Artificial Immune System For The Internet

THÈSE N^{O} 4079 (2008)

PRÉSENTÉE LE 8 AOÛT 2008

À LA FACULTE INFORMATIQUE ET COMMUNICATIONS
LABORATOIRE POUR LES COMMUNICATIONS INFORMATIQUES ET LEURS APPLICATIONS 2
SECTION DES SYSTÈMES DE COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Slaviša SARAFIJANOVIĆ

Electrical Engineer, University of Belgrade, Serbie
et de nationalité serbe

acceptée sur proposition du jury:

Prof. M. Hasler, président du jury
Prof. J.-Y. Le Boudec , directeur de thèse
Prof. A. Ijspeert, rapporteur
Prof. Ph. Oechslin, rapporteur
Prof. J. Timmis, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2008

To my 4 year old son Marko,
who has grown up along with this thesis,
and who asks me many ”pourquoi” everyday.

Abstract

We investigate the usability of the Artificial Immune Systems (AIS) approach for solving selected problems in computer networks. Artificial immune systems are created by using the concepts and algorithms inspired by the theory of how the Human Immune System (HIS) works. We consider two applications: detection of routing misbehavior in mobile ad hoc networks, and email spam filtering.

In mobile ad hoc networks the multi-hop connectivity is provided by the collaboration of independent nodes. The nodes follow a common protocol in order to build their routing tables and forward the packets of other nodes. As there is no central control, some nodes may defect to follow the common protocol, which would have a negative impact on the overall connectivity in the network.

We build an AIS for the detection of routing misbehavior by directly mapping the standard concepts and algorithms used for explaining how the HIS works. The implementation and evaluation in a simulator shows that the AIS mimics well most of the effects observed in the HIS, e.g. the faster secondary reaction to the already encountered misbehavior. However, its effectiveness and practical usability are very constrained, because some particularities of the problem cannot be accounted for by the approach, and because of the computational constrains (reported also in AIS literature) of the used negative selection algorithm.

For the spam filtering problem, we apply the AIS concepts and algorithms much more selectively and in a less standard way, and we obtain much better results. We build the AIS for antispam on top of a standard technique for digest-based collaborative email spam filtering. We notice un advantageous and underemphasized technological difference between AISs and the HIS, and we exploit this difference to incorporate the negative selection in an innovative and computationally efficient way. We also improve the representation of the email digests used by the standard collaborative spam filtering scheme. We show that this new representation and the negative selection, when used together, improve significantly the filtering performance of the standard scheme on top of which we build our AIS.

Our complete AIS for antispam integrates various innate and adaptive AIS mechanisms, including the mentioned specific use of the negative selection and the use of innate signalling mechanisms (PAMP and danger signals). In this way the AIS takes into account users' profiles, implicit or explicit feedback from the users, and the bulkiness of spam. We show by simulations that the overall AIS is very good both in detecting spam and in avoiding misdetection of good emails. Interestingly, both the innate and adaptive mechanisms prove to be crucial for achieving the good overall performance.

We develop and test (within a simulator) our AIS for collaborative spam filtering in the case of email communications. The solution however seems to be well applicable to other types of Internet communications: Internet telephony, chat/sms, forum, news, blog, or web. In all these cases, the aim is to allow the wanted communications (content) and prevent those unwanted from reaching the end users and occupying their time and communication resources. The filtering problems, faced or likely to be faced in the near future by these applications, have or are likely to have the settings similar to those that we have in the email case: need for openness to unknown senders (creators of content, initiators of the communication), bulkiness in receiving spam (many recipients are usually affected by the same spam content), tolerance of the system to a small damage (to small amounts of unfiltered spam), possibility to implicitly or explicitly and in a cheap way obtain a feedback from the recipients about the damage (about spam that they receive), need for strong tolerance to wanted (non-spam) content. Our experiments with the email spam filtering show that our AIS, i.e. the way how we build it, is well fitted to such problem settings.

Keywords

Internet, communication, email, security, network, ad hoc, routing; evolution, self, resources, protection, collaborative, filtering, detection, spam, misbehavior; artificial immune system (AIS), innate AIS, adaptive AIS, self-nonself, evolving self, danger signal theory, danger signal, PAMP signal, safe signal.

Résumé

Nous étudions l'utilisation de systèmes immunitaires artificiels (SIA) pour résoudre des problèmes choisis liés aux réseaux informatiques. Les systèmes immunitaires artificiels sont créés à partir des concepts et des algorithmes inspirés par la théorie du fonctionnement du système immunitaire humain (SIH). Nous avons considéré deux applications: premièrement la détection de mauvais comportements des relais pour le routage dans les réseaux mobiles ad hoc; deuxièmement, le filtrage de pourriel.

Dans les réseaux mobiles ad hoc, la connectivité des relais multiples est fournie par la collaboration des noeuds du réseau. Les noeuds suivent un protocole commun afin de construire leur table de routage et pour relayer les paquets des autres noeuds. Comme le protocole n'est pas contrôlé de manière centralisée, certains noeuds peuvent défaillir et ne plus suivre le protocole, ce qui peut avoir un comportement négatif sur la connectivité globale du réseau.

Nous construisons un SIA pour la détection de mauvais comportements des relais pour le routage. Nous appliquons directement les concepts et les algorithmes utilisés pour expliciter le fonctionnement du SIH. L'implémentation et l'évaluation dans un simulateur démontre que le SIA construit reproduit les effets observés dans un SIH, par exemple, la réaction secondaire plus rapide lors d'un mauvais comportement déjà observé. Par contre, l'efficacité du SIA et son utilisation en pratique sont limitées. Certains détails ne peuvent pas être pris en compte par notre approche et les besoins en calculs d'un algorithme de sélection négative sont très élevés.

Pour le problème de filtrage de pourriel, nous sélectionnons uniquement les concepts nécessaires d'un SIA et les utilisons de manière moins standard. Cette approche nous permet d'obtenir de bien meilleurs résultats. La construction de notre SIA contre le pourriel est basée sur une technique usuelle de filtrage de pourriel collaborative utilisant des digests. Mais, nous remarquons une différence technologique sous-exploitée entre les SIA et le SIH. Nous exploitons alors cette différence de manière innovante afin de réduire les ressources nécessaires en calcul. Nous améliorons également la représentation habituelle des digests utilisés dans les filtres collaboratifs de pourriel. Nous montrons que cette nouvelle représentation, utilisée conjointement avec un mécanisme de sélection négative améliore grandement les performances de filtrage du système usuel sur lequel nous avons bâti notre SIA.

Notre SIA contre le pourriel intègre plusieurs mécanismes innés et adaptifs, en particulier la sélection négative et la signalisation native ("PAMP" et signaux de danger). De cette manière, le SIA peut prendre en compte les profils des utilisateurs, leur feedback implicite ou explicite, ainsi que la *bulkiness* du pourriel. Nous montrons par simulations que le SIA dans son ensemble est extrêmement efficace pour détecter le pourriel et éviter les faux-positifs. Nos résultats montrent également que les mécanismes innés et adaptifs sont cruciaux pour l'obtention d'une excellente performance.

Nous avons développé et testé dans un simulateur notre SIA pour le filtrage collaboratif du pourriel. Mais notre solution est vraisemblablement applicable pour d'autres moyens de communication sur Internet: La téléphonie IP, le chat, les SMS, les courriels, les forums de discussions, les blogs. Pour toutes ces applications, le but est d'authoriser les communications voulues entre utilisateurs et à la fois d'empêcher les communications non-désirées d'atteindre ces mêmes utilisateurs. Le problème de filtrage a ou aura probablement un cadre identique au filtrage du courriel: le besoin d'ouverture dans le cas d'émetteurs inconnus, la *bulkiness* pour la réception de spam (plusieurs destinataires reçoivent un spam similaire), la tolérance du système en présence d'une faible quantité de spam non filtré, la possibilité implicite ou explicite d'obtenir de la part du destinataire un feedback sur le spam, et une tolérance élevée pour le contenu valide. Les expériences avec notre système de filtrage de pourriel montrent que notre SIA, en particulier la manière dont nous l'avons construit, est tout à fait appropriée dans ce cadre.

Mots clés

Internet, communication, sécurité, émail, courriel, réseau, ad hoc, évolution, *self*, authentification, identité, clef privée, clef publique, ressources, protection, collaboration, filtrage, détection, pourriel, spam, comportement inapproprié; système immunitaire artificiel (SIA), SIA inné, SIA adaptif, *self-nonself*, évolution de *self*, théorie des modèles de danger, signal de danger, *PAMP* signal, signal de sûreté.

Acknowledgments

I would like to thank my advisor, Professor Jean-Yves Le Boudec, for finding and assigning to me such an interesting and challenging topic as the application of the AIS approach for protecting computer networks shown to be. My work with him on my PhD and on my teaching-assistant duties were great opportunities. It was a pleasure for me to learn from his experience and his way of thinking.

I would also like to thank the members of my PhD jury for their helpful advice on improving the final manuscript of the thesis. The remarks from Professor Jon Timmis, a specialist in the AIS field, were especially useful in this regard.

Many thanks to the people from the LCA laboratory, for their friendship, support, always interesting discussions, playing sports together, etc. Additional special thanks go to Marc-André Lüthi for his help regarding our implementation of the AntispamLab, and to Holly B. Cogliati and Ruben Merz for helping me with the final manuscript of the thesis.

Thanks to my parents and my brother for their impact on me to, from my early life, develop optimism and trust in myself and other people, and concentrate on the good opportunities in the world around me.

Finally, but not least important, I thank my wife Nataša for her love and support during all the years we have been together, and our son Marko for giving us extra delight and motivation.

Contents

Chapter 1

Introduction

1.1 Motivation

The main goal of this thesis is to investigate the usability of the Artificial Immune System (AIS) approach for solving selected problems in computer networks. Artificial immune systems are created by using the concepts and algorithms inspired by the theory of how the human immune system (HIS) works. The motivation for applying the AIS approach comes from the following two facts.

The first fact is the ability of the HIS to recognize various body invaders, such as viruses, bacteria and other pathogens, and to clear them from the body. It creates the immune cells that are able to recognize and react against the pathogens, but that are self-tolerant to the body at the same time. While doing that, the HIS shows some desirable properties, such as: self-organization (absence of a centralized control), an ability to learn and protect against new pathogens, a memory and fast response to previously encountered pathogens, an ability to incorporate the feedback from the protected system, adaptation to changes in the protected system.

The second fact is that in computer networks we can recognize many problems resembling those faced by the HIS. Therefore, we are interested in the solutions to these problems, which exploit the analogy to the HIS and possibly inherit the properties exhibited by the HIS. We consider two applications: detection of routing misbehavior in mobile ad hoc networks, and email spam filtering.

1.2 AIS Approach and Our Design Principles

We provide a detailed background and state-of-the-art information about the common AIS approach in Chapter 3. In summary, the AIS approach is typically used as follows. The items or events to be detected (or classified) are represented in a multi-dimensional data space. The AIS algorithms are then used to build artificial antibodies that are (usually) also represented as points in the data representation space and that are able to recognize the items or events of a given class by similarity matching.

The antibodies are usually generated randomly and trained on the examples of ”good” data (also known as *self*) that should not be detected by antibodies - by deleting those antibodies that match any of the self examples. The process of eliminating self-reactive antibodies is known as a negative selection algorithm. The remaining antibodies are thus potentially able to detect any anomalous data that is not similar to the self examples (also known as *non-self*). The use of similarity in matching is very important for the generalization ability of the system (i.e. learning behind simple memorizing of examples) - the system should be self-tolerant (avoid detection) not only to presented self examples, but also to the data similar to the presented self examples.

New antibodies are often created by multiplication and mutation of the antibodies that have already proved to be useful in detection (clonal selection algorithm). The antibodies produced by negative and/or clonal selection can be used directly for detection, in which case the AIS does anomaly detection.

Additional controls may however be used in order to activate antibodies. These additional controls are called innate signaling mechanisms. When there is damage to the protected system (or when interesting items or events are verified, in another class of applications), so-called danger signal is generated that activates those antibodies that recognize the cause of the damage (the verified interesting items/events). A similar activation of the antibodies happens when predefined patterns are recognized in the monitored items or events (so called ”PAMP” signal). On the contrary, recognition of some predefined ”safe” patterns may be used for deletion of (self-reactive) antibodies (so called ”safe” signal). If these additional controls are used the AIS is able to do more specific detection (classification) then simple anomaly detection. An AIS with the innate signalling is expected to better adapt to the changing non-self in the system, and to be able to tolerate self in the system that is not predefined by self examples.

The antibodies that prove to be especially useful in detection may be assigned a longer lifetime, i.e. become ”memory” antibodies, ready to recognize repeated similar damaging (or interesting) items or behavior.

The AISs are built with many variations in using of the above explained mechanism. Different design principles, i.e. different selection, modification and use of the above listed mechanisms, are usually dictated by the problem being solved. The previous findings by the AIS community that clarify the properties of the various AIS mechanisms are also an important factor for determining the design principles for a concrete application. Interesting examples from the AIS literature about the various use of the above listed AIS mechanisms are overviewed in Chapter 3, along with their demonstrated properties.

Here we list the design principles that we follow when designing the AISs for the two problems that we consider.

AIS design principles used for routing misbehavior detection. Routing misbehavior detection was the first problem that we tried to solve by using the AIS approach. At that time, many properties and constraints of the AISs were not well known or were not emphasized enough in the related literature. Therefore we build the AIS by directly applying the above described mechanisms. We adapt an existing simple AIS data representation format to represent the routing protocol events that can be observed for the nodes in the network. We create antibodies by generating them randomly and then applying the negative selection. We also add clonal selection and the use of an automated danger signal. We experiment with with the negative selection by modifying it to use dynamically updated self examples, with the goal to make it more adaptive to the changes in normal profile of routing behavior. For this we introduce the concept of a *virtual thymus* within the AIS. Our ultimate goal is to build an AIS effective in detecting many types of misbehavior that do not need to be imagined and specified in advance.

AIS design principles used for collaborative spam detection. Although we managed to make a working AIS that detects well some types of the routing misbehavior, we realized that some particularities of the problem (and therefore many types of misbehavior) cannot be easily accounted for by the approach.

Looking for a real computer networking problem that could be well fitted to the AIS approach, we found the collaborative filtering of email spam to be a promising candidate. There already existed a classic (not AIS-based) collaborative spam detection scheme used in real-life antispam solutions and based on a similarity-preserving data representation, the kind of the representation suitable for use within AISs. Additionally, the emailing system is, similarly to the human system, able to accept small damage (small amount of unfiltered spam) and to provide a feedback about this damage (such feedback is already used by other than AIS-based spam filtering), thus being very suitable for use of the analogy to the danger signal in the HIS. A strong analogy is also possible between the PAMP signal in the HIS and the bulkiness of spam (both being patterns strongly related to the items that should be detected by antibodies). Classic signatures on the emails (though not widely used) and authentication of the emails based on these signatures, as well as replying to the received emails, can be seen as an equivalent to the safe signals assigned to the received emails. Finally, the dynamically updated examples of self (good emails) are suitable to be obtained from the outgoing and replied emails.

Our goal is to exploit all the above listed analogies in order to see if we can make an efficient AIS for collaborative spam filtering. This implies the job of integrating both the adaptive and innate AIS mechanisms, which has not been done by many AIS practitioners, with a few exceptions that are summarized in Section 3.4.6. Our goal is also to evaluate the effect of both the adaptive and innate mechanisms on the overall AIS performance.

Based on the warnings from the related AIS literature (see e.g. Frieitas and Timmis [23], recently update to [24]), and on our experience with the AIS for routing misbehavior detection, we build the AIS for antispam by paying much more attention to the data representation and by using the AIS algorithms much more selectively and in a less standard way (as compared the routing misbehavior case): (1) We analyze and try to refine the existing representation used by a classic collaborative detection scheme on top of which we build our AIS, in order to make it less vulnerable to the obfuscation by spammer and more suitable for the use within the AIS that we build; (2) We exploit a technological difference between AISs and the HIS and avoid the creation of antibodies by random generation and negative selection (which is usually inefficient, especially in cases of a high-dimensional data representation as is the one that we use); instead, we create the candidate antibodies from the observed data (emails) and apply the negative selection to these antibodies. It might be that this way of creating antibodies is not used in the HIS simply for evolutionary and technological reasons, though in principle it would be possible; (3) We do not use clonal selection; the (above mentioned) way we create antibodies should implicitly provide the first function of the clonal selection - an adaptive and efficient search in the data representation space where the antigens are being observed (as opposed to producing many non-useful new antibodies by generating them randomly); the second function of the clonal selection in the HIS is to produce *many* similar antibodies, because many copies of the pathogen get spatially distributed in the body (or a part of the body) and a close antibody-to-antigen contact is needed for the recognition, and also because the antibodies are spent in the reactions against the pathogens; in the AISs we usually do not have the spatial component and the antibodies are reusable.

1.3 Outline of the Thesis

The thesis is organized as follows.

In Chapter 2 we give a high-level overview of how the HIS works. The overview is based on the models that look at the HIS from an information-processing perspective.

In Chapter 3 we overview the standard AIS building blocks. We also provide the state of the art on how these blocks have been used within some representative AISs, and how they impact the properties of those AISs. We review the related work specific to the considered applications in the chapters related to that applications: the literature related to collaborative or AIS-based spam filtering in Chapter 4, and the literature related to the routing misbehavior detection in the Chapter 7.

In Chapter 4 we show how we build an AIS for collaborative spam bulk detection by following the design principles announced in the Introduction. We build the AIS for antispam on top of a well-known classic scheme for collaborative detection of spam bulkiness that uses similarity digests. For the initial assessment of the abilities of the system, we implement it in a simulator and provide a basic-evaluation results of the overall system and of its innate and adaptive parts when used separately, all under one chosen spammer model (spam obfuscation).

In Chapter 5 we investigate the representation of email content used by the AIS from Chapter 4. The representation used is a modified version of the representation used by the referent classic scheme. In particular, we investigate and compare the resistance of the two representations to different amounts of obfuscation under a chosen spammer model. As an example of a realistic spammer model, we use the addition of random text.

In Chapter 6 we qualitatively and quantitatively investigate the impact of the negative selection itself on the digest-based collaborative spam filtering. For this, we compare the cases when the negative selection is added to the referent classic collaborative detection scheme and when it is not. The obtained conclusions should thus be applicable to any collaborative digest-based spam filtering system that would incorporate the negative selection, including the AIS that we build in Chapter 4.

In Chapter 7 we show how we apply the negative selection AIS algorithm to detect routing misbehavior in mobile ad hoc networks. Ad hoc networks are those in which nodes collaboratively build their local routing tables and use these tables to forward packets of other nodes, thus providing the multi-hop communication in the network. We consider the case of a mobile but stationary ad hoc network. We also investigate the effect of adding to the negative selection one more AIS algorithm, namely clonal selection. We evaluate the system within a simulator.

In Chapter 8 we investigate the use of additional (as compared to the system from the Chapter 7) AIS algorithms, that should allow the AIS for routing misbehavior detection to adapt to the non-stationary changes in the protected system. In particular we include the use of the danger signal and virtual thymus. Again, we evaluate the system within a simulator.

Chapter 9 concludes the work presented in the thesis. We summarize the most important findings and reevaluate our design principles in the light of the obtained results (findings).

Chapter 2

Human Immune System: Background

This chapter aims at providing a non-familiar reader with a basic understanding of how the HIS works. Knowing and understanding the main HIS mechanisms and principles should make the understanding of their artificial counterparts, that we build in the rest of the thesis, much easier and more intuitive. Another option is to temporarily skip this chapter, as we refer to its relevant parts when making analogies to the HIS in the rest of the thesis.

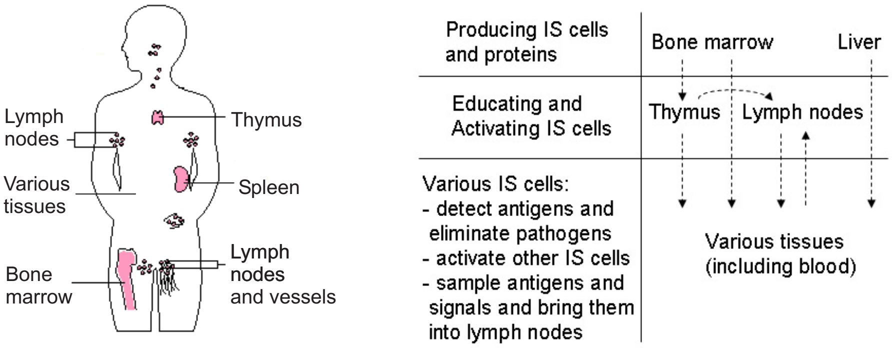
We give a brief and high-level algorithmic overview of how the human immune system (HIS) works, based mainly on the explanations found in [49] and in-there referred papers that cover well the self-nonself, infectious non-self and danger theories of the HIS. These are mutually non-excluding and complementary theories that represent the state of the art in modelling and explaining in a relatively simple way the main logic behind the complex details of the workings of the HIS.

2.1 HIS Organs, Cells and Working Principles

2.1.1 HIS Organs and Cells

The main function of the HIS is to protect the body against different types of pathogens, such as viruses, bacteria and parasites, and to clear it from debris. The core part of the HIS consists of a large number of different innate (evolutionary predetermined) and acquired (developed in adaptation to experienced infections or vaccinations) immune cells, commonly called lymphocytes. In order to provide detection and elimination of the pathogens and of the damaged, mutated or dead body cells, and to clean the body from debris, lymphocytes interact with each other and with other molecules, proteins, cells and microorganisms found in the body or belonging to the body itself.

The lymphocytes are found in the lymphatic system, which consists of IS organs such as bone marrow, the thymus and lymph nodes, and of the lymphatic vessels that connect these organs (Figure 2.1(a)). From the lymphatic system the lymphocytes move into other body subsystems (blood tissue and tissues of other body organs and systems), and possibly return back, while accomplishing their functions (Figure 2.1(b)).



Producing IS cells and proteins	Bone marrow	Liver
Educating and Activating IS cells	Thymus	Lymph nodes
Various IS cells: - detect antigens and eliminate pathogens - activate other IS cells - sample antigens and signals and bring them into lymph nodes	Various tissues (including blood)	

(a) (b)

Figure 2.1: **HIS organs, production and education of HIS cells (lymphocytes), detection of pathogens:** (a) The figure on the left shows in color the organs of the IS where majority of the lymphocytes are produced and educated; (b) The table on the right summarizes the functions that various IS organs have within a lymphocyte life cycle.

To accomplish their job properly the lymphocytes need to be created to act on pathogens and other unwanted material, but at the same time to not act against the body cells and molecules needed for normal functioning of the body.

Most of the lymphocytes are produced in the bone marrow (complement proteins, that are also part of the IS, are produced by liver). Certain types of the produced lymphocytes are already able for detection and elimination of the pathogens and go to the various tissues where they perform these actions. Other lymphocytes need to be additionally educated before they become active, and only then become able to do detection of pathogens or to co-stimulate other lymphocytes to do so.

Processes related to the education of lymphocytes happen in the thymus, lymph nodes and the tissues. Some lymphocytes are able to sample antigens (parts of pathogens) in the tissues and, based on various signals sensed from the environment (damage, infection, normal cell death), present the sampled antigens in various context inside lymph nodes to cause activation or suppression of other lymphocytes (part of the education process). Another part of the education process happens in the thymus, where majority of self-reactive lymphocytes are eliminated before even reaching lymph nodes and various tissues.

2.1.2 ”Information Processing” by Antigen Recognition and Signals Exchange

The ability of the IS cells to recognize antigens (molecular and protein patterns belonging to the pathogens or normal cells), and their ability to receive various signals from the environment and from other IS cells are in the core of the processes of proper development of IS cells and detection of pathogens. The antigens are recognized and the signals are received by means of so-called receptors.

Antigen receptors are proteins on the surface of IS cells that are capable of chemically binding to antigens. Whether chemical binding takes place between an antigen receptor and an antigen depends on the complementarity of their three-dimensional chemical structures. If it does, the antigen receptor and the antigen are said to be “cognate”. Because this complementarity does not have to be exact, an antigen receptor may have several different cognate antigens (generalization property). The binding between the antigen receptors and the antigens is a crucial step in mounting the immune reaction against the pathogens. What exactly happens after binding depends on additional control signals exchanged between different IS cells.

Similar like antigen receptors, signal receptors of an IS cell are also able to chemically bind to molecules secreted by or found on other IS or body cells. However, this binding involves recognition of special molecules (or proteins) that are interpreted by the IS cell as well known instructions for further actions to be performed by that cell. Some IS cells are able not only to receive but also to synthesize such signals, in response the observed antigens or other signals and to deliver them to other IS cells (or even to body cells), whether disposing them on its surface or secreting them as soluble molecules (or proteins).

Recognition of antigens and signals usually changes the current state and activity of the IS cells, and that is how the ”information processing” is done, and this is true for the cells of both the innate and adaptive IS parts (functional division of the IS is defined in Section 2.2). However, while the recognition ability of the innate IS cells is fixed, and therefore constrained to a predetermined set of pathogen patters this part of the IS can recognize, the production and development of the adaptive IS cells is controlled in such a way that their recognition abilities adapt for detection of also previously not seen pathogen patterns.

While some antigen and signal recognitions cause further instructions (signals) for development of IS cells, other are executive and cause destruction and elimination of pathogens (sometimes including destruction and elimination of infected or damaged body cells).

2.2 Functional Architecture of the IS

The first line of defense of the body consists of physical barriers: skin and the mucous membranes of the digestive, respiratory and reproductive tracts. They prevent the pathogen from easily entering the body.

The innate immune system is the second line of defense. It protects the body against common bacteria, worms and some viruses, and clears it from debris. It also interacts with the adaptive immune system, signaling the infections or the presence of damage in self cells and activating the adaptive IS.

The adaptive immune system learns about invaders and tunes its detection mechanisms in order to provide an effective response even against previously unknown pathogens. It provides an effective protection against viruses even after they enter the body cells. It adapts to newly encountered viruses and memorizes them for more efficient and fast detection in the future.

2.3 The Innate Immune System

The innate immune system consists of macrophages cells, complement proteins, natural killer cells, and dendritic cells. As mentioned in the previous section (Section 2.2), the innate immune system accomplishes two functions: detect the pathogens and clear them from the body, and control and activate adaptive HIS cells to do so.

2.3.1 Detection and Removal of Pathogens

The most known innate IS cells that directly participate in detecting and removing the pathogens are macrophages and complement proteins. Macrophages are big cells that are attracted by bacteria and engulf it in the process called "phagocytosis". Complement proteins can also destroy some common bacteria.

Reactions by the innate cells to pathogens are activated by chemical recognition of pathogen specific patterns by the innate cells. As mentioned previously, antigens recognition ability of the innate IS cells is predetermined and doesn't change over the life time of an organism (it is determined by evolutionary mechanisms).

The reactions are also controlled by the presence of protective proteins (decay acceleration factor (DAF) proteins) that are found on the surface of some body's cells, but not on the surface of the cells of distant organisms (including pathogens). DAF proteins protect body's cells against those complement system cells that recognize and attack a wide range of protein and molecular patterns, and therefore can attach to the body's cells as well. However the presence of self-protective DAF proteins prevents the reaction that would follow the attaching event.

2.3.2 Controlling the Adaptive IS

Both macrophages and complement proteins send signals to other immune cells when there is an attack. The complement proteins do so by secreting molecules that attract macrophages. Macrophages are able to re-stimulate T cells (important cells of the adaptive IS part) at at the sites of infection, by acting as activated antigen presenting cells (APCs).

Professional APCs, able to initially co-stimulate (activate) T cells found in lymph nodes are called dendritic cells (DCs). The way how DCs get activated and able to co-stimulate (activate) the cells of the adaptive immune system is discussed more in detail in Section 2.4.

2.4 Interaction Between the Innate and Adaptive IS Parts

As mentioned in Section 2.3, in addition to reacting to the pathogens itself, the innate HIS part also controls workings of the adaptive part.

The two mostly accepted (and not mutually excluding) theories of how the innate IS part controls the working of the adaptive IS part are the previously mentioned ”infectious non-self” and ”danger signal” models and theories of the HIS.

Before these two theories appeared, the understanding of what the main function of the HIS is and how this function is achieved was based on ”self-nonself” theory. According to this theory the innate and adaptive parts work more or less independently from each other.

According to the two new theories that are built on top of the main self-nonself theory mechanisms, the IS protects the body against infectious non-self and against entities that cause damage to the protected organism. And the innate IS controls the adaptive part to accomplish this job correctly.

We constrain our explanation of the interaction between the innate and adaptive IS parts to the signals from the innate IS cells that activate T cells of the adaptive IS. This signalling seems to be one of crucial components for proper work (and understanding) of the overall IS. However, it should be mentioned that there are various other (also important) signals that are exchanged between the innate and adaptive parts in both directions that we omit in this overview in order to simplify the otherwise very complex overall picture of the HIS.

2.4.1 Self-Nonself Theory

According to this theory, as mentioned previously, the innate and adaptive parts work more or less independently from each other. The innate IS cells are evolutionary predetermined to react to some pathogens and be tolerant to body’s self cells, and this reaction is very fast.

Unlike the innate IS cells, the adaptive IS cells can be produced that react against new, previously unseen pathogens, but this reaction takes some time to develop (reaction to previously experienced pathogens is somewhat faster due to the immune memory). The adaptive IS cells are educated to be tolerant to body’s self cells; according to the self-nonself theory this is accomplished only (in principle) by the negative selection in the thymus (explained in Section 2.5), which produces self-tolerant T cells. Adaptation to the pathogen then happens through the process of clonal selection of B cells (explained in Section 2.5), and is controlled by the T cells. Some T cells also directly react against pathogens.

This model failed to explain many immunological evidence, and has been modified a few times. In fact, its initial version did not include control of B cells by T cells, and the inclusion of this control was the first modification. The second modification actually included some interaction between the innate and adaptive part, requiring the activation of T cells by recognition of the antigens presented by innate IS cells called antigen presenting cells (APCs). However, as the antigen presenting was modelled as non-specific (APCs would present any antigens they find around), the modification did not provide any additional control mechanisms that could explain *what* the IS reacts against and what it is self-tolerant to.

2.4.2 Infectious-Nonself Theory

According to the infectious non-self theory, dendritic cells (DCs) of the innate IS are equipped with pattern recognition receptors (PRRs) that are able to chemically match molecular patterns found on most of the pathogens (called PAMPs - pathogen associated molecular patterns) but not found on body cells [51]. DCs are found resting (being non-active) in various tissues. The matching between PRRs of a DC and PAMPs of a pathogen (*infectious signal*) will activate the DC to collect surrounding antigens (including those of the pathogen) and move to the lymph nodes (see bottom part of Figure 2.2).

In a lymph node the DC displays (by use of so-called MHC molecules) the collected antigens on its surface making them available for recognition by non-activated (naive) T cells. Additionally, it also express co-stimulatory molecules on its surface that are responsible for activating T cells. Central point of the infectious nonself theory is that only PAMP-activated DCs (those that recognized infectious nonself patterns) express this type of co-stimulation signals. A T cell that matches the antigens presented by the PAMP-activated DC (this corresponds to signal 1t in Figure 2.2) will also receive co-stimulation signal (signal 2t in Figure 2.2) and become active T cell. Activated T cell participates in detection and elimination of the pathogens having cognate antigens or co-stimulates other adaptive IS cells to do so.

DCs that are not activated by PAMP recognition can also collect antigens (e.g. antigens from body cells that died normally) and present them in lymph nodes. The absence of the infectious signal is ”understood” by the DCs as a safe context. Therefore in this case the co-stimulatory molecules expressed by the DCs when they present the collected antigens are absent (absence of signal 2t) or different (which can also be considered as absence of signal 2t), which causes the suppression of T cells that match these antigens. In this way the innate system induces both the activation of the adaptive response to infectious non-self and the so-called peripheral tolerization (suppression) of T cells.

It is important however to notice that the overall tolerization results from both this peripheral tolerization and from the central tolerization of T cells that happens, as explained in Section 2.5.3, in the thymus. Indeed, it may happen that a PAMP-activated APC also picks up and presents some self antigens, but as the T cells are negatively selected in the thymus they are unlikely to match these antigens (absence of signal 1t). As both signals 1t and 2t are needed for activation of naive T cells, the overall chance of survival and activation of self-reactive IS cells is very small [51]. It seems that the IS can not provide full self-tolerance, but it uses various mechanism for minimizing self-reactivity and the damage that it causes to the body while reacting to the pathogens.

In summary, the infectious-nonself theory provides (building on top of the self-nonself model) a new control for directing the adaptive IS *what* it should react against (against infectious nonself), and the APCs do the main job in controlling the adaptive IS to focus on the antigens correlated (in time and space) to the pathogens. Whereas the innate APCs recognize pathogens based on their generic features, the adaptive IS cells are developed that recognize the pathogens based on their specific antigens.

2.4.3 Danger-Signal Theory

This theory is similar to the infectious nonself theory explained in the previous section. The main difference is in the mechanism of how the APCs are activated, i.e. what the IS is designed to react against.

According to the danger-signal theory an APC gets activated when it receives so-called *danger signal*. The danger signal is generated when there is some damage to body cells, which is usually due to pathogens. As an example, the danger signal is generated when a cell dies before being old, due to infection by a virus; the cell debris are different when a cell dies of old age (apoptosis) than when it is killed by the pathogen (necroses). APCs have receptors able to sense (recognize) the presence of necrotic debris. This recognition is called reception of the danger signal.

An APC that receives the danger signal will become active. It will further behave in the same way as PAMP-activated APCs (see previous section), and thus co-stimulate naive T cells able to recognize the antigens related to the cause of the damage, i.e. presented in danger context (along with the activation signal). In absence of the danger signals, APCs present collected antigens but do not express (or express different) co-stimulatory molecules, and therefore induce peripheral tolerization of T cells that match these antigens.

The process of collecting and presenting antigens in the safe context (when signal 2t is absent) seems also to be triggered by the protected system [19]. When a body cell dies by apoptosis, it secrets ”molecules of normal death” (*safe signal*). DCs can recognize these molecules, upon which they get activated to engulf the dead body cell (or cells), move to the lymph nodes, and present the antigens of the dead cell (or cells) in the safe context. This causes the previously explained peripheral tolerization of T cells, in this case of self-reactive T cells.

It should be noticed that according to the infectious-nonself model the innate IS is able to recognize predefined pathogen-specific patterns (PAMPs). This recognition is evolutionary developed and fixed within the lifespan of an organism, which in a way also predefines what the IS is be able to fight against. On the contrary, with the danger model the IS should be able to fight against anything (there is no predefined-patterns based dependance) that causes damage to the body, by guiding the adaptive part to develop IS cells that are specific to the antigens that the innate part finds being related to the damage (related by temporal and spacial proximity to the danger signals). The danger model seems to be free from the assumption that predetermined knowledge about pathogens is needed for mounting a reaction against them, which is the crucial difference when compared to the infectious-nonself model.

The infectious-nonself and the danger-signal models seem to be complementary rather than mutually exclusive explanations of how the HIS works, and both build on top of the basic mechanisms conceived in the self-nonself model.

2.5 The Adaptive Immune System

The adaptive IS consists of two main types of cells. They are B cells and T cells. Both B and T cells are covered with or able to secret antigen receptors. Antigen receptors on B cells are called BCRs (B cell receptors), and on T cells they are called TCRs (T cell receptors). Unlike T cells, B cells are also able to secret antigen receptors. Antigen receptors secreted by B cells are called antibodies. Both B and T cells are also equipped with various signal receptors and are able to receive or send various signals, depending on their current state.

Time-line of the development and activation of the adaptive IS cells and their use in detection of pathogens, including role of the innate IS cells in this process, is illustrated in Figure 2.2 and explained in Sections 2.5.1-2.5.6. The figure and the explanations are at the level of main antigen and signal recognition events and by these events triggered processes, omitting less relevant processes and details. They are aimed for understanding of the main logic behind the ”information representation and processing” performed by the HIS.

2.5.1 Generation of Cells of the Adaptive IS

One B cell is covered by only one type of antigen receptors (BCRs), but two B cells may have very different BCRs. As there are many B cells (about 1 billion fresh cells are created daily by a healthy human), there is also a large number of different BCRs at all times. How is this diversity of antigen receptors created and why do antibodies not match self antigens ? The answer is in the process of creating B cells. B cells are created from stem cells in the bone marrow by a rearrangement of the genes in immature B cells. Stem cells are generic cells from which all immune cells derive. The rearrangement of genes provides diversity of B cells, with potential of matching many different antigens, both those from pathogens and those from body’s cells.

Before leaving bone marrow, B cells have to survive a negative selection: If the antibodies of a B cell match any self antigen (antigens characteristic to body’s cells) present in the bone marrow during this phase, the cell dies. The cells that survive are more likely to be self tolerant, and move to the various tissues to participate in recognition and detection of pathogens.

