, reducing costs for developing and running mobile applications. It was considered as a new technology to experience a variety of low-cost mobile services and promise as a green IT solution for researchers [13].

Rapid advances in mobile computing (MC) [2] will become a powerful trend in the development of information technology as well as business and industry. However, devices are facing many challenges in terms of resources (battery life, storage and bandwidth) and communications (mobile and security) [6]. Significant limited resources impede the improvement of service quality. Cloud computing (CC) has been widely recognized as the next generation computer infrastructure. CC provides a number of benefits by allowing users to use infrastructure (servers, networks, and storage), platforms (intermediary services and operating systems), and software with low cost. The servieces are usually provided by cloud computing providers such as Google, Amazon, and Salesforce. In addition, CC allows users to use resources in an on-demand fashion. As a result, mobile applications can be quickly released and released with minimal management effort. With the explosion of CC applications and support for a wide range of user services, mobile phones (MCC) were introduced as an integration of cloud computing into the mobile environment. MCC brings new services and utilities to mobile users to take full advantage of cloud computing.

Dinh Thi Thanh Uyen, Thai Nguyen University of Agriculture and Forestry, Vietnam.

Trinh Van Ha, Thai Nguyen University of Information Technology and Communications, Thai Nguyen City, Vietnam.

Duong Thuy Huong, Thai Nguyen University of Information Technology and Communications, Thai Nguyen City, Vietnam.

Vu Van Dien, Thai Nguyen University of Information Technology and Communications, Thai Nguyen City, Vietnam.

Protecting user privacy and data/applications from attackers is a key to establishing and maintaining consumer trust in the mobile platform, especially at MCC.

Mobile devices such as cell phones, PDAs, and smartphones are exposed to many security threats such as malicious code (eg viruses, Trojans). In addition, mobile phones with integrated global positioning system (GPS) may cause privacy problems for subscribers. Two key issues are as follows:

1. Security for mobile applications: Installing and running security software such as Kaspersky, McAfee and AVG are antivirus programs on mobile devices that are simple ways to Detect security threats (eg viruses, malicious codes) on devices. With the advantages of GPS navigation devices, the number of mobile phone users using location based services (LBS) has increased. However, LBS faces a privacy issue when mobile phone users provide personal information like their current location. This problem becomes worse if the opponent knows important information of the user.

2. Data security on Clouds: Although both mobile phone users and application developers benefit from storing large amounts of data/applications on a cloud, they should be careful about deal with data/applications about their integrity, authentication, and electronic signatures.

With an increasing number of cloud services, the demand for access to data resources (eg images, files, and documents) increases. As a result, a method to deal with is, storing, managing, and accessing data resources in the cloud will become a significant challenge. However, processing data resources in the cloud is not an easy problem due to low bandwidth, mobility, and limitations of the resource capabilities of mobile devices while while the system must still ensure safety. This paper proposes a technique for balancing the performance and security of MCs using the immunology algorithm.

The rest of the paper is organized as follows. In the next section, we present the background of an immune algorithm and its abilities for distributed environment. Section 3 presents our experiments. Section 4 concludes the paper and discusses some possible future works.

II. NEGATIVE SELECTION ALGORITHM

In this paper, Artificial Immune System (AIS) [4], a multidisciplinary research area that combines the principles

of immunology and computation, is used for experiments on the proposed representation.

AIS is inspired by the observation of the behaviors and the interaction of normal component of biological systems - the self -and abnormal ones - the nonself. Negative Selection Algorithm (NSA) is a popular model of AIS mainly designed for one-class learning problems.

Given a collection of self patterns S, a typical NSA comprises of two phases: detector generation and detection [4]. In the detector generation phase (Fig. 1.a), the detector candidates are generated randomly and censored by matching them against given self samples taken from the set S. The candidates that match any element of S are eliminated and the rest are kept and stored in the set D. In the detection phase (Fig. 1.b), the collection of detectors are used to distinguish self (system components) from nonself (outlier like viruses, worms, etc.). If an incoming data instance matches any detector, it is claimed as nonself, and it is claimed as self otherwise.

From a machine learning perspective, negative selection is usually described as an anomaly detection technique. Since its introduction, NSA has been a source of inspiration for many computing applications, especially for intrusion detection [4], [9].

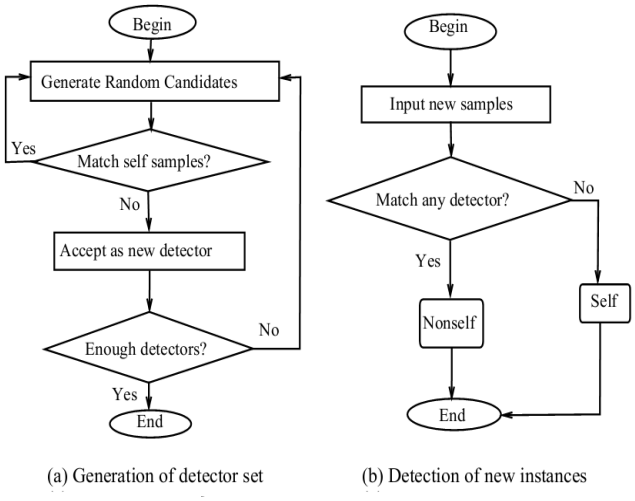

Fig. 1. Outline of a typical negative selection algorithm [4].

With respect to binary-based AIS using discrete detector set, to the best of our knowledge, the only algorithm for generating a perfect and discrete set of detectors was proposed by T. Stibor in [8] and by S. T. Wierzchoń in [10].

In the below figure, we illustrate the process for generating a detector set by using the number of five contiguous matches required for a match. The string to be protected is logically segmented into five equal-length "self" strings. To generate the detector, random strings are produced and matched against each of the self strings. The first two strings, 00000111 and 00000010, are eliminated because they both match self string 00000110 at at least five contiguous positions. The string 11101001 fails to match any string in the self at at least five contiguous positions, so it is accepted into the detector set.

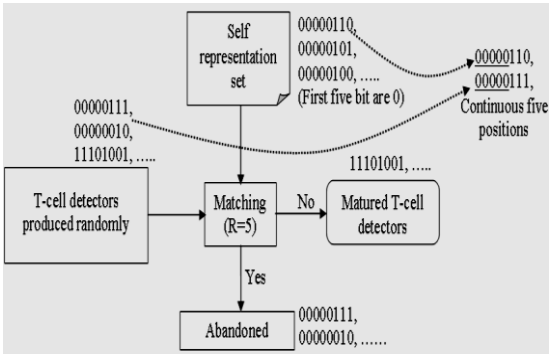

Figure 2. Illustration of generating Detector set

Each copy of the detection algorithm is unique: Most protection schemes need to protect multiple sites. In these environments, any single protection scheme is unlilely to be effective, because once a way is found to avoid detection at one site, then all sites are vulnerable. It is important to provide each protected location with a unique set of detectors. This implies that even if one site is compromised, other sites will remain protected.

Detection is probabilistic: One consequence of using different sets of detectors to protect each entity is that probabilistic detection methods are feasible. This is because an intrusion at one site is unlilely to be successful at multiple sites. By using probabilistic methods our system can achieve high system-wide reliability at relatively low cost (time and space). The price, of course, is a some-what higher chance of intrusion at any one site.

A robust system should detect (probabilistically) any foreign activity rather than looking for specific known patterns of intrusion. Most virus detection programs work by scanning for unique patterns (e.g., digital signatures) that are known at the time the detection software is distributed. This leaves systems vulnerable to attack by novel means. Like other change detectors, our algorithm learns what self is and notices (probabilis-tically) any deviation from self.

It is useful to know the probability P_M that random strings match at at least r contiguous locations. If m = the number of alphabet symbols, l = the number of symbols in a string (length of the string), and r = the number of contiguous matches required for a match, then we have [7]:

$$P_M = m^{-r} [(1- r)*(m - 1)/m +1]$$

Since detection is probabilistic, we need to make accurate estimates of these probabilities for different configurations of the change-detection system. Suppose that we have some string that we want to protect. This string could be an application program, some data, or any other element of a computer system that is stored in memmory. Using the NSA algorithm described above, we would like to estimate the number and size of detector strings that will be required to ensure that an arbitrary change to the protected string is detected with some fixed probability. We make the following definitions and calculations:

N_{Ro} = The number of initial detector strings (before censoring).

N_R = The number of strings after censoring (size of the repertoire).

N_S = The number of self strings.