B cells are however not fully self tolerant, because not all self antigens are present in bone marrow. Additional self tolerance is provided by T cells: Before a

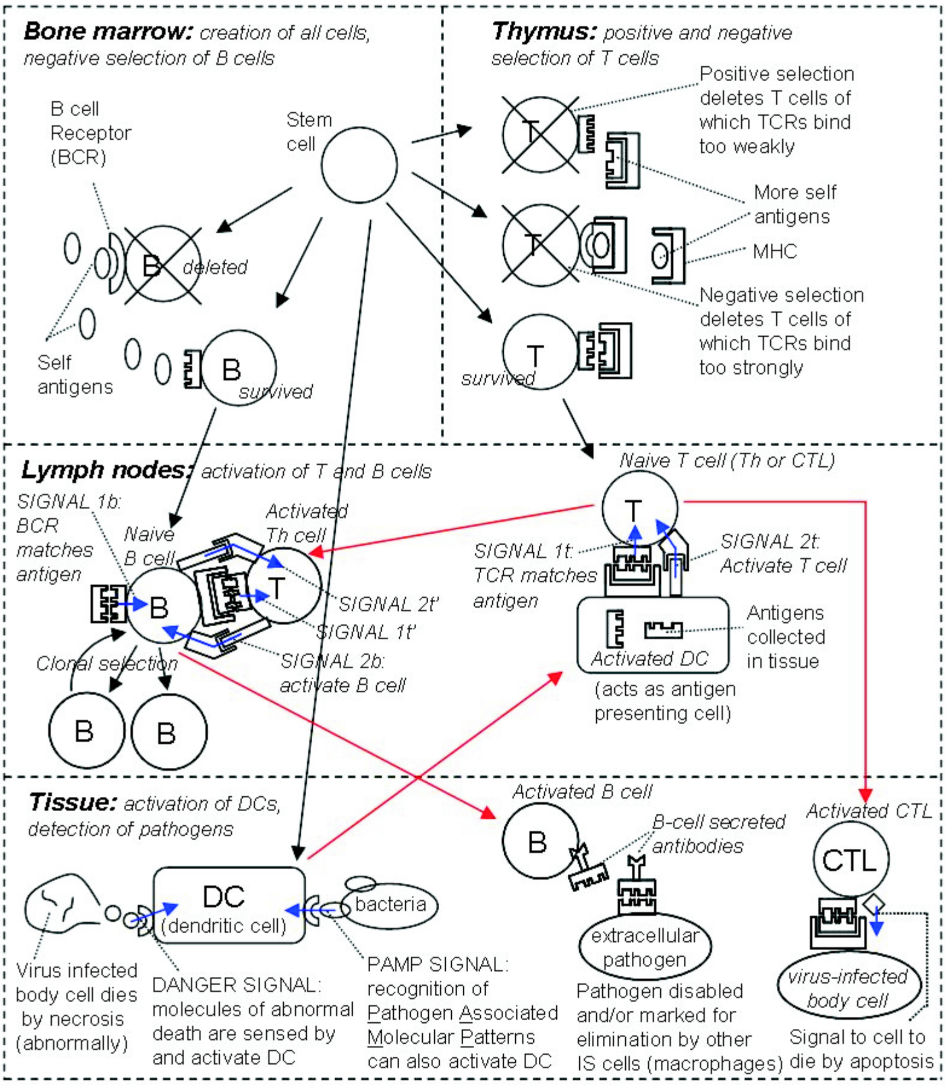

Bone marrow: creation of all cells, negative selection of B cells
B cell Receptor (BCR)
Stem cell
B deleted
Self antigens
B survived
Thymus: positive and negative selection of T cells
Positive selection deletes T cells of which TCRs bind too weakly
More self antigens
MHC
Negative selection deletes T cells of which TCRs bind too strongly
T survived
Lymph nodes: activation of T and B cells
SIGNAL 1b: BCR matches antigen
Naive B cell
Activated Th cell
SIGNAL 2t'
SIGNAL 1t'
SIGNAL 2b: activate B cell
Clonal selection
Naive T cell (Th or CTL)
SIGNAL 1t: TCR matches antigen
SIGNAL 2t: Activate T cell
Activated DC
Antigens collected in tissue
(acts as antigen presenting cell)
Tissue: activation of DCs, detection of pathogens
Activated B cell
B-cell secreted antibodies
Activated CTL
CTL
DC (dendritic cell)
bacteria
extracellular pathogen
virus-infected body cell
Virus infected body cell dies by necrosis (abnormally)
DANGER SIGNAL: molecules of abnormal death are sensed by and activate DC
PAMP SIGNAL: recognition of Pathogen Associated Molecular Patterns can also activate DC
Pathogen disabled and/or marked for elimination by other IS cells (macrophages)
Signal to cell to die by apoptosis

Figure 2.2: **Development and activation of adaptive IS cells, including role of the innate IS cells in this process.** Adaptive IS cells are generated in bone marrow and have random antigen receptors. They are negatively selected in bone marrow or thymus to die if their antigen receptors recognize samples of self antigens (central tolerance mechanism). Negative selection of T cells is more strict, as more self antigens are presented in the thymus then in bone marrow. T cells are also first positively pre-selected for the ability to recognize antigens presented by MHC molecules. These so-called naive B and T cells are not able to react against pathogens. They move to the lymph nodes where they are activated or suppressed. Activation is initiated by dendritic cells (DCs), the innate IS cells that act as antigen presenting cells in lymph nodes. DCs collect antigens in the tissue and present them (using MHC molecules) in lymph nodes along with the context in which they were collected, i.e. along with an activation signal (signal 2t) if they were collected in infectious (PAMP-related) or danger (damage related) context, else without the activation signal. T cells that recognize the presented antigen (signal 1t) and receive signal 2t are activated. T cells that receive signal 1t without signal 2t are suppressed (peripheral tolerance). Activated T cells either (CTLs) directly participate in elimination of the pathogen or (helper T cells) activate other cells to do so. B cells are activated only if they both recognize an antigen (signal 1b) and receive an activation signal (signal 2b) from a T cells cognate (signals 1t' and 2t') to the same antigen. Activated B cells are multiplied and tune their recognition to the pathogen in the process of clonal selection. They secret antibodies (antigen receptors) that act against pathogens.

B is able to secret antibodies and actively participate in detection and elimination of the antigens, it has to receive a co-stimulation from a T cell. The process of co-stimulation (activation) is further discussed in Section 2.5.4.

T cells are created in the bone marrow in a similar way as B cells, but they move (via blood stream) to the thymus and undergo the process of negative selection in there. In the thymus much more self antigens participate in the process of negative selection then in the bone marrow, which is why T cells provide the additional self-tolerance.

2.5.2 Positive Selection of T Cells in the Thymus

When T cells coming from the bone marrow via blood stream enter the thymus, they are first positively selected for the general ability to match antigens presented by MHC (Major Histocompatibility Complex) molecules.

Regarding MHC presentation, for this short overview it is only important to know that MHC molecules are molecules of a cell used by that cell to present (display) on its surface the antigens synthesized inside the cell (MHC type 1) or the antigens eaten or matched by the cell and processed for external presentation (MHC type 2). Looked more in detail, MHC actually presents smaller parts of the antigens, called peptides. The different roles of MHC type 1 and MHC type 2 presentation will become more clear from the explanations of Section 2.5.4.

In the process of positive selection, only T cells that own TCRs that at least weakly recognize at least one of the (self) antigens displayed by MHC in the thymus are selected to survive, and other T cells die. According to an explanation found in [73], purpose of positive selection is to allow survival only to those T cell that will later in the detection process be focussed (able to match) only to the antigens presented by MHC molecules (at that time a strong matching to the peptide plus MHC is required). Probably for this reason, positive selection is also known under name MHC restriction.

2.5.3 Negative Selection of T Cells in the Thymus

T cells that survive positive selection move to the part of the thymus where they undergo negative selection process. In this process T cells matching strongly enough some of the antigens (presented by the MHC molecules) are instructed to die (therefore the term negative selection).

As most of the self antigens are presented in the thymus during the negative selection process, the surviving T cells are likely to be self-tolerant. In this way achieved tolerance by the immune system to the body is called *central tolerance*. It can be seen as a high decrease in probability that the T cells match self antigens when they are later used for detection in the tissues.

It is important to mention here that the immune system uses additional self-tolerance mechanisms: The T cells that developed in the thymus need to be co-stimulated (activated) by the signals received from the innate IS cells before they can actively participate in detection and elimination of the pathogens. According to the newer IS theories (discussed more in detail in Section 2.4), the function of the immune system is not only to achieve tolerance to predefined self (which would mean that ”what is protected” is predetermined and the immune system should react against all the rest), but also to achieve tolerance to newly acquired self (”what is protected”, or at least what is not reacted against, can be dynamically determined within the life time of the protected system).

However, the negative selection is still very important as it ensures self-tolerance to at least some of the protected system (body) antigens. As the self-tolerance control provided by the innate IS cells signalling is probabilistic (as explained later), the negative selection certainly decreases the total self reactivity caused by the immune system.

It should also be mentioned that the T cells that leave thymus are at that time specialized to function as either killer T cells (also known as Cytotoxic Lymphocytes, CTLs) or helper T cells (Th), i.e. to perform one of the two very different functions, as it is explained in the next section. What exactly controls the differentiation of T cells into CTL or Th cells is not well understood [73], but also seems to not be important for understanding the overall functioning of the IS.

Self presented in the thymus. The way how the self antigens are provided that are presented during the negative selection in the thymus is still an open question in biological research. Also, older and simplified models of the HIS assume that only self antigens are presented in the thymus during the continuous process of negative selection. However this assumption has been negated in the recent literature about the HIS, where we find that ([73], pages 85-87; [27]): (1) both self and non-self antigens are presented in the thymus; the rules about how antigens can enter the thymus from the blood are unclear; (2) the thymic dendritic cells that present antigens survive for only a few days in the thymus, so they present current self antigens; if a non-self antigen is picked up for presentation during an infection, it will be presented only temporarily; once the infection is cleared from the body, freshly made antigens will no longer present the foreign antigen as self.

2.5.4 Activation or Suppression of Adaptive IS Cells, and Detection of Pathogens

B cells produced in the bone marrow and T cells produced in the bone marrow and additionally developed in the thymus are not fully mature, i.e. they are not not activated and able to perform detection of pathogens. They are often called *naive IS cells*.

Maturation and activation of naive IS cells happens mainly in the lymph nodes and spleen, as these organs are also main destination for the innate IS cells that come from the sites of infections and present to the naive lymphocytes the antigens they collected at these sites, along with a signal that determines the context in which that antigens were collected.

T cells. The way how the innate IS cells called APCs process inflammation and damage signals and to these signals related antigens, and then activate T cells in lymph nodes, is explained in detail in Section 2.4, and also illustrated in Figure 2.2 (signals 1t and 2t). Both CTLs and Th cells are activated in this way, with a subtle but important difference: CTLs are able (thanks to their so-called CD8 receptors) to see antigens presented by MHC type 1, and Th cells are able (thanks to their so-called CD4 receptors) to see antigens presented by MHC type2 molecules. This difference allows the CTLs and Th cells to properly accomplish their very different functions, as explained in what follows.

When viral antigens picked up by an APC multiply inside that APC, they will be presented by MHC type 1 on the surface of the APC, and therefore activate CTL cells. Activated CTLs move out of lymph nodes and go to the various tissues and find the sites of infection. When they recognize the cognate antigens displayed on the surface of virus infected body cells by MHC type 1 molecules, they secret molecules (called perforin) that causes apoptotic death of the virus infected cell.

The ability of the CTLs to cause apoptotic and not necrotic type of death of the virus infected cells is very important. During apoptotic death the content of the cell is not released to the environment but is nicely packaged by the cell and cleaned from the body by other cells of the IS (called macrophages). This prevents further spread of the virus from the killed infected cell. As the virus is eliminated that was already inside the cell, this type or IS response is called *cellular response*.

In the other case, when the antigens presented by the APC are parasitic or bacterial they will be presented by MHC type2 (raw picked up viral antigens are also presented in this way), and therefore activate Th cells. Main role of Th cells is to co-stimulate (activate) B cells, which upon activation secret antibodies. Antibodies are the main effector in detection of intercellular pathogens (including viruses). Detection and clearing of intercellular pathogens is usually called *humoral response*.

B cells. For a B cell to become activated, it must receive both signal 1b (recognize a cognate antigen) and signal 2b (co-stimulatory signal from an activated Th cell), as illustrated in Figure 2.2.

Activation of a Th cell by an APC causes the Th cell to move to the part of the germinal node where B cells mature. There, the T cell interacts with B cells and multiple signals are exchanged during this interaction that cause activation of B cells. After some antigen receptors of a B cell match antigens of a pathogen or self cell (signal 1b) that antigens are processed and presented on the surface of the B cell (by MHC type 2 molecules). Recognition of the presented antigen by a Th cell (signal 1t’) makes the contact between the two cells stronger and causes an additional interaction between the cells. In this interaction the two cells mutually co-stimulate each other (signals 2t’ and 2b), and the B cell gets activated. The T cell also secrets some cytokines that have an active role in the activation of the B cell (not shown on the Figure 2.2 due to the lack of space).

An important mechanism that lessens the chance for self-reactivity of B cells to come into the effect is the constrained trafficking pattern of the naive B cells. Naive B cells are limited to lymph nodes and are rarely found in the tissues. As the self antigens that are not enough presented during the negative selection of B cells in the bone marrow are also rarely found in the lymph nodes, signal 1 will also be rare during maturation of B cells in lymph nodes. However, event if the B cell recognized self antigen (signal 1b), due to the T cells tolerance mechanisms it is a high probability that a T cell cognate for that antigen will not be there to co-stimulate (activate) the B cell (the signal 2b will be absent) and the self-reactive B cell will die.

Initial activation of B cells triggers in the lymph node the process of producing new B cells, that will be able to match the pathogen better. This process consists of clonal multiplication of the activated B cells, rapid mutation of the clones, and selection of the higher affinity clones for repeated activation. The process is called clonal selection and is additionally explained in Section 2.5.5. The outcome of the clonal selection process are highly specific activated B cells. These cells become either antibody-secreting plasma B cells or memory B cells, move to the tissues (most plasma B cells actually stay in the lymph nodes and only the antibodies they secret move to the tissues), and perform their function there.

Plasma B cells secret antibodies that bind to the cognate antigens on the intercellular pathogens found in various tissues. Antibodies stick to the and pathogen mark the pathogen for removal by the innate IS cells called macrophages. Additionally, in some cases the binding itself already disables the functionality of the pathogen. Role of the memory B cells is explained separately in Section 2.5.6.

Some additional details about B and T cell activation and suppression. In the previous part of this section we explain (and illustrate in Figure 2.2) only the most important interdependent signals and cells activation mechanisms. In the next paragraphs we shortly mention few other important activation mechanisms.

For some classes of antigens, B cells may be activated without the signal 2b from Th cells. An interesting class are the antigens for which *clustering* of many BCR-to-antigen recognitions on the surface of a B cell causes signal 2b to be delivered to the B cell, and thus activate the B cell.

Once naive T cells are activated in the lymph node and move to the tissues, they can be re-activated by other sources of signal 2t then getting it from the dentritic cells. In the tissues, the main APCs that supply signal 2t and re-activate T cells are the macrophages and experienced (previously activated in a lymph node) B cells.

The requirements for activation of memory B and T cells are also different then for the naive B and T cells, and are explained in Section 2.5.6.

While the immune system uses various (central and peripheral) self-tolerance mechanisms to keep the probability of creation and activation of self-reactive IS cells very small, such IS cells can still occasionally be produced. E.g. during an infection and damage caused by a pathogen, in addition to collecting antigens of the pathogen, dendritic APCs may also pickup some self antigens in the proximity of the infection (damage) and present them in lymph nodes. If a T cell with cognate

TCRs managed to escape deletion in thymus (not all self antigens are presented during the negative selection in the thymus) and reach the lymph node it might be activated. This causes self-reactivity by the IS to the body. However, once the pathogen is cleared from the body the innate innate IS cells will stop to supply co-stimulatory signals to the adaptive IS cells. Repeated detection of self-antigens without received co-stimulatory signals will cause self-reactive IS cells to die (be suppressed).

2.5.5 Clonal Selection

After initial activation of B cells in lymph nodes (explained in the previous section), B cells enter the process of clonal selection. In this process new B cells are produced, that are more specific to the currently ongoing pathogen and in a number that suits the size of the infection.

The clonal selection process goes as follows. An initially activated B cell divides into a number of clones with similar but not strictly identical BCRs (antibodies). Statistically, some clones will match the pathogen that triggered the clonal selection better than the original B cells and some will match it less well. If the pathogens whose antigens triggered clonal selection are still present, they will continue to trigger cloning new B cells that match well the pathogen.

The process continues producing B cells more and more specific to present pathogens. B cells that are specific enough become *memory B cells* and do not need co-stimulation by signal 2b in the future. This is a process with positive feedback and it produces a large number of B cells specific to the presented pathogen. The process stops when pathogens are cleared from the body.

B cells can begin clonal selection without confirmation by signal 2b, but only in the case when matching between B cell antibodies and antigens is very strong. This occurs with a high probability only for memory B cells, the cells that were verified in the past to match nonself antigens.

2.5.6 Memory

Memory B cells produced in the process of clonal selection in the lymph nodes live a long time and they are ready to react promptly to the same cognate pathogen in the future. Whereas the first time encountered pathogens require a few weeks to be learned and cleared by the IS, the secondary reaction by memory B cells takes usually only a few days. Fast secondary reaction may be explained by the fact that memory B cells do not require the signal 2b in order to be activated.

Some of the activated T cells also become memory T cells and live longer. They speed up the secondary response by the IS to the repeated encounter of the same pathogen, because they do not need re-activation by dendritic APCs and can be easily re-stimulated by B cells and macrophages (who also can act as APCs).

Memory feature of the HIS is important for multiple reasons. As the pathogen is cleared much faster during the second and later encounters, the damage it can cause to the body is much smaller then during the first encounter. It also allows for use of *vaccination*, i.e. administering to the body weaken or ineffective antigens of a pathogen or similar artificially made antigens so that the IS is given chance to develop the appropriate immune memory specific to the pathogen. Immune memory and quick secondary response also save resources of the IS needed to clear second and subsequent infections by the repeated or a similar pathogen.

2.6 Summary and Conclusion

The HIS can be seen as an information processing and learning system that is well adapted to the kind of problems it needs to solve. In this section we summarize its working mechanisms and principles from the information processing and learning perspective.

The HIS uses (evolutionary) predetermined knowledge about some pathogens, contained in the recognition ability of the innate IS cells. Surface of these cells is equipped with antigen receptors that are able to chemically recognize antigens (molecular and protein patterns) on the surface of the pathogens, which causes detection and elimination of the pathogens. As the innate IS cells are developed independently of the pathogens, the reaction by the innate IS part to the pathogens it can recognize is immediate, efficient and fast.

Additionally, the HIS has adaptive mechanisms to develop and activate IS cells that recognize not previously seen (e.g. mutated) versions of the pathogens as well as new (evolutionary unknown) pathogens. The ability of the HIS to cover practically any pathogens (i.e. any antigen patterns) comes from the fact that the adaptive IS cells are generated at random, more precisely with random antigen receptors (but the receptors of one generated IS cell are all equal). In addition to producing useful IS cells that have potential to match pathogens, the random generation per se also produces IS cells with antigen receptors that would match the cells of the body (self-reactive IS cells). However the IS uses various mechanisms to prevent further development and activation of the self-reactive IS cells.

Most of the newly generated self-reactive IS cells are deleted by comparing the newly generated IS cells to the antigen samples known to be from the body cells, and deleting those that match (negative selection) before they can be used for detection. Self-tolerance achieved in this way is known as *central tolerance*. If the job of the HIS would be to only prevent self-reactivity against (by samples) predefined body self, negative selection would be enough to accomplish that goal.

However, it seems the HIS is designed to allow the body to be also self-tolerant to the self antigens that are not presented during the negative selection. This is important as indeed not samples of all body’s self are prepared (explicitly predefined) and available for negative selection. Additionally, the HIS learning mechanisms prevent self-reactivity to some new antigens that initially were not part of the body. As the HIS treats these antigens the same way as the antigens that initially were part of the body, we could say that the IS in a way allows the protected system (the body) to acquire (learn) new self. This self-tolerance to not explicitly defined or new self is achieved by allowing the IS cells produced in negative selection to become active and able for detection only if their antigen receptors pass some additional tests. These tests include interaction between the pathogen, innate IS, adaptive IS, and in some cases the protected system itself. The activation happens only if the antigen receptors of the IS cell are able to match antigens that are proven to be correlated either to the damage caused by pathogens or to the pathogens themselves.

Function of correlating antigens to the pathogens and damage is accomplished by the innate IS cells. There are two different mechanisms by which this is done. The first mechanism works as follows. The innate IS cells are able to recognize some molecular patterns found on many pathogens (called Pathogen Associated Molecular Patterns, PAMPs) but usually not found on self cells. Activated by this recognition the innate IS cells collect antigens from these pathogens and present them to the adaptive IS cells along with the activation signal. This will activate those adaptive IS cells that already passed negative selection and recognize the presented antigens.

The just described mechanism is practically based on predefined knowledge for pattern (PAMP) recognition. But instead of using this pattern-recognition knowledge directly for detection of pathogens, the information about the suspected antigens as well as detection duty is passed to the centrally tolerated adaptive IS cells. This makes a lot of sense, because two controls that prevent self-reactivity are combined in a way that requires both to fail in order for self-reactivity to happen. As previously explained negative selection deletes most, but not all self-reactive IS cells. The explained mechanism for correlating antigens might also by mistake collect and present some self antigens. However the chance is small for both exceptions to happen at the same time and for the same antigen. With this mechanism, the IS is concentrated at detecting infectious non-self, where infectious is defined by predefined receptors on the innate cells that recognize PAMPs, and non-self is defined by self samples used in negative selection.

Passing from the innate to the adaptive IS the antigen information about pathogens recognized initially by PAMPs is important for one more reason - detection of viruses that manage to enter body cells. Once the virus is inside a body cell, it can be recognized only be means of antigens displayed by MHC molecules of the cell.

Regarding the PAMP-recognition based correlation of antigens to pathogens, it is logical to ask why the pathogens do not (did not) simply evolve to not display PAMPs. The answer is in the fact that PAMPs are inherent to pathogens functionality (and are not inherent to self cells), in sense that pathogens having not these molecules would be less functional or not functional at all. The evolutionary pressure on pathogens is not only to try to be less detectable by the IS, but also to be able to perform their functions efficiently.

The second mechanism for correlating antigens to pathogens (and activating corresponding adaptive IS cells) is based on the ability of innate IS cells to receive signals (in form of molecules) directly from the protected system (body). When damaged by a pathogen, body's cells release molecules that signal this damage. These molecules (called danger signal) are sensed by innate IS cells and activate them to engulf the neighboring antigens (including antigens of the pathogen that caused the damage), and present them, together with a signal for activation, to the adaptive cells that already passed negative selection. Further consequences are the same as with PAMP-based activation.

With both mechanisms for activating adaptive IS cells, self-tolerance achieved by deletion of self-reactive IS cells (that survived negative selection) due to recognition of self antigens but lack of activation signal (the antigen is neither correlated to PAMPs nor to damage) is called *peripheral tolerance*.

We can notice and important principal difference between the PAMP-based and danger signal based activation mechanisms. PAMP-based (infectious non-self based) activation requires at least some predefined knowledge about pathogen (contained in receptors able to recognized PAMPs), and is thus constrained about which pathogens it can cause activation for. It however does not require any (initial) damage to be made by pathogens to the protected system in order to instruct adaptive IS cells to start reaction against the pathogens.

On the other side, danger signal based activation does require that first a damage to the protected system happens (at least a small damage) before this mechanism can be triggered. It however does not require any predetermined knowledge about pathogens, and allows the HIS to recognize any mutated or new pathogens once the pathogens start making damage to the protected system (body). Fortunately, due to huge numbers of body's cells of the same functionality and due to cells renewal, the body can easily tolerate small damage and loss of some of its cells.

Once the adaptive IS cells are activated, clonal selection fine tunes receptors to better match antigens of the pathogen, and produces large number of IS cells specific to the pathogen in order to clear it from the body (as usually many copies of the same pathogen enter the body or are produced inside the body due to pathogen multiplication). When a pathogen enters the body for the first time, this process takes rather long (usually few weeks), and it is called *primary response*. The process stops when the pathogen is cleared from the body. Debris produced by the process are cleared by the innate immune system.

Some of the IS cells with high specificity to the pathogen become *memory* cells. Memory cells live long and they are ready to react promptly to the same cognate pathogen in the future. Whereas first time encountered pathogens require a few weeks to be learned and cleared by the IS, the *secondary response* by memory cells takes usually only a few days.

In conclusion, we can see that the HIS demonstrates the unique information processing and learning mechanisms that allow it to efficiently protect the body against known and new pathogens. Its processing is manly based on pattern recognition, signals exchange, and state and activity change caused by these signals. The information processing is distributed over many individual cell to cell or cell to environment interactions, and learned knowledge is contained in many individual cells. The system has a unique way to combine predetermined knowledge about the protected system (negative selection, central tolerance) and about the pathogens (PAMPs recognition, peripheral tolerance) in order to minimize self-reactivity (mistake of reacting against the protected system). It is able to learn completely new pathogens by using feedback from the damaged protected system (danger signal) and correlating this feedback to the cause of the damage. It seems that the ability of the protected system (body) to tolerate small damage and to give some feedback about damage is crucial for the ability of the HIS to both learn new pathogens (and react against them) and at the same time tolerate new antigens that are not harmful (and that are potentially useful) for the body.

Chapter 3

AISs Background and State of The Art

This chapter is aimed at providing a non-AIS-specialist reader with a basic understanding of the standard AIS building blocks and the state of the art on their use, so that the reader is able to understand the design options and choices that we present when explaining AISs that we build within our work covered in this thesis.

We first outline the most common framework used for building AISs. Then we overview the standard AIS building blocks. We explain separately for each block its main function and mechanisms, and provide the analogy the HIS. For each building block we try to provide (and cite) the most important examples of its previous use and the existing findings (advantageous and disadvantageous properties) that one should have in mind when considering use of the block.

3.1 The Common AIS Framework

The most commonly used framework for engineering AISs was recognized and presented by de Castro and Timmis [14]. It is illustrated in Figure 3.1.

The framework involves three relatively independent steps (layers) in building an AIS: data representation, definition of affinity measures, and use of standard or modified AIS algorithms.

In the next two sections we overview some standard solutions and results for the three layers. We overview the standard solutions for the innate and adaptive subsystems separately, as they (the standard solutions) are qualitatively different for the two AIS subsystems.

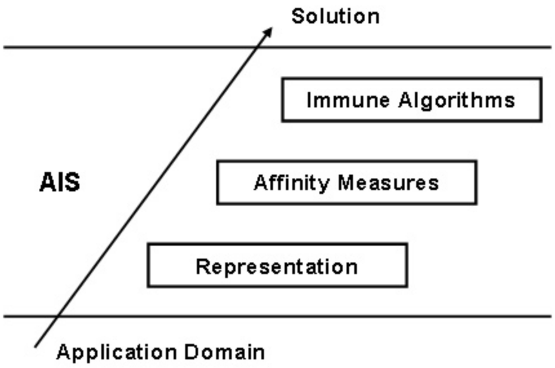


Figure 3.1: **A layered framework for engineering AISs ([14])**. Data representation defines how the observed application data (e.g. email messages) is transformed into a data format convenient for processing by AIS algorithms (e.g. binary strings). The used data format need to be such to enable measuring of the affinity (e.g. similarity) in the data representation space, as measuring of the affinity is in the core of recognition and classification by AIS algorithms (at least this is true for adaptive AIS algorithms). While standard or somewhat modified AIS algorithms usually may be applied, data representation is application specific and usually needs to be done by an expert. There are some standard affinity measures, but a proper choice depends on the application and used data representation.

3.2 Adaptive-Part AIS Building Blocks

When building an AIS, we call *adaptive* those mechanisms that use negative selection and/or clonal selection inspired algorithms to produce AIS analogs of the HIS antibodies (and other antigen receptors), i.e. to produce the patterns able to match bad-content patterns (e.g. in spam detection) or bad-behavior patterns (e.g. in computer networks intrusion detection) observed in the systems being protected (or monitored). Additional concepts or algorithms may be included in the process of producing the AIS antibodies, as explained in some of the following paragraphs of this section.

It should be mentioned that the AIS algorithms (and systems) may be applied not only for detection (this thesis focuses on and mainly talks about detection, which is in a way a two-class classification), but also for multi-class classification (as in [82]), for optimization (as in [52, 10]), etc. A detailed review of the AIS application areas can be found in [34, 35].

3.2.1 Data Representation

The formatted data on which so-called adaptive algorithms of an AIS operate are named *antigens* and *antibodies*. These are the direct-analogy counterparts of the HIS antigens and antigen receptors (antibodies are a subclass of antigen receptors) explained in the Section 2.5.

The term artificial antibody (or simply antibody when the artificial context is clear) is often used for both artificial analogs of HIS antibodies (antigen receptors secreted by B cells of the HIS) and artificial analogs of HIS antigen receptors (antigen receptors found on surface of T cells of the HIS). The reason for this is probably that often these two types of cells are not separately represented in AISs, but a unified data pattern (called antibody) is used on which various algorithms are performed that are linked to both B and T cells. In this thesis we follow the same practice and only use the term artificial antibody (or simple antibody).

The data representation defines how the observed application data (e.g. email messages) is transformed into *antigens* - a data format convenient for processing by AIS algorithms, and what is the format of the *antibodies* produced by the AIS algorithms. Antigens and antibodies need to use a format (or formats) that allows measuring their mutual affinity (e.g similarity), as measuring of the affinity is in the core of recognition and classification by AIS algorithms (see Figure 3.2).

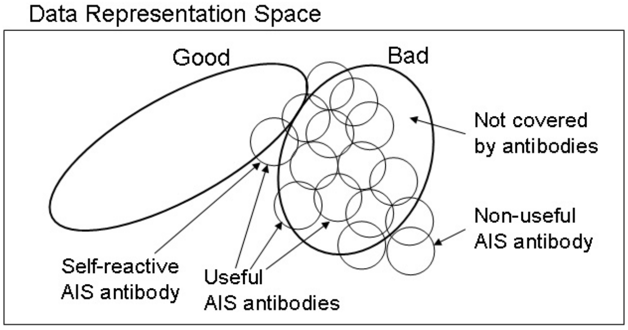


Figure 3.2: **Data representation and detection in AISs**. Antigens and antibodies are usually represented as data points in a multidimensional space. The job of the AIS is to distinguish between (usually) two classes of data, bad class (analogy to pathogens) and good class (analogy to body cells). This is done by creating antibodies, which are the data points in the centers of the spheres shown in the figure. When a new data (antigen) is observed, if it is within the sphere radius from an antibody (has high affinity to it), it is detected to belong to the bad class.

For the detection (classification) illustrated in the Figure 3.2 to work well, the data representation need to preserve important aspects of similarity (or complementarity) of the observed data, by mapping similar observed data into similar antigens. This way the data classes are well separated in the data representation space, which allows for an easier creation of antibodies that cover well the bad class and avoid (are not self-reactive to) the good class. This also allows for the generalization, i.e. the antibodies are able, usually to a good extent, to avoid detection of unseen good data and detect unseen bad data after being trained on non-exhaustive data examples.

An example of the antigen and antibody format, that is often used and preserves well the similarity of the raw data, is a vector of real valued features of the observed data. And an example of an usually appropriate affinity measure is Euclidian distance. In the related AIS work, the AIS antibodies and antigens are found to be represented in various ways: binary strings, strings of characters, real valued vectors, vectors of words, sets of words (fixed or variable size), etc.

A suitable data representation is usually dependent on the specific problem to which the AIS is applied. It is much defined intuitively and depends on the AIS system designer. It usually needs to be done by an expert in the application domain, as it may have a big impact on the performance of the system. In a recent critical overview [24], Freitas and Timmis point out that most AISs are built without giving enough attention on the importance and possible impact of the data representation (see ”inductive bias” discussion in their paper). Not only choice of a good and proper data representation format may matter, but a special expert care should also be devoted to the understanding and modelling properly the observed data, and when possible also impact the selection of the raw data that will be observed and used for modelling the observed system. Finally, a proper transformation of the data into a data representation space should be used that keeps the important data properties and data similarity.

When different parts of an antigen represent represent various properties of the observed data, these parts are usually called *genes*. Antigens may consist of fixed genes (which take different values for different observed data that is transformed into antigens), or the genes may change by being evolutionary selected from a predefined genes library [8]. It has been demonstrated that use of the gene libraries could be beneficial for enhancing the representation coverage [58](which would be useful e.g. when good representation is not easy to be found by an expert), and for adaptation of the representation to the evolving pathogens that might change their main types and diversity over the time [57].

3.2.2 Affinity Measure

The matching between the AIS antibodies and AIS antigens is usually by complementarity or similarity of their patterns, mimicking the chemical matching between antibodies and antigens in the HIS (see Section 2.5).

While in the HIS the affinity is determined by complementarity of the molecular and protein patterns, AISs usually evaluate the similarity of the compared patterns. Though choice between complementarity and similarity measures seems to be a pure formalism, Hart et al. show [33] that this choice may also impact the resulting system performance (at least in the context of the immune-network AIS algorithms they analyze). Use of similarity as the affinity measure is also easier to understand when designing data representation for information processing problems such are pattern recognition, clustering, and anomaly detection.

There are few standard similarity measures, including euclidian distance, hamming distance, and r-contiguous bits. The proper choice is application and data representation dependant. E.g. if the data is represented as a real vector, Euclidian distance is often applied as the similarity measure. In that case one should take care about proper choice of the similarity threshold in function of the dimensionality of the data representation (Stibor et al. [76]), and about proper choice of the used norm (1-norm, 2-norm, or n-norm) depending on the data being analyzed (see Freitas and Timmis [23]).

If the data is represented as a binary vector, usually r-contiguous (r contiguous values are equal in the compared vectors; the rule also applies to the case of integer vectors) or Hamming distance matching rules are used. However, one should be aware that r-contiguous rule and its variants introduce a positional bias problem, because with this rule the position of the genes within antigens matters in possibly unwanted ways, as shown by Frietas and Timmis [23]. In their more recent work [24], the same authors list many publications in which r-contiguous or a similar position-biased rule was used without a proper justification (possibly weakening the performance of the system), and suggest that r-contiguous matching rule should be avoided unless the nature of the problem suggest use of data representation with positional bias.

3.2.3 Algorithms of the Adaptive AIS Subsystem.

Usually standard AIS algorithms may be used while engineering an AIS, though their customization may be beneficial in some cases (as we show it is the case for a problem that we consider in this thesis). What we overview in the following sections are standard (most commonly used) *adaptive* AIS algorithms.

3.2.4 Negative Selection.

Negative selection is an adaptive immune system algorithm that allows learning based on the good content (behavior) data examples only (Figure 3.3). The algorithm is directly inspired by the negative selection of the immune system T cells in the thymus (explained in Section 2.5). The same process of the negative selection, as seen in the data representation space is shown in the Figure 3.5. Figure 3.5 also illustrates that a large part of the antigen and antibody representation space may correspond to impossible antigens (that can not be observed in the considered system). As nonself space is usually unspecified, the candidate antibodies must be generated randomly from the whole data representation space, which decreases the efficiency of the generation of useful antibodies.

Many AISs make their ultimate classification decisions based only on the detections (Figure 3.4) by the antibodies that are produced using only negative selection (or negative selection enhanced with clonal selection; clonal selection is explained in the next section). In these cases, the AIS is practically able only to learn and distinguish between the self represented by self examples provided for learning (negative selection), and the rest. What it does is practically an anomaly or novelty detection with respect to the self antigens provided for learning, and the generalization it achieves in this learning is tuned by means of negative selection

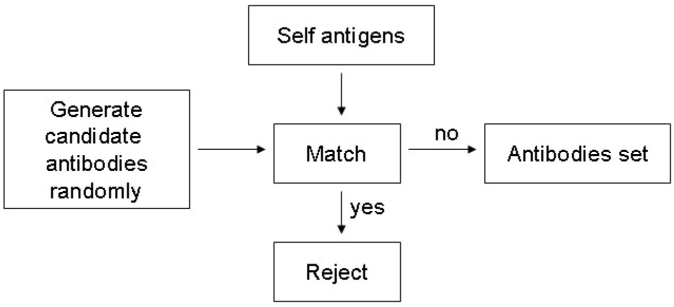


Figure 3.3: **Negative selection: algorithmic point of view** (Forrest et al. [21]). When the negative selection is used alone, the candidate antibodies are usually created at random, and then compared to the antigens produced from a set of good content (behavior) examples (*self* antigen examples). Those antibodies that match any of the self antigen examples are deleted, and the remaining antibodies are used for matching and detection of new antigens that are to be classified (Figure 3.4).

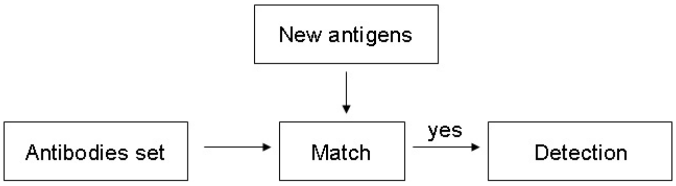


Figure 3.4: **Detection in a negative selection based AIS.**

and detection matching thresholds, which are in general different (in Figure 3.5 the two thresholds are the same and are represented by the radiuses of the corresponding spheres).

Negative selection have been first proposed by Forrest [22] within the context of anomalous program behavior detection. Since then it has been used for various anomaly detection applications, most notably for network intrusion detection as in works by Hofmeyer et al. [36, 37] and by Kim and Bentlay [40]. Other AIS (and non-AIS) algorithms can and have been used together with negative selection, or even have being integrated with it (an example is work by Kim and Bentley [42]).

Arguing on the usefulness of the negative selection, one could first ask why at all creating antibodies, i.e. why not simply keeping self data examples and classifying as anomalous those new data that is not similar to any of the self examples. Ebner et al. [18] argued that, even if the antibodies are to be created, keeping those (among randomly generated) that match self training data (a positive selection) is preferable, as the self space is usually smaller then the nonself space. In this case the purpose of creating antibodies would be to summarize possibly large number of overlapping training self examples by a smaller number of antibodies. New observed data (antigen) would be classified as anomalous if it matches none of the antibodies.

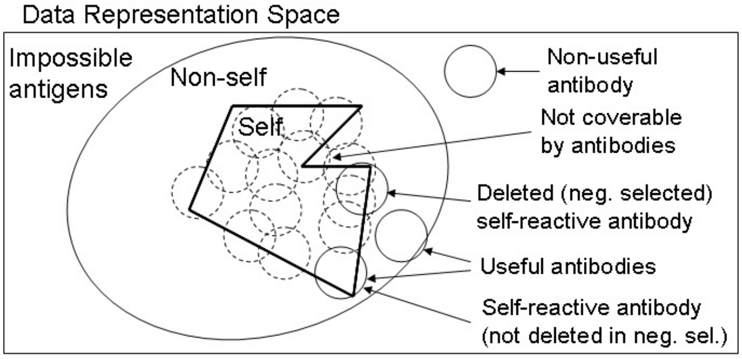

Data Representation Space
Impossible antigens
Non-self
Self
Non-useful antibody
Not coverable by antibodies
Deleted (neg. selected) self-reactive antibody
Useful antibodies
Self-reactive antibody (not deleted in neg. sel.)

Figure 3.5: **Negative selection: data representation space point of view.** Antigens (resp. antibodies) are represented by the data points in the centers of the dashed (resp. solid) line spheres. The antigens that belong to the set of all possible good content (behavior) are usually called *self*. All other possible antigens are called *non-self*. For the clarity of the figure, we show only few representative antibodies generated (or deleted) during the negative selection. In practice they are generated in a number large enough to cover well (probabilistically) the non-self space. As mentioned previously, an antibody matches an self antigen (and gets deleted) if the the antigen is within the sphere radius of the antibody. Usually not all self antigens are available during the negative selection (as shown in the Figure). Therefore, some of the randomly generated antibodies will survive that are self-reactive.

Moreover, with positive selection the candidate antibodies would not have to be generated at random, but could be created from the self training data points, which is additional computational advantage. Following on this idea and looking in a wider context leads to another and very strong logical argument against the negative selection (citation from [25]): *”...since methods like ANN and support vector machines (SVM) can form a hypersurface that can divide a state space into two (or more) subspaces, what is the point of randomly generating multiple detectors that are likely to leave gaps or contain redundancies and therefore provide less accurate classification?”*. As also pointed in [25], some answers can be found in Dasgupta and Forrest [12]: *”(i) having multiple detectors allows a non-self event to be approximately located in the state space, and (ii) detectors can be created with requiring any knowledge of the state space, unlike standard techniques.*”

Esponda et al. [20] compare positive and negative selection and show that for each there are cases in which it is more suitable, depending mainly on the application.

It has been shown in the related literature that systems built using the negative selection might also face serious scalability problems, because the random generation of the detectors becomes computationally infeasible and therefore practically inefficient if the antigens belong to a large-dimension space [40, 77]. For the same reason, Kim and Bentley suggest [40] that the negative selection should rather be used as an operator within the clonal selection algorithm, instead of being the main mechanism for generating new detectors. In their recent critical overview of the existing AISs, Freitas and Timmis [24] also analyze state of the art on negative selection, and suggest that the negative selection principle should not be used alone, but should rather be combined with more adaptive processes.

3.2.5 Clonal Selection

The clonal selection AIS algorithm is aimed at better populating with antibodies the non-self space around currently observed non-self antigens. It creates new candidate antibodies by slightly randomizing copies (clones) of the existing antibodies that prove to be useful for detecting currently observed non-self antigens, rather then generating them completely at random. This is analogous to the proliferation and mutation of the best fitting (being able to recognize currently present pathogens) B cells in the HIS.

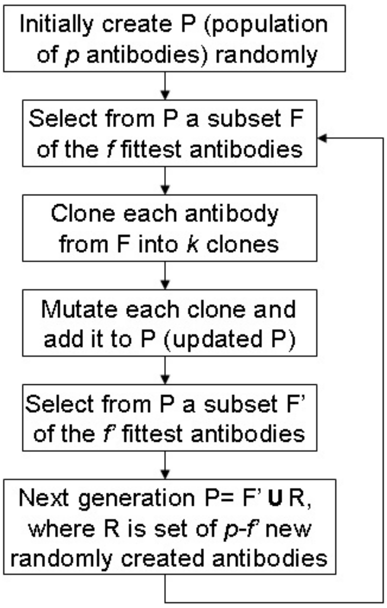

Initially create P (population of *p* antibodies) randomly
Select from P a subset F of the *f* fittest antibodies
Clone each antibody from F into *k* clones
Mutate each clone and add it to P (updated P)
Select from P a subset F' of the *f'* fittest antibodies
Next generation P= F' **U** R, where R is set of *p-f'* new randomly created antibodies

Figure 3.6: **A basic clonal-selection based AIS algorithm:** CLONALG for optimization (de Castro and Zuben [15]). Selection of the fittest antibodies, their proliferation (cloning) and mutation provide new antibodies specialized to further explore the space where the antigens are detected. Addition of a proportion of new random antibodies in each round of the algorithm allows the system to keep randomly exploring the antigen space. The figure shows a version of the algorithm for a continuous optimization (without a stopping logic).

A well known basic artificial version of the clonal selection was proposed by de Castro and Von Zuben [15] (Figure 3.6). The algorithm creates and adaptively optimizes a set of antibodies, e.g. to be able to well recognize a set of antigens. There is a version of the same algorithm adapted for pattern matching (de Castro and Von Zuben [16]),in which for each presented pattern the currently best fitting antibody is remembered in a set antibodies forming a subset of memory antibodies (called originally memory cells).

An interesting modification of the CLONALG algorithm is the algorithm by Kim and Bentley [42], which they apply for intrusion detection. They also create new antibodies from the best fitting existing once (using genetic operators), but add a negative selection operator for deleting the self-reactive antibodies created in this way. As specified in the paper, the algorithm evaluates the fittest antibodies on preselected non-self antigens. Therefore the algorithm learns using both self and non-self training data. It seems to be a quite promising extension of the clonal selection, as it provides means for both avoiding self-reactivity and adaptation of antibodies to non-self antigens (the later provides better use of resources then e.g. when the negative selection is used alone).

3.2.6 Memory

Analogously to the memory B cells of the adaptive IS (see Section 2.5), the AIS antibodies that prove to be useful in detecting non-self antigens may be assigned a longer life time, i.e. become memory antibodies. This allows for a faster *secondary response* if same or similar non-self antigens are observed in the future [36], the effect also known in the HIS [27, 62] (in the HIS, this effect is sometimes also induced by vaccination).

As noticed by Kim and Bentley [43], immune memory in AISs is often achieved implicity. Instead of having a separate class of antibodies being memory antibodies, only one class of antibodies is used. However some antibodies live longer as they are better at detecting antigens and therefore at surviving antibody population updates in adaptive AIS algorithms that delete the least useful antibodies. Gaspar and Collard [26] demonstrated that such AISs might fail to maintain memory if a part of non-self antigens dynamically changes and if antibody resources are limited. The reason for this is that the newly generated antibodies receive more stimulation from the current antigens and thus cause deletion of ”implicit” memory detectors (as the number of antibodies is constrained). Repeating but less frequent non-self antigens will be purely covered by antibodies in such a case, though they obviously deserve covering by longer leaving antibodies. This suggest use of a separate population of memory antibodies.

Implicit AIS memory, with a memory antibodies population separated from other antibodies, was used e.g. by Hofmeyr, and by Kim and Bentley. In Hofmeyr’s AIS [36] the memory antibody are assigned an infinite life time, but are deleted if found to be self-reactive. Also, to keep the memory population size limited, when it this limit is reached some randomly chosen memory antibodies are deleted. Kim and Bentley [43] investigate the role of the immune memory in systems with changing self profile (having in aim intrusion detection as a potential application).

They find that use of finite life-time memory antibodies is more appropriate in such a scenario, in addition to the mechanism of deletion of the self-reactive antibodies, which they also apply.

3.3 Self-Nonself Versus Danger Theory Based AIS Approaches

AISs built using only the adaptive-part AIS building blocks are able to discriminate between the self (represented by self data examples) and all the rest (called nonself). Consequently, they are called self-nonself based AISs. They are build using analogy to the self-nonself model of the HIS described in Section 2.4.1. Self-nonself based AISs essentially do anomaly detection.

Danger theory based AISs are those that, in addition to possibly using the adaptive part AIS building blocks (or some of them), also use the signaling mechanisms of the innate AIS part explained in Sections 3.4.2-3.4.5. Danger theory based AISs are able to do more specific detection (classification) then simple anomaly detection. They are expected to better adapt to the changing non-self in the system, and to be able to tolerate self in the system that is not predefined by self examples.

3.4 Innate-Part AIS Building Blocks

Similar like in the HIS (see Sections 2.2, 2.4), AIS innate mechanisms are also used to accomplish two rather distinctive functions: to directly perform the detection (classification) of the antigens and therefore the pathogens to which these antigens belong (the ”pathogens” being e.g. computer viruses and warms, email spam, network intrusions, defective software or hardware behavior, etc., and the antigens parts of them), or to do innate signalling, i.e. to instruct and assist the adaptive AIS mechanisms for doing that.

In both cases the innate mechanisms are in principle predetermined (if the analogy to the HIS is followed strictly), in sense that e.g. if they use pattern matching then the set of patterns that they can match is fixed (constant, unlike in the adaptive AIS part), and if they use rules then the rules are also fixed within one AIS software (hardware) version. The patterns and rules may be updated, but the updates are developed outside of the AIS.

The following sections give more explanation about the various innate AIS mechanisms, including the overview of their previous use.

3.4.1 Innate Detection

Innate AISs might be able to directly classify (or detect) observed pathogens or their antigens behavior or patterns, by use of the rules or pattern definitions fixed during the AIS design phase (or by an software/hardware update). A good example of the innate detection is the use of the human-predefined patterns and rules for detection of spam emails in SpamAssassin’s antispam software [74]. An AIS example with the innate part based on the fixed rules is the system for fault prediction in the refrigerating systems by detecting their defective behavior [78]. By detecting defective behavior predefined by the experts, the innate part of the system build in [78] allows the adaptive part to concentrate better on learning and detecting unpredictable defective behavior patterns.

3.4.2 Innate Signalling

Possible advantages of using innate AISs to process various signals from the environment (from the protected system and from ”pathogens”) and to use the processed signals in order to control the workings of adaptive AISs were first explicitly recognized and pointed to by Aickelin and Cayzer in 2002 [1]. The paper advocates use of analogy with the danger signal from Matzinger’s model of the HIS [50, 49].

The role of the danger signal in the HIS is algorithmically described in Section 2.4.3 of this thesis. The main promise of building danger signal based AISs (as advocated by Aickelin and Cayzer) was that such AISs should be able to detect (or classify) a subclass of non-self data (non-self is predefined by self examples and use of negative selection) that has an additional feature of being related to a danger signal, as opposed to the classical adaptive AISs without the danger signal mechanism that are only able to distinguish self (predefined by self examples) from all the rest (non-self). The meaning and nature of the danger signal can be very different and depend on the application. Danger signal can be for example a signal of damage caused by dangerous computer programs in case of intrusion detection systems, or it can be a signal indicating user’s interest in received information in case of filtering interesting information (e.g. interesting web pages).

In the case of intrusion detection in computer systems for example, the ability of detecting only dangerous newly-behaving (dangerous non-self) programs instead of detecting all newly-behaving (all non-self) programs would be an obvious qualitative advantage. It would allow the system to be useful even if not all self (behavior of good programs) can be predefined in advance, as the system is expected to be tolerant to the non-predefined self (non-predefined behavior of good programs).

Since the initial proposal by Aickelin and Cayzer, the approach gained on popularity and AISs have been built that exploit analogy to all the three main HIS signalling mechanisms (described in Sections 2.4.2 and 2.4.3 of this thesis): PAMP signal, danger signal, and safe signal. In the following sections we first explain these signals in the context of AISs and give some typical examples of how they were defined. Then we overview how these signals were jointly used to control the adaptive IS part or even to directly do detection or classification.

3.4.3 PAMP (Infectious-Nonself) Signal

We explain the PAMP-based signalling in the HIS in detail in Section 2.4.2. In summary, the PAMP signalling is based on the recognition of evolutionary predefined patterns (Pathogen Associated Molecular Patterns, or PAMPs) on the pathogens. PAMPs are often found on pathogens because these patterns are inherent to the pathogens functionality. Antigens of the PAMP-recognized pathogens are presented to the (previously negatively selected, see Section 3.2.4) cells of the adaptive IS along with a signal for activation of the cells that are able to recognize the presented antigens. The activated adaptive IS cells become able to recognize the pathogen even if its PAMPs become unavailable for observation (e.g. when a virus is inside a body cell, for more details about this read about cellular response in Section 2.5.4).

Similar like in the HIS, in AISs the PAMP-based pattern recognition differs from the recognition of antigens by antibodies (as explained in Section 3.2.1, the term antibody includes all antigen receptors of the adaptive AIS part, both those that correspond to the HIS antibodies secreted by B cells and those that correspond to the HIS antigen receptors found on T cells) both in the processes of creating the PAMP receptors and antibodies and in the consequences of the recognition events. While the antibodies (antigen receptors) are generated randomly, then selected based on a knowledge about the protected system, and upon activation used for the final recognition of the pathogen based on its antigens (as explained in Section 3.2.3), the PAMP receptors are generated based on some general knowledge about the pathogens and are mainly used for activating the antibodies.

One of few AISs that explicitly state use of PAMP signals is the system for computer intrusion detection by Greensmith et al. [32]. The system is aimed for detection of anomalous processes (and therefore suspicious programs) running on a computer within a user session. In their system, an increased number of ”destination unreachable” errors on a computer is used as a PAMP pattern related to outgoing scans of Internet addresses done by Internet worms (malicious programs that spread themselves over Internet without users intervention) and by some bots (programs that infect multiple computers and allow remote use of the resources on these computers by the attacker). This seems to be a good example of the PAMP signal, as the considered pattern is predefined and inherent to functionality of the considered classes of malicious programs (worms and bots) and it is not characteristic for majority of normal programs. The same AIS uses yet another PAMP signal that indicates high proportion of open and closed connections with respect to the exchanged traffic, feature that is again characteristic for malicious port scanning activities.

3.4.4 Danger Signal

We explain in detail the role of the danger signal in the HIS in Section 2.4.3. In summary, the danger signal is generated when there is a damage to the protected system (body cells). The signal triggers presentation of the antigens found in the proximity of the damage (thus including the antigens of the pathogen that caused the damage) to the (previously negatively selected, see Section 3.2.4) cells of the adaptive IS along with a signal for activation of the cells that are able to recognize the presented antigens.

Similarly, the danger signal in AISs is an indication of a damage to the experienced system, and must be such to give some information about what is the cause (or what are possible causes) of the damage. If the strict analogy is followed to the HIS, the danger signal is generated by the protected system, and has nothing to do with (pre-predetermined or learned) directly observable features of the pathogens.

In their AIS for intrusion detection, Greensmith et al. [32] use as danger signals the increased volume of the total traffic and the ratio of the number of connections-establishing packets to the overall number of exchanged packets (both per time unit). Both signals are based on analysis of statistics of the values under normal network usage and when there is an intrusion. If the analogy to the HIS is followed strictly, the signals seem to be more appropriate to be called PAMP signals then danger signals. Especially the second signal is rather a pattern characteristic to malicious programs then an evidence of a damage in the system.

In his thesis [79] Twycross describes an AIS for detection of abnormal usage activities of a server program running on a computer. He declares as the danger signals the anomalous values of CPU, memory and files usage by the monitored server. The amounts of the used CPU, memory and files are declared anomalous if they have not been observed during the preliminary training phase.

As noticed by Greensmith in her thesis [29], the way how Twycross generates the innate signals looks more like recognition of the patterns characteristic for the abnormal usage (that is to be detected) then like a sign of a damage in the system. Greensmith suggest that labelling the signals as PAMP signals instead as danger signals would be more appropriate in this case. Looking for a possible approval for naming the signals as danger signals in this case, we can notice the following. Unlike the antigens of the monitored server, which represent the structural properties of the behavior of the server (for the definition and use of the antigens in Twycross's system see Section 3.4.6), the signals can be seen as external to the behavior of the server. If the computer resources would be considered as the protected system and if their anomalous use would be considered as a damage to the protected system (which we are not sure is the best description of the purpose of the AIS), then it would be logical to call the signals danger signals.

A good example of the danger signal can be found in the work of Secker et al. [71], who make a system for filtering documents as being interesting to the user or not. The danger signal is defined as the absence of the user's interest in the offered documents (web pages or received emails). Because obtaining the web search results or emails that the user is not interested in does indeed damage the user in a way, it makes sense to consider the user as the protected system in this application. As both the protected system and the damage to it that generates the danger signal can be clearly identified in this case, the analogy to the HIS seems to be completely appropriate.

3.4.5 Safe Signal

We explain in detail the role of the safe signal in the HIS in Section 2.4.3. In summary, the safe signal is represented by ”molecules of normal death” secreted by body cells when they die normally. The signal triggers presentation of the antigens of the cell to the (previously negatively selected, see Section 3.2.4) cells of the adaptive IS without a signal for activation. Consequently, the adaptive IS cells that are able to recognize the presented antigens get suppressed.

One of few AISs that explicitly state use of safe signals is the previously mentioned system for computer intrusion detection by Greensmith et al. [32]. In their system the safe signals are generated when the networking traffic is stable between a computer and the Internet, and when the moving average of the packets size has normal (not very low) values. The authors see use of the safe signals as a way to counteract (unwanted) effects of other signals, i.e. to reduce misdirection.

3.4.6 Joint Use of the Innate Signals and Interaction With the Adaptive Part

Greensmith et al. [31] propose an algorithm, called Dendritic Cell Algorithm (DCA), for processing the innate signals mentioned in the previous sections along with antigens (explained later) to which they are related. The DCA uses a linear combination of the signals to periodically determine current values of the ”activation” or ”suppression” context in which antigens (processes) are observed. If enough signals is observed within a limited time (which the authors call the dendritic cell maximum life time), the prevailing context is used to classify the involved antigens. There are some additional details in the algorithm, such as use of ”inflammatory” signal for multiplication of effects of other signals, and use of multiple ”dendritic cells” that preform the explained signals and antigen processing (the classification results are averaged over multiple dendritic cells).

Using the DCA, Greensmith et al. build the previously mentioned AIS for detection of anomalous processes in computers [32, 29]. In their system, the antigens are IDs of computer processes that invoke system calls, and the DCA classifies the processes (running programs) as anomalous (suspicious) or not. The AIS does not have an adaptive part, and the classification is done directly by the DCA. The AIS is very light in terms of needed computation resources, which is usually not the case with AISs that use adaptive AIS algorithms (producing adaptive artificial antibodies is usually computationally intensive, see Section 3.2.3). However, the AIS exhibits rather high misdetection rates if the monitored process does not use many system calls or if active normal and anomalous processes are analyzed simultaneously, which questions its usability in realistic settings.

There are few other AISs that use the DCA, including the AIS for misbehavior detection in sensor networks by Kim et al. [41], the AIS for SYN scan detection by Greensmith and Aickelin [30], and the AIS used as a robotic classifier by Oates et al. [54]. All of them perform classification using the DCA only, and do not have an adaptive AIS part.

Very few AISs integrate the innate and adaptive AIS parts and use the innate IS mechanisms for controlling the development of the adaptive IS antibodies. The most explicit and noticeable effort in this direction is done by Twycross et al. while building the previously mentioned AIS for computer intrusion detection [79, 80] (or more precisely the AIS for detection of abnormal usage activities of a server program running on a computer). In their AIS the antigens are structured data records about system calls made by the monitored server.

Instead of creating antibodies, the authors apply the negative selection directly to the observed antigens, by deleting those antigens that match any of the self antigens collected during a preliminary training phase in which any anomalous use of the server is granted to be absent. They (the authors) observe that this is equivalent, though not completely analogous, to the central tolerization of the T cells in the HIS (central tolerization of T cells is explained in Section 2.5.3). The antigens that survive the negative selection are classified by use of the innate signals, in a way similar to the DCA (which is in a way equivalent to the peripheral tolerization in the HIS explained in Section 2.4).

While the approach by Twycross et al. seems to be very promising, as it tries to exploit advantages of both innate and adaptive mechanisms, the evaluation results for the application that they consider (intrusion detection) show very high false positive and false negative ratios. The approach has one additional drawback in that, due to the way the danger signals are defined, it doesn't allow for online learning (when abnormal use of the server is not granted to be absent) of legitimate changes in the normal usage of the monitored server.

In their system for email classification [71] (mentioned in the previous section), Secker et al. use the danger signal for co-stimulating clonal selection, and that is how the antibodies are activated and how the new antibodies are produced in their system. The co-stimulation happens when the user explicitly shows no interest in the emails previously classified as junk. The AIS is compared to a Bayesian classifier. Its classification accuracy is shown to be comparable to the used Bayesian classifier. The AIS is however faster in learning sudden changes in the user's interest. As the AIS has too high rate of misclassification of good emails, it is not appropriate to be used as a classical email filter. It could however be applied for finding interesting emails among those that have been declared as junk by standard email filers (and that are usually indeed junk for an average recipient).

On top of the algorithm developed by Secker et al. for the (above mentioned) AIS for email classification, Ayara et al. build the AIS for prediction of failures of ATM machines [3]. They enrich the system by adding generation of new antibodies from the antigens that correspond to the unpredicted failures.

Yet another AIS that integrates the innate and adaptive mechanisms is the AIS for cooperative worm detection by Kim et al. [46]. The AIS incorporates the functions of the T cells and DCs of the HIS. The system is however specified only at a high level and is not evaluated.

The AIS for information filtering by Chao and Forrest [9] also integrates adaptive and innate AIS functionalities, although the authors do not describe their system using the (only recently popular) danger theory terminology. In their AIS the non-activated antibodies are ”implicit” (not instantiated) and correspond to the incoming antigens. When the user protected by the AIS receives unwanted information that the AIS did not filter, it provides a feedback to the system. The antigen corresponding to the received information will be transformed into an activated antibody, which corresponds to the reception of a costimulation signal and activation of an antibody in the HIS. The authors also use negative selection to prevent creation of those antibodies that correspond to pre-declared ”good” information. The authors do not discuss that in this way they actually avoid random creation of new antibodies, which could cause inefficiency in the use of negative selection in their AIS.

3.5 Summary and Conclusion

The information processing principles and algorithms used for explaining how the HIS works have been applied for building many AISs aimed at solving similar technical problems. Computer intrusion and malware detection, information filtering, and data classification are typical AIS application areas, though there are many other.

Majority of the AISs use analogy to the adaptive HIS algorithms, namely negative selection and clonal selection. Negative selection, the crucial adaptive AIS algorithm for achieving tolerance to the protected system, is shown however to be computationally infeasible except for small and simple problems. Regarding adaptive AISs, it has been recently noticed that most of the built AISs did not put enough attention to the quality of the used representation and affinity measures, though these may be crucial for building a successful AIS.

Recently, AISs using innate mechanisms or combining innate and adaptive mechanisms became popular, promising better computational feasibility and qualitatively new learning capabilities.

Despite the unique and promising information processing AIS mechanisms, the recent reviews from the field [24, 25] agree however that AISs were not yet demonstrated as a clearly superior approach over the competing machine learning and statistical techniques, for an important real problem or a class of problems. The most often mentioned possible reasons for this are: the previously mentioned computational inefficiency of negative selection and not careful use of data representation and affinity measures; not proper choice of the problems that would fit well the AIS mechanisms and their possible advantages; not enough comparison of the built AISs to the competing approaches.

Chapter 4

AIS For Collaborative Spam Detection (AIS-CSD)

In this chapter we present our AIS for collaborative spam bulk detection. We build the system on top of a well known classic scheme for collaborative detection of spam bulkiness, by adding AIS algorithms to it. The original scheme uses similarity digests for finding emails that belong to the same bulk. Similarity digests are the binary strings algorithmically computed from email content or parts of email content. They preserve content similarity, in sense that if two text are more similar their digests are likely to also be more similar.

For initial assessment of the abilities of the system, we implement it in a simulator and provide a basic-evaluation results of the overall system. We also evaluate the effects that the innate and adaptive AIS parts have on the functioning of the overall AIS. All the evaluations are under one chosen spammer model (spam obfuscation).

4.1 Introduction

4.1.1 Related Work and Unsolved Problems

One of main problems not solved by the existing similarity-hashing based [11] and other collaborative content filtering methods [13, 61, 84] is that the representation of the email content used for spam filtering is vulnerable to the random or aimed text additions and other text obfuscation, which leaves many obfuscated spam emails undetected.

Also, the results of the queries to the collaborative-filtering databases have usually very constrained impact to the detection decision. For example, when computing an email spamminess score, the product offered to the students and employees at EPFL (our university), which is based on SpamAssassin [74], weights the Razor's result [61] approximately the same as the rule "subject is all big letters", and much less then the Bayesian score [28]. This is probably done in order to avoid false detection of good emails, because the used similarity-hashing representation is not precise enough for distinguishing well spammy patterns from normal email content.

Although the general idea of exchanging the exact or similarity signatures derived from the emails for spam bulk detection has been well known for years [81, 53], we do not find solutions that successfully address the above explained problems.

Damiani et al. [11] investigate the vulnerability of the Nilsimsa representation [53] (used by DCC [13]) and show the results that suggest that the representation becomes completely non-useful if enough random text is added by the spammer to the different copies of the same original spam message. They also find that, in case when the hashing function used by the filters is known, the spammer can defeat Nilsimsa representation by adding a much smaller (20 times smaller) amount of text.

Interestingly, though their results show major weaknesses of Nilsimsa, the authors comment the results only in the region of small random additions for which the representation is still good, i.e. the additions being up to 3 times longer then the spammy message. Nothing prevents the spammer from adding more text and moving into the region where the representation does not work well, which could happen already with the added random text 5 times longer then the spammy message. The problem here is that the signature is computed from all, or predefined but variable in length, parts of the email. This gives enough room to the spammer for effective add-random-text and/or add-chosen-text obfuscation. Our system is designed to avoid such problems.

Prakash and O'Donnell [60] describe a digest-based reputation system called CNC (Cloudmark Network Classifier). CNC uses similar mechanisms like the scheme analyzed by Damiani et al., but in the CNC system some users report about the received emails whether they are spam or not. A reputation of the reporters is maintained and used to aggregate the reports on the same signatures (the system uses exact matching between the similarity preserving signatures). The reporting of non-spam messages in their system is called ”negative assertions”. The negative assertions are used with the purpose of avoiding false detections of bulky good email. CNC however does not process the signatures locally before submitting them to the central database (all signatures from one email are declared either spammy or innocent).

Regarding the existing use of the artificial immune system algorithms for spam filtering, we find that both the email representation and the algorithms are crucially different from our solution. Oda and White [55] use a word-based representation. They compute scores based on both good and bad words present in the email, which is, the same as Bayesian filtering methods, vulnerable to the additions of good words or phrases.

The representation used by Secker et al. [71], another AIS based approach, is also word-based and not resistant to the letter-level obfuscation because the exact matching is used. It should be noticed however that their system is built with the aim of classifying emails as being interesting or not to a particular user, which is somewhat different task from the spam filtering. Their AIS learns about uninteresting (junk) email experienced by one user, by using the clonal selection AIS algorithm (an overview of the clonal selection is given in Section 3.2.4). It discovers the repeated spam patterns and could be used for detecting repeated spam (assuming use of an improved email representation that is resistant to spam obfuscation). On the contrary, the system has no built-in mechanisms to detect new spam content based on the network-wide bulkiness of spam, although the bulkiness offers a strong spam evidence that should certainly be exploited.

Another type of content-based filtering is Bayesian filtering, originally proposed by Graham [28]. A good feature of Bayesian filters is that they adapt to the protected user's profile, as they are trained on the good and bad email examples of the protected user. The disadvantages are vulnerability to the addition-of-good-words attack and absence of mechanisms to exploit bulkiness of new spam. The system has only a "local" view of a new ongoing spam bulk.

Usually the Bayesian filtering and collaborative filtering are done separately, and then the results are combined, along with results from other methods, for the final decision making. It might be advantageous for collaborative filtering if some local spamminess processing is done before the information is exchanged for the collaborative filtering, which the existing systems do not take into account.

The only solution known to us that uses the signatures on the strings of fixed length is the work by Zhou et al. [84], a peer to peer system for spam filtering. However, their signatures are exact and are not similarity signatures, as required by the rest of their system to work. Even modest spam obfuscation is able to alter some of the bits of such generated signatures, which prevents their system from detecting spam bulks. Their analysis results in a different conclusion, because they use rather unrealistic obfuscation (which alters the created signatures with a very small probability) to test their solution.

4.1.2 Our AIS Approach and Design Choices

Following our design principles announced in Chapter 1, we choose the components of our AIS for antispam in the following way.

Data Representation. The antigens and antibodies in our system use a (modified) format of similarity signatures used by a standard collaborative spam detection scheme [11]. The standard similarity signatures are generated from the predefined email parts or the complete email, and transformed into binary strings by use of a similarity hashing technique (as explained in detail in Section 5.1.4). This representation satisfies the AIS requirement of preserving the similarity of the original data (email text). Similar email texts results in similarity signatures having smaller Hamming distance then the similarity signatures of unrelated texts. Therefore the direct use of this representation for building an AIS is possible.

However, as we would like to apply the AIS processing on the signatures before exchanging them for collaborative spam bulkiness detection, and also as we notice a potential vulnerability of the above mentioned signatures, we assume that creating multiple similarity signatures from the email parts sampled at random email positions could be more appropriate to do. This assumptions are analyzed in detail and verified in Chapters 5 and 6. We also slightly alter the similarity hashing method itself, as explained in Section 4.2.3.

Having the signatures created from multiple random email parts and applying negative selection AIS mechanism on these signatures distinguishes our system from the DCC and CNC collaborative filtering schemes discussed in the previous section, as discussed in the next paragraphs.

In the remaining sections of this chapter, we use the term *proportional signatures*, which corresponds to the antigens and candidate antibodies in the AIS terminology, and we use the term *detectors*, which correspond to naive and activated antibodies in the AIS terminology.

Generation of antibodies and negative selection. We exploit a technological difference between AISs and the HIS and avoid the creation of antibodies by random generation and negative selection (which is usually inefficient, especially in cases of a high-dimensional data representation as is the one that we use). Instead, we create the candidate antibodies from the observed data (emails) and apply the negative selection to these antibodies. It might be that this way of creating antibodies is not used in the HIS simply for evolutionary and technological reasons, though in principle it would be possible.

While the ”negative assertions” declare all the signatures from one email as spammy or normal, our system is more precise and selects (by use of negative selection) which signatures should be used further, and additionally processes them (creation and local processing of antibodies) before deciding to exchange them for collaborative filtering. DCC doesn’t use any mechanism similar to negative selection or negative assertions.

We do not use clonal selection because the (above mentioned) way we create antibodies should implicitly provide the first function of the clonal selection - an adaptive and efficient search in the data representation space where the antigens are being observed (as opposed to producing many non-useful new antibodies by generating them randomly). The second function of the clonal selection in the HIS is to produce *many* similar antibodies, because many copies of the pathogen get spatially distributed in the body (or a part of the body) and a close antibody-to-antigen contact is needed for the recognition, and also because the antibodies are spent in the reactions against the pathogens; in the AISs we usually do not have the spatial component and the antibodies are reusable.

Use of innate signals. We use the two types of innate signals for controlling the activation of antibodies: the PAMP signal and the danger signal. The role of these signals in modelling the workings of the HIS is explained in Sections 2.4.2 and 2.4.3. Previous use of PAMP and danger signals in AISs is summarized in Sections 3.4.3, 3.4.4 and 3.4.6.

PAMP signal. PAMP signal in our AIS is measure of the bulkiness of emails. The bulkiness is a pattern characteristic to spam and usually not characteristic to

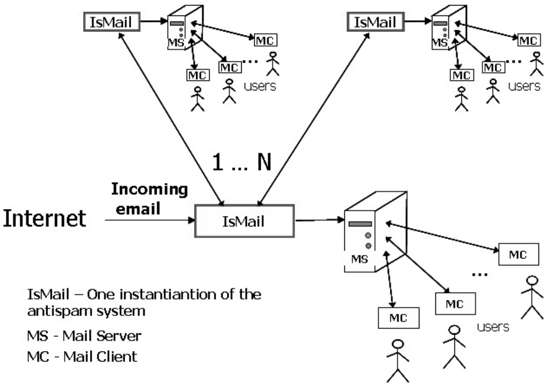


Figure 4.1: The position of an antispam system with respect to other antispam systems, the protected email server and Internet.

good emails. Bulkiness is also the pattern that is functionally important for spamming and can not be easily avoided by spammers. The analogy to the PAMP signal in the HIS is therefore very strong.

In our system, in order for the PAMP signal to be used, antibodies need to be created before the damage to the protected system (user receives unfiltered spam) happens. That is why we create the new antibodies from all the observed antigens (and not only upon the damage, as done in [9]).

Danger signal. We assume that users in the system have and use ”delete as spam” button. When a user receives an unfiltered spam - that corresponds to the damage in the HIS, and when it press ”delete as spam button” - that corresponds to the danger signal in the HIS.

4.2 Description of the System

4.2.1 Where Do We Put the Antispam System

The antispam system, which filters the incoming e-mails for the users having their accounts on the same e-mail server, is placed in front of that e-mail server towards its connection to the Internet (Figure 4.1). This is the logical place of the filter, though the deployment details might differ a bit.

The antispam system designated to one e-mail server and its users can be an application added to the e-mail server machine, or it can be a computer appliance running such an application. A few such antispam systems can collaborate with each other, and each of them is also interfaced to the email server and accounts it protects. The collaboration to other antispam systems can be trusted, like in the case of few antispam systems administered by the same authority, or evaluated by the antispam system and correspondingly adapted, as it would probably be the case in a self-organized collaboration of antispam systems with no inherent mutual trust.

4.2.2 What the system does, inputs, outputs.

The antispam system decides for the incoming emails whether they are spam or not. If enough evidence is collected that an e-mail is spam, it is either blocked or marked as spam and sent to the e-mail server for easy sorting into an appropriate folder. Otherwise, upon a maximum allowed delay by the antispam system or upon a periodic or user-triggered send/receive request from the user's email client to the email server, the email is passed unchanged to the e-mail server.

The first-type inputs into the antispam system are incoming e-mail messages, before they are passed to the e-mail server.

The second-type inputs to an antispam system come from the access by the antispam system to the user accounts it protects. The antispam system observes the following email-account information and events for each protected email account: text of the e-mails that the user sends; text of the e-mails that the user receives and does an action on them; the actions on the e-mails processed by the antispam system and received by the user, i.e. not filtered as spam, including deleting a message, deleting a message as spam, moving a message to a folder; the actions on the e-mails processed by the antispam system and filtered as spam, which could happen very rarely or never depending on the user's behavior and performances of the antispam system; the send/receive request from the email client of the user to the e-mail server; email addresses from user's contacts. We assume that some of the users protected by the antispam system have "delete" and "delete-as-spam" options available from its e-mail client for deleting messages and use them according to their wish, but this assumption could be released and another feedback could be incorporated from the user actions on his emails, like moving the emails to good folder for example or simply deleting the emails. Here "delete" means move to "deleted messages" folder, "delete-as-spam" means move to "spam messages" folder. We also assume that all the e-mails that the user still did not permanently delete are preferably on the e-mail server, so the antispam system can observe the actions taken on them. Here "permanently delete" means remove from the e-mail account. The messages could be all moved to and manipulated only on the e-mail client, but then the client should enable all the actions on the e-mails to be observed by the antispam system.

The third-type inputs to the antispam system are messages coming from collaborating antispam systems. The messages contain useful information derived from the strings sampled from some of the e-mails that have been either deleted-as-spam by the users having accounts on the collaborating antispam systems or found by local processing as being suspicious to represent spammy part of an email from a new spam bulk. The third-type inputs to the antispam system are especially useful if there is small number of the accounts protected by the system. One of the factors that determine the performances of an antispam system is the total number of the active accounts protected by the antispam system and its collaborating systems.

The main output from the antispam system are the decisions for the incoming emails whether they are spam or not.

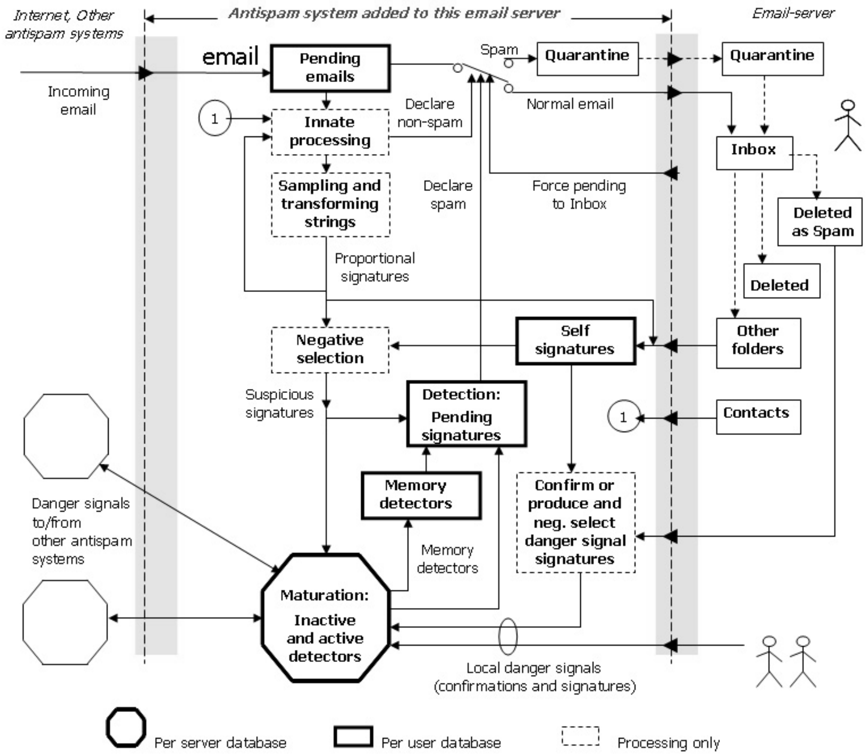


Figure 4.2: Internal architecture of the antispam system.