P_M = The probability of a match between 2 random strings.

f = The probability of a random string not matching any of the N_S self strings = $(1 - P_M)^{N_S}$.

P_f = The probability that N_R detectors fail to detect an intrusion.

If P_M is small and N_S is large, then $f \approx e^{P_M N_S}$ and

$$N_R = N_{Ro}\, f,\ P_f = (1— P_M)N_R$$

If P_M is small and N_R is large, then $P_f \approx e^{P_M N_R}$.

Thus, $N_R = N_{Ro}\, f = -\ln(P_f)/P_M$

Then we get the following: $N_{Ro} = -\ln(P_f)/P_M/(P_M(1-P_M)^{N_S})$

This formula allows us to predict the number of initial strings (N_{Ro}) that will be required to detect a random change, as a function of the probability of detection (1 - P_f), the number of self strings being protected (N_S), and the matching rule (P_M). N_{Ro} is minimized by choosing a matching rule such that $P_M \approx 1/N_S$.

III. EXPERIMENTS

Table 1 illustrates the effect of varying r and l on P_M for different values of m. The first row shows the configuration we have used in most of our experiments. Setting r = 8 corresponds to a one-byte change. The first four rows of the table show the linear increase in P_M as the length of the string (l) increases. Rows two and five show the exponential decrease in P_M as r increases. Finally, the last eight rows show the dramatic effect on P_M of increasing the alphabet size.

m	r	l	P_M
2	8	16	0,019531250
2	8	32	0,050781250
2	8	64	0,113281250
2	8	128	0,238281250
2	16	32	0,000137329
2	16	64	0,000381470
2	16	128	0,000869751
4	16	256	0,000000042
4	8	32	0,000289917
4	8	64	0,000656128
4	8	128	0,001388550
4	16	32	0,000000003
4	16	64	0,000000009
4	16	128	0,000000020
4	16	256	0,000000042
8	16	32	0,000000000
8	16	64	0,000000000
8	16	128	0,000000000
8	16	256	0,000000000

Table 1: Example values of P_M for varying values of m (alphabet size), r (number of contiguous matches required for a match), and l (string length).

Based on the above analysis, we can make several observations about the algorithm:

1. It is tunable: we can choose a desired probability of detection (P_f), and then estimate the number of detector strings required as a function of the size of N_S (the strings to be protected). Since an increased probability of detection results in increased computational expense (due to the increased size of Ro and R), one can choose a desired probability of detection by determining (a) how fatal a single intrusion would be, and (b) how much redundancy exists in the system.

2. N_R is independent of N_S for fixed P_M and P_f. That is, the size of the detector set does not necessarily grow with the number of strings being protected. This implies that it is possible to protect very large data sets efficiently.

3. If N_R, P_f, and P_M are fixed, then N_{Ro} grows exponentially with N_S. This exponential factor is unfortunate in one respect, but it does imply that once a set of detectors has been produced (say, using a supercomputer) that it would be virtually impossible for a malicious agent to change self and then change the detector set so that the change was unnoticed. N_R, can be controlled by choosing $P_M = 1/N_S$.

4. Detection is symmetric: Changes to the detector set are detected by the same matching process that notices changes to self. This implies that when a change is detected there is no a priori way to decide if the change was to self or to the detectors. The advantage is that self confers the same protection to the detector set that the detector set provides to self.

An experimental prorogram illustrates the approach methods. The implemented algorithm takes a set of self S (incoming data), P_M, number of ditributed sites, l and r as inputs and popular performance metric True Positive, False Negative, False Negative, True Negative with number of attacks as outputs.

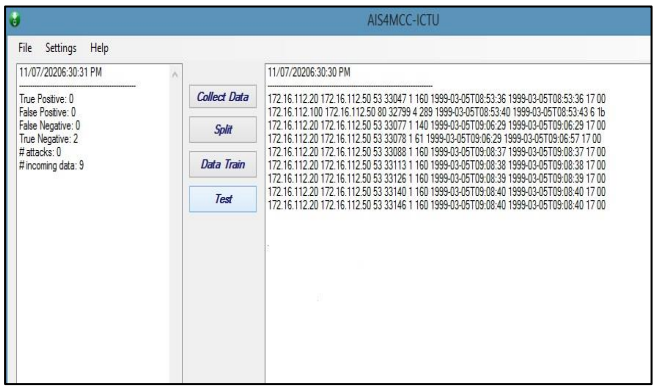

Figure 3. Illustration of detecting attacks from distributed sites

IV. CONCLUSIONS

Mobile cloud computing is one of the future mobile technology trends because it combines the advantages of mobile computing and cloud computing, thus providing optimal services to users.