Another output are the collaborating messages sent to other antispam systems. These messages contain useful information derived from the strings sampled from some of the e-mails that has been deleted-as-spam by the users having accounts on the antispam system, or are locally found to be bulky. If the collaboration is self-organized and based on evaluated and proportional information exchange, the antispam system has to create these outgoing collaborating messages in order to get similar input from other antispam systems.

4.2.3 How the System Does Its Job - Internal Architecture and Processing Steps

Internal architecture and processing steps of the antispam system are shown on Figure 4.2. Each block represents a processing step and/or a memory storage (database). All the shown blocks are per user and are shown for only one user on the figure, except the “Maturation” block which is common for all the users protected by the same antispam system. The following processing tasks are done by the system.

Incoming emails are put into the pending state by the antispam system, until the detection process decides if they are spam or not, or until they are forced to an

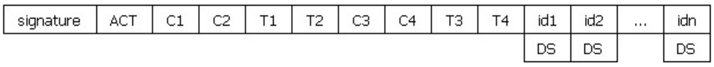
Inbox by pending timeout, by periodic request from the mail client, or by a request from the user. The innate processing block might declare an email as non-spam and protect it from further processing by the system. If an email is found to be spam, it is quarantined by the antispam system or it is marked as spam and forwarded to the email server for an easy classification. Otherwise it is forwarded to the email server and goes directly to the Inbox. The user has access to the quarantined emails and can force some of them to be forwarded to the Inbox, but is not required to do so.

A pending email that is not protected by the innate part is processed in the following way. First, the text strings of predefined length are sampled from the email text at random positions. Then, each sampled text string is converted into the binary-string representation form called proportional signature ("binary peptide"). The details on creating the proportional signatures are given in Section 4.2.3. To continue reading this section, you just need to know that similar strings generate similar proportional signatures, i.e. their signatures have small hamming distance, and that unrelated strings with very high probability result in not similar proportional signatures (big hamming distance). This explains why the term proportional signature is used.

Each proportional signature is passed to the negative selection block. Another input to the negative selection block are so called self signatures, the signatures obtained in the same way as the proportional signatures of the considered incoming email, but with the important difference that they are sampled from the e-mails that the user implicitly declared as non-spam (e.g. outgoing emails). In the negative selection block, the proportional signatures of the considered incoming email that are within a predefined negative-selection-specific similarity threshold of any self signature are deleted, and those that survive become so called suspicious signatures.

Each suspicious signature is duplicated. One copy of it becomes naive detector (naive antibody) and is passed to the maturation block, and another copy is the observed antigen and it is passed to the detection block. In the HIS the antigens are not negatively selected before detection. In our case we do it simply because it is convenient to reuse the (negative selection) processing that anyway has to be done for antibodies. Each suspicious signature passed to the detection block is stored there as a pending signature. It is compared against already existing memory and active detectors and against the new active and memory detectors potentially made during the email pending time. If a suspicious signature is matched (found to be within a predefined detection-specific similarity threshold) by an active or memory detector, the corresponding email is declared as spam. The pending signatures are kept only as long as their corresponding email is pending.

The active detectors used in the detection process are produced by the maturation (block) process. The inputs to this process are the above mentioned suspicious signatures (naive detectors), local danger signatures and remote danger signatures. The local danger signal signatures are created in the same way like the suspicious signatures, but from the emails being deleted as spam by the users protected by the



signature	ACT	C1	C2	T1	T2	C3	C4	T3	T4	id1	id2	...	idn
										DS	DS		DS

Figure 4.3: Syntax of a detector. ACT stands for activated/non-activated and this bit shows the state of the detector. C1 is the counter of the bulkiness of suspicious signatures (it serves as the receptor of the PAMP signal). C2 is the counter of the local danger signal signatures (it serves as the receptor for measuring the level of the received danger signal), i.e. the signatures generated from emails deleted as spam by users and negatively selected against user specific self signatures. Ti is time field for validity date of counter Ci. "id" is a local (server wide) identification of the protected user account that received email from which the signature originates, and is useful when deciding how and which users this signature might impact once it becomes activated (explained later). DS is so called danger signal bit of a local clustered signature. It is set to 1 if its corresponding signature comes from an email deleted as spam, else it is set to 0.

antispam system. The remote signatures are obtained from collaborating antispam systems.

Except upon start of the system, when it is empty, the maturation block contains so called inactive and active detectors. When a new suspicious signature is passed to the maturation block, it is compared using a first maturation-similarity threshold against the signatures of the existing inactive detectors in the maturation block. Syntax of a detector is shown on the Fig 4.3. If the signature is not matching any of the existing inactive detectors signatures, it is added as new inactive detector to the maturation block. If it is matching an existing inactive detector, the status of that detector (the first that matched) is updated, by incrementing its counter C1, refreshing its time field value T1, and adding the id of that user (C1 is the counter of the bulkiness of suspicious signatures; it serves as the receptor of the PAMP signal). The same happens when a local danger signature is passed to the maturation block, the only difference is that, if matching, C2 and T2 are affected instead of C1 and T1 and DS bit is set to 1. C2 is the counter of the local danger signal signatures (it serves as the receptor for measuring the level of the received danger signal). Upon refreshing, the T2 is typically set to a much later expiration time then it is the case with T1. The same happens when a remote danger signature is received from a collaborating system, with a difference that id and DS fields (see next paragraph for the explanation of different fields) are not added and the affected fields are only C3, C4, T3, T4. Local suspicious and danger signatures are passed to the maturation block accompanied by id value, and remote danger signatures do not have the id value but have its own C3 and C4 fields set to real number values (could be binary too), so the local C3 and C4 counters may be incremented by one or by values dependant on these remote incoming signature counters.

Whenever an inactive detector is updated, a function that takes as input the counters of this detector is called that decide about a possible activation of the detector (in the current simple implementation we use a threshold for each counter independently). If the detector is activated, it is used for checking the pending signatures of all the local users' detection blocks (1 per user). We call this recurrent detection of pending email messages. Optionally, only the detection blocks could be checked for which id is added to the detector.

Upon the activation of a detector, its signature is copied to the memory detectors databases of those users that had their id added to the detector and appropriate DS bit set to 1. Memory detectors are also assigned a life time, and this time is longer then for the activated detectors.

Whenever a new detector is added or an existing is updated by the local suspicious or danger signature, a function is called that takes as inputs C1 and C2 and decides if a signature should be sent to a collaborating system (in a simple implementation the counters may trigger the actions independently).

Both the inactive and active detectors live until all the lifetimes (T1-T4) are expired. The old proportional signatures and detectors in different blocks are eventually deleted, either because of expired life time or need to make space for those newly created.

Transforming the strings into the proportional signatures

There are several reasons and goals to transform the sampled text strings into binary representation. First, in order to preserve privacy, it is important to hide the original text when exchanging the information among the antispam systems. To achieve this we use one way hash functions when transforming text string into its binary equivalent. Second, it is important that the similarity of the strings, as it would be perceived by the reader, is kept as similarity of the corresponding binary patterns that is easy to compute and statistically confident. Similarity might mean small hamming distance, for example. "Statistically confident" means that the samples from unrelated emails should with very high chance have the similarity smaller than a given threshold, while the corresponding samples from the different obfuscations of the same spam email, or from similar spam emails, should with high chance have the similarity above the threshold. "Corresponding" means that they cover similar spammy patterns (expressions or phrases) that exist in the both emails. Third, the binary representation should be efficient, i.e. it should compress the information contained in the text string and keep only what is relevant for comparing the similarity. Last, but not least important, the binary representation should provide possibility to generate the random detectors that are difficult to be anticipated and tricked by the spammers, even if the source code of the system is known to the spammers.

To achieve the above listed goals, we design the representation based on so called similarity hashing. Our custom hashing is similar to the Nilsimsa [53], with important differences. It is illustrated on the Fig 4.4. The input is a string of the

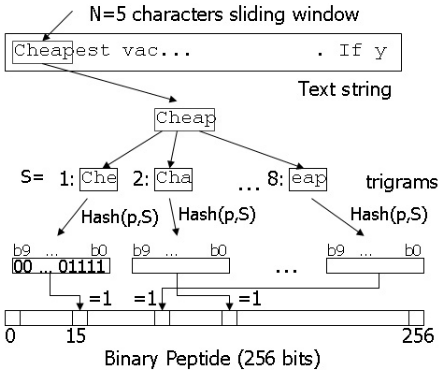


Figure 4.4: Hashing of a sampled text string into the proportional signature (also called binary peptide). Collaborating systems must use the same signature length (typically 256, 512 or 1024 bits).

fixed length (sampled at a random position from the email). The sliding window is applied through the text of the string. The window is moved character by character. For each position of the window 8 different trigrams are identified. A trigram consists of three characters taken from the predefined window positons. Only the trigrams containing the characters in the original order from the 5-character window and not spaced more then by one character are selected. Then a parametric hash function is applied that transforms each trigram into the integer from 1 to M, where M is the size of the binary representation that must be the same for all the collaborating systems. The bit within the binary string ”proportional signature” indexed by the computed integer is set to 1. The procedure is repeated for all window positions and all trigrams.

Unlike the Nilsimsa method that accumulates the results within the bins of the proportional signature, and then applies a threshold to set most populated beans to 1 and other beans to 0, we just do overwrite a bit if it is already set, i.e. we fill the proportional signature as if we would fill a Bloom filter. In the used transformation, M is determined as the smallest value that provides desirable small contention in the Bloom structure. It is important to notice that the hash function could be any mapping from the trigrams on the $1 - M$ interval, preferably with a uniform distribution of values for randomly generated text. The parameter *p* on the figure controls the mapping. Preferably, the hash function produce the same values for the trigrams containing the same set of characters, in order to achieve robustness against obfuscations that reorder letters within words.

Use of the Bloom filter rule for setting of the signature bits prevents from deleting (by text additions) the bits that correspond to the spammy patterns. Contrary, with a method like Nilsimsa it is possible to add text that will overweight the spammy phrase trigrams and prevent them of being shown up in the signature.

4.3 Evaluation

Evaluation of the system is done using a custom simulator made in C programming language. It simulates configurable number of servers and email users per server, and implements the main functions of the proposed antispam system (some details are missing, like detector timers for example, which were not important for the performed short simulations). User's behavior of sending and reading emails is simulated using a random (uniform) distribution of time between reading new emails and sending random number of emails. The recipients are chosen at random. Network delay is also a parameter (though its impact is small).

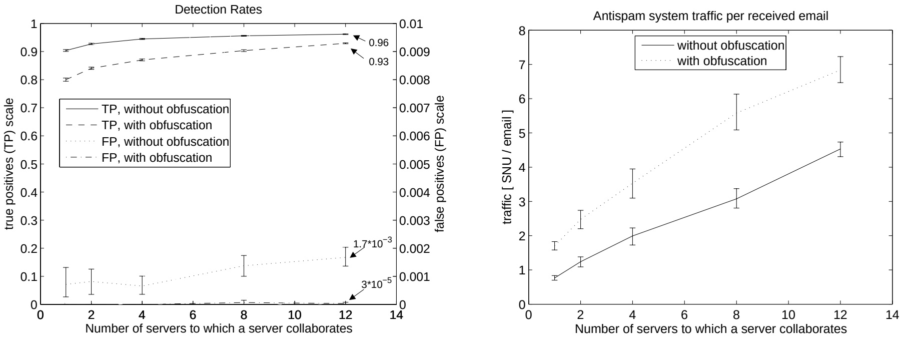
We tested how number of systems to which a system collaborates impacts the detection results. We did it for two cases: without obfuscation, when spammer sends many identical copies of the same spammy message, and with obfuscation, when spammer changes each copy from the spam bulk. We tested the system only for one obfuscation model in which letters inside words are rearranged completely randomly for each new copy (such text is still readable by humans). Spammer sends spams in bulks, as that is standard spamming-business model.

We used spam and ham (not easy or hard ham) sets from SpamAssassin Corpus [75] of emails for the evaluation. The length of simulation we used was 2h, as constrained with the number of messages from the used corpus. All the experiments are repeated 20 times (seeding the random number generator with different valuse), and the average values are computed and shown along with the 95% confidence intervals.

The default parameters of the system used in the simulations are as follows (we did not optimize parameters; we used the first set of the parameters that worked reasonably well). The length of sampled strings is 64 characters, the length of binary peptides and detectors is 256 bits. The finite life time of the detectors is not simulated. These time constants should anyway be longer then the length of the simulation we were able to perform (due to limited amount of emails in the used corpus). So no detectors were deleted during the simulation. The detectors are activated if C2=1 or if C1=2. The detectors are sent to the collaborating systems when C2=1 (by that time C2 may reach a value higher than 1; we did not try to initiate exchange based on more complicated functions of C1 and C2). In this initial testing we also do not use separate counters for C3 and C4, but instead we use C1 and C2 (respectively). The threshold for negative selection is 0.7 (possible range is 0 to 1, 0 corresponding to all bits different, and 1 corresponding to all bits equal between the compared signatures). Detection threshold is 0.85 (possible range 0 to 1).

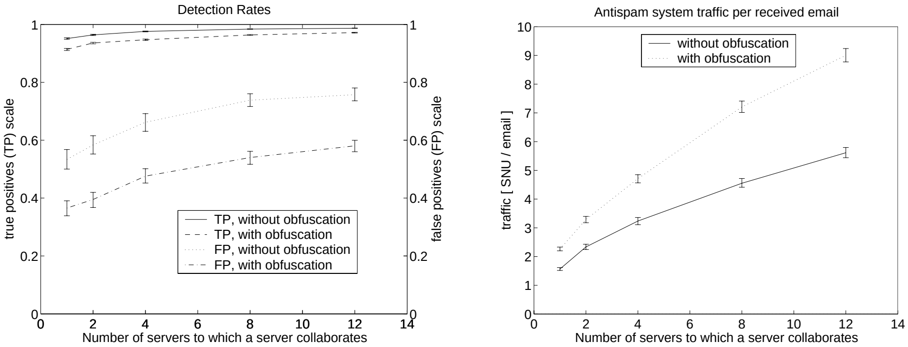
4.4 Results Discussion

From Figure 4.5(a) we can see that collaboration to up to 10 other antispam systems already gives good results, and that the systems copes well against the tested



(a) (b)

Figure 4.5: Complete AIS: (a) Detection results; (b) Traffic exchanged for collaboration. SNU stands for Standard Nilsimsa Unit = 256 bits, i.e. the traffic is measured relatively to the amount of traffic produced by a Nilsimsa query to centralized database upon receiving an email.

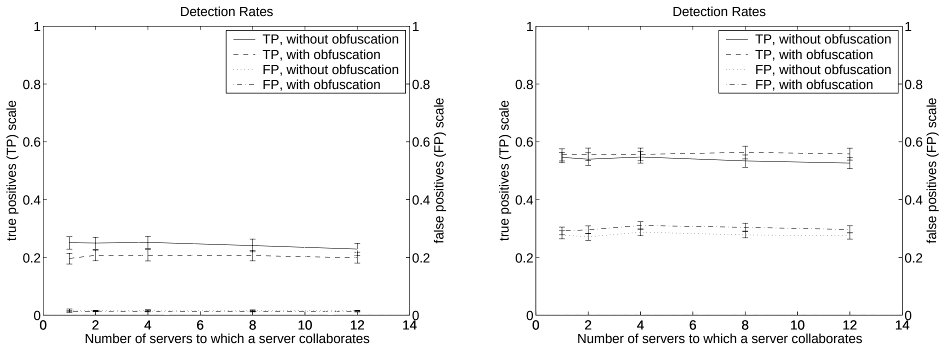


(a) (b)

Figure 4.6: Innate AIS used without adaptive AIS part (negative selection turned off): (a) Detection results; (b) Traffic exchanged for collaboration.

obfuscation. We were surprised that False Positives is bigger with non-obfuscated messages, but we found that this detections happen with detectors that correspond to header fields of emails. This can be explained with the fact that we did short simulations and during that time self examples are still not learned well as number of good email examples we start with is limited. Obfuscation of messages lessen this artifact. We expect that this artifact will go away in longer simulations and with larger initial number of emails in Inboxes.

From Figure 4.5(b) we can see that the traffic created by an antispam system upon receiving an email is only few times larger then for making one nilsimsa database query, which is very moderate usage of the traffic. Upon inspecting number of created candidate detectores and number of exchanged detectors, we found that less than 10% is is exchanged from those created. This is due to the control done by negative selection and maturation processes that put away the detectors



(a) (b)

Figure 4.7: Adaptive AIS used without innate part (PAMP signal and danger signal turned off): (a) Standard value (0.85) of the detection threshold; (b) Decreased value (0.7) of the detection threshold.

that correspond to the parts of the emails that are likely to be normal text (or obfuscation component that is usually random) and allow the collaborating systems to concentrate on further processing of the suspicious patterns.

From Figure 4.6 we can see that the negative selection is crucial for proper work of the system. If it it turned off, the false positives become to high. This is inline with the findings of Chapter 6 that analyzes the effect of negative selection on a basic collaborative detection scheme. As expected, turning off the negative selection increases the collaboration traffic (the additionally exchanged signatures are actually those that pollute the system and increase false positives).

From Figure 4.7(a) we can see however that use of the negative selection alone, i.e. when the innate signals (PAMP and danger signal), does not produce enough good detectors and spam detection is very low (bulkiness and feedback from users are not exploited). Decreasing the detection similarity threshold does not help, as it causes increase in the false positives (Figure 4.7(b)).

4.5 Conclusion

The initial evaluation shows that the system achieves promising detection results under modest collaboration, and that it is rather resistant to the tested obfuscation. Both the innate and adaptive parts are crucial for a good detection results.

The very promising true and false positive detection results become more intuitive, as well as a possible placement of our AIS within the antispam solutions employed in practice, when our AIS is compared to other antispam techniques that use similar mechanisms. We recall here that our AIS incorporates an evaluation of the spam bulkiness (in our AIS terminology this is an innate mechanism called PAMP signal) and that it incorporates the feedback from the email users about unfiltered spam (in our AIS terminology this is an innate mechanism called danger signal). It also takes into account the (dynamically built) content profiles of the users, by use of the negative selection algorithm (an adaptive AIS mechanism), applied in a non-standard way in the process of creating antibodies (spam detectors).

If we look at other antispam techniques that use similar mechanisms, the situation is as follows. Other AIS-based antispam solutions, such as the work of Oda and White [55, 56] and of Bezerra et al. [4], use adaptive AIS algorithms and sometimes incorporate feedback from the users but do not exploit bulkiness of spam. This could be the reason our AIS outperforms the AISs of Oda and White and of Bezerra et al., in both the true and false positive detections. The numbers should however be taken with some reservation, due to unidentical experimental settings. Whereas, the classic collaborative detection schemes like those presented by Damiani et al. [11] and by Zhou et al. [84] do exploit the spam bulkiness, but they do not benefit from the possibility of using the negative selection or a similar mechanism[1].

The antispam solutions employed in practice achieve good performance usually by using independently multiple local processing techniques (e.g. Bayesian or neural networks based; the above mentioned AISs could be incorporated as well) and network based techniques (IP-address based black and white lists, previously mentioned collaborative content-based techniques, social network based methods), and then combining the results obtained from each technique. Some of these techniques also use the feedback from the users. Although it is clear (from the results of Chapter 6) that the antispam solutions used in practice can benefit at least from equipping the classical bulkiness evaluation with the negative selection mechanism, the promising results obtained in Chapter 4 for our AIS that integrates processing of the multiple spam aspects (content, bulkiness, user feedback) suggest that the AIS could potentially be a good replacement for multiple techniques that separately exploit these spam aspects.

The use of the artificial immune system approach to represent observed and processed information (suspicious signatures) and a learned state (signatures that became active or memory detectors) enables efficient information exchange among collaborating systems. Only the relevant (locally processed) and independent small units of information (the proportional signatures) need to be exchanged to achieve a distributed and collaborative detection of email spam. These units are independent and contain a *summarized partial information* observed and processed by a local system. With the use of classical neural networks as a competitive approach, the locally learnt information specific to one input pattern is distributed among many links of the local neural network and only the classification output from the local network is available - which does not allow for the simple exchange of summarized information learned locally from multiple strongly correlated observed input patterns.

[1] ”Negative assertions” used by the Cloudmark’s collaborative filtering scheme (summarized in Section 4.1.1) is the only mechanism we are aware of that is somewhat similar to the negative selection; there are however fundamental differences in the details and provided functionality between the ”negative assertions” mechanism and our use of negative selection, as discussed more in detail in Section 4.1.2

An important point that remains to be evaluated experimentally is the *dynamic* of the response to a new spam bulk by the network of antispam systems. According to the design, we know that during a response to a new spam bulk more resources are used, and that upon creating enough detectors in response to this spam bulk and distributing them within the antispam network, all (or huge majority) of the collaborating users are protected from the remaining spams from that bulk and from repeated similar spams (note that this is very similar to the inflammation and win over a virus by the human immune system). So, it is important to determine the resources needed and the ability and limits of the system to cope with spam under "stress" spamming conditions, when maybe the goal of the attacker is not only to get spam through into the Inboxes, but also to defeat the antispam system(s) (and its reputation) by putting it out of its normal working mode. That is serious reason for trying to understand and control the mechanisms that start and stop the "inflamation" (reaction to spam bulks) in the network of antispam systems.

Chapter 5

Data Representation in AIS-CSD

In this chapter we investigate the representation of email content used by the AIS from Chapter 4. The used representation is a modified version of the representation used in a well known classic collaborative detection scheme. Evaluation of the original representation has been published in a paper that is often cited in the related antispam literature. This representation have been used in some important real world antispam solutions.

In particular, we investigate and compare the resistance of the two representations to different amounts of obfuscation under a chosen realistic spammer model. Our goal is to understand how the representations behave under the obfuscation, and to justify their advantages and disadvantages. As an example of a realistic spammer model, we use addition of random text, but we also comment on expected behavior under other realistic spammer models.

5.1 Introduction

5.1.1 Background on Collaborative Spam Detection Using Similarity Digests

An important feature of spam, which can be exploited for detecting it easier, is its bulkiness. A spam bulk mailing consists of many copies of the same original spam message, each sent to a different recipient or group of recipients. The different copies from the same bulk are usually obfuscated, i.e. modified a bit in order to look different from each other. Spammers apply obfuscation in order to make collaborative spam detection more difficult.

Indeed, in collaborative spam detection it is important to have a good technique for determining which emails belong to the same bulk. This allows, after observing an initial portion of a bulk, for the bulkiness scores to be assigned to the remaining emails from the same bulk. If the collaborative spam detection is based purely on the evaluation of bulkiness, each recipient must be equipped with a white lists of all the bulky sources from which she or he wants to receive emails.

Having a good technique for determining which emails belong to the same bulk also allows for the individual evidence of spamminess to be joined, if such evidence is generated by collaborating filters or users for some of the emails from an initial portion of the bulk. The observed bulkiness and the estimated spamminess of a bulk can then be used to better filter the remaining emails from the same bulk. Collecting and using the evidence of spamminess is especially useful if the reputation of spam reporters is evaluated and used, in which case the collaborative detection may be relatively safe to use even if the recipients are not equipped with white lists of the bulky sources from which they want to receive emails.

A good source of the evidence of spamminess, which is increasingly used in practice, are the emails tagged as spam by those users that have and use a ”delete-as-spam” button in their email-reading program. Automated and probabilistic tagging is also possible, e.g. by use of Bayesian filters' scores, or by use of ”honey pot” email accounts that are not associated to real users but only serve to attract unsolicited bulk emails.

5.1.2 Importance of Digest-Based Collaborative Spam Detection

There are two big classes of techniques for collaborative communication filtering in the Internet, including the email filtering. One class is based on protecting (e.g. white-listing) the good sources of communication. It exploits reputation of users or source IP addresses. Examples are PGP certificates, reputation of IP addresses, or reputation of domains accompanied by use of domain authentication. However, this class only protects the communications that has been already learned to be good.

The second class of collaborative email filtering consists of collaborative detection of bad email sources and collaborative detection of bad (spam) content.

In practice it is usually necessary to accept new communications from the users or sources for which such a reputation is not yet built, i.e. there is need for accepting and learning ”new” good communications. Regarding email, IP black lists block most of the spam already at the connection level. The white and black lists are usually applied first as they are both fast and allow pre-filtering the email stream already at the connection level, which is good for lowering usage of the communication resources. However, due to the use of dynamic domains and botnets by spammers, significant amount of spam is not blocked by black lists, and use of content based filtering techniques is still very useful.

Collaborative detection of bad content is especially useful for detecting and blocking bulk spam (most of spam is sent in bulk). It allows for relatively early detection of new bulks that contain not previously observed spam message (phrase).

5.1.3 Existing digest-based approaches

A well-known technique for detecting whether emails belong to the same spam bulk is presented and evaluated in the ”OD-paper” by Damiani et al. [11] (OD stands for Open Digest). OD-paper is often cited in the literature related to digest-based collaborative spam filtering, as it gives very positive results and conclusions on the resistance of the technique to the increased obfuscation-effort by spammers. It also shows that the technique is expected to have very low false-positives.

The technique produces similar digests out of similar emails, and uses them to find out which emails belong to the same bulk. The digests are produced from the complete email or from the complete predefined parts of the email. The digest queries are submitted to a global database, and the replies indicate the number of similar messages (queries) observed by the database. The details on producing the digests and similarity matching between the digests are explained in 5.1.4. It is important to mention that such a technique is implemented by DCC [13], and that the DCC database of digests is used by SpamAssassin [74] (a very popular open-source antispam software integrated in many spam filters).

The peer-to-peer system for collaborative spam filtering by Zhou et al. [84] is another well-known and often cited digest-based antispam technique. It uses multiple digests per email, created from the strings of fixed length, sampled at random email positions. They apply however the exact matching instead of a similarity matching between the digests, as required by the rest of their system to work. Even modest spam obfuscation is able to alter some of the bits of such generated digests, which prevents their system from detecting spam bulks. Their analysis results in a different conclusion, because they use rather unrealistic obfuscation (which alters the created digests with a very small probability) to test their solution.

The system that we propose in the previous chapter produces multiple digests per email, from the strings of fixed length, sampled at random email positions, and it uses similarity matching. Additionally, it uses artificial immune system algorithms to process the digests before and after exchanging them with other collaborating systems, in order to control which digests will be activated and used for filtering of the incoming emails. The system shows good performances in detecting spam bulk under a specific spammer model, but an additional evaluation is needed for more general conclusions about its abilities. As the factorial analysis is missing, it is not clear whether the observed good performances are due to the way the digests are produced (e.g. as compared to the standard digest from the OD-paper [11]), or due to the advanced algorithms used by the system.

The direct comparison of the above explained different ways of producing the digests from emails, according to our best knowledge, has not yet been scientifically evaluated.

5.1.4 Nilsimsa Hashing For Determining Similarity Between Emails

Single digest per email. The open-digest technique from the OD-paper represents an email by a 256-bits digest. The transformation is performed using Nilsimsa hashing [53]. This is a locally sensitive hash function, in sense that small changes in the original document may impact only few bits of the digest. That means that similar documents will have similar digests, in sense of a small Hamming distance between them. With the standard hash functions small changes in the original document usually result in a digest that is completely different from the digest of the original document.

Nilsimsa Hashing. OD-paper gives a detailed description of the Nilsimsa hashing. In summary, a short sliding window is applied through the email. For each position of the window, the trigrams from the window are identified that consist of the letters from the predefined window positions (that are close to each other, but not only consecutive-letters trigrams are used). The collected trigrams are transformed, using a standard hash, to the positions between 1 and 256, and the accumulators at the corresponding positions are incremented. Finally, the accumulators are compared to the mean or to the median of all the accumulators, and the bits of the digest are set to 0 or 1, depending on whether the corresponding accumulators are bellow or above the threshold.

Such digest are often called ”open digests” because: a) the digests computation method is assumed to be publicly known; b) the used similarity hashing hides original email text, so the privacy of the content is preserved even if the digests are openly exchanged for collaborative filtering.

Nilsimsa Compare Value (NCV) between two digests is defined to be equal to the number of the equal bits at the same positions in the two digests, minus 128 (for the digests of 256 bits). We use NCV as the measure of the similarity of the two emails from which the two digests are produced[1]. The higher NCV indicates the higher similarity of the texts from which the digest are computed. NCV of two random unrelated text-strings is random and centered around 0. Distribution of NCV for two readable (meaningful) but unrelated texts from the same language is slightly shifted to the right (in our examples centered around 30). If the texts are related (contain similar or identical parts) NCV takes values considerably higher then 30, and if the texts are completely identical NCV is equal to 128.

Alternative-digests email-to-email NCV. Our AIS for collaborative spam detection introduced in Chapter 4 uses multiple digests produced from the randomized-position and fixed-length samples. In order to evaluate similarity between two emails when multiple digests per email are used, we define the NCV between two emails to be the maximum NCV over all the pairs of the digests between the two compared emails. Such defined email-to-email NCV shows how similar are the most similar parts in the two emails.

5.1.5 This Chapter Contributions

Re-evaluation of the digests from OD-paper

We first repeat and then extend some of the open-digest paper [11] experiments, using the simplest spammer model from that paper. More precisely we re-consider the experiments with spammer which obfuscates emails by addition of random

[1] NCV is first defined and used in open-digest paper [11]

characters. We find that some of the most important conclusions of the open-digest paper are rather miss-leading.

Evaluation of the alternative digests

For fairness of the comparison between the digests from OD-paper and the alternative digests, we evaluate both cases for the same simple detection algorithm (digest queries to a common database) that was the basis for the original OD-paper experiments.

We show that the alternative technique greatly improves the resistance of spam detection against increased obfuscation effort by spammers, while keeping misdetection of good emails at a similar level. Based on the observed results, we discuss possible additional modifications and algorithms that could be added on top of the modified digest technique to further improve its filtering performance.

5.2 Revisiting Results and Conclusions of OD-paper

OD-paper assumes use of a database of digests from ham and spam emails, e.g. created out of those emails that are observed recently in the emailing network. It compares the emails to be filtered to the emails from the database. As good emails are unrelated to each other and to spam emails, their digests are expected to match with the digest from the database with a small probability. On the other side, the digests from spam emails should with a high probability match the digests in the database that come from the same spam bulk. Spam digests are also expected to not match many of the digests that come from other emails and other bulks. Therefore, the evaluation metrics must be slightly differently computed for evaluating the detection of bulky spam emails then for evaluating the misdetection of good emails. Some possible evaluation metrics and the used evaluation metrics are discussed in Section 5.2.2. The performed experiments and the computed evaluation metrics are detailed in Sections 5.2.3-5.2.5.

5.2.1 Considered Spammer Model

OD-paper evaluates the spam detection technique explained in Section 5.1.3 against few spammer attacks: addition of random characters, aimed addition of characters that takes into account the details of how the digests are produced, replacement of words by synonyms, and perceptive substitution of characters.

In this chapter we do all evaluations using the first spammer model (addition of random characters).

5.2.2 Metrics Used to Evaluate The Open-Digest Technique From The OD-paper

To assess the ability of spam bulk detection and good email misdetection, as the first option, one could simulate the real scenario of submitting the digests of the emails to be filtered to a database, and receiving back the counters of how many digests in the database is matched by the submitted digests. And then comparing the counters to their ideal values (0 - for ham emails; number of earlier emails from the same bulk - for spam emails).

The second option would be to do email-to-email comparisons and estimate the probabilities of matching between unrelated emails (e.g. between a ham email and emails from the database), and between related emails (spam emails from the same bulk). Then, knowing a possible size of the digest database, one could calculate the probabilities of the counter values returned by the database upon a digest-query.

The OD-paper uses the second option for estimating false detection of good emails. It also uses the second option to evaluate detection of spam bulk, but instead of showing the probabilities of email-to-email matching it shows the average of the Nilsimsa Compare Values between the compared digests.

Computing the average NCV makes sense only if spam emails are compared only to the spam emails from the same bulk. Actually, the OD-paper experiments for evaluating detection of spam bulk are done exactly that way.

In addition to repeating the OD-paper experiment that evaluates spam bulk detection in the same way as it is done originally in OD-paper, we also evaluate spam detection in the case when the queried database contains both spam and good emails, in order to make the evaluation more realistic and complete. In the mixed database case it doesn't make sense to use the average of all computed NCVs as the metric (bulk detection results would be polluted by results of NCV scores against unrelated ham and spam emails). Instead, we compute average of the *maximum NCVs* each spam email scores against the emails of the database (as a simple and logical alternative). In parallel, we also evaluate *email-to-email matching probabilities*, and we also show the *histogram of email-to-email NCVs* (for these two metrics, in the case of spam-detection evaluation, we account only for comparisons against the emails from the same bulk - otherwise bulk-matching results would be masked by unrelated-emails-matching results). Another possibility would be, for example, to count the number of the NCV scores above a given threshold - in which case it would make sense to account for the comparisons against all the emails from the database (similar to the ”maximum NCV” case).

As it is shown in the rest of this chapter, use of adequate metrics is important for correct and reliable assessing of the properties of the evaluated technique from the performed experiments.

5.2.3 ”Spam bulk detection” experiment ($SPAM - SPAM\_BULK$)

This section gives the details of the experiment in which we reproduce the result from the Figure 2 of the OD-paper (spam bulk detection under the”random addition” attack).

The $SPAM - SPAM\_BULK$ experiment:

- 20 emails are sampled randomly from the spam database (we use 20030228_spam_2.tar.bz2 spam repository[2] from the Spamassassin public corpus [75]);
- a pair of obfuscated copies is created form each of the 20 emails (random characters are added at the end of the email, amount being a ratio of the original emails size);
- the digests of the two copies from a pair are compared to each other by computing the NCV (Nilsimsa Compare Value[3]).
- mean and 95% confidence are calculated out of the 20 NCVs.
- the above steps are repeated, except the first one, for other values of the ratio of added characters (using also the values larger from those evaluated in the OD-paper).

The results of this experiment are shown in Figure 5.1.

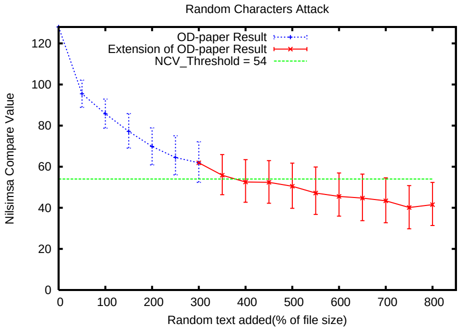


Figure 5.1: **Repeated and extended OD-paper result for bulk spam detection under the ”adding random text” spammer model.** Repeated OD-paper experiment (dotted blue line) recreates pretty well the curve from the Figure 2 of the OD-paper. Extension of the same experiment (solid red line) for higher values of the percent of added random text (as compared to non-obfuscated spam) indicates that the open-digest technique actually is vulnerable to random text additions - opposite to the conclusion from the OD-paper.

Parameters and results discussion

NCV comparision threshold. The used NCV comparison threshold is the same as in the OD-paper(the dashed green line of Figure 5.1). The OD-paper authors

[2] OD-paper used the spam repository from SpamArchive (www.spamarchive.org) which is not any more available on the Internet. We also were not able to obtain it from the authors of OD-paper. Thus we decided to use a SpamAssassin repository.

[3] See footnote 2

calculate and suggest the value 54 as the value that should ensure low misdetection of good emails (this is discussed more in detail within the experiment dedicated to the evaluation of the misdetection of good emails, Section 5.2.5).

Recovered OD-paper experiment. The dotted blue line fits very well the result from the Figure 2 of OD-paper, which suggests that we recreated the OD-paper experiment properly[4] (i.e. the NCVs in this experiment are ”spam to spam from the same bulk”, as we assumed).

Extended experiment: different conclusions. Based on the observation that the average NCV is above the matching threshold for the obfuscation ratios they tested, the authors of OD-paper claim the good resistance of the detection against increased obfuscation efforts of spammers. Our extension of the experiment shows that using even slightly higher obfuscation ratios then those tested in OD-paper brings the average NCV (solid red line) bellow the threshold, which invalidates the reasoning of the above mentioned OD-paper claim, as this small additional obfuscation effort is easy for spammers to perform.

NCV metric usability. However, though we agree that the average NCV gives some indication about the resistance of the detection to the increased obfuscation efforts by spammers, we suggest and show in the following experiments and figures that the use of additional metrics such are probability of email-to-email matching (on the level of the digests) and the histogram of NCVs allows us to much better see the qualitative and the quantitative impact of the obfuscation to detection of spam and misdetection of good emails.

5.2.4 ”Spam bulk detection” experiment ($SPAM - DB$)

This section gives the details of the experiment in which we test the digests of spam emails against a database of the digests of both spam and good emails. Such database of digests is used in the experiment done in OD-paper for evaluating false-positives (comparison of good emails to the database of digests), which means that is should also be used for evaluating the true-positives, because in reality there is only one database - in reality the filter doesn’t know in advance (of course) whether a newly received email is spam or ham. However, we can define some of the evaluation metrics to consider only the comparisons of the emails from the same bulk, and we do so.

The $SPAM - DB$ experiment:

- a 100-emails database (DB set) is created by sampling randomly 50 spam emails from 20030228_spam_2.tar.bz2 SpamAssassin’s spam repository[5] and

[4] The OD-paper does not specify this experiment in detail, and we were not able to obtain the original code of the OD-paper experiments from the web or from the OD-paper authors.

[5] See footnote 3

50 ham emails from 20021010_easy_ham.tar.bz2 SpamAssassin's ham repository[6]; whenever an email is sampled, it is removed from the original database.

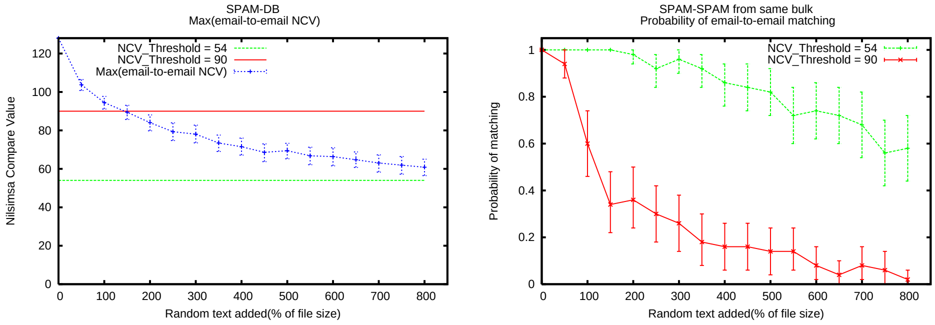
- the 50 spam emails from DB set are copied into another set (set of spam ”to be checked”, i.e. the SPAM set)
- the 100 spam emails from DB and SPAM sets are obfuscated by adding random characters to the end of email in the amount which has a specified ratio to the amount of characters in the original email
- the 100-emails database DB is converted into a 100-digest database (DB' set), by producing a digest from each email;
- for each of the 50 emails from the SPAM set the following step is performed;
- the digest of the email is compared to all the digests from DB', the NCV is computed for each comparison, the maximum NCV is computed over the comparisons, NCVs obtained for the comparisons to the spam emails that origin from the same original spam email (i.e. belong to the same spam bulk) are compared to the threshold (54), and the number of matched emails (when $NCV >= 54$ there is a matching) is divided with the number (lets call it n) of the emails in DB that belong to the same bulk, in order to obtain an estimate of the probability for that email to match other spam email from the same bulk (in our case the estimated probability will be a binary number, because n is equal to 1, due to the experiment design).
- mean and 95% confidence interval are calculated out of the 50 results obtained for the SPAM emails (for each spam from SPAM set we obtained a maximum NCV and an indicator (0 or 1) of matching to another email from the same bulk)
- the above steps are repeated, excluding steps 1 and 2, for other values of the ratio of added characters.

Parameters and results discussion

The obtained mean and CI for the maximum NCV are shown, in function of the obfuscation ratio, in Figure 5.2(a). The probability of matching between a spam email from SPAM set - and an email from DB that is from the same bulk - is shown in Figure 5.2(b) (mean and CI, in function of the obfuscation ratio), for the two values of the matching threshold. The histogram of spam to spam from the same bulk NCVs is shown in Figure 5.3(a) (for non obfuscated spam emails) and in Figures 5.3(b)-5.3(d) (for slightly, moderately and heavily obfuscated spam emails).

Max(email-to-email NCV) metric. It should first be well understood that the the means and CIs of the maximum NCVs (Figure 5.2(a)) that the spam-email digests scored against the digests from the database is practically a very similar

[6] OD-paper used the legitimate messages from comp.risks (www.usenet.org) which are not any more available. We also were not able to obtain them from the authors of OD-paper. Thus we decided to use a SpamAssassin repository.

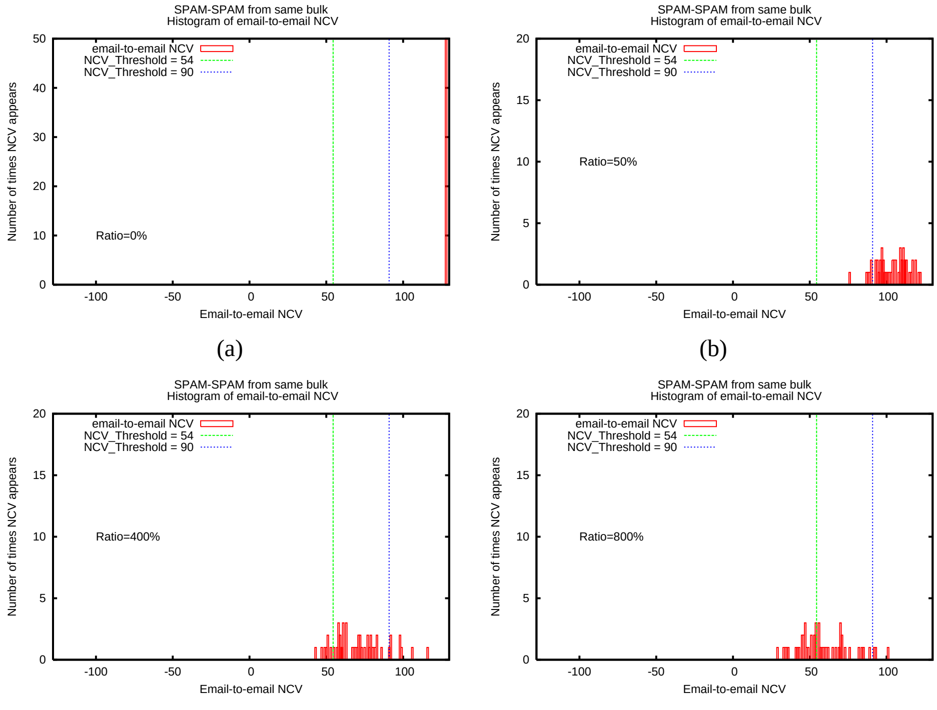


(a) (b)

Figure 5.2: OD-paper digest technique: Bulk spam detection under the ”adding random text” spammer model. The case in which the database DB of digests contains both spam and ham. We see the technique is vulnerable to the increased effort by spammer, but the impact of the increased obfuscation can not be correctly concluded from simple extension of the OD-paper experiments and by looking at only the averaged NCVs. The probability of email-to-email matching is much more informative and allows for a more proper qualitative and a more detailed quantitative assessing of the performances of the detection scheme. The observed probability of email-to-email results suggest that the increased obfuscation will substantially increase the number of emails from the bulk that pass the users' filters before the critical number of the digest from the bulk is collected in the digests database, especially for the NCV compare limit values that are noticable higher then 54 (e.g. for 90). It is shown in the next experiment (Section 5.2.5) that for a low misdetection of good emails NCV compare limit value must be much higher then 54.

metric to the means and CIs of NCVs shown in Figure 5.1 (the recreated and extended OD-paper result). We recall that in the $SPAM - SPAM\_BULK$ experiment (Figure 5.1) the spam-email digests are compared only to the digests from the same spam bulk, and the mean observed values are shown in the figure. Here, in the $SPAM - DB$ experiment (Figure 5.2(a)), the spam-email digests are compared to the digest from the database, and the database digests come from good emails, unrelated spam emails, and spam emails from the same bulk. The $Max()$ operator will in most cases select those NCVs that spam emails scored against spam emails from the same bulk, because the unrelated emails usually have much lower mutual NCVs.

However, we can notice that the NCV curve on the Figure 5.2(a) has higher values than the NCV curve on the Figure 5.1. The difference is especially evident for higher obfuscation ratios. The reason for this difference is a relatively wide distribution of NCVs even between unrelated emails (which can be seen e.g. from the Section 5.2.5 experiment, Figure 5.5). So, for high obfuscation ratios, for which the mean NCV between spam emails from the same bulk becomes low, there is a high chance that some spam emails will score a higher NCV value against one of many unrelated emails from the database will (then that NCV value gets picked up by The $Max()$ operator), then against just one considered obfuscated copy from the same bulk. This suggests use of higher NCV threshold values, in order to eliminate or minimize the effect of the mentioned ”wide NCV distribution” phenomena. The mentioned ”wide NCV distribution” phenomena is even more relevant for misdetection of good emails and is additionally discussed in Section 5.2.5.



(a)

(b)

(c)

(d)

Figure 5.3: OD-paper digest technique: Impact of increased obfuscation by spammer on mutual matching of emails from the same bulk. The increased ratio of added random text renders most of the digest to become not useful for spam bulk detection. NCV threshold value must be high to the low misdetection of good emails constraint (as shown in Section 5.2.5).

NCV threshold values. All the metrics are computed for the two values of the NCV threshold, 54 and 90. The NCV threshold 54 allows comparison of the results and conclusions with those from OD-paper. We consider one additional threshold value (90) in order to illustrate the effect of the threshold change on both detection of spam (this experiment) and misdetection of good emails (the experiment of Section 5.2.5). We pickup the second value to be much higher then 54, because in the next experiment (Section 5.2.5) we find that for the NCV threshold value 54 the misdetection of good emails is very high. In our experiments we do not optimize the NCV threshold value. This could be done by applying the standard procedure for optimizing parameters of spam filters, i.e. by finding the NCV threshold value that achieves an appropriate compromise between the misdetection of good emails and non-detection of spam.

Vulnerability to obfuscation: NCV versus probability of email-to-email matching results. If we try to even only qualitatively conclude about the bulk spam detection vulnerability to the obfuscation, by looking only at the means and CIs of the maximum NCVs that spam emails scored against the emails from the database (the comparison is on the level of the digests), we can see (Figure 5.2(a)) that for the NCV threshold 54 the CIs are always above the threshold . We could conclude that the detection is well resistant to the obfuscation. For the NCV threshold 90, the complete CI is above the threshold for the obfuscation ratio up to 100%, but the complete CI is bellow the threshold already at the obfuscation ratio 200%, and stays bellow the threshold for all higher obfuscation ratios. Following the reasoning used in OD-paper, we could (wrongly) conclude that, for the NCV threshold 90, the detection is well resistant to the obfuscation ratios up to 100%, and that it is not resistant for the obfuscation ratios above 200%.

However, if we look at the estimated probability of email to email from the same bulk matching (Figure 5.2), we can have more correct qualitative and even very good quantitative conclusions about the detection efficiency and resistance to the obfuscation. For the NCV threshold 90, the probability of matching is effectively rather similar for the obfuscation rations 100% and 200%, in the sense that in both cases only few digests from a bulk in the database would ensure high probability that at least one, and actually a large portion of them of them, match the new digests from the same bulk. From the probability of email to email detection results it is possible to compute the number of the the digest from a spam bulk that have to be collected in the database in order to achieve a specified high probability for the new digests from the same bulk to match a specified number of the digests from the database (requiring more than one match may be needed in order to achieve low misdetection of good emails).

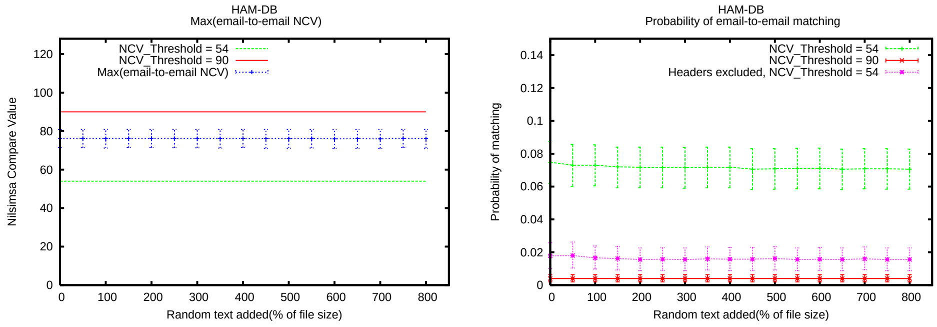
Already visually and without detailed computation we can see that stronger obfuscation will not completely prevent detection of spam, but will substantially increase the number of spam emails from a bulk that will bypass the detection.

Obfuscation under x-ray: NCV histogram. Figures 5.3(a)-5.3(d) show visually what happens with the digest under the obfuscation of spam emails. Strong obfuscation renders most of the digests from a bulk to become not-useful for matching other digests from the same bulk (the NCVs bellow the threshold). One should be aware that only high NCV threshold values are acceptable for practical use, as required for low misdetection of good emails (as shown in Section 5.2.5).

5.2.5 ”Good emails misdetection” experiment ($HAM - DB$)

This section gives the details of the experiment in which we test the digests of good emails against a database of the digests of both spam and good emails. The same experiment is done in OD-paper for evaluating false-positives. Only the used email repositories are different, and the number of the compared emails is different, which should normally not affect the results (due to the experiment design), except for one of the used metrics that is dependant on the number of compared emails.

The $HAM - DB$ experiment:



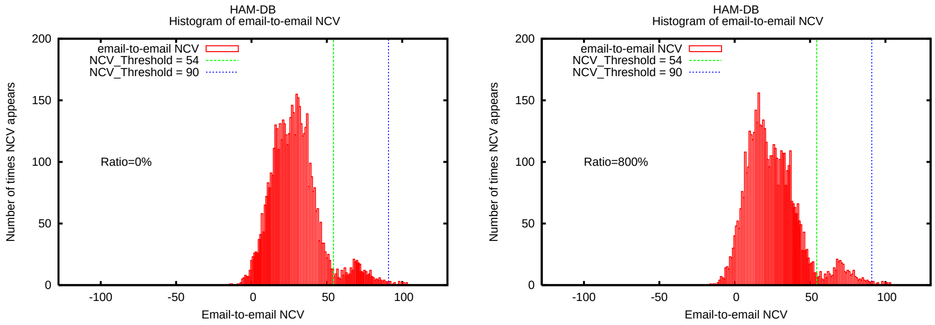
(a) (b)

Figure 5.4: OD-paper digest technique: misdetection of good emails under the ”adding random text” spammer model. We consider the case in which the database DB of digests contains both spam and ham digests (the same case is considered in the OD-paper). We can see that for TH=54 (the only case evaluated in OD-paper) misdetection is much higher then advocated (and partially experimentally supported) in OD-paper.

- a 100-emails database (DB set) is created by sampling randomly 50 spam emails from 20030228_spam_2.tar.bz2 SpamAssassin’s spam repository[7] and 50 ham emails from 20021010_easy_ham.tar.bz2 SpamAssassin’s ham repository[8]; whenever an email is sampled, it is removed from the repository.
- the 50 spam emails from DB set are obfuscated by adding random characters to the end of email in the amount which has a specified ratio to the amount of characters in the original email
- the 100-emails database DB is converted into a 100-digests database (DB’ set), by producing a digest from each email;
- another 50 hams (hams ”to be checked”, i.e. the HAM set) are sampled from the ham repository (whenever an email is sampled, it is removed from the repository).
- for each of the 50 emails from the HAM set the following step is performed;
- the digest of the email is compared to all the digests from DB’, the NCV is computed for each comparison, the maximum NCV is computed over the comparisons, each NCV is compared to the threshold (54), and the number of the matched emails (when $NCV >= 54$ there is a matching) is divided with 100 in order to obtain an estimate of the probability for that email to match other unrelated ham and spam emails.
- mean and 95% confidence interval are calculated out of the 50 results obtained for the HAM emails (for each ham from HAM set we obtained a maximum NCV and a probability of matching emails from DB)
- the above steps are repeated, excluding steps 1 and 4, for other values of the ratio of added characters.

[7] See footnote 3

[8] See footnote 7



(a) (b)

Figure 5.5: OD-paper digest technique: Impact of the NCV threshold on the (undesirable) matching of good-email digests to the database of digest from both spam and good emails. We can see that the NCV threshold should be set very high in order to provide low misdetection of good emails, much higher then the value TH=54 suggested by OD-paper for good detection of bulk spam. However, very high values of the NCV threshold are not possible to use as they increase the vulnerability of spam detection to the increased obfuscation by spammer (as shown in Section 5.2.4, Figure 5.2). We can also see that the obfuscation of spam has no effect on the misdetection of good emails.

Parameters and results discussion

The obtained mean and CI for the maximum NCV are shown, in function of the obfuscation ratio, in Figure 5.4(a). The probability of matching between a good email from HAM set and an email in the DB database of spam and good emails is shown in Figure 5.4(b) (mean and CI, in function of the obfuscation ratio), for the two values of the matching threshold. The histogram of email-to-email NCVs is shown in Figure 5.5(a) (for non obfuscated spam emails in DB) and in Figure 5.5(b) (for heavily obfuscated spam emails in DB).

Misdetection of good emails is much higher then advocated in OD-paper. Though we implement the same experiment as the one described in OD-paper[9,10] (and detailed here in Section 5.2.5), we do not obtain the zero false positives (misdetection of good emails) result of the OD-paper. Only the used email repositories and the number of mutually compared digests in our experiment are different then in the OD-paper experiment, but that should normally not have big effect on the results of the experiment. Even if the misdetection probability would be very small, as OD-paper experiment uses higher number of digest comparisons it should easier discover the cases of misdetection. However OD-paper states that such cases were not observed in their experiment, though in our experiment the observed ratio

[9] OD-paper does not specify whether the digests are produced from the complete emails (including headers) or only from the contents of the emails. Therefore we perform the experiment both with and without exclusion of headers, for the same NCV threshold value (54) as the one used in OD-paper, but in both cases the misdetection of good emails is non-zero, contrary to the finding of OD-paper.

[10] As elimination of headers or any other preprocessing of the emails before computing the digests is not mentioned in OD-paper, and as the headers are usually present in the the databases used for evaluation of emails, in the remaining experiments we compute the digests from the complete emails. We were not able to obtain the original OD-paper code and email repositories (see footnotes 3, 5, and 7). However, the well recovered curve from Figure 2 of OD-paper (our Figure 1) suggests that the digests of OD-paper are also computed from the complete emails.

of matching between a good email digest and a digest from the DB database is around 2% if the headers are excluded when producing the digests (the doted pink line on the Figure 5.4(b)), and it is about 7.5% if the digests are produced from the complete emails (the dashed green line on the Figure 5.4(b)), with the same NCV threshold value (54) as in the corresponding OD-paper experiment.

A possible reason for the different results could be a bug in one (or both) of the experiment implementations. However we observer the scheme under multiple metrics and find all of them consistent and logical within each experiment, as well as if we compare them between the different experiments (as discussed in the ”Parameters and results discussion” sections).

Another possible reason could be that one of the experiments uses the non-representative repository of good emails. However, among two real repositories, the one that shows that an email filtering technique might produce good emails misdetection (with a probability that is rather high) is more relevant - an email filtering technique is rather not good if it produces (high) misdetection of good emails in some cases (testing repositories), even if there are cases (testing repositories) in which it does not produce misdetection of good emails (unless it is possible to identify classes of users that correspond to the ”good”-case ham repository, so that at least these users may use it safely).

Possibility to decrease misdetection of good emails is very limited. Similar as in the previous experiments, we cannot tell a lot about the effectiveness of the filtering scheme from the NCV means and CIs (Figure 5.4(a)), as well as about the effect of different NCV threshold values. Actually, in this experiment, from Figure 5.4(a) we can at least conclude that the obfuscation does not impact the misdetection of good emails.

From the Figure 5.4(b) we can see that changing the NCV threshold from 54 to 90 decreases the observed ratio of unwanted digests matching by an order of magnitude. But the ratio achieved with NCV threshold 90 is still to big for a practical use, and the problem here is that a further increase of the NCV threshold would make the spam detection even more vulnerable at the higher obfuscation ratios (see Figure 5.2(b)).

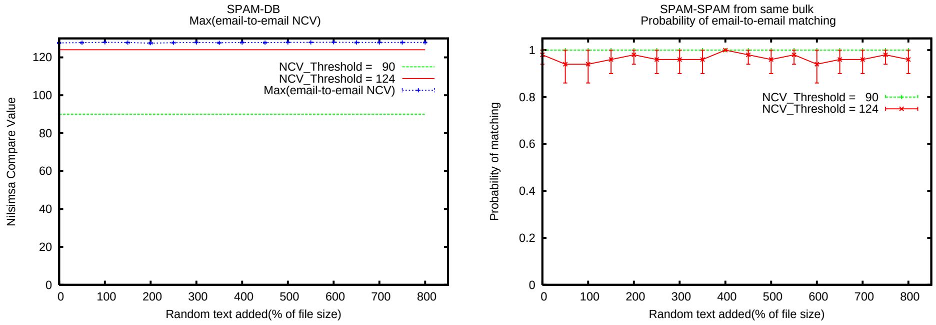
Shifted and wide NCV histogram phenomena. The NCV histograms between good emails and the emails from the DB database (Figure 5.5) show that the digests from non related emails are far from bit-wise random, as the complete histograms are shifted to the right and not centered around the zero. This comes from the fact that the trigrams from the used language are not uniformity distributed.

The digests are also not independent, as the histogram is not completely gaussian and contains an additional local maximum. The dependence might come from the fact that there are many good emails that quote other emails (or their parts) in the replies, or simply there are unrelated good emails that discuss around the same currently popular topic. Quoting is especially exhibited a lot in the discussions over the mailing lists. The dependence normally makes the histogram wider than it would be for independent good emails. A lot of the right part on the Figures 5.5(a)-5.5(b) is populated, which implies use of high NCV threshold values to ensure low misdetection of good emails, and makes a lot of the digests computed out of spam bulk emails (Figures 5.3(a)-5.3(d)) ineffective for mutual matching, especially under a strong obfuscation of spam emails.

What we can learn from the NCV histograms. The NCV histograms and the above analysis show that the assumptions used in OD-paper, for an approximate estimation of the misdetection probability by use of the Binomial probability distribution ($B(n, p)$, with n=256, and p=0.5), are far from being even approximately correct.

The above histogram analysis also gives some suggestions on how the conflict between the low misdetection of good emails and good detection of spam bulk requirements could possibly be lessen. One suggestion, in a search for a way to ensure a smaller overlap between the HAM-DB NCV histograms and the SPAM-SPAM_BULK NCV histograms, is to try to use longer digests (e.g. of 512 bits, instead of 256 bits).

Another suggestion, which we actually evaluate (Section 5.3), is to try to make the digests more specific by computing them not from complete emails (with or without headers), but from smaller email parts, e.g. from the strings of the constrained length sampled from the emails.

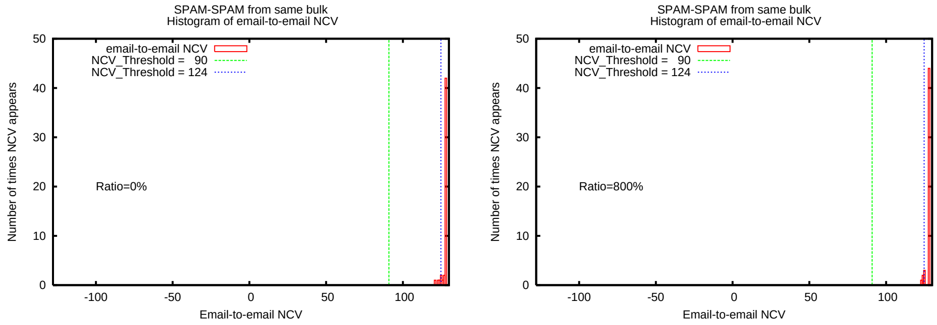


(a) (b)

Figure 5.6: Alternative digest technique: Bulk spam detection under the ”adding random text” spammer model. We can see from (b) that the spam bulk detection is very resiliant to the increased obfuscation effort by spammer. We can see from (a) that, similar as in the previous experiments, the average NCV is not a very informative metric.

5.3 Alternative to the original open-digests

As it is demonstrated in the previous experiments, the original open digest technique is vulnerable to a simple obfuscation by spammer. In this section we consider use of digests that are created from the strings of fixed length, sampled from an email at random positions. The constrained length of the samples from which the digests are produced should make it more difficult for spammers to easily ”hide” spam text by adding a lot of random text into the email. Sampling strings at random



(a) (b)

Figure 5.7: Alternative digest technique: Impact of increased obfuscation by spammer on mutual matching of emails from the same bulk. A random pair of emails from the same bulk match each other (with a high probability, as shown in Figure 5.6(b)) through the digests that are very specific to each other, and what is the most important the effective digests stay very specific even in the presence of very strong obfuscation under the considered spammer model. This allows for setting of the NCV threshold to high values, as needed for low misdetection of good emails.

positions should make the digests less predictable by the spammer, which should make aimed attacks (analyzed in OD-paper) less efficient. We experimentally compare the two ways of producing digests under the same conditions, i.e. assuming the same simple detection algorithm (the one described in Section 5.1.3).

OD-paper also considers use of multiple digests per email, but these are not created from strings of fixed length, and are not randomized. They are created the same way as in the single hash case, but only the different hash functions are used for each (the set of the used hash functions is assumed to be fixed and globally known).

5.3.1 Sampling strings and producing digests

The sampled strings are 60 characters in length, which looked to us as a good compromise between covering the spam phrases, and not giving to the spammer a lot of space for obfuscation. The initial string is sampled starting at a uniform random position between 1 and 30. Each new string starts at position increased by 30 plus a random number between 1 and 30. We have chosen the parameters intuitively and didn't optimize them.

The digests are produced out of the sampled strings using the same Nilsimsa similarity hashing as in the single digest experiments.

5.3.2 Email-to-email comparison

We keep the same experiments as in the case with single digest, with the only difference in email to email comparison. We define, for the considered multiple digest approach, the NCV between two emails to be the maximum NCV between all the pairs of the digest (produced from the sampled strings) of the two compared emails. The goal is to score how much similar are the most similar parts in the two emails.

5.3.3 ”Spam bulk detection - new digest” experiment ($SPAM-DB$, $new\ digest$)

The experiment is the same as in the corresponding single digest case, i.e. as specified in Section 5.2.4, with the only difference in how the digests are produced and how the email to email comparison is performed (which is explained in Sections 5.3.1 and 5.3.2).

Parameters and results discussion

The results are shown in Figures 5.6 and 5.7.

NCV threshold values. We perform the experiments for the NCV threshold value 90, as it looked as a more reasonable choice between the two values we used in the single digest experiments. After observing the NCV histogram results, we find that there is a space for an additional increase of the NCV value. In order to illustrate the effect of the NCV change in the new digest experiments, we also perform them with the NCV threshold value 124. Again, we do not optimize the threshold.

Spam detection is efficient and very resistant to the obfuscation. As we can see from Figure 5.6(b), the spam detection is very efficient and resistant to the increased spam obfuscation. From Figure 5.7 we see that the digests useful for spam bulk detection became (as compared to the single digest case) very specific and practically not affected by the increased obfuscation (for the considered spammer model). As now the digests encode email parts of the constrained (actually fixed) length, they cannot be polluted a lot by the considered random characters addition attack (the behavior should be similar for the random readable (good) text addition attack).

5.3.4 ”Good emails misdetection - new digest” experiment ($HAM-DB$, $new\ digest$)

The experiment is the same as in the corresponding single digest case, i.e. as specified in Section 5.2.5, with the only difference in how the digests are produced and how the email to email comparison is performed (which is explained in Sections 5.3.1 and 5.3.2).

Parameters and results discussion

The results are shown in Figures 5.8 and 5.9.

Misdetection of good emails is similar as in OD-paper. While the alternative digests provide bulk spam detection that is more efficient and much better resistant to the considered obfuscation, the obtained good email misdetection results are similar as in the single digest case (NCV threshold values 124 and 90 in the alternative digests case correspond to the threshold values 90 and 54 in the single digest case, respectively). However the observed email-to-email matching ratio is rather too high for a practical use and needs to be further decreased.

Opened a new dimension for further decrease of the misdetection of good emails. We made the alternative digests from the relatively short strings of fixed length in order to make them more specific and able to show whether two emails contain very similar parts (e.g. a message the spammer wants to get through within obfuscated emails) or not. This helped decreasing the overlap between the HAM-DB NCV histograms and the SPAM-SPAM_BULK NCV histograms.

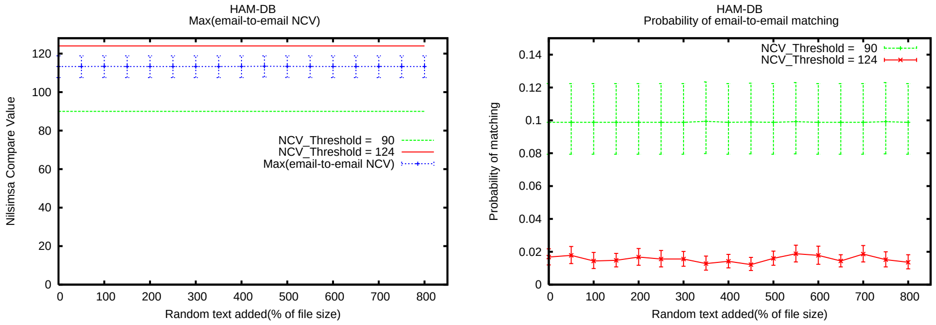
When using the alternative digests, the fact that different digests encode different parts of an email allows to further lessen the overlap, and so to lessen the conflict between the good detection of spam and low misdetection of good email requirements. This should be possible to achieve e.g. by using so called ”negative selection” AIS algorithm (see for example [45], [63]; AIS stands for Artificial Immune Systems) to first compare the digests computed from the newly received emails to a database of known good digests (”SELF” database), and eliminate those new digests that match, and only then compare the remaining digests from the new email (if any) with those in the database built through the collaborative bulk detection. In practical terms, this should directly decrease the misdetection of good emails. In the terms of the NCV histograms, this would cause most of the right part or the HAM-DB NCV histogram to disappear.

An important fact here is that, as the negative selection is applied on all new emails - so also on spam emails, the digests of spam emails that have a lot of good looking content (e.g. intentionally added by spammers) would also often be deleted, and these spam emails not detected (see ”if any” in the previous paragraph). However, with the relatively short alternative digests deleting of the digests that include the ”innocent” email patterns still leaves the high probability for survival of the digests that include the ”novel” (not known as innocent) and possibly spammy patterns, which means preserving the detection of spam bulk at a good level. The possibility to keep some and eliminate other email parts opens a new dimension within the collaborative spam bulk detection.

Examples of innocent patterns that can be pre-collected or automatically extracted and updated to the ”SELF” database are sender’s email client information and used email formatting information, which are often present in email headers. Also, it is possible to build the ham content profile of a user and use it in the negative selection, when filtering emails for that user.

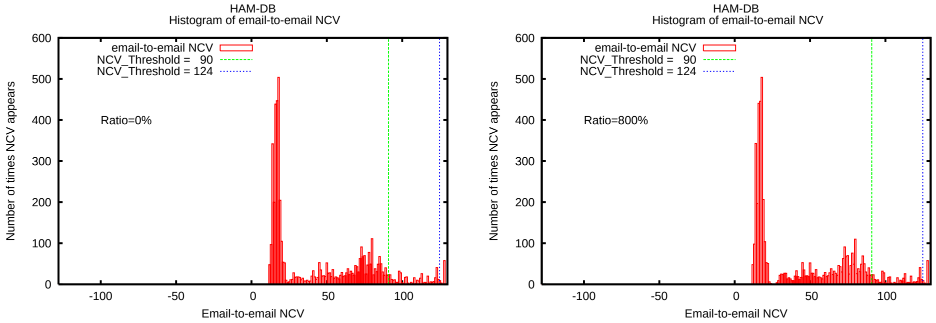
5.4 Conclusion

Improved detection resistance to obfuscation. We repeated and extend some of the open-digest paper [11] experiments, using the simplest spammer model from



(a) (b)

Figure 5.8: Alternative digest technique: misdetection of good emails under the ”adding random text” spammer model. We consider the case in which the database DB of digests contains both spam and ham digests (the same case is considered in the OD-paper for the OD-paper digest). For the values of the NCV threshold that provide good spam bulk detection (which is resilient to the increased obfuscation - this was not possible to achieve with OD-paper digest), the misdetection has very similar values to those obtained with the OD-paper digest technique.



(a) (b)

Figure 5.9: Alternative digest technique: Impact of the NCV threshold on the (undesirable) matching of good-email digests to the database of digest from both spam and good emails. We see that the NCV threshold should be set very high in order to provide low misdetection of good emails, but with the alternative digests that is possible due to good detection of spam with high threshold values. We can also see that the obfuscation of spam has no effect on misdetection of good emails.

that paper. We found that the conclusions of the open-digest paper are rather missleading. Contrary to the findings of the OD-paper, the original digest technique actually is vulnerable to the increased obfuscation that is rather easy by the spammer to perform. We also found that the misdetection of good emails is much higher then advocated (and partially experimentally supported) in OD-paper, for the NCV threshold that they propose and use (though a higher threshold is also not a good solution, as it would make the obfuscation of spam even easier). We proposed and evaluated, under the same spammer model, a modified version of the original digest technique. The modified version greatly improves the resistance of spam detection against increased obfuscation effort by spammers, while keeping misdetection of good emails at a similar level.

Alternative digests provide a good promise for further decreasing the misdetection of good emails: negative selection. Due to the property that different digests encode different parts of an email, the alternative digests look very promising regarding the possibility for inclusion of additional algorithms into the basic detection scheme considered in this chapter. E.g., so called negative selection algorithm could be added to first compare the digests computed from the newly received emails to a database of known good digests (”SELF” database), and eliminate those new digests that match, and only then compare the remaining digests from the new email (if any) with those in the database built through the collaborative bulk detection. This is aimed at decreasing the misdetection of good emails. The NCV histogram results suggest that such a technique would be much more effective with the alternative digests then with the digests as in the OD-paper (see Section 5.3.4 for a detailed explanation).

Negative selection should be quantitatively evaluated for its advocated ability to decrease the misdetection of good emails.

Other spammer models. In this chapter we did consider only one obfuscation (spammer model), the one to which the OD-technique looked very vulnerable, and we experimentally confirmed the vulnerability. While we show that the alternative digests are resistant to the considered obfuscation, they should be also tested under other obfuscation techniques, e.g. those considered in the OD-paper.

Digest length. As we already mentioned, it would be interesting to see whether use of longer digests (e.g. of 512 or 1024 bits, instead of 256) would help to decrease the overlap between the HAM-DB NCV histograms and the SPAM-SPAM_BULK NCV histograms, and so lessen the the conflict between the low misdetection of good emails and good detection of spam bulk requirements.

Similarity hashing variants. With a goal to better counter the aimed addition attack (described in OD-paper, not evaluated here), it would be interesting to investigate the effect of replacing the use of the accumulators within the similarity hashing (used to produce the digests, and explained in detail in Section 5.1.4) by a bloom-filter like procedure for setting the bits of the digest. With this procedure, the bits to which the trigrams point are set to 1, regardless whether they were set previously or not; collisions are allowed but should not be too many.

Chapter 6

Negative Selection in AIS-CSD

In this chapter we qualitatively and quantitatively evaluate the contribution of the negative selection AIS algorithm alone to the functioning of our AIS. For this, we compare the cases when the negative selection is added or not to a simple collaborative detection scheme based only on the digest queries to a common database.

We first demonstrate how the negative selection impacts unwanted random matching between unrelated emails. Then we transform these results in order to show how use of the negative selection actually impacts the misdetection of ham and detection of spam bulk. We also comment on the possible impact of different spammer models, and on applicability of the results for other detection schemes than the considered simple use of the digest queries to a common database.

6.1 Introduction

6.1.1 FP-TP Conflict In Digest-Based Collaborative Spam Detection

In content-based collaborative spam detection, usually a database of previously observed emails or email digests is formed, and queried upon receiving new emails. This allows, after observing an initial portion of a bulk, for the bulkiness scores to be assigned to the remaining emails from the same bulk. Spam will be distinguished by having ”many (more then a threshold) similar emails have been observed” score, while a ham query should score ”less then the (same) threshold similar emails has been observed”. This also allows the individual evidence of spamminess to be joined, if such evidence is generated by collaborating filters or users for some of the emails from an initial portion of the bulk. A good source of the evidence of spamminess, which is increasingly used in practice, are the emails tagged as spam by those users that have and use a ”delete-as-spam” button in their email-reading program. Automated and probabilistic tagging is also possible, e.g. by use of Bayesian filters’ scores, or by use of ”honey pot” email accounts that are not associated to real users but only serve to attract unsolicited bulk emails.

It should be mentioned that if the collaborative spam detection is based purely on the evaluation of bulkiness, each recipient must be equipped with a white lists of all the bulky sources from which she or he wants to receive emails. If trustworthy spamminess reports are used, the scheme may be usable even without the white lists.

Whether some of the previously observed emails are tagged as spam or not, for the collaborative detection to work it is necessary to have a good technique for determining similarity between the emails and find which emails (probably) belong to the same bulk, i.e. spam emails from the same bulk should ”match” each other with a high probability, and ham emails should with a high probability not match unrelated ham and spam emails.

In Chapter 5, it has been indicated and partially demonstrated that collaborative spam bulk detection by use of similarity digest may be efficient even under strong obfuscation by spammer, i.e. when the spammer modifies the different copies from a bulk in order to hide their mutual similarity from the automated tools. However, the same evaluations suggest that for the settings of the parameters that provide good matching between the emails from the same bulk, the unwanted random matching between ham emails and unrelated ham and spam emails stays pretty high. This directly translates into a need for use of higher bulkiness thresholds in order to ensure low false positive (FP) detection of ham, which implies that larger initial parts of spam bulks will not be filtered, i.e. true positive (TP) detection will not be very high (FP-TP conflict).

The above facts suggest need for use of additional algorithms (as opposed to simple counting of similar digests in the database), in order to lessen the FP-TP conflict and make the technique more useful.

6.1.2 Our Approach For Lessening FP-TP Conflict

In this Chapter we follow our suggestion from the Chapter 5 and evaluate the impact of the negative selection on lessening the FP-TP conflict (FP-TP conflict is explained in Section 6.1.1).