Applications supported by mobile cloud including mobile commerce, mobile learning, and mobile healthcare have been more popular, showing the applicability of MCC into a variety of mobile services.

This article has provided an overview of AIS, especially NSA and its applications in MCC security. If the process of generating detectors is costly, it can be distributed to multiple sites because of its inherent parallel characteristics. In the future, we will develop this research for incorporating into real MCC.

ACKNOWLEDGEMENT

This research is based upon work supported in part by Information and Communication Technology University, Thai Nguyen University for research project "Mobile cloud technology and learning applications".

REFERENCES

[1] P. D'haeseleer, "An immunological approach to change detection: theoretical results," Proceedings 9th IEEE Computer Security Foundations Workshop, Kenmare, Ireland, 18-26, 1996.

[2] J. Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches. in Wireless Communications and Mobile Computing, Wiley, 2013.

[3] Haag, Charles R., Gary B. Lamont, Paul D. Williams, and Gilbert L. Peterson. "An artificial immune system-inspired multiobjective evolutionary algorithm with application to the detection of distributed computer network intrusions." In International Conference on Artificial Immune Systems, pp. 420-435. Springer, Berlin, Heidelberg, 2007.

[4] Z. Ji. Negative Selection Algorithms: from the Thymus to V-detector. PhD thesis, The University of Memphis, 2006.

[5] J. Kim, P. J. Bentley, U. Aickelin, J. Greensmith, G. Tedesco, and J. Twycross. Immune system approaches to intrusion detection - a Review. Natural Computing, 6:413–466, 2007.

[6] White Paper, "Mobile Cloud omputing Solution Brief," AEPONA, November, 2010.

[7] J. K. Percus, Percus, and A. S. Perelson. Predicting the size of the antibody combining region from consideration of efficient self/non-self dis-crimination. Proceedings of the National Academy of Science, 90:1691-1695, 1993.

[8] T. Stibor, K. M. Bayarou, and C. Eckert, "An investigation of R-chunk detector generation on higher alphabets," in Genetic and Evolutionary Computation Conference (GECCO), vol. 3102 of Lecture Notes in Computer Science, pp. 299–307, 2004.

[9] H. Yang, T. Li, X. Hu, F. Wang, Y. Zou, "A survey of artificial immune system based intrusion detection", The Scientific World Journal, 2014.

[10] S. T. Wierzchoń. Generating optimal repertoire of antibody strings in an artificial immune system. In IIS'2000 Symposium on Intelligent Information Systems, pp. 119–133, 2000.

[11] W. Zheng at el, A Rapid r-continuous Bits Matching Algorithm for Large-scale Immunocomputing, in Proceedings of the International Conference on Computer Science and Software Engineering, 431- 434, 2008.

[12]. D. Y. Yeung, Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models", Pattern Recognition, vol. 36, no. 1, pp. 229–243, 2003.

[13] http://www.mobilecloudcomputingforum.com/

BIOGRAPHY

Dinh Thi Thanh Uyen is an IT lecturer at Thai Nguyen University of Agriculture and Forestry. She finished her master course on Computer science at S.N.R Sons - India in 2011. She has taught a wide variety of courses for UG students and guided several projects.

Trinh Van Ha is a lecturer in Faculty of Information Technology - University of Information Technology and Communications, Thai Nguyen. He finished his master course on Computer science at Thai Nguyen University in 2008. He has taught a wide variety of courses for UG students and guided several projects. He has published several papers in national and international journals. His research interests are network architecture and computer security.

Duong Thuy Huong is a lecturer in Faculty of Information Technology - University of Information Technology and Communications, Thai Nguyen. She finished her master course on Computer science at Thai Nguyen University in 2014. She has taught a wide variety of courses for UG students and guided several projects.

Vu Van Dien is a lecturer in Faculty of Information Technology - University of Information Technology and Communications, Thai Nguyen. He finished his master course at Hanoi University of Technology in 2016. He has taught a wide variety of courses for UG students and leaded several projects. His research interests are network architecture and computer security.