Use of negative selection algorithm in this case simply means that the digests from the newly received emails are first compared to a database of the digests created from a representative set of good emails (so called SELF database), and those that match any of the SELF database digests are deleted, and only the remaining digests are used for querying the collaborative database of previously observed digests (Figure 6.1).

In our experiments we form the SELF set by taking randomly given number of good emails from a part of the used ham corpus (the other two ham corpus parts are used for simulating incoming ham that is to be filtered and for simulating the digests that are created from previously observed emails and still present in the collaborative-detection database). In practice SELF could be automatically updated (e.g. the emails we send and reply to could be used to update the SELF set).

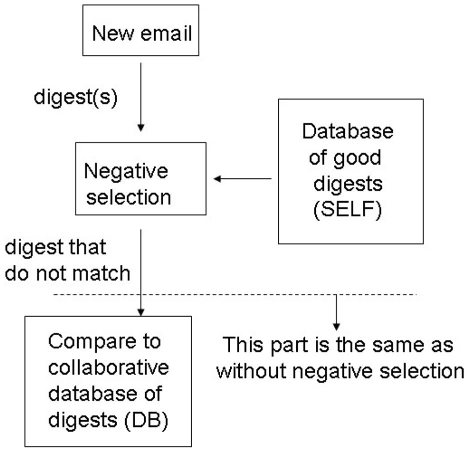

New email
digest(s)
Negative selection
Database of good digests (SELF)
digest that do not match
Compare to collaborative database of digests (DB)
This part is the same as without negative selection

Figure 6.1: Adding negative selection to the detection process

We consider only the case of collaborative spam bulk detection that uses multiple digests produced from the strings of fixed length sampled at randomized positions within email, because use of single digest per email is not resistant to increased spam obfuscation (as shown in Chapter 5). We consider the same spammer model as the one used in Chapter 5, i.e. addition of random text to the original spam message.

We perform the same experiment as the one used in Chapter 5, but now with and without addition of the negative selection. We evaluate spam bulk detection by estimating the probability for the emails from the same bulk to match each other, and we evaluate misdetection of good emails by evaluating the probability for ham emails to match unrelated ham and spam emails.

Examples of digests that could cause unwanted matching between ham and ”unrelated” emails, and that the negative selection is expected to eliminate, are the digests produced from sender’s email-client information and used email-formatting information, which are often present in email headers, or the digest produced from the interesting textual content that is being epidemically forwarded to the friends (ham examples may also be dynamically collected, e.g. from authenticated and reputable users). There are also some good phrases that may often be used in the good emails.

6.2 Experimental Evaluation Of Negative Selection

In this section we give additional details of the approach and its experimental evaluation introduced in Section 6.1.2.

6.2.1 Experiment Setup

The performed experiment (with and without use of negative selection) is illustrated in Figure 6.2.

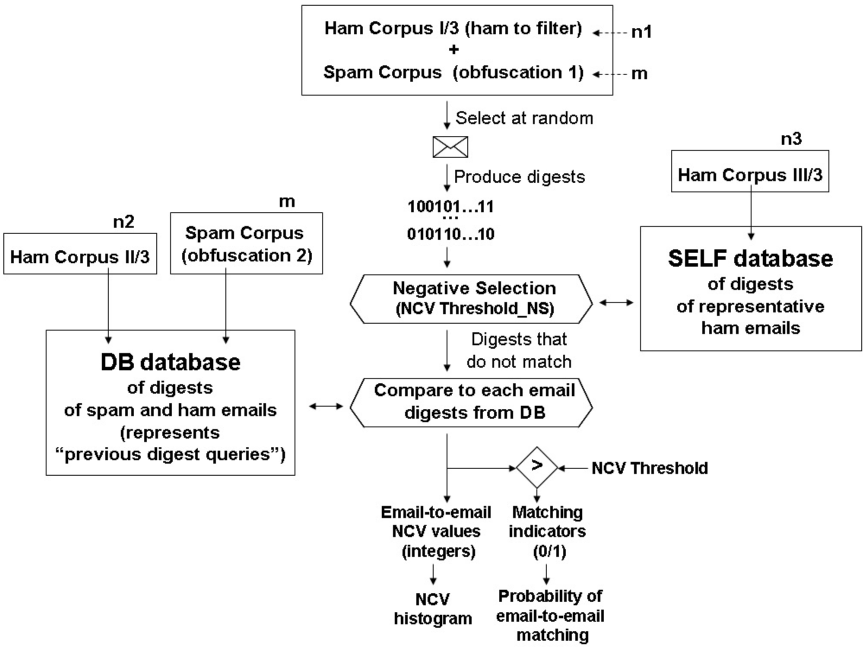


Figure 6.2: Experiment without/with negative selection. The experiment determines the probabilities of matching between ham or spam emails and the emails from the collaborative-detection database DB. If the negative selection step is used, some of the digests from some emails will be deleted before comparing the email to the database. We also look at the NCV histograms in order to better understand what happens without/with negative selection. The ni ($i = 1, 2, 3$) and m are the number of emails used in the comparisons (in this experiment these values are set to 20).

The experiment determines the probabilities of matching between ham or spam emails and the emails from the collaborative-detection database DB. More precisely, each of n1 ham emails (that simulate newly received emails) is compared to each of n2 ham and m spam emails from the collaborative-detection database DB, in order to estimate the probability of matching between newly received ham emails and unrelated ham and spam emails previously (within a time window) observed by the collaborative-detection database DB. Also, each of m spam emails is compared to one obfuscated copy in DB database that originates from the same original spam message (two obfuscated copies from the same bulk), in order to estimate the probability of spam emails to match other spam emails from the same bulk. We used $ni, m = 20$. For all estimated probabilities we also compute confidence intervals. In Section 6.2.3 we show how these estimated email-to-email matching probabilities may be translated into ham misdetection and spam bulk detection ratios.

Emails are selected randomly from the corresponding databases. Spam emails are optionally obfuscated. The comparison is on the level of similarity digests and is expressed in form of email-to-email NCV (this similarity measure is defined in Section 5.1.4).

Optionally, negative selection is applied to remove some of the email digests before comparing that email to the emails from the database DB. The negative selection is aimed at removing those digests that show tendency to match the digests from good emails, i.e. they are removed if they match any of the SELF-database digests (SELF-database digests are produced from emails known to be ham).

In all experiments we use the detection NCV threshold 90, and when the negative selection is applied we use the negative selection NCV threshold 50. We choose these values as they provide representative results, but we do not optimize them.

Evaluation Metrics. In addition to estimating *email-to-email matching probabilities*, we also show the *histogram of email-to-email NCVs* (in the case of spam-detection evaluation, we account only for comparisons against the emails from the same bulk - otherwise bulk-matching results would be masked by unrelated-emails-matching results). Histograms are useful for understanding what exactly happens with the digests and for explaining the results.

6.2.2 Results Discussion

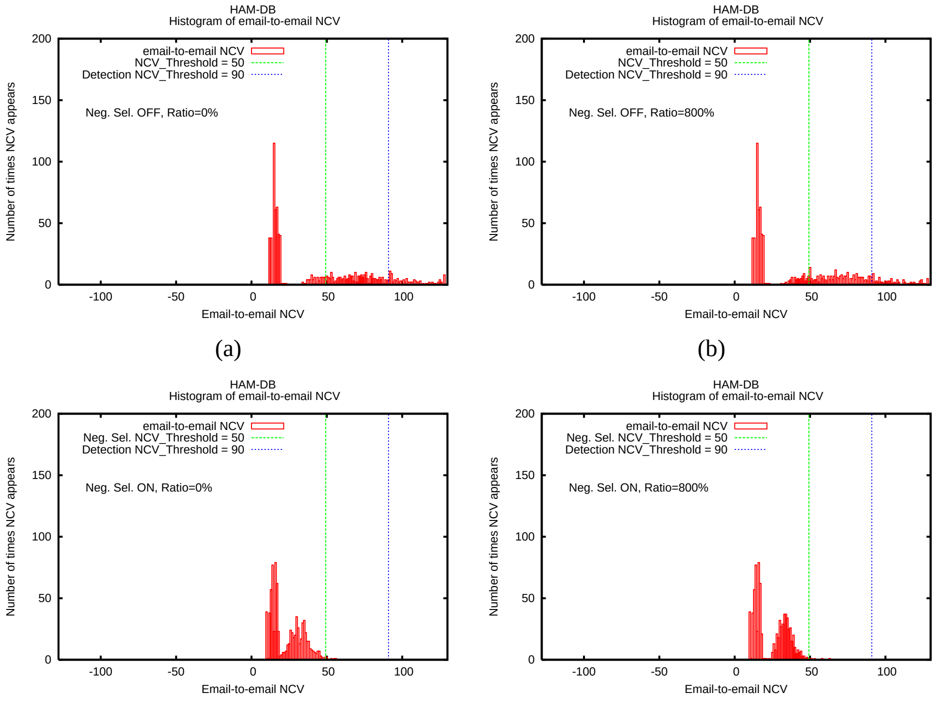
Impact of Negative Selection on Ham misdetection

From Figures 6.3(a) and 6.3(b) we can see that, when the negative selection is not used, the histogram of the NCV values between ham emails and unrelated ham and spam emails is very wide and have a tail in the right part of the figures, which explains the high values of the probability of (unwanted) matching between the ham emails and unrelated ham and spam emails (dashed green line of the Figure 6.4). From Figures 6.3(a), 6.3(b) and 6.4 we can also see that the obfuscation doesn't impact the the probability of (unwanted) matching between the ham emails and unrelated ham and spam emails.

From Figures 6.3(c) and 6.3(d) we can see that, upon applying the negative selection, most of the ham digests that cause unwanted matching between the ham emails and unrelated ham and spam emails are deleted (remaining NCV similarity between unrelated emails is bellow the detection threshold).

Actually we observe no matching between ham emails and unrelated ham and spam emails from the database DB (solid red line of the Figure 6.4) overlaps with x axes). However, when one type of output is not observed in repeated binary experiment (in our case matching of unrelated emails upon applying the negative selection), that doesn't mean that the probability of that event is zero. It might be very low to be detected with a relatively small number of experiment repetitions. The upper value of the 95% confidence interval to which the real value of the probability belongs may be computed using the following formula [59]:

$$CI_{upper} = 1 - (\alpha/2)^{(1/n)} \qquad [1]$$



(a) (b)

(c) (d)

Figure 6.3: Impact of Negative Selection on misdetection of ham: NCV histograms of (unwanted) random matching of ham emails to emails in the collaborative-detection database DB.

In our case $\alpha = 0.05$ (as we want to compute 95% confidence interval), and $n = 800$ (as we compare 20 ham emails to 40 emails from the database DB). We obtain $CI_{upper} = 0.0046$. The lower limit of the confidence interval is equal to 0. In order to more precisely estimate the real value of the probability, larger number of comparisons should be performed. However the determined CI already shows that negative selection causes a large decrease of the probability for ham emails to match unrelated ham and spam emails from DB.

Impact of Negative Selection on Spam Bulk Detection

From Figure 6.5(a) we see that even under strong obfuscation (ratio=800%) the matching between the emails from the same bulk is very strong (high email-to-email NCV values), i.e. the considered digests are very resistant to the increased obfuscation (this has already been shown in Chapter 5).

From Figure 6.5(b) we see that turning on the negative selection does not have visible effects on the matching between the spam emails from the same bulk.

For the used detection NCV thereshold 90, in cases with and without negative

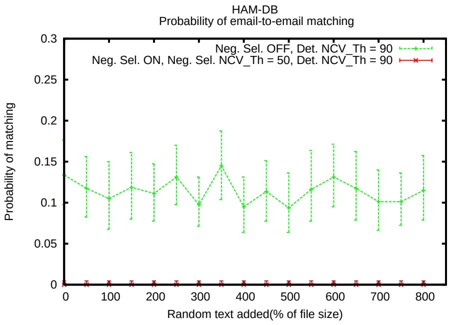
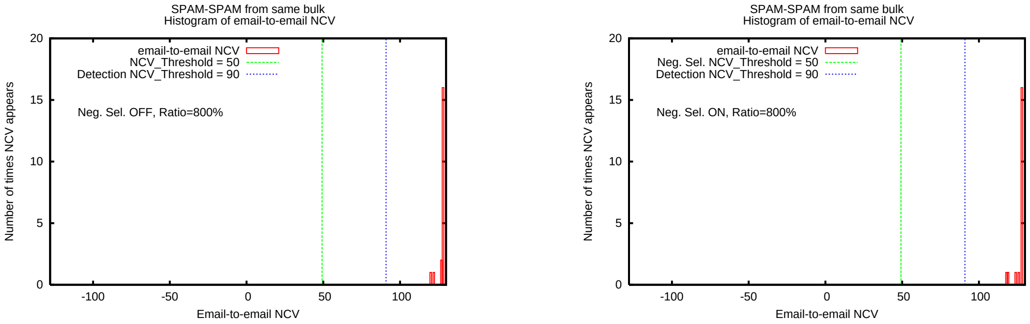


Figure 6.4: Impact of Negative Selection on misdetection of ham: the probability of (unwanted) random matching of ham emails to emails in the collaborative-detection database DB.



(a) (b)

Figure 6.5: Impact of negative selection on detection of spam bulk: NCV histograms of matching spam emails to the emails from the same bulk.

selection, all observed email-to-email NCV values are above the threshold and the observed matching ratio is equal to 1 (so we do not plot this value).

6.2.3 Transforming (estimated) Email-to-email Matching Probabilities Into FP and TP

To determine whether an email is spam or not based on its observed bulkiness, the number of the emails that it matched in the database DB should be compared to a bulkiness threshold. If this number is above the threshold, that is indication the email is probably spam, else it is probably ham.

If the number of the emails in the database DB is equal to N, and the probability of matching between a ham and un unrelated ham or spam email is equal to p, and assuming the result of email comparisons between a ham email and the emails from the database are approximately independent and identically distributed binary variables, the probability of the query score for a ham email to be above the bulkiness threshold th, which is at the same time the probability of misdetection of ham emails, is equal to:

$$FP = 1 - BinoCdf(th, N, p), \qquad [2]$$

where BinoCdf is the well known binomial cumulative distribution function.

For example if we consider the case of N=100000, for the two values of p that we obtained in the performed experiments (see Figure 6.4): p1=0.1 without negative selection and p2=0.0046 with negative selection, we obtain the probability of ham misdetection in function of the bulkiness threshold as shown in the Figure 6.6 (the curves are computed and shown only for the probabilities that are not bellow the precision of the computer).

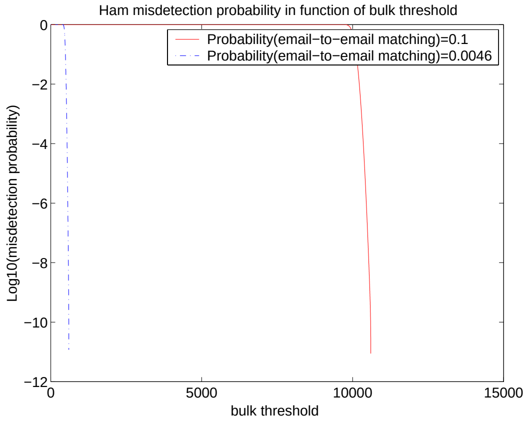


Figure 6.6: Determining bulkiness threshold that satisfies a low ham misdetection requirement: Ham misdetection probability in function of bulkiness threshold.

We see that, in order to achieve very small ham misdetection probability, negative selection allows use of about 20 times smaller bulkiness threshold. This directly translates into approximately 20 times smaller initial portion of the bulk that will not be filtered until enough evidence is collected. In the case with negative selection the real probability of matching between ham emails and unrelated ham or spam emails might be even smaller the used value p2=0.0046 (we used the upper limit of the confidence interval), the gain might be even bigger.

6.2.4 Expected behavior under other Spammer Models

It would be interesting to evaluate the negative selection under different spammer models, especially under those that decrease the probability p of matching between emails from the same bulk (obfuscation usually do not impact the probability of matching between ham and unrelated ham and spam emails). In that case the initial part of the bulk that would not be filtered (using bulkiness threshold that provides small ham misdetection) would be approximately increased by factor $1/p$.

6.3 Conclusion

In this chapter we demonstrate how by use of the negative selection algorithm the unwanted random matching between unrelated emails may by decreased at least by an order of magnitude, while preserving the same good matching between the emails from the same bulk. We also show how this translates into an order of magnitude (at least) less undetected bulky spam emails, under the same ham misdetection requirements.

As the negative selection mechanism achieves its effect through the decrease of unwanted matching between the digests from unrelated emails and can be added as a module, its benefit should hold if added to other digest-based collaborative detection schemes (including peer-to-peer schemes and the schemes that use various additional inputs (e.g. user feedback) and algorithms as opposed to a simple counting of bulkiness).

Chapter 7

Routing misbehavior: stationary self

In this chapter we show how we apply the negative selection AIS algorithm for detecting routing misbehavior in a mobile ad hoc network. We also investigate effect of adding to the negative selection one more AIS algorithm, namely clonal selection.

Ad hoc networks are those in which nodes collaboratively build routing information and use it to forward packets for each other, thus achieving multi-hop communication. We consider the case of a mobile but stationary ad hoc network.

We evaluate the system within a simulator and show impact of some parameters on its behavior.

7.1 Introduction

7.1.1 Problem Statement: Detecting Misbehaving Nodes in DSR

Mobile ad-hoc networks are self organized networks without any infrastructure other than end user terminals equipped with radios. Communication beyond the transmission range is made possible by having all nodes act both as terminals and information relays. This in turn requires that all nodes participate in a common routing protocol, such as Dynamic Source Routing (DSR) [39]. A problem is that DSR works well only if all nodes execute the protocol correctly, which is difficult to guarantee in an open ad-hoc environment.

A possible reason for node misbehavior is faulty software or hardware. In classical (non ad-hoc) networks run by operators, equipment malfunction is known to be an important source of unavailability [38]. In an ad-hoc network, where routing is performed by user provided equipment, we expect the problem to be exacerbated. Another reason for misbehavior stems from the desire to save battery power: some nodes may run a modified code that pretends to participate in DSR but, for example, does not forward packets. Last, some nodes may also be truly malicious and attempt to bring the network down, as do Internet viruses and worms. An extensive list of such misbehavior is given in [5]. The main operation of DSR is described in Section 7.1.3. In our simulation, we implement faulty nodes that, from time to time, do not forward data or route requests, or do not respond to route requests from their own cache.

We consider the problem of detecting nodes that do not correctly execute the DSR protocol. The actions taken after detecting that a node misbehaves range from forbidding to use the node as a relay [48] to excluding the node entirely from any participation in the network [5]. In our work we focus on the detection of misbehavior and do not discuss the details of actions taken after detection.

However, the actions do impact the detection function through the need for a secondary response. Indeed, after a node is disconnected (boycotted) because it was classified as misbehaving, it becomes non-observable. Since the protection system is likely to be adaptive , the 'punishment" fades out and redemption is allowed [6]. As a result, a misbehaving node is likely to misbehave again, unless it is fixed, for example by a software upgrade. We call primary [resp. secondary] response the classification of a node that misbehaves for the first [resp. second or more] time; thus we need to provide a secondary response that is much faster than primary response.

We chose DSR as a concrete example, because it is one of the protocols being considered for standardization for mobile ad-hoc networks. There are other routing protocols, and there are parts of mobile ad-hoc networks other than routing that need misbehavior detection, for example medium access control protocols. We believe the main elements of our method would also apply there, but a detailed analysis is for further work.

7.1.2 Traditional Misbehavior Detection Approaches

Traditional approaches to misbehavior detection [48, 5] use the knowledge of anticipated misbehavior patterns and detect them by looking for specific sequences of events. This is very efficient when the targeted misbehavior is known in advance (at system design) and powerful statistical algorithms can be used [7].

To detect misbehavior in DSR, Buchegger and Le Boudec use a reputation system [5]. Every node calculates the reputation of every other node using its own first hand observations and second hand information obtained from others. The reputation of a node is used to determine whether countermeasures against the node are undertaken or not. A key aspect of the reputation system is how second hand information is used, in order to avoid false accusations [5].

The countermeasures against a misbehaving node are aimed at isolating it, i.e., packets will not be sent over the node and packets sent from the node will be ignored. In this way nodes are stimulated to cooperate in order to get service and maximize their utility, and the network also benefits from the cooperation.

Even if not presented by its authors as an artificial immune system, the reputation system in [5, 7] is an example of a (non-bio inspired) immune system. It contains interactions between its healthy elements (well behaving nodes) and detection and exclusion reactions against non-healthy elements (misbehaving nodes). We can compare it to the natural *innate* immune system (Section 2.3), in the sense that it is hardwired in the nodes and changes only with new versions of the protocol.

Traditional approaches miss the ability to learn about and adapt to new misbehavior. Every targeted misbehavior has to be imagined in advanced and explicitly addressed in the detection system. This is our motivation for using an artificial immune system approach.

7.1.3 DSR: basic operations

DSR is one of the candidate standards for routing in mobile ad hoc networks [39]. A "source route" is a list of nodes that can be used as intermediate relays to reach a destination. It is written in the data packet header at the source; intermediate relays simply look it up to determine the next hop.

DSR specifies how sources discover, maintain and use source routes. To discover a source route, a node broadcasts a route request packet. Nodes that receive a route request, add their own address in the source route collecting field of the packet, and then broadcast the packet, except in two cases. The first case is if the same route request was already received by a node; then the node discards the packet. Two received route requests are considered to be the same if they belong to the same route discovery, which is identified by the same value of source, destination and sequence number fields in the request packets. The second case is if the receiving node is destination of the route discovery, or if it already has a route to the destination in its cache; then the node sends a route reply message that contains a completed source route. If links in the network are bidirectional, the route replies are sent over the reversed collected routes. If links are not bidirectional, the route replies are sent to the initiator of the route discovery as included in a new route request generated by answering nodes. The new route requests will have the destination be the source of the initial route request. The node that initiates an original route request gets usually more route replies, every containing a different route. The replies that arrive earlier than others are expected to indicate better routes, because for a node to send a route reply, it is required to wait first for a time proportional to the number of hops in the route it has as the answer. If a node hears that some neighbor node answers during this waiting time, it supposes that the route it has is worse then the neighbor's one, and it does not answer. This avoids route reply storms and unnecessary overhead.

After the initiator of route discovery gets a first route reply, it sends data over the obtained route. While packets are sent over the route, the route is maintained, in such a way that every node on the route is responsible for the link over which it sends packets. If some link in the route breaks, the node that detects that it cannot send over that link should send error messages to the source. Additionally it should salvage the packets destined to the broken link, i.e., re-route them over alternate partial routes to the destination.

The mechanisms just described are the basic operation of DSR. There are also some additional mechanisms, such as gratuitous route replies, caching routes from forwarded or overheard packets and DSR flow state extension [39].

7.2 Design of Our Detection System

7.2.1 Overview of Detection System

Every DSR node implements an instance of the detection system, and runs it in two phases. In an initial phase, the detection system learns about normal behavior of the nodes with respect to the DSR protocol. During this phase, the node is supposed to be in a protected environment in which all nodes behave well. From received or overheard packets, the node observes the behavior of its neighbors and represent it by the antigens (Section 7.2.3). At the end of this learning phase, the nodes runs the negative selection process and creates its antibodies, which we call detectors (Section 7.2.3). In general terms, the first phase implements a special form of supervised learning, where only positive cases are used for training.

After the learning is done, the node may leave the protected environment and enter the second phase where detection and classification are done. In this phase, the node may be exposed to misbehaving nodes. Detectors are used for checking if newly collected antigens represent the behavior of good or bad nodes. If an antigen, created for some neighbor during some time interval, is detected by any of the detectors, the neighbor is considered to be suspicious in that time interval. If there are relatively many suspicious intervals for some node, that node is classified as misbehaving (Section 7.2.4). In the node that did detection, this classification event triggers the clonal selection process (Section 7.2.5). In this process the node adapts its detectors to better detect experienced misbehavior. This results in a better response if the same or similar misbehavior is encountered again. In general terms, the second phase implements a form of re-enforcement learning.

The detailed algorithm is given in the appendix.

7.2.2 Mapping of Natural IS Elements to Our Detection System

For the representation of self-nonself and for the matching functions, we start from the general structure proposed by Kim and Bentley [40], which we adapt to our case.

The elements of the natural IS used in our detection system are mapped as follows:

- Body: the entire mobile ad-hoc network
- Self Cells: well behaving nodes
- Nonself Cells: misbehaving nodes
- Antigen: Sequence of observed DSR protocol events recognized in sequence of packet headers. Examples of events are “data packet sent”, “data packet

received", "data packet received followed by data packet sent", "route request packet received followed by route reply sent". The sequence is mapped to a compact representation as explained in Section 7.2.3.

- Antibody: a pattern with the same format as the compact representation of antigen (Section 7.2.3).
- Chemical Binding: binding of antibody to antigen is mapped to a "matching function", as explained in Section 7.2.3.
- Bone Marrow: Antibodies are created during an offline learning phase. The bone marrow (protected environment) is mapped to a network with only certified nodes. In a deployed system this would be a testbed with nodes deployed by an operator; in our simulation environment, this is a preliminary simulation phase.

7.2.3 Antigen, Antibody and Negative Selection

Antigens

Antigens could be represented as traces of observed protocol events. However, even in low bit rate networks, this rapidly generates sequences that are very long (for a 100 seconds observation interval, a single sequence may be up to 1 Gbit long), thus causing generation a large number of patterns prohibitive. This was recognized and analyzed by Kim and Bentley in [40] and we follow the conclusions, which we adapt to our case, as we describe now.

A node in our system monitors its neighbors and collects one protocol trace per monitored neighbor. A protocol trace consists of a series of data sets, collected on non-overlapping intervals during the activity time of a monitored neighbor. One data set consists of events recorded during one time interval of duration Δt ($\Delta t = 10$s by default), with an additional constraint to maximum N_s events per a data set ($N_s = 40$ by default).

Data sets are then transformed as follows. First, protocol events are mapped to a finite set of primitives, identified with labels. In the simulation, we use the following list:

A= RREQ sent
B= RREP sent
C= RERR sent
D= DATA sent and IP source address
is not of monitored node
E= RREQ received
F= RREP received
G= RERR received
H= DATA received and IP destination address
is not of the monitored node

A data set is then represented as a sequence of labels from the alphabet defined above, for example

$$l_1 = \texttt{(EAFBHHEDEBHDHDHHDHD,..)}$$

Second, a number of "genes" are defined. A gene is an atomic pattern used for matching. We use the following list:

Gene1= #E in sequence
Gene2= #(E*(A **or** B)) in sequence
Gene3= #H in sequence
Gene4= #(H*D) in sequence

where #('sub-pattern') is the number of the sub-patterns 'sub-pattern' in a sequence, with * representing one arbitrary label or no label at all. For example, #(E*(A **or** B)) is the number of sub-patterns that are two or three labels long, and that begin with E and end with A or B. The genes are used to map a sequence such as l_1 to an intermediate representation that gives the values of the different genes in one data set. For example, l_1 is mapped to an antigen that consists of the following four genes:

$$l_2 = \begin{pmatrix} 3 & 2 & 7 & 6 \end{pmatrix}$$

Finally, a gene value is encoded on 10 bits as follows. A range of the values of a gene, that are bellow some threshold value, is uniformly divided on 10 intervals. The position of the interval to whom the gene value belongs gives the position of the bit that is set to 1 for the gene in the final representation. The threshold is expected to be reached or exceeded rarely. The values above the threshold are encoded as if they belong to the last interval. Other bits are set to 0. For example, if the threshold value for all the four defined genes is equal to 20, l_2 is mapped to the final antigen format:

$$l_3 = \begin{pmatrix} 0000000010 & 0000000010 & 0000001000 & 0000001000 \end{pmatrix}$$

There is one antigen such as l_3 every Δt seconds, for every monitored node, during the activity time of the monitored node. Every bit in this representation is called a "nucleotide".

Antibody and Matching Function.

Antibodies have the same format as antigens (such as l_3), except that they may have any number of nucleotides equal to 1 (whereas an antigen has exactly one per gene). An antibody matches an antigen (i.e. they are cognate) if the antibody has a 1 in every position where the antigen has a 1. This is the same as in [42] and is advocated there as a method that allows a detection system to have good coverage of a large set of possible nonself antigens with a relatively small number of antibodies.

The genes are defined with the intention to translate raw protocol sequences into more meaningful descriptions.

Negative Selection

Antibodies are created randomly, uniformly over the set of possible antibodies. During the off-line learning phase, antibodies that match any self antigen are discarded (negative selection).

7.2.4 Node Detection and Classification

Matching an antigen is not enough to decide that the monitored node misbehaves, since we expect, as in any AIS, that false positives occur. Therefore, we make a distinction between detection and classification. We say that a monitored node is **detected** (or "suspicious") in one data set (i.e. in one interval of duration Δt) if the corresponding antigen is matching any antibody. Detection is done per interval of duration Δt (=10s by default). A monitored node is **classified as "misbehaving"** if the probability that the node is suspicious, estimated over a sufficiently large number of data sets, is above a threshold. The threshold is computed as follows.

Assume we have processed n data sets for this monitored node. Let M_n be the number of data sets (among n) for which this node is *detected* (i.e. is suspicious). Let $\theta_{\max}$ be a bound on the probability of false positive detection (detection of well behaving nodes, as if they are misbehaving) that we are willing to accept, i.e. we consider that a node that is detected with a probability $\leq \theta_{\max}$ is a correct node (we take by default $\theta_{\max} = 0.06$). Let α (=0.0001 by default) be the classification error that we we target. We classify the monitored node as misbehaving if

$$\frac{M_n}{n} > \theta_{\max}(1 + \frac{\xi(\alpha)}{\sqrt{n}}\sqrt{\frac{1-\theta_{\max}}{\theta_{\max}}}) \tag{7.1}$$

where $\xi(\alpha)$ is the $(1-\alpha)$-quantile of the normal distribution (for example, $\xi(0.0001) = 3.72$). As long as Equation (7.1) is not true, the node is classified as well behaving. With default parameter values, the condition is $\frac{M_n}{n} > 0.06 + 0.88\frac{\xi(\alpha)}{\sqrt{n}}$. The derivation of Equation (7.1) is given in the appendix.

7.2.5 Clonal Selection

The clonal selection mechanism with negative selection works as follows. A fraction W (by default 15%) of the best scored detectors of the node that receive signal 1b enter the clonal selection process, in which each of them will produce one new additional detector. Every detector that enters the process is cloned; the clone is mutated, and if it matches some of the self-antigens in the node's collection, it is eliminated (negative selection test). This is repeated until one mutated clone is generated that survives the negative selection test. Mutation is defined as random flipping bits of the detector, which occurs with some small probability p (by default 0.1). The newly generated detectors are substituted to the same number of this node's detectors; the detectors that had the worse score in the detection of misbehaving node that triggered the process are eliminated.

For example, let d be a node's detector that is selected to undergo clonal selection. A possible clonal selection process scenario is as follows.

$$d = (\ 1100010110 \quad 1110110010 \quad 0010100000 \quad 1000001101\ )$$

From the set (S) of self antigens that the node has collected during the learning phase, let a_1 and a_2 are two examples:

$$a_1 = (\ 0000000010 \quad 0000000010 \quad 0000001000 \quad 0000000100\ )$$
$$a_2 = (\ 0000000010 \quad 0000000010 \quad 0000001000 \quad 0000001000\ )$$

We see that d does not match a_1 or a_2 (the matching rule is given in Section 7.2.3); it does not match any of the antigens from S, by the rule how it is generated. Let d_1 be the first mutated clone of d, which happens to differ from d in 3 bits:

$$d_1 = (\ 1100010110 \quad 1110110011 \quad 0010101000 \quad 1000001001\ )$$

Because d_1 matches a_2, d_1 is deleted and another mutated clone d_2 is created, which happens to differ from d in 4 bits:

$$d_2 = (\ 1100010110 \quad 1100110010 \quad 0011100000 \quad 0000011101\ )$$

We see that d_2 does not match a_1 or a_2, and assuming that it does not match any other antigen from the set S, d_2 becomes a valid detector, and will substitute a badly scored one from the set S.

Because the length of the detectors is 40 bits and the mutation probability per bit is 0.1, mutated clones will differ from original detectors from which they are created in 4 bits in average, before negative selection. When detectors are created in learning phase they differ in average in 20 bits before negative selection. This means that clonal selection increases the percentage of the detectors that are similar but slightly different to good scored ones. It is intuitively clear from these facts, and is confirmed by the experiments, that the clonal selection improves the detection results.

7.2.6 System Parameters

In this implementation, the default values of parameters (Table 8.1) are chosen from extensive pilot simulation runs, as a compromise between good detection results and a small memory and computation usage by the detection system.

7.3 Simulation Results

7.3.1 Description of Simulation

Experimental Setup

We have implemented our detection system in Glomosim [83], a simulation environment for mobile ad-hoc networks. We use the simulator's DSR code and modify

Table 7.1: Detection System Parameters

Parameters	**Default values**
maximal number of self antigens collected for learning	450
maximal time for collecting self antigens for learning	200 s
number of detectors	300
number of genes in an antigen	4
number of nucleotides per gene	10
max. number of protocol events recorded in a data set	40
maximal time for collecting a data set (an antigen)	10 s
accepted misbehavior threshold $\theta_{\max}$	0.06
targeted classification error ratio α	0.0001
percentage of detectors selected for clonal selection	0.15
probability of mutation per bit	0.1

it only to allow nodes' misbehavior. The detection system code that we add can be run in two versions: with or without clonal selection mechanism.

We simulate a network of 40 nodes. The simulation area is a rectangle with the dimensions of 800 m x 600 m. The nodes are initially deployed on a grid with the distance between neighbors equal to 100 m. The mobility model is random way-point. The speed of nodes is a parameter, and we set it to be constant for one simulation run. The radio range is 380 m. Traffic is generated as constant bit-rate, with packets of length 512 bytes sent every 0.2-1 s.

We run two series of experiments:

- (Without clonal selection:) In the first set of experiments, clonal selection is disabled. The initial set of detectors generated during the learning phase is used unchanged during the detection and classification phase.
- (With clonal selection:) Then we run a second set of experiments, now with clonal selection. Every experiment consists of four consecutive phases, in order to obtain primary and secondary responses. The first phase is learning an initial set of detectors (as in the first experiment). The second phase is detection and classification, but now the detectors start to change and adapt to misbehavior during the phase. This allows us to measure the primary response metrics. In the third phase, nodes do not misbehave, and the system forgets about previous detections, but the set of detectors obtained at the end of the second phase is kept unchanged. The fourth phase is again a detection and classification phase, with the same misbehavior as in the second phase, but the initial set of detectors is now different (it is the set that is achieved at the end of the second phase). This gives the secondary response metrics. The conditions are otherwise the same as in the first set of experiments.

We performed 20 independent replications of all experiments, in order to obtain good 90% confidence intervals.

For all the simulations, the parameters have the default values (Table 8.1), unless otherwise mentioned.

Misbehavior

is implemented by a modifying the DSR implementation in Glomosim. We implemented two types of misbehavior: (1) non-forwarding route requests and non-answering from its route cache and (2) non-forwarding data packets. A misbehaving node does not misbehave for every packet. In contrast, it does so with fixed probabilities, which are also simulation parameters (default values are 0.6). In our implementation, a node is able to misbehave with different probabilities for the two types of misbehavior. In the simulation we always set both misbehavior probabilities to the same value (what is not necessary). The number of misbehaving nodes is also a simulation parameter (default value is 5).

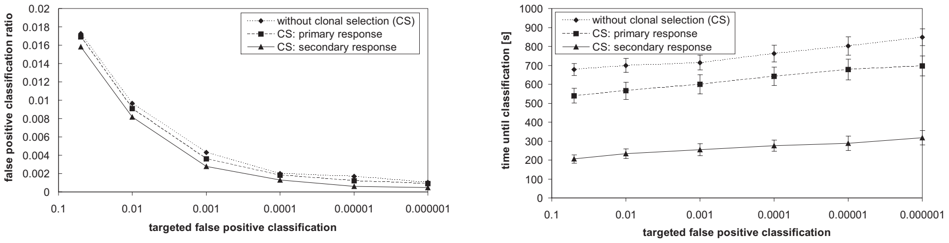
Performance Metrics

We show simulation results with the following metrics:
Average Time until Correct Classification: the time until a given node, that is running the detection algorithm, classifies a given misbehaving node as misbehaving for the first time (after a sufficiently large number of positive detections occurred, see Equation (7.1)), averaged over all pairs $\langle$node i that is running the detection algorithm, node j that is classified as misbehaving by node $i\rangle$. When we talk about **"response time"** of the detection system, we refer to this metric.
True Positive Classification Ratio: the percentage of misbehaving nodes that are classified as misbehaving.
False Positive Classification Ratio: the percentage of well behaving nodes that are mistakenly classified as misbehaving.

7.3.2 Simulation results

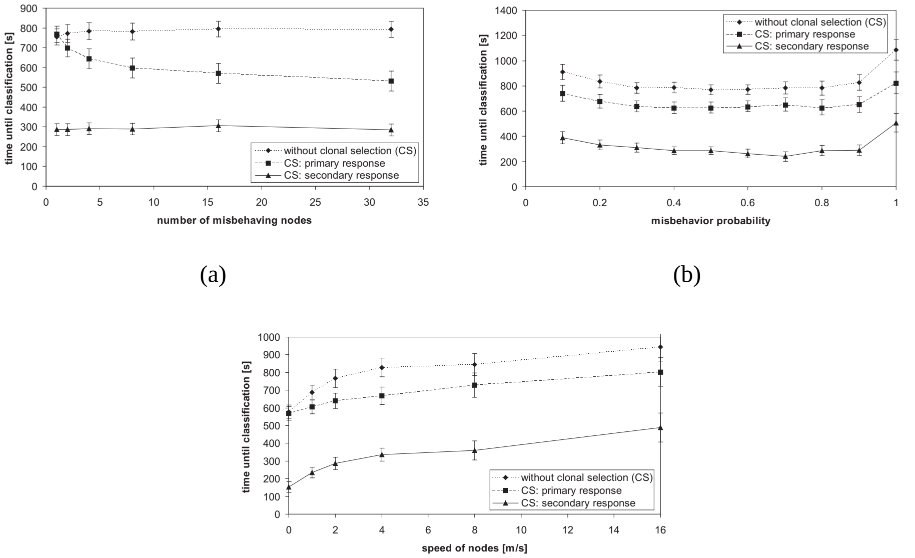
Classification capabilities. For all simulation runs, all misbehaving nodes are detected and classified as misbehaving by all their neighbors in the static case and by all nodes in the cases with mobility. The main effect on other classification metrics is by the parameter α, the classification error tolerance (Figure 7.1). By decreasing the value of α, the false positive classification ratio decreases quickly to very small values, below the value 0.002 (Figure 7.1(a)), causing a relatively small increase in the time needed for true positive classification (Figure 7.1(b)). This indicates that it is possible to choose α in the order of 0.0001 and obtain both a very small value of false classification ratio and a small time until classification.

Impact of clonal selection. Clonal selection has a significant effect on the secondary response time, which is decreased by a factor $3 - 4$ (Figure 7.1(b)). The false positive classification ratio is also decreased a little (Figure 7.1(b)). Another



(a) (b)

Figure 7.1: The AIS main metrics: (a) Average ratio of misclassification for well behaving nodes (false positive) and (b) average time until correct classification for misbehaving nodes versus target false positive classification probability α, for tree cases: without clonal selection (CS), with CS-primary response, with CS-secondary response. Comment: true positive classification ratio is equal to 1, what is not shown on the graphs.



(a) (b)

(c)

Figure 7.2: Behavior of the detection system with parameters change: Time until true positive classification with respect to: (a) number of misbehaving nodes; (b) misbehavior probability; (c) speed of nodes. In all cases target misclassification probability parameter α is set 0.0001. We do not put obtained false positive ratios on the graphs, because they vary in the vary small range between 0.001 and 0.003.

positive effect of clonal selection is a slight decrease in primary response time (Figure 7.1(b), Figure 7.2). This slight decrease is more pronounced when the number of misbehaving nodes is large. This is because it is likely that a node that sees a specific misbehaving node for the first time was already exposed to other misbehaving nodes before. If there is only one misbehaving node, this effect does not occur (Figure 7.2(a)).

Effect of other parameters. Figure 7.2 shows that parameters other than α have a limited effect on the time until classification. The effect of these parameters on the false positive classification ratio is even smaller. The values of obtained false positive classification ratios vary in the very small range between 0.001 and 0.003 (as mainly determined by the value of the parameter α ($\alpha = 0.0001$)) and this is the reason we do not show them on the graphs.

7.4 Conclusions

In this chapter we show that learning and good classification by use of negative selection AIS algorithm is possible for the considered protocol misbehavior. However, time needed by the system to classify the behavior of the node is very high, if a very small probability of miss-classification is required. Without additional mechanisms, node could simply misbehave and change its identity e.g. each 100s, making the probability to be detected low.

Also, it should be mentioned the representation that we use (sequence of observed protocol events) doesn't capture many possible misbehavior of the node. E.g. instead of sending a large packet, a node could simulate that by sending a small one - the representation of the node does not capture such a misbehavior. Increasing the number of nucleotides of an antigen, in order to represent more possible misbehavior types could easily lead into the known problem of low efficiency of the used negative selection.

Chapter 8

Routing Misbehavior: Non-Stationary Self

In the previous chapter we use the negative selection and clonal selection AIS algorithms, basically following the self-nonself AIS approach. In this chapter we exploit the danger theory based AIS approach, and extend the system from the previous chapter by adding to it the danger signal, an innate AIS mechanism explained in Section 3.4.4. The danger theory approach is expected to be more suitable for non-stationary environments (the two approaches are defined and compared in Section 3.3). We integrate the danger signal with the adaptive AIS part in a common way, i.e. use it for activating and clonally selecting antibodies produced by the negative selection algorithm. Additionally, we try a non-standard use of the danger signal - for providing dynamically the self examples used in the negative selection.

8.1 Unsolved AIS Problems That We Address in This Chapter

Eliminating need for preliminary learning in protected environment. Most of the AISs require a set of self examples to be collected in advance in order to start producing detectors. To collect self examples, the system to be protected must be run in a protected environment, in which nonself is not present. In most of the cases in practice, such a protected environment is either inconvenient or impossible to provide.

Capability to learn changing self. Even if an initial set of self examples is available, but a dynamic updating of this set is not provided, the AIS will not operate correctly if self changes over time. The problem is that detectors produced in the process of negative selection will be self-reactive. False positives caused by these detectors can be decreased by requiring the existence of a correlated danger signal in order to have detection, but this makes AISs more sensitive to the wrong danger signals. Fully reliable danger signals are often difficult to find in concrete

AIS applications, which is one of the reasons why the danger signal in not much used in the AIS literature. This would especially be the issue in distributed environments, for example in our misbehavior detection problem. We beleive that for a low false-positive detection, both a successful dynamic description of self and a danger signal should be used.

The problem of dynamically providing self examples in an automated way (without human feedback) is not much investigated in the related literature. The only known system that provides dynamic description of self (without a human feedback) is that of Somayaji and Forrest [72]. Their system achieves tolerance to new normal behavior in an autonomous way, but it works only under the assumption that new misbehavior always shows new patterns that are more grouped in time than new patterns of normal behavior; if the patterns of a new misbehavior are first sparsely introduced in the system, the system will become tolerant to that misbehavior. For their concrete application, as well as for many other AIS applications, such an assumption does not hold.

Kim and Bentley [42, 43, 44] define and investigate in detail the dynamic clonal selection algorithm. They show that the algorithm enables an AIS to learn changing self if the examples of the current self are provided, but they do not tackle the problem of providing the self examples in an automated way. We believe that adding a component to an AIS that solves this problem would enable the AIS to autonomously learn about and defend a changing protected system.

Mapping from matching to detection. Matching between an antigen and an antibody is not enough to cause detection and reaction in the HIS [73, 27] (antigens are proteins that cover the surface of self and nonself cells; antibodies are proteins produced by the HIS, capable of chemically binding to nonself antigens). The clustering of the matches on the surface of an immune cell and an additional danger signal are required for detection. These additional requirements can be viewed as detection control mechanisms aimed at decreasing false positives. These mechanisms are still not appropriate in existing AISs. The high false-positives detection rate is a common unsolved problem of many AISs, although it seems not to be so with the HIS.

8.2 Our AIS Approach for Protecting Dynamic Self

Our goal is to make an AIS that, analogously to its natural counterpart [50], automatically learns and detects both already encountered and new nonself, but becomes tolerant to any previously unseen self.

Our AIS learns the dynamic self, produces the detectors and does the detection using four AIS concepts: “virtual thymus”, “clustering”, “danger signal” and “memory detectors”.

Virtual Thymus. “Virtual Thymus” (VT) is a novel concept that we introduce in [66]. It uses observed antigens and generated DSs (the concept of the DS is defined below) and continuously provides self antigen examples that represent the dynamic self of the protected system. It solves the problem of learning a changing self and eliminates the need for a preliminary training in a protected environment. Apart from the negative selection, the way our VT works does not have analog in the existing theories of the HIS. It is important to mention that it is not yet clear to the immunologists how the repertoire of the antigens presented in the thymus during negative selection is formed (see the last paragraph of Section 2.5.3).

Clustering. Clustering is also a novel concept that we introduce in [66]. It maps matches between the detectors and antigens into the detection decisions in a way that constrains false-positive detection probability and minimizes the time until detection under this constraint. The term clustering comes from immunology, where it denotes grouping of matches between antibodies and antigens that is required for recognition of a pathogen.

Danger Signal. The Danger Signal (DS) is generated when there is a damage to the protected system. It correlates the damage with the antigens that are in some sense close to the damage, showing that these antigens might be the cause for the damage. We use the DS as a control mechanism for the detection and clonal selection, with the aim to decrease the false positives, which is already proposed but not implemented in the existing AISs. This use of the DS, and the way how it is generated is analogous to the DS and Antigen Presenting Cells (APC) in Matzinger's model [50] of the HIS. As mentioned above, we also use the DS to implement VT, which is a novel use of the DS. We have to mention that this use of the DS doesn't have its analog in the existing theories and models of the HIS.

Memory Detectors. As already mentioned in Section 8.1, the detectors that prove useful in the detection undergo the clonal selection process and become memory. They provide a fast response to the already experienced nonself. Clonal selection and memory detectors are well investigated in the related literature [42, 43, 44, 17].

8.3 Overview the AIS that we build

8.3.1 AIS Building Blocks

The four main AIS concepts that we use (Virtual Thymus, Clustering, Danger Signal and Memory Detectors) are implemented within the six building blocks (Figure 8.1), at every network node.

The Mapping and Danger Signal (DS) blocks are interfaces to the system to be protected and to AISs in other nodes. The mapping block transforms observed routing protocol events into internal AIS representation of the behavior (Antigens). The DS block transforms the experienced degradation of the communication quality of this node into the danger signal, and exchanges the signal with other nodes. The Virtual Thymus (VT) uses danger signal to dynamically select self examples from the collected antigens. It uses the selected self antigens for negative selection of the detectors produced by the Bone Marrow block. The Clonal Selection block

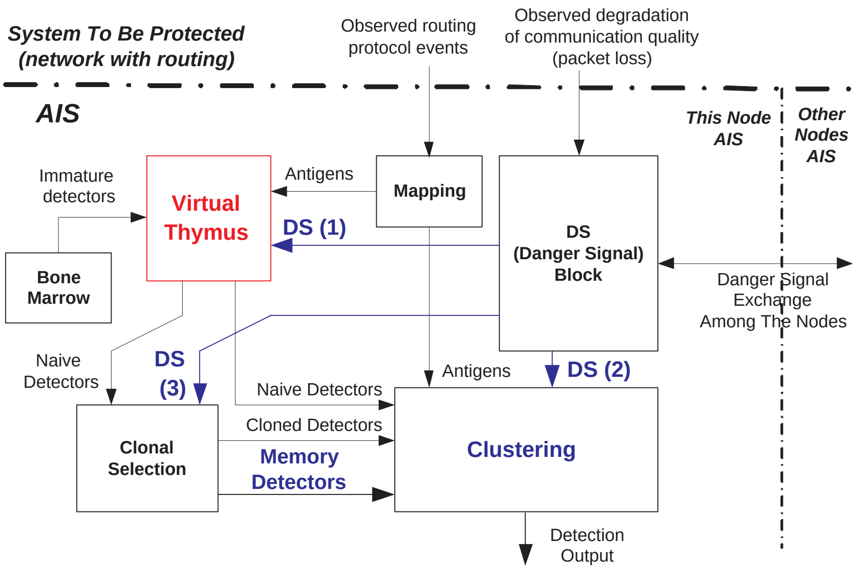


Figure 8.1: AIS building blocks in one network node. *Note:* DS(1), DS(2) and DS(3) are all the same danger signal, indexing only emphasizes three different uses of it.

multiplies and refines useful detectors, if they get costimulated by the danger signal. The Clustering block defines whether matching between the detectors and the observed antigens results in the detection of the corresponding node or not. The result depends on the numbers of observed and matched antigens for that node, on the types of the detectors that matched (whether they are memory or not), and on the presence of correlated danger signals.

8.3.2 How The AIS Works.

Here we explain how the AIS works by putting a node equipped with the AIS in the network scenarios typical for invocation of certain blocks or concepts. For the details, please see the pseudo code of the six AIS building blocks (Appendix B) and the values of the used parameters (Table 8.1 in Section 8.4).

Bootstrap. When a node joins the network for the first time, it does not have any preloaded knowledge or data examples about normal behavior or misbehavior in the network. It starts its AIS in the *initial self learning phase*. In this phase the node observes antigens and uses the VT to collect self examples. It uses the DS block to provide DSs needed for correct functioning of the VT. In this phase Clonal Selection and Clustering blocks are turned off. This is done in order to avoid initial false positives until the number of the self antigen examples stabilizes. Once the number of self examples reaches a threshold, the node turns on the Clonal Selection and Clustering blocks, i.e., it enters *detection phase* and stays in this phase all the time. During the detection phase the node continues to observe antigens, produce DSs, and dynamically update self examples.

If the node leaves the network and joins it again, it may start with saved self examples and enter detection phase immediately.

It is important to notice that there are no assumptions on the behavior of the other nodes during the initial learning phase (some of them may misbehave). Also, the node starts to participate in routing and use the network to communicate already in the initial learning phase.

Observing Antigens. The node continuously observes routing protocol events for all its neighbors and temporarily records them within subsequent time intervals of predefined length (a system parameter). Two nodes are neighbors if they are within radio range of each other. Examples of protocol events are route request received, data packet received, data packet forwarded. Protocol events collected for one neighbor within the last time interval are compressed into a compact form, called antigen. The antigen represents the behavior of the observed node in that time interval. The details of mapping from the observed protocol events to the antigens are given in Section 8.3.4.

Generating, Transmitting and Receiving the DS. When a node communicates with another node, it normally receives acknowledgments for its data packets, that are sent by the destination. The danger signal is generated by a node when it experiences a packet loss, i.e., when it does not receive the acknowledgment (packet loss is seen as a damage to the protected system). The DS is then sent over the route on which the packet loss took place, and received by all the nodes that can overhear it. A typical scenario is shown on the Figure 8.2.

The DS contains the information about the (approximate) time of the packet loss, and about the nodes on the route over which the lost packet was sent. So, the receivers of the danger signal are able to correlate it with the collected antigens that are close in time and space (on the same route) to the experienced damage (i.e. to the packet loss).

It is possible that the DS is generated even if there is no any misbehavior, because a packet was lost due to a excessive collision of radio signals, or break of the used route due to the mobility, which normally happens in mobile ad hoc networks. Our system is robust to such false DSs, as explained in the next three paragraphs.

Use of the DS in The VT: Achieving The Dynamic Self. To define dynamic self in our system, we extend the notion of self from the behavior specified by the routing protocol (DSR, for example) to any interactive node behavior that does not have negative effects on the normal network trafficking, i.e. does not cause packet losses. As a packet loss, we count any case in which the packet does not arrive

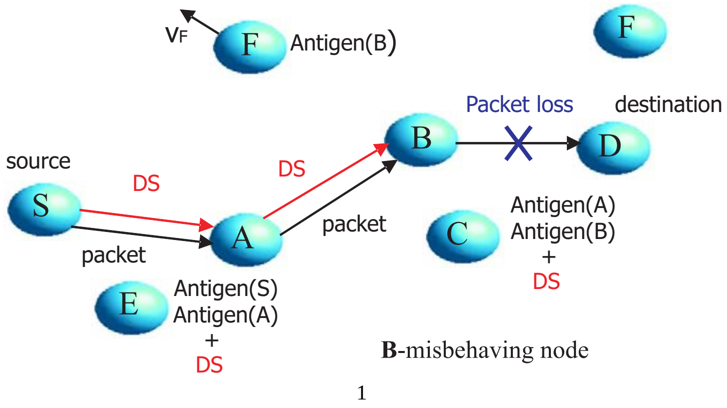


Figure 8.2: **Generating, transmitting and receiving the danger signal (DS)**: node S is sending data packets to D over an established route; there is a packet dropped by node B; the source S of the packet does not receive the acknowledgment from the destination D; S generates and sends the danger signal over the route on which the loss took place; the nodes that receive the danger signal are A, B, C and E. In this scenario, node F moves away and do not receive the danger signal.

at the destination, or the acknowledgment from the destination about receiving the packet does not reach the source, or there is a high delay in any of these packets.

How the node learns the dynamic self is shown on the Figure 8.3.

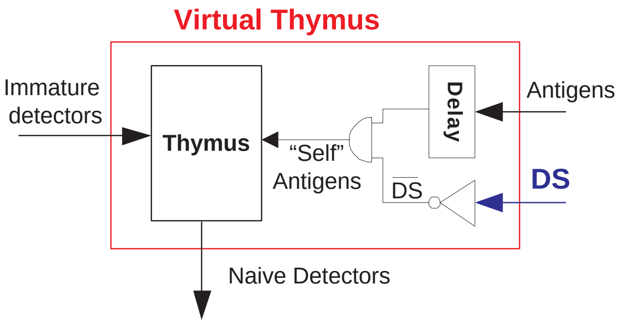


Figure 8.3: **Virtual Thymus:** Self to be used in the dynamic process of clonal selection is dynamically collected from the observed antigens that are not correlated with any danger signal.

Whenever the node observes a new antigen, it sends it towards the thymus with a delay that is large enough to receive all the DSs that could be related to that antigen. If none of potentiality received DSs correlates this antigen to a damage experienced in the network, the antigen is considered as self and passed into the thymus if needed. Once the number of self antigen examples reaches its stable value (it is the end of the initial self learning phase), a new example is added only when an old one leaves the thymus.

Producing New Naive Detectors in VT. New naive detectors are produced in the VT all the time in a process of dynamic negative selection.

In the initial self learning phase, new random detectors are generated by the Bone Marrow and negatively selected with all collected self antigens. When new self antigen is collected, all existing detectors are negatively selected only with that new self antigen. The detectors are not used for the detection in this initial phase. Note that the method is computationally more convenient and not more costly then collecting all self examples and then creating and checking all the detectors.

After the initial learning phase, a new self antigen that enters the thymus is used to negatively select only newly created immature detectors, but not those that already passed the negative selection (and became naive).

The number of naive detectors stays approximately constant. A naive detector that does not score enough matches to undergo clonal selection dies after a fixed life time (a parameter). In parallel, if the number of naive detectors is smaller then a maximum value (which is a parameter), new naive detectors are generated by the negative selection in the VT. The naive detectors that score many matches and receive costimulations by the DSs for that matches undergo clonal selection and become memory. The detector is first duplicated. One copy becomes a memory detector, while another is hypermutated and becomes new naive detector.

Finite-Time Antigen Presentation in The VT and Wrong DSs. A collected self-antigen example is used in the VT for a finite time, then it leaves the VT and is replaced with fresh one. This aging and continuous updating provides the VT with the current version of the "dynamic self", and makes AIS less sensitive to nonself antigens that are mistakenly selected as self examples (because the DS was mistakenly absent). If a nonself antigen is mistakenly picked up and used to represent self, new naive detectors that match it will not be produced until this antigen leaves the VT. During this time period, the detectors produced before the nonself antigen is picked up are still able to match that and similar antigens.

Wrong DSs that are mistakenly or intentionally generated may prevent self antigen to be used as an example. This is not a problem for the VT as there would be other self examples that are collected. It is also possible to counteract misbehaving nodes that try to maliciously generate many fake DSs and prevent other nodes from collecting enough self examples (an attack to the AIS). In our case such a malicious node may be detected because it sends too many DSs, though it should change a path that is not working for its data packets if it really has a reason to send danger signals (in current AIS simulation this type of misbehavior is not implemented). In general, as this is a known attack specific to VT, a specific predefined detector(s) may be designed for it in a concrete AIS.

It is important to note that the joint use of the "dynamic self" and danger signal brings new quality to the robustness of the AIS: false positive matches by detectors are partially compensated by the absence of the danger signal; temporary presence of wrong danger signal does not cause detection if there is no corresponding false positive matching by current detectors whose production was indirectly controlled by danger signals in the past; if the danger signal is currently mistakenly absent to costimulate detection for a nonself antigen, memory detectors for which correct education was assisted by the danger signal in the past can do detection.

Matching. Once the detection phase starts, every new observed antigen is checked with the existing detectors for matching. Matches for the antigens observed for one node are temporarily stored and used for clustering and detection (next paragraph).

Clustering And Use of The DS for Detection. In order to have detection of a node as misbehaving, our AIS requires more antigens of that node to be matched by the detectors. After every new antigen observed for a node, the matching information is used together with the matching information previously collected for that node within some time window, and a decision is made if there is detection or not (we call this clustering; the term comes from immunology, see Section 2.5.4).

The method is time adaptive. If most of the observed antigens are matched, fewer observations are required for detection. The method may also be tuned to achieve a tradeoff between probability of false-positive detection and the time until detection. The details of the method, including the clustering formula, are given in Section 7.2.4. In practice, clustering is done for the matches from the finite time window. The size of the window can always be chosen (experimentally) large enough to cover majority of the detection cases. Then, intuitively, the clustering formula approximately holds.

The matches by the naive detectors are clustered separately from the matches by the memory detectors. The parameters used for the clustering differ in the two cases as well. As the memory detectors are better educated, they require smaller number of clustered matches for the same false positive probability, and so provide a faster detection.

Additional control for decreasing false positives is to require a correlated danger signal for observed antigen, in order to count the matching of that antigen for clustering.

Similar methods are used in the HIS, clustering of the matches on the surface of an immune cell and an additional danger signal are required for detection ([73, 27]).

Use of The DS for Clonal Selection: Producing Memory Detectors. The detectors that have many matches costimulated by the danger enter the process of clonal selection. They are cloned, i.e. multiplied and randomly modified. The clones that receive costimulation many times (above a threshold) are promoted into memory.

Use of Memory Detectors. Analogously to the HIS, our memory detectors do not require the DS to verify their matches to the antigens. Less clustering is also required for the detection, comparing to the naive detectors and their clones that still did not become memory.

Memory detectors are long lived, in order to provide fast response to the experienced misbehavior. As the number of memory detectors is limited, we keep those that are unique. From those that are not unique, we keep last used ones. Concretely, when a new memory detector is created and the number of the existing memory detectors is maximum possible, we choose an old memory detector to delete as follows. We find existing memory detectors that are able to match certain number (that is a parameter) of the latest antigens matched by the new detector before it became memory (these are similar to the new memory detector), and from these we delete the one that is not used for the longest time. If we do not find such similar ones, then we delete the one that is not used for the longest time among all the old detectors.

DS is not needed for matches by memory detectors to be counted for the detection. But a systematic absence of DSs for similar antigens matched by a memory detector means that the memory detector is probably reactive to the new self, and it is deleted. For the details of the testing see Appendix B.

8.3.3 Mapping of HIS Elements to Our AIS

The elements of the natural IS used in our detection system are mapped as follows:

- Body: the entire mobile ad-hoc network.
- Self Cells: well-behaving nodes.
- Non-self Cells: misbehaving nodes.
- Antigen: (AIS) antigen, which is a sequence of observed DSR protocol events recognized in the sequence of packet headers and represented by binary strings as explained in Section 8.3.2 and with the details given in Section 7.2.3 (representation is adopted from [44]).
- Antibody: detector; detectors are binary strings produced in the continuous processes of negative selection and clonal selection; ideally, they “match” non-self antigens (produced by misbehaving nodes) and do not match self antigens.
- Chemical binding of antibodies to antigens: “matching function” between detectors and antigens, defined in detail in Section 7.2.3.
- Detection: a node detects a neighbor as misbehaving if the node’s detectors match relatively many of the antigens produced by that neighbor (clustering) and if it receives danger signals related to those antigens.
- Clustering: clustering of matching antibodies on the immune system cell surface is mapped to the clustering of matches between detectors and antigens in time for a given observed node.
- Aging of the immune cells: finite life time of the detectors

- Necrosis and apoptosis: packet loss.
- Danger signal: the danger signal in our framework contains information about the time and nodes correlated with a packet loss.
- Antigen presenting cell: transmission of the danger signal.
- Thymus: The virtual thymus is a set of mechanisms that provide (as explained in Section 8.3.2) the presentation of the current self in the system during the continuous negative selection process.
- Memory cells: memory detectors; detectors become memory if they prove to be useful in detection; they differ from normal detectors by longer lifetime and lower clustering required for detection.

8.3.4 Mapping behavior to Antigens and Use of Clustering

The mapping used here was adopted from Kim and Bentley [44], and is the same as in the previous Chapter, where it is specified in detail. Used clustering method is also the same as in the previous chapter.

8.4 Performance Analysis

8.4.1 Analyzed Factors and Experiments

We analyze the effects of turning on/off complete components of the system on its performance metrics defined in the next section. Concretely, we analyze the effects of: (1) substitution of the preliminary learning phase in a protected environment by the virtual thymus, in case of stationary normal behavior; protected environment means that misbehavior is absent from the network; (2) use of the danger signal for detection control; (3) use of memory detectors; (4) substitution of the preliminary learning phase in the protected environment by the virtual thymus, in the case of normal behavior that changes with time. The changing self is implemented by increasing the amount of the data generated by the nodes at the middle of the simulation (after 30 minutes of simulated time).

Clustering is used in all the experiments, as we already have shown its advantage over simple matching in Section 7.2.4.

The values of the system parameters used in the simulation are given in Table 8.1. The same default values are used in all the experiments. We have found the default values by pilot runs. To learn about the parameters, study the pseudo code of the AIS building blocks (Appendix B) and how the AIS works (Section 8.3.2).

8.4.2 Performance Metrics

The metrics we use are: (1) time until detection of a misbehaving node; (2) true-positive detection, in form of the distribution of the number of nodes which detect a misbehaving node; (3) false-positive detection, in form of the distribution of the number of nodes which detect a well-behaving node. The metrics are chosen from

Table 8.1: AIS Parameters

Parameter	Default Value
AntigenCollectionTime	10 s
DelayBufferSizeMax	1200
AntigenTowardsVTMin	70 s
StoringTimeDS	11 s
AntigPresentTimeVT	250 s
MaxNumberOfAntigensVT	1200
MaxNumberNaive	1000
MaxNaiveTime	500 s
ThresholdCS	20
ProbaPerBit	0.08
MemoryGroupingParameter	1
MemoryTestTriger	15
NumberOfTestSetsMax	30
MemoryDetTestSize	20
MemoryConfidenceMax	0.5
DetByMemoryTimeWindow	18 s
DetByNaiveTimeWindow	40 s
ThetaMemory	0.01
AlphaMemory	0.0001
ThetaNaive	0.06
AlphaNaive	0.0001

a reputation system perspective; we see the use of a reputation system [6] as a way to add a reactive part to our AIS.

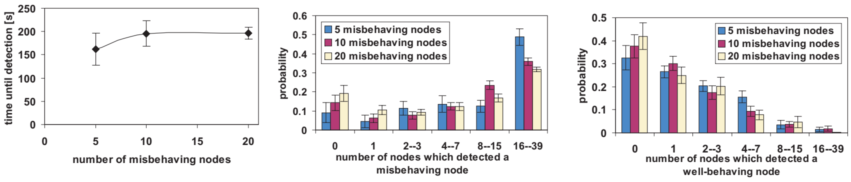
8.4.3 Simulation: General Settings and Assumptions

The simulation is done in Glomosim network simulator [83]. There are 40 nodes in the simulation, of which 5-20 nodes are misbehaving. Mobility is the random way point, speed is 1m/s, without pauses. The simulation area is 800x1000 m, and the radio range is 355 m. A misbehaving node exhibits both types of misbehavior: 1) it does not forward data packets or 2) it does not answer or forward route request messages; when a misbehaving node has a chance to misbehave, it does it with certain probability (0.6 by default), that is also a parameter.

Note that the clustering rule given by Equation (7.1) holds for an infinite time simulation, in which every misbehaving node is eventually detected after a long time (zero false negatives) buy every other node in the network (because of the finite simulation area and RWP mobility). When we stop the simulation, for every encountered and not yet detected misbehaving node we count one false negative.

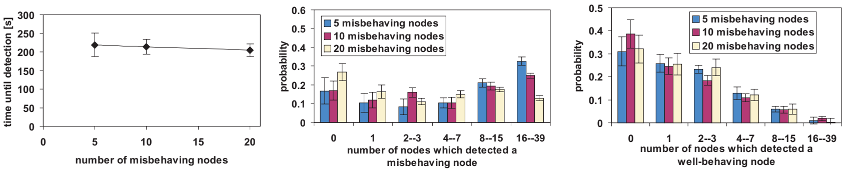
8.4.4 Simulation Results

All the results are average values of 20 runs, with 90 % confidence intervals for the mean values.



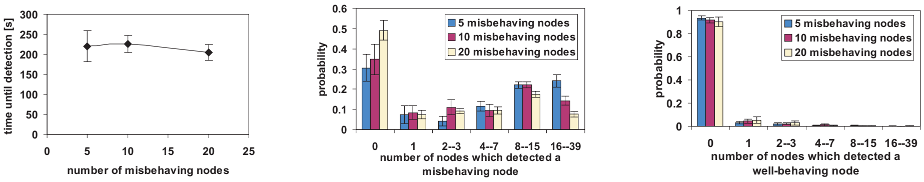
(a) (b) (c)

Figure 8.4: Use of the preliminary learning phase: (a) time until detection, (b) correct detections and (c) misdetections.



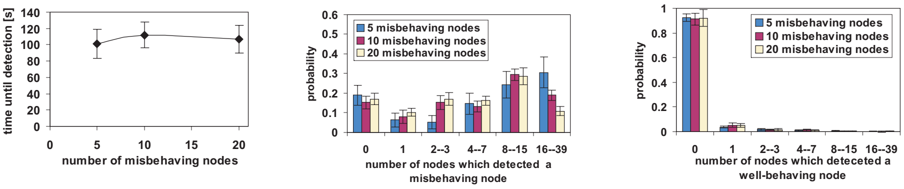
(a) (b) (c)

Figure 8.5: Use of the virtual thymus instead of the preliminary learning phase: (a) time until detection, (b) correct detections and (c) misdetections.



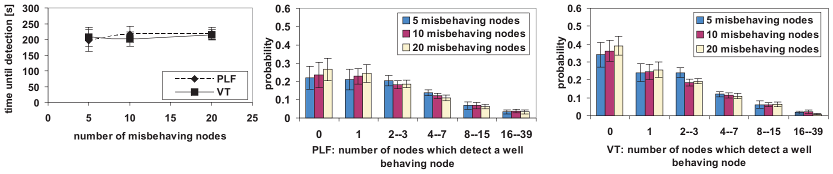
(a) (b) (c)

Figure 8.6: Use of the danger signal for detection decision making: (a) time until detection, (b) correct detections and (c) misdetections.



(a) (b) (c)

Figure 8.7: Use of memory detectors: (a) time until detection, (b) correct detections and (c) misdetections.



(a) (b) (c)

Figure 8.8: Virtual thymus versus preliminary learning phase, the effect of change of normal behavior during the use of the AIS: (a) time until detection, (b) misdetections: preliminary learning phase, (c) misdetections: virtual thymus.

Virtual thymus versus preliminary learning phase in the protected environment: From the Figures 8.4 and 8.5 we see that the preliminary learning phase can be substituted by the virtual thymus. Time until detection and the false positives are similar in both cases, while the false negatives are slightly worse in the case with the virtual thymus. This result proves that VT enables the system to learn the protected system self instead of using provided self examples.

The danger signal used for detection control has a large impact in decreasing false positives (Figures 8.6(c) and 8.7(c)).

The use of the memory detectors significantly decreases the time until detection (Figure 8.7(a)), and also improves true-positive detection (Figures 8.6(b) and 8.7(b)). Such impact of the memory is already shown in the related literature [43]. In our case it is the direct consequence of the lower value of the clustering parameter Θ_{max} used for memory detectors (compared to the naive detectors).

Better response to evolving self: From Figure 8.8 we see that the VT outperforms the solution with the preliminary learning phase in the case of self that has changed during the simulation. For the preliminary learning phase solution, false positive probability increased substantially and become larger then for the VT solution, when the stable normal behavior is substituted by the normal behavior that changes in the middle of the simulation. The reason is that with the preliminary learning phase the initial set of self examples is continued to be used for the negative selection after the self has changed. With VT, examples of new self are learned after the changing point, which directly impacts the false positives.

8.5 Conclusions

From our results we conclude that the examined mechanisms: the virtual thymus, the clustering, the danger signal and the use of memory detectors can be successfully applied to our problem.

The use of the “virtual thymus” brings two qualitative advantages compared to the standard AIS: (1) it eliminates the need for the preliminary learning phase in the protected environment, i.e., enables use of the AIS in applications for which a protected environment is not possible; (2) it provides the AIS with the ability to learn self that can change with time; this provides smaller false positives comparing to the case with preliminary training phase, as shown in our experiment (in which self changes in the middle of the simulation).

Clustering achieves low false positives by increasing the time until detection, but this price is paid only when a misbehavior is experienced for the first time. The later encounters are solved faster by the memory detectors that require less clustering.

We find a simple danger signal in our system, and show how it is useful in controlling detection decisions (decrease of false positives). We also show the use of the DS to implement the VT (the central component of our solution).

8.6 Discussion and Future Work

We expect that the combination of the four concepts bring some additional advantages that have not been analyzed by the simulation.

As explained in Section 8.3.2, in cases of not previously seen self, the self-learning ability of the virtual thymus should improve robustness of the self-tolerance to the wrong danger signals, compared to the thymus with the predefined self examples collected in the preliminary learning phase. This feature is especially important in applications in which a reliable danger signal is difficult to provide (this is the case with our system, see Section 8.3.2). The effect of wrong danger signals caused by misbehaving nodes and impact of the VT on this effect are not experimentally evaluated here, and it remains as future work. But we should mention that even though wrong DSs (both missing and incorrect) are inherently present (they are not intentionally generated) in our system, the AIS still works well.

We have defined the genes manually, in the design phase. It is possible to automate this process by calculating the correlations of automatically generated gene-candidate pairs on an observed self behavior. Gene candidates can also be formed automatically from the set of observable protocol events. Our solution for collecting self behavior examples in an unprotected environment makes this method more promising.

Chapter 9

Conclusion

9.1 Findings

In this section we summarize the most important findings from the work presented in the thesis.

Finding 1 ([69], Chapter 5)**: A randomized-position fixed-length sampling of email text makes digest-based collaborative spam-bulk detection resistant to some important obfuscation techniques used by spammers.** We found only one published work that uses such digests for collaborative spam detection [84], but it uses exact matching instead of similarity matching between the digests. To the best of our knowledge, we are the first to evaluate such digests under similarity matching, and demonstrate their advantage for email spam filtering over the digests previously evaluated in the literature [11] (which are created from predefined email parts of variable length). This way of producing digests should be also useful for other cases of distributed bulk detection (e.g. distributed detection of fast-spreading computer viruses and worms[1].)

Finding 2 ([70], Chapter 6)**: The negative selection AIS mechanism improves the filtering performance of digest-based collaborative spam detection by an order of magnitude or more.** More precisely, under the fixed requirements for good email misdetection, when added to a simple and standard bulkiness-detection scheme, the negative selection decreases the amount of undetected spam bulk by an order of magnitude or more. In the experiments where we demonstrate this, we use the improved representation mentioned in *Finding 1*.

To the best of our knowledge, we are the first to propose and evaluate the use of negative selection within a collaborative spam detection scheme[2]. As the negative

[1] For the viruses and worms, the digests may be produced e.g. from the executable-file content or from observed-execution logs.

[2] ”Negative assertions” used by the Cloudmark’s collaborative filtering scheme (summarized in Section 4.1.1) is the only mechanism we are aware of that is somewhat similar to the negative selection; there are however fundamental differences in the details and provided functionality between the ”negative assertions” mechanism and our use of negative selection, as discussed more in detail in Section 4.1.2

selection mechanism achieves its effect through the decrease of unwanted matching between the digests from unrelated emails and can be added as a module, its benefit should hold if added to other digest-based collaborative detection schemes (including peer-to-peer schemes and the schemes that use various additional inputs (e.g. user feedback) and algorithms as opposed to a simple counting of bulkiness).

Finding 3 ([64, 63], Chapter 4)**: We found an innovative way to efficiently incorporate the negative selection AIS mechanism within our AIS for antispam.** The inefficiency of the negative selection algorithm that assumes random creation and the negative selection of adaptive antibodies has been already recognized [40, 77] and addressed in the AIS literature. Instead of creating new candidate antibodies randomly, Kim and Bentley [42] create them from the existing antibodies by use of the clonal selection algorithm, and they apply the negative selection (as an operator) to eliminate self-reactive antibodies. With the same goal of avoiding the inefficient creation of new antibodies, Secker et al. [71] create new antibodies by using the clonal selection algorithm and do not apply the negative selection at all (as not required by the application they consider).

Our solution is most similar to that of Chao and Forrest [9], who create new antibodies from the antigens that receive a co-stimulation (the co-stimulation in their system can be seen as a danger signal) from the users receiving unwanted information[3]. We create new candidate antibodies from all the newly observed antigens, and we apply the negative selection on these. As discussed in Section 4.1.2, this allows our antibodies to receive costimulation not only by a danger signal, but also by a PAMP signal (and potentially by ”safe” signals).

It should also be mentioned that our use of negative selection is not equivalent to the direct (pre-)classification of the antigens (digests created from the incoming emails) by their comparison to the good-email digest examples (which would be similar as done by Twycross [79] in his AIS for intrusion detection). Indeed, our AIS creates antibodies, processes them further, exchanges (some of) them with other collaborating systems, and remembers (some of) them for future use.

Finding 4 ([64, 63], Chapter 4)**: The AIS for antispam with integrated innate and adaptive parts is very good both in detecting spam and in avoiding the misdetection of good emails. Our experiments show that both the innate and adaptive components are crucial for achieving the good performance.** The very promising true and false positive detection results become more intuitive, as well as a possible placement of our AIS within the antispam solutions employed in practice, when our AIS is compared to other antispam techniques that use similar mechanisms. We recall here that our AIS incorporates an evaluation of the spam bulkiness (in our AIS terminology this is an innate mechanism called PAMP signal) and that it incorporates the feedback from the email users about unfiltered spam (in our AIS terminology this is an innate mechanism called danger signal).

[3] Secker et al. [71] create the initial set of antibodies and Ayara et al. [3] create part of new antibodies in a way similar to that of Chao and Forrest, i.e. from the antigens confirmed to be nonself.

It also takes into account the (dynamically built) content profiles of the users, by use of the negative selection algorithm (an adaptive AIS mechanism), applied in a non-standard way in the process of creating antibodies (spam detectors).

If we look at other antispam techniques that use similar mechanisms, the situation is as follows. Other AIS-based antispam solutions, such as the work of Oda and White [55, 56] and of Bezerra et al. [4], use adaptive AIS algorithms and sometimes incorporate feedback from the users but do not exploit bulkiness of spam. This could be the reason our AIS outperforms the AISs of Oda and White and of Bezerra et al., in both the true and false positive detections. The numbers should however be taken with some reservation, due to unidentical experimental settings. Whereas, the classic collaborative detection schemes like those presented by Damiani et al. [11] and by Zhou et al. [84] do exploit the spam bulkiness, but they do not benefit (see *Finding 2*) from the possibility of using the negative selection or a similar mechanism[4].

The antispam solutions employed in practice achieve good performance usually by using independently multiple local processing techniques (e.g. Bayesian or neural networks based; the above mentioned AISs could be incorporated as well) and network based techniques (IP-address based black and white lists, previously mentioned collaborative content-based techniques, social network based methods), and then combining the results obtained from each technique. Some of these techniques also use the feedback from the users. Although it is clear (from the results of Chapter 6) that the antispam solutions used in practice can benefit at least from equipping the classical bulkiness evaluation with the negative selection mechanism, the promising results obtained in Chapter 4 for our AIS that integrates processing of the multiple spam aspects (content, bulkiness, user feedback) suggest that the AIS could potentially be a good replacement for multiple techniques that separately exploit these spam aspects.

Finding 5 ([64, 63], Chapter 4)**: While building our AIS for antispam we noticed that the AIS approach, due to its data representation, seems to be very suitable for the collaborative distributed information processing tasks, and especially for collaborative distributed detection tasks.** The AIS uses a data representation that keeps the local knowledge in the form of the independent units of summarized (processed) information that can be exchanged between the collaborating spam detection systems (which is not the case with e.g. neural networks). This allows for the processing performed locally by one collaborating system to be reused by other collaborating systems. Apart from increasing the quality of the information exchanged for collaboration, this also lessens its amount.

The possible advantages of this unique AIS feature (the ability of creation and exchange of independent units of summarized knowledge) should however be evaluated by means of comparative experiments.

Finding 6 ([47, 67, 66, 68], Chapters 7 and 8)**: A straightforward application of the AIS concepts and algorithms for detecting routing misbehavior in**

[4] See footnote 2.

mobile ad-hoc networks leads to the system that mimics well most of the effects observed in the HIS. However, its effectiveness and practical usability are very constrained. A reason for this is that a detailed representation of all the relevant data in the exchanged packets (relevant regarding the correctness of the nodes behavior) would be of a very high dimension, for which use of the negative selection is computationally infeasible (as reported in the related work [40, 77]). The restricted representation that we use captures only one part of the possible misbehavior types.

We did not try to fix the usability problem by building a better (high-dimensional) representation and by using the negative selection in the way it was done for the antispam problem. The main reason is that we did not find a simple generic way to build a good representation for this application, and that we did not want to spend too much time crafting manually the representation for one of many candidate protocols for routing in mobile ad hoc networks.

In the AIS for routing misbehavior detection we faced an additional problem: when a node observes its neighboring node, it sees only a part of the space and packet exchanges relevant to that neighbor, thus it receives low quality (incomplete) input information about the behavior of its neighbors.

9.2 Re-Evaluation of the Used Design Principles

In this section we re-evaluate the design principles that were announced in the Introduction of the thesis, with respect to the obtained results (findings).

9.2.1 Routing Misbehavior Detection Case

Following the most commonly used AIS approach, we were able to build the AIS for routing misbehavior detection that mimics well most of the effects observed in the HIS, e.g. a faster secondary reaction to the already encountered misbehavior. Due to the constrained observability of nodes by each other, and thus blurred input data, the detections based on short observations were not reliable. Applying a statistical decision-method based on multiple, consequent, short observations allowed for the noise to be filtered and for the classification to be reliable, though a bit slow. The principle used that requires multiple, in time clustered, detections to occur before a node is classified as misbehaving is similar to the clustering of detections between antigens and antibodies on the surface of immune system cells that may activate the immune system cell and so verify the detection of the pathogen. In our case this principle proved to be crucial for achieving a good classification.

Regarding other used AIS algorithms, we show by experiments that each of them improves performances of the system. However these contributions are relatively small when compared to the basic solution that uses the negative selection and clustering of the short-observations based detections. Though we were able to identify and implement an appropriate danger signal (appropriate with respect to the analogy with the HIS, as also noticed by Freitas and Timmis in their recent review on AISs [24]), the positive impact of the danger signal proved to be very modest. The reason is probably the fact that the used danger signal is not very precise in identifying the antigens related to the damage. It points only to the path on which a packet is lost and not to the exact node that is responsible for the packet loss.

9.2.2 Collaborative Spam Detection Case

In the case of the AIS for antispam, the approach of combining both the innate and adaptive mechanisms prove to be crucial for the good performance of the system. We also show that important benefits are achieved by devoting special attention to the data representation.

By using an innovative (though similar to some previous solutions, see Section 4.1.2) way of incorporating the negative selection into the process of creating new antibodies, we avoid the computational constraints usually faced by the AIS systems that apply the negative selection on the randomly generated antibodies. As in our case the representation (data space) is of a very high dimension (256 bits), the random generation of antibodies would make the system very inefficient. On the contrary, a rough estimation from the simulations (Section 4.4) suggests that the AIS for antispam with incorporated negative selection is computationally efficient.

9.3 Contributions

9.3.1 Solving Real Problems

Based on the obtained simulation results, our AIS for antispam seems to be a very promising candidate for a real-life implementation (Chapter 4, [64, 63]). The system design also seems to be appropriate for filtering other types of Internet communications.

We additionally show that an existing digest-based scheme for collaborative spam filtering, used within practical antispam solutions, could directly benefit from the use of our improved representation of digests and from the addition of the negative selection [69, 70]. These results are expected to hold for other collaborative spam detection schemes as well.

Within the course of this thesis, we also develop AntispamLab [65, 2] - a tool for realistic and automated testing of email spam filters. Prototyping our AIS for antispam and comparing it to other real-life antispam solutions by using the AntispamLab tool thus seems to be the most natural continuation of our work.

9.3.2 Contribution to the AIS field

Our contributions to the field of AISs are as follows. We show an example of the application (collaborative spam filtering), which is especially suitable for an integrated use of both innate and adaptive AIS mechanism, and we build an appropriate AIS in which both the innate and adaptive mechanisms are crucial for the overall good performance. The AIS is also a good example of a very strong analogy to the HIS for most of the used mechanisms, and especially for the used innate signals (danger signal and PAMP signal).

We also provide an innovative example of incorporating the negative selection in a computationally efficient way, by generating new antibodies on-the-fly when new antigens are observed, and applying the negative selection on these antibodies. This way of generating new antibodies is the most similar to the idea presented in [9] by Chao and Forrest, though in their system new antibodies are generated only when a co-stimulation for a new antigen is provided by the user when he receives unwanted information and declares it as such (which in a way corresponds to a damage and generation of the danger signal). When the antibodies are generated as in our system, they may be also co-stimulated by the PAMP signal (an important costimulation signal in our system).

9.4 Overall Conclusion

The success of applying the AIS approach depends heavily on the application used. In the case of routing misbehavior we faced various difficulties that prevented us from building a usable system. The main obstacle to apply the AIS approach successfully was the difficulty of building a good representation of the observed behavior that would capture well the misbehavior of the nodes.

On the contrary, the AIS approach shows to be very suitable for collaborative spam filtering. In particular, the following features of the problem make it a very appropriate candidate to be addressed by the AIS approach: need for openness to unknown senders (creators of content, initiators of the communication), bulkiness in receiving spam (many recipients are usually affected by the same spam content), tolerance of the system to a small damage (to small amounts of unfiltered spam), possibility to implicitly or explicitly and in an inexpensive way obtain a feedback from the recipients about the damage (about spam that they receive), need for strong tolerance to wanted (non-spam) content.

We were able to build an AIS for antispam that shows very promising detection results. Our design included improving the existing data representation, an innovative use of negative selection (an adaptive AIS mechanism), identification appropriate innate signals in the system, and integration of the adaptive and innate AIS parts. The experimental results shown that both the innate and adaptive parts are crucial for the good performance of the system.

Many other types of Internet communications show features similar to those listed above for the email filtering problem, which suggests that our AIS design could also be applicable for filtering these communications.

Appendix A

Derivation of Equation 7.1

We model the outcome of the behavior of a node as a random generator, such that with unknown but fixed probability θ a data set is interpreted as suspicious. We assume the outcome of this fictitious generator is iid. We use a classical hypothesis framework. The null hypothesis is $\theta \leq \theta_{\max}$, i.e., the node behaves well. The maximum likelihood ratio test has a rejection region of the form $\{M_n > K(n)\}$ for some function $K(n)$. The function $K(n)$ is found by the type-I error probability condition: $\mathbb{P}\{M_n > K(n)\}|\theta) \leq \alpha$, for all $\theta \leq \theta_{\max}$, thus the best $K(n)$ is obtained by solving the equation

$$\mathbb{P}(\{M_n > K(n)\}|\theta_{\max}) = \alpha$$

The distribution of M_n is binomial, which is well approximated by a normal distribution with mean $\mu = n\theta$ and variance $n\theta(1-\theta)$. After some algebra this gives $K(n) = \sqrt{n}\xi\sqrt{\theta_{\max}(1-\theta_{\max})} + n\theta_{\max}$, from which Equation (7.1) derives immediately.

Appendix B

Pseudo Code of The AIS Blocks from Figure 8.1

```
//All the blocks execute in parallel
//In our case, this is implemented using continues-time event based simulation
//Every Send() has a corresponding Receive() in another block(s)


// Mapping block
Initialize the variables; set timerMB=0; While (TRUE){
    if (timerMB<AntigenCollectingTime){
        Update list of the neighbors (i.e. the nodes within the radio range);
        Collect observed protocol events for every neighbor separately;
    }else{
        Transform the collected events into antigens;
        Send antigens towards Virtual Thymus,i.e. into the Delay Buffer;
        Send antigens to Clustering Block;
        timerMB=0;
    }//end if else
}//end while


// Danger Signal (DS) block
Initialize the variables; While (TRUE){
    Receive DS from other nodes that you overhear and store it;
    if (experienced own packet loss){
        Generate a DS and store it;
        Send the danger signal over the route on which the loss happen;
    }//end if
    Delete stored DSs older then StoringTimeDS;
}//end while


// Bone Marrow block
isThereProdced=0; While (TRUE){
    if (!isThereProduced){
        Produce a new random detector;
        isThereProduced=1;
    }//end if
}//end while


// Virtual Thymus (VT) block
```

```
Initialize the constants; phaseVT=Initial;
numberOfImmatureDetectors=0;
ReserveDelayBufferSpace(DelayBufferSizeMax); While (TRUE){
    Accept the antigen into the Delay Buffer if room, otherwise drop it(){
        if (there is a free space in the buffer) accept the antigen;
        else if (the oldest antigen is delayed>AntigenTowardsVTMin)replace the oldest one;
        else drop the antigen;
    }//end Accept...()
    Drop antigens from the Buffer if delayed more then AntigPresentTimeVT;
    Drop antigens from the buffer if correlated with currently stored DSs;
    if (phaseVT==Stationary){
        Delete the antigens from the VT that are presented longer then AntigPresentTimeVT(){
            do not delete more then currentNumberOfAntigensInDelayBuffer;
            delete the oldest ones;
        }//end Delete...()
        while (numberOfAntigensInVT<MaxNumberOfAntigensVT){
            Take the youngest enough delayed antigen from the the Delay Buffer, if any;
            if (the antigen is not correlated with stored DSs){
                Put the antigen into VT.
            }else{
                Delete the antigen;
            }//end if else
        }//end while
        while (numberOfNaiveDetectorsCS<MaxNumberNaive){
            Receive new immature detector;
            Negatively select new immature detector into naive;
            if (not deleted) move naive detector into Clonal Selection block;
        }//end while
    }else{
        while(numberOfImmatureDetectors<MaxNumberNaive){
            Receive new immature detector;
        }//end while
        while (numberOfAntigensInVT<MaxNumberOfAntigensVT){
            Take the youngest enough delayed antigen from the the Delay Buffer, if any;
            if (the antigen is not correlated with stored DSs){
                Put the antigen into VT.
            }else{
                Delete the antigen;
            }//end if else
            Negatively select the immature detectors in the thymus by new antigen;
            while(numberOfImmatureDetectors<MaxNumberNaive){
                Receive new immature detector;
                Negatively select new immature detector with antigens in VT if any;
            }//end while
            if (numberOfAntigensInVT==MaxNumberOfAntigensVT){
                PhaseVT=stationary;
                Change state of immature detectors into naive;
                Move naive detectors into Clonal Selection block;
                while (numberOfNaiveDetectorsCS<MaxNumberNaive){
                    Receive new immature detector;
                    Negatively select new immature detector into naive one;
                    if (not deleted) move naive detector into Clonal Selection block;
                }//end while
            }//end if
        }//end while
    }//end if else
}//end while


// Clonal Selection  block
Initialize the variables; numberOfNaiveDetectorsCS=0; While
(TRUE){
    ///create naive detectors, do clonal selection, promote naive to memory
    Delete naive detectors older then MaxNaiveTime;
    Receive next matching event info for innate detectors, from Clustering block;
    if (matchingEventIsReceived){
        if (the matching is costimulated by DS){
            score(corresponding naive detector)++;
            if (score(corresponding naive detector)>TresholdCS){
                score(corresponding naive detector)=0;
                Make a copy of the detector;
                Promote the detector into a memory detector;
                Decide which one of the old memory detectors to delete(){
                    find set S of those that can detect last MemoryGroupingParameter ...
                    ... antigens detected by the new detector before it became memory
                    if (the set S is not empty) {
                        from the set S, choose the one that is not used for ...
                        ... the longest time;
                    }else{
                        from all old detectors, choose one that is not used for ...
                        ... the longest time;
                    }//end if else
                }// end Decide...()
                Delete the chosen old memory detector;
                Do randomly-flip-bits hypermutation for the detector copy (ProbaPerBit);
            }//end if
        }//end if
        matchingEventIsReceived=0;
    }//end if
    ///test memory detectors
    Receive next matching event info for memory detectors, from Clustering block;
    if (the detector is not under testing){
        if (DS is missing for the matching){
            increase missingDSsCounter(the detector);
            if (missingDSsCounter(the detector)== MemoryTestTrigger){
                put the detector under testing;
            }//end if
        }//end if
    }else{
        // the detector is under testing
        //for this detector next matches, form sets of mutually similar antigens
        //first element of a set has a missing DS, other elements are similar to the first one
        if (the DS is missing and testing sets are not formed){
            form a testing set and put this antigen in it as a first element;
        }else if (the antigen is different from a set first element in not more then one bit){
            add the antigen to the set;
        }else if (number of the sets formed<NumberOfTestSetsMax){
            form a new testing set and put this antigen in it as a first element;
        }//end if else
        //testing a set corresponds to testing of the detector if it is probably self reactive
        if (a test set length==MemoryDetTestSize){
            if (more then MemoryDetConfidence \% of the set elements is missing DS){
                delete this detector and all its test sets;
            }//end if
        }//end if
        if (number of tested sets==NumberOfTestSetsMax/2){
            stop testing this detector and delete all its testing sets;
        }//end if
    }//end if else
}//end while


// Clustering  block
Initialize the variables; While (TRUE){
    Receive new antigen from the Mapping block;
    Match antigen by memory detectors;
    Store the matches within the time window DetByMemoryTimeWindow;
    Apply clustering for matches by memory detectors(TetaMemory, AlfaMemory);
    Match antigen by naive detectors;
    Evaluate is there is a DS that costimulates the matching;
    Store the matches within the time window DetByNaiveTimeWindow;
    Apply clustering for costimulated matches by naive detectors(TetaNaive, AlfaNaive);
}//end while
```

Publications

Conference papers

- Resolving FP-TP Conflict in Digest-Based Collaborative Spam Detection by Use of Negative Selection Algorithm
  Sarafijanovic, Slavisa ; Perez, Sabrina; Le Boudec, Jean-Yves. Accepted for CEAS-2008, The Fifth Conference on Email and Antispam, Mountain View, California, USA, August 21-22, 2008.
- Improving Digest-Based Collaborative Spam Detection
  Sarafijanovic, Slavisa ; Perez, Sabrina ; Le Boudec, Jean-Yves. In Proceedings of MSC-2008, The 2008 MIT Spam Conference, Cambridge, Massachusetts, USA, March 27-28, 2008.
- Artificial Immune System For Collaborative Spam Filtering
  Sarafijanovic, Slavisa ; Le Boudec, Jean-Yves. In proceedings of NICSO-2007, The Second Workshop on Nature Inspired Cooperative Strategies for Optimization, Acireale, Italy, November 8-10, 2007., Acireale, Italy, November 8-10, 2007. In Studies in Computational Intelligence, num. ISSN: 1860-949X, Springer Verlag, 2008.
- AntispamLab - A Tool for Realistic Evaluation of Email Spam Filters
  Sarafijanovic, Slavisa ; Hernandez, Luis ; Naefen, Raphael ; Le Boudec, Jean-Yves. In Proceedings of CEAS-2007, The Fourth Conference on Email and Antispam, Mountain View, California, USA, August 2-3, 2007.
- An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors
  Sarafijanovic, Slavisa ; Le Boudec, Jean-Yves. In Proceedings of ICARIS-2004, 3rd International Conference on Artificial Immune Systems, Catania, Italy, p. 342-356, 2004
- An Artificial Immune System for Misbehavior Detection in Mobile Ad Hoc Networks with both Innate, Adaptive Subsystems and with Danger Signal
  Sarafijanovic, Slavisa ; Le Boudec, Jean-Yves. In proceedings of AISB-2004, Symposium on The Immune System and Cognition (ImmCog-2004), Leeds, UK, p. 45-46, 2004
- An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks

Le Boudec, Jean-Yves ; Sarafijanovic, Slavisa. Presented at: Bio-ADIT 2004 (The First International Workshop on Biologically Inspired Approaches to Advanced Information Technology), Lausanne, Switzerland. In: Bio-ADIT 2004 (The First International Workshop on Biologically Inspired Approaches to Advanced Information Technology), p. 96-111, 2004

Journal papers

- An Artificial Immune System Approach with Secondary Response for Misbehavior Detection in Mobile Ad-Hoc Networks
  Sarafijanovic, Slavisa ; Le Boudec, Jean-Yves. In: IEEE Transactions on Neural Networks, Special Issue on Adaptive Learning Systems in Communication Networks, vol. 16, num. 5, p. 1076 - 1087, 2005
- An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors
  Sarafijanovic, Slavisa ; Le Boudec, Jean-Yves. In: International Journal of Unconventional Computing, vol. 1, p. 221-254, 2005

Patents

- Method to Filter Electronic Messages in A Message Processing System
  Sarafijanovic, Slavisa ; Le Boudec, Jean-Yves. US Patent No. 2008/0059590 A1, filed September 5, 2006, published March 3, 2008.

Curriculum Vitæ

Slavisa Sarafijanovic was born in 1973 in Tuzla, Yugoslavia. He earned ”Diploma Electrical Engineer in Telecommunications” degree from The Faculty of Electrical Engineering, University of Belgrade, in January 2000. He completed the Predoctoral School in Computer, Communication and Information Science at EPFL, Lausanne, 2002/2003.

Since September 2003 he has worked toward his PhD at the Laboratory for Computer Communications and Applications (LCA) at EPFL, under the supervision of Professor Jean-Yves Le Boudec. His current work is on the use of the Artificial Immune System (AIS) approach to protect communication in computer networks, with emphasis on routing misbehavior detection and email spam filtering.

Bibliography

[1] U. Aickelin and S. Cayzer. The danger theory and its application to artificial immune systems. Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS 2002), pp 141-148, Canterbury, UK.

[2] AntispamLab. Project web page. http://lcawww.epfl.ch/ssarafij/antispamlab/, March 2007.

[3] M. Ayara, J. Timmis, R. deLemos, and S. Forrest. Immunising Automated Teller Machines. In Proceedings of ICARIS 2005, the 4th International Conference on Artificial Immune Systems, Banf, Canada. LNCS 3627. pp. 404-417. C.Jacob Et. al. (Eds) 2005.

[4] G. B. Bezerra, T. V. Barra, H. M. Ferreira, H. Knidel, L. N. deCastro, and F. J. V. Zuben. An Immunological Filter for Spam. In Proceedings of ICARIS 2006, the 5th International Conference on Artificial Immune Systems, Oeiras, Portugal. LNCS 4163. pp. 446-458. Bersini and Carneiro (Eds) 2006.

[5] S. Buchegger and J.-Y. L. Boudec. Performance Analysis of the CONFIDANT protocol: Cooperation of nodes - Fairness In Distributed Ad-Hoc Networks. In *Proceedings of IEEE/ACM Symposium on Mobile Ad-Hoc Networking and Computing (MobiHOC)*, Lausanne, CH, June 2002. IEEE.

[6] S. Buchegger and J.-Y. L. Boudec. A Robust Reputation System for Mobile ad hoc Networks. Technical Report IC/2003/50, EPFL-DI-ICA, Lausanne, Switzerland, July 2003.

[7] S. Buchegger and J.-Y. L. Boudec. The Effect of Rumor Spreading in Reputation Systems for Mobile Ad-hoc Networks. In *Proceedings of WiOpt '03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks"*, Sophia-Antipolis, France, March 2003.

[8] S. Cayzer and J. Smith. Gene Libraries: Coverage, Efficiency and Diversity. In Proceedings of ICARIS 2006, the 5th International Conference on Artificial Immune Systems, Oeiras, Portugal. LNCS 4163. pp. 136-149. Bersini and Carneiro (Eds) 2006.

[9] D. L. Chao and S. Forrest. Information Immune Systems. In the Proceedings of ICARIS 2002, The First International Conference on Artificial Immune Systems, pp. 132-140, Canterbury, England, 2002.

[10] G. Coelho and F. J. V. Zuben. omni-ainet: An immune-inspired approach for omni optimization, ser. LNCS, H. Bersini and J. Carneiro, Eds., vol. 4163. Springer, pp. 294-308, 2006.

[11] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. An open digest-based technique for spam detection. In *Proceedings of The 2004 International Workshop on Security in Parallel and Distributed Systems*, San Francisco, CA, USA, September 2004.

[12] D. Dasgupta and S. Forrest. An Anomaly Detection Algorithm Inspired by the Immune System. In D. Dasgupta (Eds), Artificial Immune Systems and their Applications, Springer-Verlag, pp. 262-277, 1999.

[13] DCC. http://www.dcc-servers.net/dcc/, Feb 2008.

[14] L. N. deCastro and J. Timmis. Artificial Immune Systems: A New Computational Intelligence Approach. Springer, 2002.

[15] L. N. deCastro and F. VonZuben. The clonal selection algorithm with engineering applications. In Proceedings of GECCO 2000, Workshop on Artificial Immune Systems and Their Applications, p. 36-37, 2000.

[16] L. N. deCastro and F. VonZuben. Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation, 6(3):239-251, 2002.

[17] L. N. deCastro and F. J. VonZuben. Learning and Optimization Using the Clonal Selection Principle, IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems, 6(3), pp. 239-251.(2002).

[18] M. Ebner and et al. On the use of negative selection in an artificial immune system. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pages 957-964, New York, NY. Morgan Kaufman Publishers, San Francisco, CA.

[19] A. Edinger and C. Thompson. Death by Design: apoptosis, necrosis and autophagy. Current Opinion in Cell Biology, 16(6):663-669, 2004.

[20] F. Esponda, S. Forrest, and P. Helman. A formal framework for positive and negative detection. IEEE Transactions on Systems, Man, and Cybernetics, 34(1):357-373, 2004.

[21] S. Forrest, S. A. Hofmeyr, and A. Somayaji. Computer Immunology. Communications of the ACM, 40(10), p. 88-96, 1997.

[22] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri. Self-Nonself Discrimination in a Computer, Proceeding of 1994 IEEE Symposium on Research in Security and Privacy, Los Alamos, CA: IEEE Computer Society Press, 1994.

[23] A. A. Freitas and J. Timmis. Revisiting the Foundations of Artificial Immune Systems: A Problem Oriented Perspective. In J. Timmis, P. Bentley, and E. Hart, editors, Proceedings of the 2nd International Conference on Artificial Immune Systems, volume 2787 of Lecture Notes in Computer Science, pages 229-241. Springer, September 2003.

[24] A. A. Freitas and J. Timmis. Revisiting the Foundations of Artificial Immune Systems for Data Mining. IEEE Transactions on Evolutionary Computation 11(4) pp. 521-540 (2007).

[25] S. M. Garrett. How do we evaluate artificial immune systems? *Evol. Comput.*, 13(2):145–177, 2005.

[26] A. Gaspar and P. Collard. From GAs to artificial immune systems: Improving adaptation in time dependent optimization. In Proceedings of the Congress on Evolutionary Computation, volume 3, pages 1859-1886, Mayflower Hotel, Washington D.C., USA. IEEE Press, 1999.

[27] R. A. Goldsby, T. J. Kindt, B. A. Osborne, and J. Kuby. Immunology, 5th edition, W. H. Freeman and Company, 2003.

[28] P. Graham. A plan for spam, http://www.paulgraham.com/spam.html, 2002.

[29] J. Greensmith. The Dendritic Cell Algorithm, PhD Thesis, University of Nottingham, UK, 2007.

[30] J. Greensmith and U. Aickelin. SYN Scan Detection and the DCA, GECCO 2007, London, UK, pp 49-56.

[31] J. Greensmith, U. Aickelin, and S. Cayzer. Introducing Dendritic Cells as a Novel Immune-Inspired Algorithm for Anomaly Detection, In Proceedings of the Fourth International Conference on Artificial Immune Systems (ICARIS 2005), LNCS 3627, pp 153-167, Springer-Verlag, Banff, Canada, 2005.

[32] J. Greensmith, U. Aickelin, and J. Twycross. Articulation and Clarification of the Dendritic Cell Algorithm, Proceedings of the 5th International Conference on Artificial Immune Systems (ICARIS 2006), LNCS 4163, pp 404-417, Springer-Verlag, Oeiras, Portugal, 2006.

[33] E. Hart and P. Ross. Studies on the Implications of Shape-Space Models for Idiotypic Networks. In Proceedings of ICARIS 2004, the 3rd International Conference on Artificial Immune Systems, Catania, Sicily, Italy. LNCS 3239. pp. 413-426. G. Nicosia et al. (Eds) 2004.

[34] E. Hart and J. Timmis. Application areas of AIS: The past, the present and the future, In Proceedings of ICARIS 2005, the 4th International Conference on Artificial Immune Systems, Banff, Alberta, Canada, August 14-17, 2005.

[35] E. Hart and J. Timmis. Application areas of AIS: The past, the present and the future, Applied Soft Computing, vol. 8, no. 1, pp. 191-201, 2008.

[36] S. Hofmeyr. An Immunological Model of Distributed Detection and it's Application to Computer Security. *PhD thesis*, Department of Computer Sciences, University of New Mexico, USA, April 1999.

[37] S. A. Hofmeyr and S. Forrest. Architecture for an Artificial Immune System. Evolutionary Computation 7(1):45-68. 2000.

[38] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot. *Analysis of Link Failures in an IP Backbone*. Proceeding of IMW 2002. ACM Press. Marseille (France). November 2002.

[39] D. Johnson and D. Maltz. The dynamic source routing protocol for mobile ad hoc networks. *Internet draft, Mobile Ad Hoc Network (MANET) Working Group*, IETF, February 2003.

[40] J. Kim and P. Bentley. Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection: *Genetic and Evolutionary Computation Conference 2001* (GECCO-2001), San Francisko, pp. 1330-1337, July 7-11.

[41] J. Kim, P. Bentley, C. Wallenta, M. Ahmed, and S. Hailes. Danger is ubiquitous: Detecting malicious activities in sensor networks using the dendritic cell algorithm. In Proc. of the 5th International Conference on Artificial Immune Systems (ICARIS), LNCS 4163, p. 390-403, 2006.

[42] J. Kim and P. J. Bentley. Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Clonal Selection with a Negative Selection Operator. The *Congress on Evolutionary Computation* (CEC-2001), Seoul, Korea, pp.1244-1252, May 27-30, 2001.

[43] J. Kim and P. J. Bentley. Immune Memory in the Dynamic Clonal Selection Algorithm. *Proceedings of the First International Conference on Artificial Immune Systems* (ICARIS) Canterbury, pp.57-65, September 9-11, 2002.

[44] J. Kim and P. J. Bentley. A Model of Gene Library Evolution in the Dynamic Clonal Selection Algorithm. *Proceedings of the First International Conference on Artificial Immune Systems* (ICARIS) Canterbury, pp.175-182, September 9-11, 2002.

[45] J. Kim and P. J. Bentley. An evaluation of negative selection in an artificial immune system for network intrusion detection. In L. Spector, E. D. Goodman,

A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 1330–1337, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

[46] J. Kim, W. O. Wilson, U. Aickelin, and J. McLeod. Cooperative Automated Worm Response and Detection ImmuNe ALgorithm(CARDINAL) Inspired by T-Cell Immunity and Tolerance. In Proceedings of ICARIS 2005, the 4th International Conference on Artificial Immune Systems, Banf, Canada. LNCS 3627. pp. 168-181, C. Jacob et al. (Eds) 2005.

[47] J.-Y. Le Boudec and S. Sarafijanovic. An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks. In the Proceedings of Bio-ADIT 2004, The First International Workshop on Biologically Inspired Approaches to Advanced Information Technology), Lausanne, Switzerland, p. 96-111, 2004.

[48] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of MOBICOM 2000*, pages 255–265, 2000.

[49] P. Matzinger. The Danger Model in it's Historical Contex. *Scandinavian Journal of Immunology*, 54:4-9, 2001.

[50] P. Matzinger. Tolerance, Danger and the Extended Family. *Annual Review of Immunology*, 12:991-1045, 1994.

[51] R. Medzhitov and C. A. Janeway. Decoding the patterns of self and nonself by the innate immune system. Science, 296:298-300, 2002.

[52] G. Nicosia. Immune algorithms for optimization and protein structure prediction, PhD dissertation, University of Catania, 2004.

[53] Nilsimsa. http://lexx.shinn.net/cmeclax/nilsimsa.html, Sep 2006.

[54] R. Oates, J. Greensmith, U. Aickelin, J. Garibaldi, and G. Kendall. The Application of a Dendritic Cell Algorithm to a Robotic Classifier, In the Proceedings of ICARIS 2007, the 6th International Conference on Artificial Immune Systems, San Paulo, Brazil, 2007.

[55] T. Oda and T. White. Developing an immunity to spam. In: Genetic and Evolutionary Computation Conference, Chicago(GECCO 2003), Proceedings, Part I. Volume 2723 of Lecture Notes in Computer Science, 2003, 231-241.

[56] T. Oda and T. White. Immunity from Spam: An Analysis of an Artificial Immune System for Junk Email Detection. In Proceedings of ICARIS 2005, the 4th International Conference on Artificial Immune Systems, Banf, Canada. LNCS 3627. pp. 276-289, C. Jacob et al. (Eds) 2005.

[57] M. Oprea and S. Forrest. Simulated evolution of antibody gene libraries under pathogen selection. IEEE International Conference on Systems, Man, and Cybernetics, 11-14 Oct 1998, Volume 4, Issue , p. 3793-3798, 1998.

[58] A. S. Perlson, R. Hightower, and S. Forrest. Evolution (and learning) of v-region genes. Research in Immunology Vol. 147, pp. 202-208 (1996).

[59] A. M. Pires. Confidence intervals for a binomial proportion: comparison of methods and software evaluation. In *In Klinke, S., Ahrend, P. and Richter L. (editors), Proceedings of the Conference CompStat 2002*, August 24-28, 2002.

[60] V. V. Prakash and A. O'Donnell. Fighting Spam with Reputation Systems. In ACM Queue 3(9):36-41, 2005.

[61] Razor. Razor project web page (Sep 2006), http://razor.sourceforge.net/.

[62] M. J. Robbins and S. M. Garrett. Evaluating Theories of Immunological Memory Using Large-Scale Simulations. In Proceedings of ICARIS 2005, the 4th International Conference on Artificial Immune Systems, Banf, Canada. LNCS 3627. pp. 193-206, C. Jacob et al. (Eds) 2005.

[63] S. Sarafijanovic and J.-Y. L. Boudec. Artificial Immune System For Collaborative Spam Filtering. In the Proceedings of NICSO 2007, The Second Workshop on Nature Inspired Cooperative Strategies for Optimization, Acireale, Italy, November 8-10, 2007.

[64] S. Sarafijanovic and J.-Y. L. Boudec. Method to Filter Electronic Messages in A Message Processing System, US Patent No. 2008/0059590 A1, filed September 5, 2006, published March 3, 2008.

[65] S. Sarafijanovic, L. Hernandez, R. Naefen, and J.-Y. L. Boudec. AntispamLab - A Tool for Realistic Evaluation of Email Spam Filters, In the Proceedings of CEAS 2007, The Fourth Conference on Email and Antispam, Mountain View, California, USA, August 2-3, 2007.

[66] S. Sarafijanovic and J.-Y. Le Boudec. An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors. In the Proceedings of ICARIS-2004, 3rd International Conference on Artificial Immune Systems, Catania, Italy, p. 342-356, 2004.

[67] S. Sarafijanovic and J.-Y. Le Boudec. An Artificial Immune System Approach with Secondary Response for Misbehavior Detection in Mobile Ad-Hoc Networks. IEEE Transactions on Neural Networks, Special Issue on Adaptive Learning Systems in Communication Networks, vol. 16, num. 5, p. 1076-1087, 2005.

[68] S. Sarafijanovic and J.-Y. Le Boudec. An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors. International Journal of Unconventional Computing, vol. 1, p. 221-254, 2005.

[69] S. Sarafijanovic, S. Perez, and J.-Y. L. Boudec. Improving Digest-Based Collaborative Spam Detection, In the Proceedings of MSC2008, The 2008 MIT Spam Conference, Cambridge, Massachusetts, USA, March 27-28, 2008.

[70] S. Sarafijanovic, S. Perez, and J.-Y. L. Boudec. Resolving FP-TP Conflict in Digest-Based Collaborative Spam Detection by Use of Negative Selection Algorithm, In the Proceedings of CEAS 2008, The Fifth Conference on Email and Antispam, Mountain View, California, USA, August 21-22, 2008. (accepted).

[71] A. Secker, A. Freitas, and J. Timmis. AISEC: An Artificial Immune System for Email Classification. In Proceedings of the Congress on Evolutionary Computation, Canberra, IEEE, 2003, 131-139.

[72] A. Somayaji and S. Forrest. Automated Response Using System-Call Delays. Proceedings of the 9th USENIX Security Symposium, The USENIX Association, Berkeley, CA (2000).

[73] L. Sompayrac. How the Immune System Works, 2nd Edition. Blackwell Publishing, 2003.

[74] SpamAssassin. http://spamassassin.org/, Feb 2008.

[75] SpamAssassin-Public-Corpus. http://spamassassin.org/publiccorpus/, Feb 2008.

[76] T. Stibor, J. Timmis, and C. Eckert. On the Use of Hyperspheres in Artificial Immune Systems as Antibody Recognition Regions. In Proceedings of ICARIS 2006, the 5th International Conference on Artificial Immune Systems, Oeiras, Portugal. LNCS 4163. pp. 215-228, Bersini and Carneiro (Eds) 2006.

[77] T. Stibor, J. Timmis, and C. Eckert. On the appropriateness of negative selection defined over hamming shape-space as a network intrusion detection system. In *Congress On Evolutionary Computation – CEC 2005*, pages 995–1002. IEEE Press, 2005.

[78] D. Taylor and D. Corne. Innate and acquired immunity in real time systems. IEEE Proceedings of the Fourth International Conference onHybrid Intelligent Systems, 5-8 Dec. 2004, p. 390-395, 2004.

[79] J. P. Twycross. Integrated Innate and Adaptive Artificial Immune Systems applied to Process Anomaly Detection. Ph.D. Thesis, University of Nottingham, UK, 2007.

[80] J. P. Twycross and U. Aickelin. An Immune-inspired Approach to Anomaly Detection. Chapter in Handbook of Research on Information Assurance and Security, Idea Publishing Group, 2007.

[81] C. W. Preventing delivery of unwanted bulk e-mail, US patent 6,330,590. (2001).

[82] A. Watkins. Exploiting immunological metaphors in the development of serial, parallel and distributed learning algorithms, PhD dissertation, University of Kent, Computing Laboratory, UK, 2005.

[83] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for parallel simulation of large scale wireless networks. *Proceedings of the 12th workshop on Parallel and Distributed Simulations-PDAS'98*, May 26-29, in Banff, Alberta, Canada, 1998.

[84] F. Zhou, L. Zhuang, B. Zhao, L. Huang, A. Joseph, and J. Kubiatowicz. Approximate object location and spam filtering on peer-to-peer systems. In *Proceedings of ACM/IFIP/Usenix Int'l Middleware Conf., LNCS 2672, pp. 1-20.*